O’ZBEKISTON RESPUBLIKASI OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI
O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI
TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI NUKUS FILIALI

DASTURIY INJINIRING KAFEDRASI

Ro’yxatga olindi:
№_____
_____
_____

2018 y «___»_____

«Tasdiqlayman»
O’quv va tarbiya ishlari bo’yicha
direktor o’rinbosari
_____X Seytkamalov
«____» _____ 2018y.

“OBYEKTGA YO’NALTIRILGAN DASTURLASH TILLARI”

O’QUV-USLUVIY MAJMUA

Bilim sohasi:	300000 – Ishlab chiqarish texnik soha
Ta’lim sohasi:	330000 – Kompyuter texnologiyalari va informatika
Ta’lim yo’nalishi:	5330600 – Dasturiy injiniring

Nukus – 2018

Fanning ishchi o’quv dasturi ishchi o’quv reja va o’quv dasturiga mos ishlab chiqildi.

Tuzuvchilar:

J. Kudaybergenov – TATU Nukus filiali “Dasturiy injiniring” kafedrasi assistenti.

Q. Jalelov – TATU Nukus filiali “Dasturiy injiniring” kafedrasi assistenti.

Taqrizchilar:

f-m.f.d prof N.U Uteuliev. – TATU Nukus filiali “Dasturiy injiniring” kafedrasi mudiri.

Fanning ishchi o’quv dasturi “Dasturiy injiniring” kafedrasining 2018 yil “__” avgustdagi ___ - son yig’ilishida muhokama qilindi va fakultet ilmiy kengashiga muhokama uchun taklif qilindi.

Kafedraning qaroriga asoslanib ayrim bo’limlarning bajarilish ketma-ketligi yoki ishchi dasturda asosida tuziladigan taqvim rejalarda o’zgarishlar bo’lishi mumkin.

Kafedra mudiri: _____ f-m.f.d prof N.U Uteuliev.

Fanning ishchi o’quv dasturi «Kompyuter injiniringi» fakultet kengashida muhokama qilindi va foydalanishga taklif qilindi. (2017 yil “__” avgustdagi __ - sonli bayonnomasi)

Fakultet kengashi raisi: _____ f-m.f.n. A.Arziev

Kelishildi:
O’quv-metodik bo’lim boshlig’i _____ Yadgarov SH.

ЎЗБЕКИСТОН РЕСПУБЛИКАСИ АХБОРОТ ТЕХНОЛОГИЯЛАРИ ВА КОММУНИКАЦИЯЛАРИНИ РИВОЖЛАНТИРИШ ВАЗИРЛИГИ

МУҲАММАД АЛ-ХОРАЗМИЙ НОМИДАГИ ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ НУКУС ФИЛИАЛИ

«ОБЪЕКТГА ЙЎНАЛТИРИЛГАН ДАСТУРЛАШ ТИЛЛАРИ» ФАНИДАН МАЪРУЗАЛАР КУРСИ

НУКУС 2018

МАЪРУЗА 1. МАВЗУ: ОБЪЕКТГА ЙЎНАЛТИРИЛГАН ДАСТУРЛАШ ТИЛЛАРИ. JAVA ДАСТУРЛАШ ТИЛИ ҲАҚИДА

JAVA тилининг асоси бу - объектга йўналтирилган дастурлаш (ОЙД) нинг принципларини амалга ошириш. Объектга йўналтирилган методика JAVA дан ажиралмас ва барча JAVA нинг дастурлари қандайдир бир объект йўналишига эгадир. Шунинг учун оддий JAVA дастурини ёзишдан олдин, ОЙД ҳақида тушунчага эга бўлишимиз керак. ОЙД - бу кучли восита бўлиб дастурлашни такомиллаштирувчи процесс. Комьпютер кашф этилгандан бошлаб мураккаб дастурларни соддалаштириш мақсади дастурлаш усуллари куп маротаба ўзгарди. Масалан биринчи компютерлар билан дастурлаштириш машинанинг қўлланиш қоидалари ёрдами билан амалга оширилганю бу усул дастур узунлигида инструкциялар бир нечта бўлганига қадар ишлади. Дастурнинг хажми ўсишига боғлиқ ассемблер тили яратилди. Бу тил дастурчиларга машина инструкциясининг символик кўринишидан фойдаланиб янада бундан каттароқ ва қийинироқ дастурларни ёзиш имкониятини берди. Дастурнинг ҳажмининг ўсишига боғлиқ турда юқори даражадаги тиллар (масалан, FORTRAN и COBOL) пайдо бўлди ва дастурчиларга ўсиб бораётган дастур қийнчиликларни енгишга ёрдам берди. Бу биринчи компьютер тиллари критик ҳолатга яқинлашганда структурали дастурлаш пайдо бўлди.

Дастурлашнинг хар бир босқичи дастурчига бундан ҳам қийинироқ дастурларни ёзиш имкониятини берувчи методларнинг ва муҳитларнинг яратилиши билан тавсифланади. Бунинг ҳар бир қадамида усулларнинг энг мувафақиятлиларига таянган ҳолда ўз элементлари билан бирга янги усуллар яратилиши билан дастурлаш саҳосини ривожлантирди. Тахминан дастурлаш ривожланишининг шундай схемаси объектга йўналтирилган дастурлашга қадар программистлар учун “охири” эди. Унинг пайдо бўлиши структурали дастурлар ҳал қила олмайдиган проектларнинг бошланишигасабаб бўлди. Дастурларнинг муракаблигини ечиш учун янги усул керак бўлди ва бу муаммонинг ечиьи ОЙД булди.

Объектга йўналтирилган дастурлаш ўзига структурали дастурлашнинг энг яхши ғояларини танлаб олди ва уларни янги концепциялар билан боғлади. Натижада дастурларни ташкил қилишнинг янада замонавий усули пайдо булди. Умуий қилиб айтадиган бўлсак дастурни иккита усулнинг бири билан ташкил қилиш мумкин: кодга (шунингдек, ҳаракатга ёки дастурда нима содир бўлса шунга) ёки маълумотларга (шунингдек, аниқ бир ҳаракатга) таянган ҳолда.

Если говорить в самых, общих чертах, программу можно организовать одним из двух способов: опираясь либо на код (т.е. на действия, или на то, что происходит в программе), либо на данные (т.е. на то, что подвергается определенному воздействию). Структурали дастурлаш усулларини фойдаланилмаганда дастурлар харакатга таянган ҳолда ташкил қилинган.

Бундай ёндашишни маълумотларга таъсир этувчи код кўринишида тасвирлаш мумкин.

Объектга йўналтирилган дастурлаш умуман бошқача ишлайди. Улар маълумотлар атрофида ташкил қилинган булиб,бундай ташкил қилишнинг калитлик принципи: маълумотларгина кодга кириш мумкинлигини бошқаради дейилади. Обектга йўналтирилган тилларда программист маълумотлар устида харакатларнинг бажарилиши учун рухсат берадиган маълумот ва кодни аниқлайди. Шундай қилиб, маълумотлар тури унга қўлланадиган операцияларни аниқ аниқлайди. Объектга йўналтирилган дастурлашни қуллаб қувватлаш учун барча ОЙД тиллари шунингдек JAVA ҳам уч турдаги характерлик белгилардан иборат: инкапсуляция, полимор-физм ва мерос.

1.1 Инкапсуляция

Инкапсуляция — бу код(ҳаракат) ва маълумотларни бирлаштирадиган ва нотўғри қулланишдан сақлайдиган дастурлаш механизми. Объектга йўналтирилган тилларда код ва маълумотларни шундай қилиб боғласа бўлади, бунда автаном қора қути пайдо бўлади. Бу қутининг ичида барча зарурли булган маълумотлар ва код жойлашган булади. Код ва маълумотларни бундай боғлашдан объект ҳосил бўлади. Бошқача айтганда объект -- бу инкапсуляцияни қуллаб қувватлайдиган элемент.

Код ва маълумотлар булар иккаласи обект тузувчилари булиб, улар ичкаридан ёпилган ва очиқ булиши мумкин. Ёпиқ код ёки ёпиқ маълумотлар объектнинг бошқа қисмларигагина белгили ва рухсат этилган. Бу ёпиқ код ёки маълумотлар шу объект ичида булмаган дастурнинг бошқа қисмларига хеч қандай алоқа қилиши мумкин эмаслигини англатади. Агар код ёки маълумотлар очиқ бўлса, у ҳолда у дастурнинг бошқа қисмлари(уларнинг объект ичида бўлганлигига қарамасдан) ҳам алоқа қила олиш мумкинчилигига эга булади. Қоида бўйича объектнинг очиқ қисми ёпиқ элементлар билан бошкарилувчи интерфейсни таминлаш учун фойдаланилади.

JAVA инкапсуляциясинингш асосий бирлиги синф ҳисобланади.Синф объект шаклини аниқлайди. У код ва маълумотлар сингари берилган маълумотларни тузатиб боради. JAVA да синф объектларни тузиш учун фойдаланилади. Объект — синфларнингнинг нусхаси. Шурдай қилиб синф объектни қандай қуришни кўрсатувчи шаблондаги элементлар йиғиндиси.

Синфни тузувчи код ва маълумотлар синф аъзолари деб номланади. Синфда аниқланган маълумотлар нусқа ўзгарувчилари (instance variable) деб аталади, шу маълумотлар билан ишловчи код эса аъзо методлари(member method) ёки соддагина қилиб методлар деб айтилади. «Метод» —бу термин бўлиб, JAVA учунқисмдастурлани англатиш учун қабул қилинган. Агар сиз C ёки C++ тиллари билан таниш бўлсангиз, эҳтимол англаган бўлсангиз JAVA да программист уларни методлар деб, C ва C++ да программист уларни функциялар деб атайди. Ўз навбатида JAVA C++нинг авлоди, “функция” термини ҳам ҳам JAVA-методи сингари қабул қилинган бўлиб фойдаланилади.

1.2 Полиморфизм

Полиморфизм (грекча polymorphism сўздан олинган бўлиб, «кўп форм»деган маънони англатади) — бу бутун синф учун бир интерфейсни яратувчи ҳаракат сифати. Полиморфизмга аитомобил рули оддий мисол бўла олади. Руль (интерфейс) бу автомобилда қандай рулмуханизмини фойдаланишга қарамасдан руль булиб қолади. Бошқача айтганда ҳар қандай ҳолатда бир хил ишлайди: сизнинг автомобилни қандай бошқарганингиз билан хам у рулдир. Шундай қилиб қандай руллик бошқариш бўлишидан қатъий назар руль чапга бурилса у автомобилни ҳамм чапга буришга мажбур қилади. Биробразли интерфейс имконияти шундан турадики, яъни агар сиз рул билан қандай ишлашни билсангиз, унда сиз ҳоҳлаган турдаги автомобилни бошқара оласиз.

Дастурлашга ҳам шундай принципни қўлланишга бўлади. Мисол сифатида учун “охирги келган-биринчи хизматда” принципи бўйича фунцияланувчи стек (stack), шунингдек хотира областини қараб ўтамиз. Майли майли сиз уч хил турдаги стекни ташкил қилиш керак бўлган дастур ёзасиз. Биринчи стек -бутун сонли қиймат учун, иккинчиси қалқиб юрувчи нуқта(точка) қиймати учун,учунчиси символлар учун. Бундай ҳолда ҳар бир стекни амалга ошириш учун хар хил турдаги сақланган маълумотлар булишига қарамасдан бир хил турдаги алгоритм фойдаланилади. Обектга йуналтирилган тил булмаган ҳолда сизга учта хар хил номларга эга бўлган “стеклар” қисм дастурининг тупламини тузишингизга туғри келарди. Бироқ полиморфизм булганлиги сабабли JAVA муҳитида барча стекнинг уч турини ҳам ўз ичига оладиган “стеклар” қисм дастурининг бир тупламини тузиш етарли. Бошқача айтганда бир стекни қўлланишни билган ҳолда бошқалари учун ҳам қўлланишга бўлади.

Полиморфизм концепцияси кўпинча қуйидаги сўзлар билан ифодаланади:

«бир интерфейс — кўп методлар». Бу белгили ҳаракатлар груҳи бажарилиши учун умумий интерфейс ишлаб чиқиш мумкинлигини билдиради. Полиморфизм дастурнинг қийинчилигини осонлаштиришга ёрдам беради бу ерда дастурчига умумий синф ҳаракати вазифаси учун битта интерфейсдан фойдаланиш имкониятини беради. Конкрет (шунингдек каракли ёки тескари ҳолатда) ҳаракатни (метод) компилятор танлайди. Дастурчига буни ўзи қилишига зарурат булмайди. Унинг вазифаси — умумий интерфейсни тўғри фойдаланиш.

1.3 Мерос

Мерос бу шундай жараён бўлиб унда бир объект иккинчи бирининг хоссасига утиши мумкин. Мерос сабабли иерархиялик классификация қўллаб қувватланади. Бошқарилувчи иерархической (келиб чиққан) классификация кўринишида катта бўлган билимлар соҳаси ташкил қилинади. Масалан,

Красный Делишес олмаси олма классификациясининг қисми ҳисобланади, ўз навбатида мевалар синфининг ҳам қисмидир ва шунингдек катта бўлган овқат синфининг ҳам қисми ҳисобланади. Шундай қилиб овқат синфи мевалар қисмсинфига(подкласс) ҳам тегишли булган белгили бир сифатга эга (еб бўладиган,иштаҳалик ва б .) бўлади. Бу сифатлардан мевалар синфи мева маҳсулотларини бошқаларидан ажратиб турадиган ўзига тан бўлган жиҳатларга (спецификалик характер) (ширинлик, ширали ва б.)эгадир Олма синфида эса олмага тан бўлган сифатлар аниқланади(теракда ўсади тропик эмас ва б.). Красный Делишес синфи дастлабки барча синфларни мерос қилиб олган ҳолда олманинг бу навига тегишли булган сифатни аниқлайди.

Агар белгиларнинг иерархик(келиб чиқиш) берилишини ҳисобга олмасак, у ҳолда ҳар бир объект учун тан булган барча унинг характеристикасининг аниқ шаклини аниқлашга тўғри келарди. Лекин обект меросига боғлиқ унинг синф ичида аълохидалигини билдирувчи сифатларинигина аниқлаш етарлидир, чунки у(объект) ўз ота-онасининг умумий белгиларини мерос қилади. Ўз навбатида мерос механизмигина бир объект умумий синф учун конкрет экземплярлигини беради.

МАЪРУЗА 2. МАВЗУ: АСОСИЙ КОНСТРУКЦИЯЛАР

Режа: Ўзгарувчилар ва константалар; Амаллар, Кўрсаткичлар ва иловалар; Дастур тузилиши; Операторлар; Номланган константалар; Типларни келтириш.

Ўзгарувчилар ва константалар

Ўзгарувчилар. Ўзгарувчи номи остига чизиш белгиси ёки лотин харфидан бошланувчи лотин харфлари, араб рақамлари ва остига чизиш хамда доллар белгилари кетма кетлиги яъни идентификатордир. Java — тилида регистр фарқ қилади. Масалан, Value и VALUE — турли идентификаторлардир.

Ўзгарувчиларнинг қуйидаги типлари мавжуддир: **byte(**байт**), char(**символ**), short(**қисқа бутун**), int(**бутун**), long(**узун бутун**), float(**хақиқий**), double(**иккиланган хақиқий**), boolean(**мантиқий**).**

JAVA тилида unsigned (ишорасиз) таърифи ишлатилмайди.

Ўзгарувчилар таърифи содда шакли:

<тип> <ўзгарувчилар_номлари_рўйхати >;

ўзгарувчиларни таърифлашда бошланғич қийматларини кўрсатиш мумкин.

<тип> <ўзгарувчилар_номлари_рўйхати >= <инициализатор>;

Бу усул инициализация дейилади.

Мисоллар:

float pi = 3.14 , cc=1.3456;

int year = 1999;

Агар ўзгарвчи қиймати кўрсатилмаган бўлса автоматик равишда ноль қиймат қабул қилади.

Константалар. Константа бу ўзгартириш мумкин бўлмаган қийматдир. Константалар бутун, символли хақиқий, мантиқий турлари мавжуд.

Бутун сонлар ўнлик, саккизлик ёки ўн олтилик саноқ системаларида берилиши мумкин. Ўнлик константалар ўнлик ракамлари кетма кетлигидан иборат бўлиб, биринчи раками 0 бўлиши керак эмас(масалан: 8, 0, 192345). Саккизлик константалар 0 билан бошланувчи саккизлик ракамларидан иборат кетма кетликдир (масалан: 016 – ўнлик қиймати 14, 01). Ўн олтилик константалар – 0x ёки 0X билан бошланувчи ўн олтилик рақамлар кетма кетлигидир (масалан: 0xA, 0X00F).

Символли константа – апострифга олинган битта (масалан: ‘ғ’,’r,’6’) ёки бир нечта (масалан: –‘\n’, ‘\0xF5’) символ. Слеш ‘\’ символидан бошланган символлар эскейп ёки бошқарувчи символлар дейилади

Хақиқий константа икки кўринишда бўлиши мумкин: фиксирланган нуқтали (масалан: 5.7, .0001, 41.) ва сузувчи нуқтали (масалан: 0.5e5, .11e-5).

Мантиқий константалар true(рост) ва false(ёлғон) қийматлардан иборат. Ички кўриниши false – 0, ихтиёрий бошқа қиймат true деб қаралади.

Сатрли константалар Java тилида жуфт қавслар ичига олинган ихтиёрий матндир (""). Сатрли константалар бир қаторга жойлашиши лозим.

Амаллар

Арифметик амаллар. Арифметик амаллар бинар ва унар амалларга ажратилади. Бинар амалларга қўшиш +, айириш –, кўпайтириш *, бўлиш / ва модуль олиш % амаллари киради.

Масалан 20/3=6; (-20)/3=-6; 5%2=1;

Унар амалларга унар минус – ва унар +; инкремент ++ ва декремент— амаллари киради. Инкремент ва декремент амаллари префикс яъни ++i, постфикс яъни i++ кўринишда ишлатиши мумкин. Масалан агар i=2,k=2 бўлса у холда 3+(++i)= 6, 3+k++=5 га тенг бўлади. Иккала холда хам i=3,k=3 га тенг бўлади.

Нисбат амаллари. Нисбат амаллари қиймати мантиқий бўлиб катта >; кичик <;катта ёки тенг >=; кичик ёки тенг <=; тенг == ; тенг эмас != амалларидан иборат.

Мантиқий амаллар. Мантиқий амаллар дизъюнкция ||, конъюнкция &&, инкор ! амалларидан иборат.

Разрядли амаллар. Разрядли амаллар Разрядли дизъюнкция |, Разрядли конъюнкция &, Разрядли XOR ^, Разрядли инкор !, Разрядли чапга суриш<< ва Разрядли ўнгга суриш >> амалларидан иборат. Масалан 5 коди 101 га тенг ва 6 коди 110 га тенг:

6&5= 4=100; 6|5= 7= 111; 6^5=3=011; ~6=4=010.

5<<2=20 ёки 101<<2=10100; 5>>2=1 ёки 101>>2=001=1.

Қиймат бериш амали. Оддий қиймат бериш амали бинар амал булиб чап операнди одатда узгарувчи унг операнди одатда ифодага тенг булади:

Ўзгарувчи_номи = ифода;

Масалан: Z=4.7+3.34; C=y=f=4.2+2.8;

Мураккаб қиймат бериш амали унар амал бўлиб қуйидаги кўринишга эга: Ўзгарувчи_номи **амал**= ифода;

Масалан: x+=4 ифода x=x+4 ифодага эквивалентдир;

x>>=4 ифода x=x>>4 ифодага эквивалентдир;

Шартли амал. Шартли амал тернар амал дейилади ва учта операнддан иборат булади: <1-ифода>?<2-ифода>:<3-ифода>. Масалан: y=a<b?a:b .

Типлар билан ишловчи амаллар. Типларни ўзгартириш амали қуйидаги икки куринишга эга:

Каноник: (тип_номи) операнд; масалан: r=(unsigned long)1;

Функционал: тип_номи (операнд); масалан: z=double(1);

Хотирадаги хажмни хисоблаш sizeof амалининг икки кўриниши мавжуд:

sizeof ифода масалан: Sizeof 3.14=8 sizeof (тип) масалан: Sizeof(char)=1

Кўрсаткичлар ва иловалар

Кўрсаткичлар таърифи. Кўрсаткичлар қиймати конкрет типдаги объектлар учун хотирада ажратилган адресларга тенгдир

Бирор ўзгарувчи адресини олиб, кўрсаткичга қиймат сифатида бериш учун адрес олиш «&» амалидан фойдаланилади. Кўрсаткич орқали ўзгарувчи қийматини олиш учун адрес бўйича қиймат олиш «*» амалидан фойдаланилади.

Кўрсаткичларга ўхшаб Иловаларнинг қийматлари хам адреслардир. Иловаларга мурожаат қилинганда автоматик равишда * қиймат олиш амали бажарилади.

Дастур тузилиши

Java тилида дастур номланган синфлардан иборат.

Мисол:

```
class HelloWorld {
public static void main (String args []) {
System. out. println ("Hello World");
}
}
```

Юқорида келтирилган мисол матнини HelloWorld.java файлга ёзиш лозим.

Бу мисолни трансляция қилиш учун javac - Java трансляторини ишга тушириш лозим:

C: \> javac HelloWorld.Java

Транслятор HelloWorld.class номли процессорга боғлиқ бўлмаган байт-кодли файл яратади. Дастур кодини ишга тушириш учун Java тили мухитига бажариш учун янги синфни юклаш лозим. Шуни таъкидлаш лозимки синф жойлашган файл эмас синф номи кўрсатилади.

C: > java HelloWorld

Hello World

Мурожаат модификатори public асосий усул main хамма синфларда кўринишини билдиради.

Махсус static сўзи ёрдамида бутун синф билан бирга ишлатиладиган ўзгрувчилар ва усуллар таърифланади. Агар усул static сўзи ёрдамида таърифланган бўлса фақат локал ва статик ўзгарувчилар билан ишлаши мумкин.

Агар main усули қиймат қайтармаса модификатор void ишлатилади.

Хамма мавжуд Java-интерпретаторлар, синф интерпретациясини main усулини чақиришдан бошлайди. Java-транслятор main усули бўлмаган синфни трансляция қилиши мумкин, лекин Java-интерпретатор main усули бўлмаган синфни ишга тушира олмайди.

Усул String args[] элементи String синфига тегишли объектлар массиви бўлган args параметрни эълон қилади .

Экранга чиқариш учун out объектининг println усулидан фойдаланилади. Объект out статик равишда System инициализация қилиниб OutputStream синфида таърифлангандир.

Номланган константалар

Java тилида константаларни белгилаш учун final калит сўзидан фойдаланилади:

```
public class Constants
public static void main(String[] args)
final double CM_PER_INCH = 2.54;
double paperWidth = 8.5;
double PaperHeight = 11;
System.out.println("Сахифа хажми сантиметрларда: "
+ paperWidth * CM_PER_INCH + "га"
+ paperheight * CM_PER_INCH);
```

Номланган константа қийматини ўзгартириш мумкин эмас. Константалар номида катта харфларни ишлатиш шарт эмас.

Java тилида бир синф ичида бир неча усулларда ишлатиладиган константалар синф константалари (class constants) деб аталади. Синф константалари static final калит сўзи ёрдамида таърифланади.

```
public class Constants2
{
public static final double CM_PER_INCH = 2.54;
public static void main(String [] args)
{
double paperWidth = 8.5;
double PaperHeight = 11;
System.out.printIn("Сахифа катталиги сантиметрларда: "
+ paperWidth * CM_PER_INCH + "га"
+ paperHeight * CM_PER_INCH);
```

Синф константалари main усули ташқарисида таърифланади. Агар константа public сифатида таърифланган бўлса унга бошқа синф усулларида мурожаат қилиш мумкин. Масалан қуйидагича:

Constants2.CM_PER_INCH.

Java тилида const сўзи хизматчи хисобланади, лекин фойдаланилмайди. Константаларни таърифлаш учун final калит сўзидан фойдаланиш зарур.

Типларни келтириш

Типларни келтириш (type casting) маълум типдаги ўзгарувчи бошқа тидаги қиймат қабул қилганда фойдаланилади. Баъзи типлар учун келтириш автоматик равишда бажарилади. В Java тилида автоматик типларни келтириш ўзгарувчи типи хажми қийматни сақлашга етарли бўлганда бажарилади. Бу жараён кенгайтириш *(widening)* ёки юксалтириш *(promotion)* деб аталади, чунки, кичик разрядли тип катта разрядли типга кенгайтирилади. Масалан int типи byte типидаги қийматни сақлашга етарли, шунинг учун типларни келтириш талаб қилинмайди. Тескариси мумкин эмас,

шунинг учун byte типидаги ўзгарувчи int типидаги қийматни қабул қилиши учун, типларни келтириш операторидан фойдаланиш лозим. Бу жараён торайтириш *(narrowing)* деб аталади, чунки трансляторга қийматни ўзгартириш хақида ошкор маълумот берилади. Бунинг учун думалоқ қавс ичида тип номи кўрсатилади. Масалан:

int a = 100;

byte b = (byte) a;

Агар сузувчи қавсли сонни энг яқин бутун сонга келтириш лозим бўлса Math.round усулидан фойдаланилади.

double x = 9.997;

int nx = (int)Math.round(x) ;

Энди nx ўзгарувчи қиймати 10 га тенг. Лекин round усулидан фойдаланилган типларни келтириш операторидан фойдаланиш лозим чунки бу усул long типидаги қийматни қайтаради.

Сонни бир типдан иккинчисига келтиришда натижа зарур диапазондан чиқиб кетиши мумкин, бу холда натижа қисқартирилади. Масалан (byte) 300 ифода қиймати 44 га тенг.

Мантиқий ва бутун типлар орасида келтириш мумкин. Баъзи холларда мантиқий қийматни бутун қийматга келтириш учун шартли ифодадан фойдаланиш мумкин, масалан b ? 1 : 0.

Ифодаларда типларни автоматик келтириш

Агар ифодада byte, short ва int типидаги ўзгарувчилар ишлатилса, бутун ифода типи int га кўтарилади. Агар ифодада бирор ўзгарувчи типи— long бўлса, бутун ифода типи long типга кўтарилади. Кўзда тутилган бўйича Java тилида хамма бутун константалар int типига эга деб қаралади. Хамма бутун константалар охирида L ёки 1 символи турган бўлса, long типига эга.

Агар ифода float типидаги операндга эга бўлса, бутун ифода float типига кўтарилади. Агар бирор операнд double типига эга бўлса, бутун ифода типи double типига кўтарилади. Кўзда тутилган бўйича Java тилида хамма сузувчи вергулли константалар double типига тегишли хисобланади. Агар хақиқий константа охирида F ёки f символи турган бўлса, float типига эга.

Назорат саволлари:

1. Асосий типларни кўрсатинг.
2. Кўрсаткич таърифини келтиринг.
3. Хамма амалларни кўрсатинг.
4. Қачон ошкор типларни келтиришга хожат йўқ?
5. Ифодаларда типлар қандай автоматик келтирилади?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 3. МАВЗУ: ОПЕРАТОРЛАР

Режа: Изохлар; Қўшма операторлар; Киритиш; Танлаш операторлари; Цикл операторлари; Ўтиш операторлари.

Изохлар

Изох оператори бажарилмайдиган оператор бўлиб блокли (Масалан **/*бу изох */**) ёки сатрли (Масалан **//бу изох)** бўлиши мумкин.

Қўшма операторлар

Қўшма операторга қўшма оператор ва блок киради. Иккала холда хам бу фигурали қавсга олинган операторлар кетма кетлигидир. Блок қўшма операторлан фарқи шуки таърифлар хам қатнашади.

Киритиш

Маълумотни "киритиш стандарт қурилмасидан " (яъни клавиатурадан) ўқиш учун мулоқот дарчаси яратиш керак. Бунинг учун JOptionPane.showInputDialog(promptString) усулидан фойдаланиш лозим. Бу усул фойдаланувчи терган сатрни қайтаради.

Масалан дастур фойдаланувчисини сўраш:

String name = JOptionPane.showInputDialog("What is your name?");

Усул қайтарган сатрни сонга айлантириш учун Integer.parseInt ёки Double.parseDouble, усулидан фойдаланиш лозим. Масалан:

String input = JOptionPane.showInputDialog("How old are you?");

int age = Integer.parselnt(input);

Агар клавиатурда 40 сони терилса, сатрли ўзгарувчи input қиймати "40" сатрга тенг бўлади. Бу сатрни Integer .parseInt усули 40 сонига айлантиради.

Дастур JOptionPane. showsInputDialog усулини чақирганда, унинг ишини тугатиш учун System.exit(0) усулини чақириб тугатиш лозим. Экранга диалог дарчасини чиқариш янги бошқариш оқимини хосил қилади. Бу оқим ўз ишини main усули билан бирга тугатмайди. Хамма оқимларни ёпиш учун System.exit усулини чақириш лозим. Бу усул бутун сонли параметр, яъни дастурдан "чиқиш кодини" қайтаради. Кўзда тутилган бўйича, агар дастур иши тўғри тугатилган бўлса, чиқиш коди нольга тенг бўлади, акс холда нольга тенг бўлмайди. Хар хил хато холатлар хақида хабар бериш учун, хар хил чиқиш кодидан фойдаланиш мумкин.

JOptionPane синфи javax. Swing пакетида аниқланган. Асосий java. Lang пакетида аниқланмаган синфдан фойдаланиш учун import директивасидан фойдаланиш лозим.

Масалан:

import javax.swing.*;

```
public class InputTest
 {
public static void main(String[] args)
 {
// Биринчи киритиш.
String name = JOptionPane.showInputDialog("Исмингизни айтинг ");
 // Иккинчи киритиш.
String input = JOptionPane.showInputDialog ("Ёшингиз нечада?");
 // Сатрни сонга айлантириш.
int age = Integer.parseInt(input);
 // Натижани консольга чиқариш.
 System.out.println("Салом, " + name +
". Кейинги йилда сизни ёшингиз тенг бўлади " + (age + 1 ) ) ;
System.exit(0) ;
}
}
```

Танлаш операторлари

Шартли оператор. Шартли танлаш операторида аввал шарт текширилади. Агар шарт рост бўлса биринчи акс холда иккинчи оператор(агар у бўлса) бажарилади.

if (ифода) 1- оператор else 2- оператор еки **if (ифода) 1-оператор**

Мисол. Символ иккилик эканлигини текширувчи дастур:

```
class IfElse {
public static void main(String args[]) { char ch = ‘1’;
String s;
if (ch==’1’ || ch == ‘0’) {
s = "binary";
} else
s= "no binary";
System.out.println( "Simvol " + s + ".");
}
}
```

Дастур бажариш натижаси:

C: \> java IfElse

Simvol binary.

Калит бўйича танлаш опертори. Калит бўйича танлаш оператори қуйидаги шаклга эга:

```
switch(<ифода>)  {
        case <1-киймат>:<1-оператор>
         ...
        default: <оператор>
         …  }
```

Калит бўйича танлаш операторида берилган ифода қиймати бирор case қийматига мос келса, шундан кейинги хамма операторлар бажарилади, акс холда default(агар у бўлса) сўзидан кейинги оператор бажарилади.

Мисол. Символ иккилик эканлигини текширувчи дастур:

```
class SwitchDemo { public static void main(String args[]) {
char ch = ‘1’;
String s;
switch (month) {
case ‘0’: // FALLSTHROUGH
case ‘1’: // FALLSTHROUGH
s = "binary";
break;
default:
s = "no binary";
}
System.out.println("Simvol " + s + ".");
}
}
```

Цикл операторлари

Олдинги шартли цикл. Қуйидаги кўринишга эга:

while (<шарт>) < цикл танаси> ;

Олдинги шартли циклда олдин шарт текширилади кейин то шарт ўлғон бўлгунча цикл танаси бажарилади.

Мисол. Цикл ёрдамида ўн сатрга «tick» сўзини чиқариш:

```
class WhileDemo {
public static void main(String args[]) {
int n = 10;
while (n > 0) {
System.out.println("tick " + n);
n--;
}
}
}
```

Кейинги шартли цикл. Қуйидаги кўринишга эга:

```
do
<цикл танаси>;
while (<шарт>);
```

Олдин цикл танаси бажарилиб, кейин шарт текширилади. Цикл то шарт ёлғон бўлмагунча давом этади.

Мисол. Цикл ёрдамида ўн сатрга «tick» сўзини чиқариш:

```
class DoWhile {
public static void main(String args[]) {
```

```
int n = 10;
do {
System.out.println("tick " + n);
} while (--n > 0);
}
}
```

Параметрли цикл. Қуйидаги кўринишга эга:

for(1-ифода; шарт; 2-ифода)
цикл танаси;

Аввал 1 – ифода бажарилади ва то шарт ёлғон бўлмагунча цикл танаси ва 2- ифода бажарилади. Ихтиёрий ифода бўш бўлиши мумкин, лекин уларни ажратувчи қавс « ; » бўлиши шарт.

Мисол. Бирдан ўнгача сонларни чиқариш:

```
class ForDemo {
public static void main(String args[]) {
for (int i = 1; i <= 10; i++)
System.out.println("i = " + i);
}
}
```

Ўтиш операторлари

Ўтиш операторлари бошқаришни шартсиз узатишни амалга оширади.

Блокдан ёки циклдан чиқиш учун **break –** операторидан фойдаланилади.

Агар циклни давом эттириш шартини цикл ўртасида текшириш зарур бўлса break операторидан фойдаланиш қулайдир.

Масалан, қуйидаги дастурда учта ички жойлашган блок мавжуд бўлиб, хар бирининг уникал белгиси мавжуд. Ички блокда жойлашган break оператори, b блокдан кейинги операторга бошқаришни узатади. Натижада икки println оператори бажарилмай қолади.

```
class Break {
public static void main(String args[]) { boolean t = true;
a: { b: { c: {
System.out.println("Before the break"); // break операторидан олдин
if (t)
break b;
System.out.println("This won't execute"); // Бажарилмайди }
System.out.println("This won't execute"); // Бажарилмайди }
System.out.println("This is after b"); // b дан сўнг
}
}
}
```

Дастур бажариш натижаси:

C:\> Java Break

Before the break
This is after b

Цикл кейинги итерациясига ўтиш учун **continue** – операторидан фойдаланилади.

Худди break оператори каби, continue операторида қайси ички цикл итрациясини тўхтатиш кераклигини кўрсатувчи белгидан фойдаланиш мумкин. Масалан кўпайтириш жадвалини учбурчак шаклда чиқариш учун белгили continue операторидан фойдаланувчи дастур:

```
class ContinueLabel {
public static void main(String args[]) {
outer: for (int i=0; i < 10; i++) {
for (int j = 0; j < 10; j++) {
if (j > i) {
System.out.println("");
continue outer;
}
System.out.print(" " + (i * j));
}
}
}
}
```

Бу дастурда continue оператори j параметрли ички цикл ишини тўхтатиб, i параметрли ташқи цикл кейинги итерациясига ўтишга олиб келади:

C:\> Java ContinueLabel
0
0 1
0 2 4
0 3 6 9
0 4 8 12 16
0 5 10 15 20 25
0 6 12 18 24 30 36
0 7 14 21 28 35 42 49
0 8 16 24 32 40 48 56 64
0 9 18 27 36 45 54 63 72 81

Назорат саволлари:

1. Изох қандай кўрсатилади?
2. Қўшма оператор таърифини келтиринг.
3. Киритиш қандай амалга оширилади?
4. Танлаш операторларини кўрсатинг.
5. Цикллар турларини кўрсатинг.
6. Нима учун break ва continue операторлари ишлатилади?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 4. МАВЗУ: МАССИВЛАР

Режа: Бир ўлчовли массивлар; Массивдан нусха олиш; Массивларни тартиблаш; Кўп ўлчовли массивлар; Команда қатори параметрлари

Бир ўлчовли массивлар

Массив индексли ўзгарувчидир.

Массив содда таърифи:

<тип> <ўзгарувчи_номи>>[<константа_ифода>] = <инициализатор>;

Массив индекслар қиймати хар доим 0 дан бошланади.

Массивга хотира ажратиш учун махсус new операторидан фойдаланилади.

int a[];

int a=new int[6];

Массив инициализация қилинганда элементлар сони кўрсатилиши шарт эмас.

Масалан:

double d[] = {1, 2, 3, 4, 5};

Бу синтаксис конструкция ёрдамида янги ўзгарувчи киритмасдан массивни қайта инициализация қилиш мумкин.

Масалан:

smallPrimes = new int{ 17, 19, 23, 29, 31, 37 };

Массивда элементлар сонини хисоблаш учун **length** усулидан фойдаланиш мумкин.

Масалан:

```
for (int i = 0; i < a.length; i++)
System.out.println(a[i]) ;
```

Массив яратилгандан сўнг унинг хажмини ўзгартириш мумкин эмас (лекин алохида элементларини ўзгартириш мумкин).

Java тилида [ ] оператори кўзда тутилган бўйича индекс ўзгариш диапазонини текширади.

Массивни икки усулда таърифлаш мумкин:

int [] a;

ёки

int a[];

Массивдан нусха олиш

Битта массивдан иккинчисига нусха олиш мумкин, лекин бу холда иккала ўзгарувчи битта массивга илова қилади.

int[] luckyNumbers = smailPrimes;

luckyNuimbers[5] = 12; // Энди элемент smailPrimes[5] тенг 12 га.

Агар массив хамма элементларидан бошқа массивга нусха олиш зарур бўлса **java.lang.System** синфининг **аггаусору** усулидан фойдаланиш лозим.

Масалан қуйидаги операторлар, икки массив яратади, сўнгра биринчи масив охирги тўотта элементидан иккинчи масивга нусха олади. Нусха олиш биринчи массив иккинчи элементидан бошланпиб, иккинчи массивга учинчи позициядан нусха олинади.

```
int[] smailPrimes = {2, 3, 5, 7, 11, 13};
int[] luckyNumbers = {1001, 1002, 1003, 1004, 1005, 1006, 1007};
System.аггаусору(smailPrimes, 2, luckyNumbers, 3, 4);
for (int i = 0; i < luckyNumbers.length; i++)
System.println(i +.": " + luckyNumbersfi]);
```

Бу операторлар бажарилиши қуйидаги натижага олиб келади.

```
0: 1001
1: 1002
2: 1003
3: 5
4: 7
5: 11
6: 13
```

Массивларни тартиблаш

Агар сонлар массивини тартиблаш лозим бўлса **java.util.Arrays** синфининг **sort** усулидан фойдаланиш лозим.

Усул массивни тезкор алгоритм асосида тартиблайди.

```
Int []a = new int[10000];
Arrays.sort(a);
```

Қуйидаги дастурда ишчи массивда лотерея учун сонлар тасодифий равишда генерация қилинади.

```
import java.util.*;
import javax.swing.*;
public class LotteryDrawing
{
public static void main(String[] args)
{
String input = JOptionPane.showInputDialog
("ҚАНЧА номерни топиш лозим?");
int к = Integer.parselnt(input);
input = JOptionPane.showInputDialog
("Энг катта номер нечага тенг бўлиши мумкин?");
int n = Integer.parselnt(input);
// массивни 1 2 3. . .n сонлар билан тўлдирамиз.
int[] numbers = new int[n];
for (int i = 0; i < numbers.length; i++)
numbers[i] = i + 1;
// к сон генерация қилиб, иккинчи массивга жойлаштирамиз.
```

```
int[] result = new int[к];
for (int i = 0; i < result.length; i++)
{
// 0 дан то n – 1 гача тасодифий индекс генерацияси.
int r = (int)(Math.random() * n);
// элементни тасодифий ячейкага жойлаштирамиз.
result[i] = numbers[r];
// охирги элементни тасодифий ячейкага кўчирамиз.
numbers[r] = numbers[n - 1];
n— ;
}
// тартибланган массивни чиқарамиз.
Arrays.sort(result);
System.out.println
("Қуйидаги комбинацияга тикинг — ачинмайсиз!")
for (int i = 0; i < result.length; i++)
System.out.println(result[i]);
System.exit(0);
}
}
```

Агар " 49 тадан 6 та", ютиш лозим бўлса дастур қуйидаги маълумотни босиб чиқаради.

```
Қуйидаги комбинацияга тикинг — ачинмайсиз!
4
7
8
19
30
44
```

Кўп ўлчовли массивлар

Java тилида чин кўп ўлчовли массивлар мавжуд эмас. Кўп ўлчовли массивлар массивлар массивлари сифатида тасвирланади. Қуйида келтирилган дастур double типидаги ўн олти элементдан иборат матрица яратиб, ноль қиймат билан инициализация қилади. Бу массив ички реализацияси double типидаги — массивлар массивидир.

```
double matrix [][] = new double [4][4];
```

Қуйида шунча хотира ажратилади, лекин иккинчи ўлчов учун хотира ошкор қадамма қадам ажратилади.

```
double matrix [][] = new double [4][];
matrix [0] = new double[4];
matrix[1] = new double[4];
matrix[2] = new double[4], matrix[3] = { 0, 1, 2, 3 };
```

Бошқа холларда агар массив элементлари олдиндан маълум бўлса, кўп ўлчовли масссивни инициализация қилиш учун new операторидан фойдаланилмайдиган қисқа ёзувдан фойдаланиш мумкин. Масалан.

int[][] magicSquare = {{16, 3, 2, },(5, 10, 11},{9, 6, 7},{4, 15, 14}};

Массив сатрларини осонгина ўзгартириш мумкин!

double[] temp = balance [i];
balance[i] = balance[i+1];
balance[i=l] = temp;

Бундан ташқари Java тилида "текис бўлмаган" яъни хар хил сатрлари хар хил узунликка эга бўлган массивлар яратиш осондир.

Нотекис массив яратиш учун хотирага сатрларни сақловчи массив яратилади.

int[][] adds = new int[NMAX+1][];

Сўнгра сатрлар яратилади.

for (n=0; n<=NMAX; n++)
adds[n] = new int[n + l] ;

Хотирада бутун массив жойлашгандан сўнг, массив элементларига мурожаат қилиш мумкин. Лекин индекслар диапазондан чиқиб кетмаслиги лозим.

for (n=0; n<odds.length; n++)
for (k=0; k<odds[n].length; k++)
// Имкониятларни хисоблаш.
adds[n][k] = lotteryOdds;

Қуйидаги мисолда i-чи сатри ва j-чи устуни кесишмасида " j та сондан і лотерея номерини танлаш " имкониятлари сони ёзилган учбурчак массив яратилади.

Массив і-чи сатрида і+1 элемент жойлашган.

```
public class LotteryArray
{
public static void main(String[] args)
(
final int NMAX' = 10;
// Учбурчак матрицани жойлаштириш.
int[][] odds = new int[NMAX + 1][];
for (int n = 0; n <= NMAX; n++)
odds[n] = new int[n + 1];
// Учбурчак матрицани тўлдириш,
for (int n = 0; n < odds.length; n++)
for (int k = 0; k < odds[n].length; k++)
{
/*
Биномиал коэффициентларни хисоблаш.
n * (n - 1) * (n - 2) * . . . * (n - k + 1)
*/
```

```
int lotteryOdds = 1;
for (int i = 1; i <= k;
lotteryOdds = lotteryOdds * (n - i + 1). / i;
odds[n][k] = lotteryOdds;
// Учбурчак массивни чиқариш.
for (int n = 0; n < odds.length; n++)
{
for (int к = 0; к < odds[n].length; k++)
{
// Чиқаришда бўшлик белгиларини жойлаштириш.
String output = " " + odds[n][k];
// Натижа кенглиги 4 символга тенг майдонга чиқарилади,
output = output.substring(output.length() - 4);
System.out.print(output);
}
System.out.println();
}
}
```

Команда қатори параметрлари

Java тилидаги хар бир дастурда String [ ] args параметрига эга main усули мавжуд. Бу параметр команда қаторида кўрсатилган аргументлардан иборат сатрлар массивини main усулига узатади.

Масалан.

```
public class Message
public static void main(String[] args)
if (args[0].equals("-h");
System.out.print ("Салом,");
else if (args[0].equals("-g"))
System.out.print("Хайр,");
// Командалар қатори қолган аргументларини чиқаради,
for (int i = 1; i < args.length; i++)
System.out.print(" " + args[i]);
System.out.print("!");
}
}
```

Агар дастур қуйидагича чақирилса

Java Message -g бевафо дунё

массив args қуйидаги элементлардан иборат бўлади.

args[0] "-g"

args[l] "бевафо"

args[2] "дунё"

Дастур қуйидаги маълумотни чиқаради

Хайр, бевафо дунё!

Java тилидаги дастурда main усулидаги args массиви дастур номини сақламайди. Масалан Message дастурни қуйидаги команда ёрдамида ишга туширилса

Java Message -h дунё

Команда қаторидаги args[0] элемент, "Message" ёки "Java" сатрига эмас "- h" сатрига тенг бўлади.

Назорат саволлари:

1. Массивни инициализация қилиш усулларини кўрсатинг.
2. Қандай қилиб массивлар формал параметр сифатида таърифланиши мумкин?
3. Нима учун new амали қўлланади?
4. Масивлардан нусха олиш ва массивларни тартиблаш қандай амалга оширилади?
5. Динамик массивлар хосил қилиш усулларини кўрсатинг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 5. МАВЗУ: САТРЛАР

Режа: Сатрлар билан ишлаш; Конструкторлар; Сатрларни яратиш; Сатрларни улаш; Символларни ажратиш; Солиштириш; Тенглик; Тартиблаш; Символ ёки остки сатрни излаш.

Сатрлар билан ишлаш

Сатрга мос келувчи синф java.lang пакетида жойлашгандир. Бу синф **String** деб аталиб, символлар ўзгармайдиган массивини объект шаклидаги тасвиридир. Бу синфда сатрларни солиштириш, излаш, маълум символлар ва остки сатрларни ажратувчи усуллар мавжуд. **StringBuffer** синфи сатрни яратгандан сўнг ўзгартириш керак бўлганда ишлатилади.

String ва **StringBuffer** final деб эълон қилинган, бу эса бу синфларнинг ҳеч биридан ҳосила синф яратиб бўлмаслигини англатади. Бу сатрларни қайта ишлаш операцияларини бажаришда самарадорликни оширувчи оптималлаштиришни айрим кўринишларини қўллаш учун қилинган.

Конструкторлар

Бошқа ихтиёрий синф холидаги каби **new** оператори ёрдамида String туридаги объектлар яратишингиз мумкин. Бўш сатр яратиш учун параметрсиз конструктор ишлатилади.

String s = new String():

Қуйида келтирилган код фрагменти **String** типидаги **s** объект яратади ҳамда уни конструкторга символли массивда параметр сифатида узатилган учта символдан иборат сатр билан инициализация қилади.

```
char chars[] = { 'a', 'b', 'c' };
String s = new String(chars);
System.out.println(s);
```

Коднинг ушбу фрагменти қуйидаги сатрни чиқаради «abc».

Ушбу конструктор учта параметрга эга:

String(char chars[], int бошланғичИндекс, int символларСони);

Ушбу усулда инициализация қилишга мисол:

```
char chars[] = { 'a', 'b', 'c', 'd', 'e', 'f' }:
String s = new String(chars,2,3);
System.out.println(s);
```

Ушбу код фрагменти қуйидаги сатрни чиқаради «cde».

Сатрларни яратиш

Java тили жуфт қўштирноқли сатр таркибли литерал кўринишдаги ёзувни бу операция учун стандарт қисқартмани ўз ичига олади. Қуйида

келтирилган код фрагменти олдинги char типидаги массив билан инициализация қилинган сатрга эквивалент.

String s = "abc";

System.out.println(s);

String объектлари билан фойдаланилувчи умумий усуллардан бири бу length дир, у сатрдаги символлар сонини қайтаради. Навбатдаги код фрагменти сатрдаги символлар сонини экранга чиқаради.

String s = "abc";

System.out.println(s.length);

Java да ҳар бир сатр литерал учун String синфининг вакили яратилади, шунинг учун бу синфнинг усулларини нафақат мурожаатли ўзгарувчилар билан балки, бевосита сатр-литерал билан чақириш мумкин. Мисол учун қуйидаги код фрагменти 3 ни чиқаради.

System.out.println("abc".Length());

Сатрларни қўшиш

String s = "He is " + age + "years old.";

Сатрни + оператори билан битта сатрга бирлаштирсак, уни ўқиб чиқиш ва тушуниш албатта олдинги кўринишдаги эквивалентларига нисбатан осон унинг

Сатрларни улаш

Қўшиш оператори + ишлатилган қуйидаги сатрни

String s = «He is » + age + " years old.";

Усуллардан фойдаланиб яратилган шаклига қараганда тушуниш осондир:

String s = new StringBuffer("He is ").append(age);

s.append(" years old.").toString();

Таъриф бўйича String синфи объектини ўзгартириш мумкин эмас. Мавжуд сатрга янги символ қўшиш ёки символни ўзгартириш мумкин эмас. Бир сатрни иккинчисига улаш хам мумкин эмас. Шунинг учун Java тили String объектини щзгартирувчи амалларни StringBuffer мос амалларига алмаштиради.

Операторларни бажариш тартиби

Охирги мисолни қайта кўриб чиқамиз:

String s = "He is " + age + " years old.";

Агар age —String эмас, int типидаги ўзгарувчи бўлса бу мисол ўзига хос хусусиятларга эга бўлади. Ўзгарувчи бутун қиймати StringBuffer синфининг append усулига узатилади. Бу усул ўзгарувчи қийматини матн шаклига келтириб, объектдаги сатрга улайди. Бутун сонлар ва сатрлар биргаликда ишлатилганда кутилмаган натижа келиб чиқиши мумкин. Масалан

String s = "four: " + 2 + 2;

Бу мисолда натижа «four: 4» эмас "four: 22" бўлади.

Бутун сонлар йиғиндиси биринчи хисобланиши учун қавслардан фойдаланиш лозим:

String s = "four: " + (2 + 2);

Символларни ажратиш

Битта символни сатрдан ажратиш учун charAt усулидан фойдаланиш мумкин. Бир неча символларни ажратиш учун getChars усулидан фойдаланиш мумкиндир. Қуйидаги мисолда String синфи объектидан символлар массивини ажратиш кўрсатилган.

```
class getCharsDemo {
public static void main(String args[]) {
String s = "This is a demo of the getChars method.";
int start = 10;
int end = 14;
char buf[] = new char[end - start];
s.getChars(start, end, buf, 0);
System.out.println(buf);
} }
```

Шунга эътибот бериш лозимки getChars усули чиқиш буферига end индексли символни ёзмайди. Шунинг учун мисода чиқарилаётган сатр 4 символдан иборат.

```
C:\> java getCharsDemo
demo
```

String синфида бутун сатрни char тиридаги массив сифатида қайтарувчи — toCharArray функцияси мавжуддир. Бу механизм альтернатив варианти сатрни byte типидаги массивга ёзиб, 16-битли символлардаги катта байтлар қийматларини ташлаб юборади. Бу усул getBytes деб аталиб, параметрлари getChars параметрлари билан бир хил маънога эга, фақата учинчи параметр сифатида byte типидаги массивдан фойдаланиш лозим.

Солиштириш

Икки сатр бир хиллигини аниқлаш учун String синфининг equals усулидан фойдаланиш лозим. Бу усулнинг альтернатив шакли equalsIgnoreCase деб аталиб, бу усулдан фойдаланилганда харфларрнинг катта кичиклиги фарқланмайди. Қуйида икки усулдан фойдаланиш кўрсатилган мисол берилган:

```
class equalDemo {
public static void main(String args[]) {
String s1 = "Hello";
String s2 = "Hello";
String s3 = "Good-bye";
String s4 = "HELLO";
System.out.println(s1 + " equals " + s2 + " -> " + s1.equals(s2));
System.out.println(s1 + " equals " + s3 + " -> " + s1.equals(s3));
```

```
System.out.println(s1 + " equals " + s4 + " -> " + s1.equals(s4));
System.out.println(s1 + " equalsIgnoreCase " + s4 + " -> " +
s1.equalsIgnoreCase(s4));
} }
```

Дастур бажариш натижаси:

```
C:\> java equalsDemo
Hello equals Hello -> true
Hello equals Good-bye -> false
Hello equals HELLO -> false
Hello equalsIgnoreCase HELLO -> true
```

String синфида equals усулининг махсус холи бўлган усуллар гурухи мавжуд. Бир сатрдаги остки сатрни иккинчи сатрдаги остки сатр билан солиштириш учун regionMatches усулидан фойдаланилади. Остки сатр параметр сифатида узатилган фрагмент билан бошланишини аниқлаш учун startsWithусулидан фойдаланилади. Сатр охири параметр билан бир хиллигини текшириш учун endsWith усули ишлатилади

Тенглик

Сатрларни солиштиришда equals усули ва оператор == турли текширишни амалга оширади. Агар equal усули символларни сатр ичида солиштирса оператор == объектларга кўрсаткичларни солиштиради ва улар бир хил объектга илова қилаётганлигини текширади. Қуйидаги мисолда сатрлар бир хил бўлгани билан, турли объектлар бўлгани учун equals ва == турли натижаларни беради.

```
class EqualsNotEqualTo {
public static void main(String args[]) {
String s1 = "Hello";
String s2 = new String(s1);
System.out.println(s1 + " equals " + s2 + " -> " + s1.equals(s2));
System.out.println(s1 + " == " + s2 + ", -> " + (s1 == s2));
} }
```

Дастур ишлаш натижаси:

```
C:\> java EqualsNotEqualTo
Hello equals Hello -> true
Hello == Hello -> false
```

Тартиблаш

Дастурда тартиблашдан фойдаланиш учун сатрларни солиштириш лозим. Бунинг учун String синфининг compareTo усулидан фойдаланиш мумкиндир. Кўп холларда иккита сатр бир хил эканлигини билишнинг ўзи етарли бўлмайди. Саралаш керак бўлган иловаларда қайси сатр иккичисидан кичик эканлигини билиш керак. Бу саволга жавоб бериш учун синфнинг усулидан фойдаланиш керак. Агар усул қайтарган бутун қиймат манфий бўлса, усул тегишли сатр параметр сатрдан кичик, мусбат бўлса катта. Агар усул қайтарган қиймат 0 бўлса сатрлар бир хил. Қуйидаги дастурда пуфакча

усулида сатрлар массиви тартибланиб compareTo усулидан фойдаланилади. Бу дастур сатрлар рўйхатини тартибланган шаклда чиқаради.

```
class SortString {
static String arr[] = {"Now", "is", "the", "time", "for", "all",
"good", "men", "to", "come", "to", "the",
"aid", "of", "their", "country" };
public static void main(String args[]) {
for (int j = 0; i < arr.length; j++) {
for (int i = j + 1; i < arr.length; i++) {
if (arr[i].compareTo(arr[j]) < 0) {
String t = arr[j];
arr[j] = arr[i];
arr[i] = t;
}
}
System.out.println(arr[j]);
}
} }
```

Символ ёки остки сатрни излаш

Маълум символ ёки остки қаторни излаш учун String синфига тегишли indexOf ва lastIndexOf усулларидан фойдаланилади. Бу усуллардан хар бири изланаётган символ индекси ёки остки сатр биринчи символи индексини қайтаради. Иккала холда хам агар излаш муваффақиятсиз тугаган бўлса -1 қиймат қайтарилади. Қуйидаги мисолда шу усуллардан фойдаланиш турли вариантлари келтирилган.

```
class indexOfDemo {
public static void main(String args[]) {
String s = "Now is the time for all good men " +
"to come to the aid of their country " +
"and pay their due taxes.";
System.out.println(s);
System.out.println("indexOf(t) = " + s.indexOf('f'));
System.out.println("lastIndexOf(t) = " + s.lastIndexOf('f'));
System.out.println("indexOf(the) = " + s.indexOf("the"));
System.out.println("lastIndexOf(the) = " + s.lastIndexOf("the"));
System.out.println("indexOf(t, 10) = " + s.indexOf('f' , 10));
System.out.println("lastIndexOf(t, 50) = " + s.lastIndexOf('f' , 50));
System.out.println("indexOf(the, 10) = " + s.indexOf("the", 10));
System.out.println("lastIndexOf(the, 50) = " + s.lastIndexOf("the", 50));
} }
```

Қуйида дастур ишлаш натижаси келтирилган. Сатрларда индекслар нольдан бошланади.

C:> java indexOfDemo

Now is the time for all good men to come to the aid of their country

and pay their due taxes.
indexOf(t) = 7
lastlndexOf(t) = 87
indexOf(the) = 7
lastlndexOf(the) = 77
index0f(t, 10) = 11
lastln
dex0f(t, 50) = 44
index0f(the, 10) = 44
lastlndex0f(the, 50) = 44

Назорат саволлари:

1. Сатр символли массивдан қандай фарқ қилади?
2. Сатрларни улаш хусусиятларини кўрсатинг.
3. Символларни олиш қандай амалга оширилади?
4. Сатрларни солиштириш қандай амалга оширилади?
5. Символни ёки остки сатрни излаш усулини кўрсатинг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 6. МАВЗУ: САТРЛАР БИЛАН ИШЛАШ ХУСУСИЯТЛАРИ

Режа: Нусха олишда сатрларни ўзгартириш; StringBuffer синфи; Конструкторлар; StringTokenizer синфи.

Нусха олишда сатрларни ўзгартириш

String синфи объектларини ўзгартириш мумкин бўлмагани учун хар гал сатрни ўзгартирганда StringBuffer объектига айлантириш, ёки бўлмаса String синфининг сатр нусхасини яратувчи усулларидан фойдаланиш лозим

substring

String синфи объектидан, **substring** усули ёрдамида остки сатрни ажратиб олиш мумкин. Бу усул берилган сатр индекслари диапазонидан янги символлар нусхасини яратади.. Керакли остки сатр фақат биринчи симоли индексини кўрсатиш мумкин. Бу холда кўрсатилган индесдан охиригача хамма символлардан нусха олинади. Бундан ташқари биринчи ва охирги индексни кўрсатиш мумкин, бу холда кўрсатилган индексдан иккинчи индексгача бўлган хамма символлардан нусха олинади.

"Hello World".substring(6) -> "World"

"Hello World".substring(3,8) -> "lo Wo"

concat

Сатрлар конкатенацияси яъни сатрларни улаш concat усули ёрдамида амалга оширилади. Бу усул String синфи янги объектини яратиб, берилган сатрни кўчиради ва усул параметрида кўрсатилган сатрни улайди.

"Hello".concat(" World") -> "Hello World"

replace

Символларни алмашириш replace усулига параметр сифатида икки символ узатилади. Сатр янги нусхасида биринчи символга мос символлар иккинчи символга алмаштирилади.

"Hello".replace('l' , 'w') -> "Hewwo"

toLowerCase и toUpperCase

Бу усуллар жуфтлиги хамма символларни қуйи ва юқори регистрга ўтказади.

"Hello".toLowerCase() -> "hello"

"Hello".toUpperCase() -> "HELLO"

trim

Берилган сатр олдидаги ва охиридаги бўшлик символлари trim усули ёрдамида олиб ташланади.

“Hello World “.trirn() -> "Hello World"

valueOf

Бирор маълумотни матнли сатрга айлантириш учун valueOf усулидан фойдаланилади. Бу статик усул Java тилидаги хамма маълумотлар турларида мавжуддир.

StringBuffer

StringBuffer — синфида сатрлар билан ишлаш учун керак бўлган усуллар мавжуддир. String синфи объектлари маълум узунликдаги ўзгармас сатрлардан иборат. StringBuffer объектлари узунлигии ва қиймати ўзгарувчан сатрлардан иборат. Java тилида иккала синфдан актив фойдаланилади, лекин дастурчилар + операторидан фойдаланилган холда String синфи объектларини ишлатадилар.

Конструкторлар

StringBuffer синфи объектини параметрларсиз яратиш мумкин, бу холда 16 символ учун жой ажратилади ва сатр узунлигини ўзгартириб бўлмайди. Бундан ташқари конструкторга буфер узунлигини кўрсатувчи бутун сон бериш мумкин. Бундан ташқари конструкторга сатр узатиш мумкин. Бу холда сатрдан объектга нусха олинади ва қўшимча 16 символга жой ажратилади. StringBuffer объекти узунлигини аниқлаш учун **length** усулидан, сатр учун ажратилган жойни аниқлаш учун **capacity** усулидан фойдаланиш мумкин. Масалан:

```
class StringBufferDemo {
public static void main(String args[]) {
StringBuffer sb = new StringBuffer("Hello");
System.out.println("buffer = " + sb);
System.out.println("length = " + sb.length());
System.out. println("capacity = " + sb.capacity());
} }
```

Дастур натижасидан кўриш мумкинки String-Buffer объектида қўшимча жой ажратилган.

```
C:\> java StringBufferDemo
buffer = Hello
length = 5
capacity = 21
```

ensureCapacity

StringBuffer объекти яратилгандан сўнг янги символларга жой ажратиш учун **ensureCapacity** усулидан фойдаланиш мумкин.

setLength

Буфердаги сатр узунлигини ўрнатиш учун setLength усулидан фойдаланиш мумкин. Агар объектдаги сатр узунлигидан катта қиймат берилса, усул янги сатр охирини нуль кодли символ билан тўлдиради.

charAt и setCharAt

Битта символни StringBuffer объектидан **charAt** усули ёрдамида ажратиш мумкин. Сатр керакли позициясига символ ёзиш учун **setCharAt** усулидан фойдаланилади. Қуйидаги мисолда иккала усулдан фойдаланиш кўрсатилган:

```
class setCharAtDemo {
public static void main(String args[]) {
StringBuffer sb = new StringBuffer("Hello");
System.out.println("buffer before = " + sb);
System.out.println("charAt(1) before = " + sb.charAt(1));
sb.setCharAt(1, 'i');
sb.setLength(2);
System.out.println("buffer after = " + sb);
System.out.println("charAt(1) after = " + sb.charAt(1));
} }
```

Дастур бажарилиши натижаси.

```
C:\> java setCharAtDemo
buffer before = Hello
charAt(1) before = e
buffer after = Hi
charAt(1) after = i
```

append

StringBuffer синфининг **append** усули одатда ифодаларда + операторидан фойдаланилганда чақирилади. Хар бир параметр учун String.valueOf усули чақирилади ва унинг натижаси StringBuffer объектига уланади. Бундан ташқари append усули StringBuffer объектига илова қайтаради. Бу эса мисолда кўрсатилганидек чақириқлар кетма кетлигини хосил қилишга имкон беради.

```
class appendDemo {
public static void main(String args[]) {
String s;
int a = 42;
StringBuffer sb = new StringBuffer(40);
s = sb.append("a = ").append(a).append("!").toString();
System.out.println(s);
} }
```

Бу мисол натижаси:

```
C:\> Java appendDemo
a = 42!
```

insert

Сатрларни улаш **insert** усули append ўхшаш бўлиб, хар бир маълумотлар типи учун бу усунинг ўз шакли бордир. Фақат append дан фарқли String.valueOf усули қайтарган символлрни, StringBuffer объекти охирига қўймайди, балки, биринчи параметри кўрсатган, буфердаги жойга қўяди. Кейинги мисолимизда "there" сатри "hello" ва "world!" орасига қўйилади.

```
class insertDemo {
public static void main(String args[]) {
StringBuffer sb = new StringBuffer("hello world !");
sb.insert(6,"there ");
System.out.println(sb);
} }
```

Бу дастур бажарилганда қуйидаги сатр чиқади:

C:\> java insertDemo

hello there world!

StringTokenizer

Матнни қайта ишлашда кўпинча матнни лексемалар - сўзлар (tokens) кетма кетлигига ажратиш керак бўлади. StringTokenizer синфи лексик тахлил деб аталувчи шундай ажратиш учун мўлжалланган. StringTokenizer синфи берилган сатрни ва ажратувчи символларни талаб қилади. Кўзда тутилган бўйича оддий ажратувчи символлар ишлатилади: бўшлик белгиси, табуляция, сатрни ўтказиш ва сатр бошига қайтариш. StringTokenizer объекти яратилгандан сўнг сатрдан кетма кет лексемаларни ажратиш учун nextToken усулидан фойдаланилади. Бошқа усул — hasMoreTokens — агар сатрда хали ажратилмаган лексемалар бўлса true қайтаради.

Қуйидаги мисолда “калит=қиймат” кўринишдаги сатрни тахлил қилиш учун StringTokenizer объекти яратилади. Чиқувчи сатрда “калит=қиймат” жуфтлиги қўш нуқта билан ажратилади.

```
import java.util.StringTokenizer;
class STDemo {
static String in = "title=The Java Handbook:" + "author=Patrick
Naughton:" + "isbn=0-07-882199-1:" + "ean=9 780078 821998:" +
"email=naughton@starwave. corn";
public static void main(String args[]) {
StringTokenizer st = new StringTokenizer(in, "=:");
while (st.hasMoreTokens()) {
String key = st.nextToken();
String val = st.nextToken();
System.out.println(key + "\t" + val);
}
}
 }
```

Назорат саволлари:

1. String ва StringBuffer қандай фарқ қилади?

2. StringBuffer синфи объектлари қандай яратилади?
3. StringBuffer синфи усулларини кўрсатинг.
4. Сатрларни конкатенация қилишда қандай усул чақирилади?
5. StringTokenizer синфи нима учун ишлатилади?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 7. МАВЗУ: СИНФЛАР ВА ОБЪЕКТЛАР

Режа: Синф; Компоненталарга мурожаат хуқуқлари; Конструктор; Синфнинг статик компоненталари.

Синф

Синтаксис бўйича, JAVA да синф – бу мавжуд бўлган типлар асосида янги яратилган структурланган тип.

Синф таърифи содда шакли:

<синф_типи> <синф_номи>{<синф_компонентлари_рўйхати>};

бу ерда:

синф_типи –**class** хизматчи сўзи;

синф_номи – идентификатор;

синф_компонентлари_рўйхати – синфга тегишли маълумотлар ва функциялар таърифи.

Функция – бу объектлар устида бажариладиган операцияларни аниқловчи синф усули.

Маълумотлар – бу объект структурасини хосил қилувчи майдон.

Синф объекти (синф нусхаси) ни таърифлаш учун қуйидаги конструкциядан фойдаланилади:

<синф_номи> <объект_номи>;

Объект орқали майдонларга ва усулларга қуйидагича мурожжат қилиш мумкин:

< объект_номи >. <майдон_номи>

< объект_номи >. <усул_номи>

Компоненталарга мурожат хуқуқлари

Компоненталарга мурожаат хукуки мурожаат спецификаторлари ёрдамида бошкарилади: **public, private, protected**.

Умумий (public) компоненталар дастурни ихтиёрий қисмида мурожаат хуқуқига эга. Улардан, ихтиёрий функция ушбу синф ичида ва синф ташқарида фойдаланса хам бўлади.

Хусусий (private) компоненталар синф ичида мурожаат хуқуқига эга, лекин синф ташқарисидан эса мурожаат килиш мумкин эмас. Компоненталардан ушбу улар тавсифланган синфдаги функция - аъзолари орқали фойдаланиш мумкин.

Химояланган (protected) компоненталар синф ичида ва хосила синфларда мурожаат хуқуқига эга.

JAVA тилида агар синф таърифида **class** сўзи ишлатилган бўлса хамма компоненталари умумий хисобланади.

Конструктор

Конструктор - бу синф объектларини автоматик инициализация килиш учун ишлатиладиган махсус компонентали функция.

Конструкторлар куриниши қуйидагича булиши мумкин:

<Синф номи> (<формал параметрлар руйхати>)
{<конструктор танаси>}

Бу компонента функция номи синф номи билан бир хил булиши лозим.

Дастурчи томонидан кўрсатилмаган холда хам new оператор ёрдамида синф объекти яратилганда ёки хотирада жойлашганда конструктор автоматик равишда чақирилади.

Конструктор объект учун хотирада жой ажратади ва маълумотлар – синф аъзоларини инициализациялайди.

Конструкторлар учун кайтарилувчи типлар, хатто void типи хам кўрсатилмайди

Конструкторлар ихтиёрий синфлар учун доимо мавжуд, лекин агарда у кўрсатилган холда тавсифланмаган бўлса, у автоматик равишда яратилади. Кўрсатилмаган холда параметрсиз конструктор ва нусха конструктори яратилади. Агарда конструктор очиқ холда тавсифланган бўлса, унда кўрсатилмаган холда конструктор яратилмайди. Кўрсатилмаган холда умумий (public) конструкторлар яратилади.

Мисол:

```
class Point { int x, y;
Point(int x, int y) {
this.x = x;
this. y = y;
}
Point() {
x = 0;
y = 0;
}
double distance(int x, int y) {
int dx = this.x - x;
int dy = this.y - y;
return Math.sqrt(dx*dx + dy*dy);
}
double distance(Point p) {
return distance(p.x, p.y);
} }
class PointDist {
```

```
public static void main(String args[]) {
Point p1 = new Point(0, 0);
Point p2 = new Point(30, 40);
System.out.println("p1 = " + pi.x + ", " + p1.y);
System.out.println("p2 = " + p2.x + ", " + p2.y);
System.out.println("p1.distance(p2) = " + p1.distance(p2));
System.out.println("p1.distance(60, 80) = " + p1.distance(60, 80));
} }
```

Қуйида дастур бажарилиш натижаси келтирилган:

C:\> java PointDist

p1 = 0, 0
p2 = 30, 40
p1.distance(p2) = 50.0
p1.distance(60, 80) = 100.0

This кўрсаткичи

Агарда конкрет объектга ишлов бериш учун синф аъзоси – функция чақирилса, унда шу функцияга объектга белгиланган кўрсаткич автоматик ва кўрсатилмаган холда узатилади. Бу кўрсаткич **this** исмига эга.

Синфни қуйидаги тасвирлаш мумкин:

```
class Point { int x, y;
Point(int x, int y) {
this.x = x;
this.y = y;
} }
```

Усулларни қўшимча юклаш

Язык Java тилида бир хил номли турли параметрлар рўйхатига эга усуллар яратиш мумкин. Бундай техника усулларни қўшимча юклаш (**method overloading**) деб аталади. Қўшимча юкланган усуллар конструкторлар бўлиши шарт эмас.

Пример:

```
class Point { int x, y;
Point(int x, int y) {
this.x = x;
this. y = y;
}
Point() {
x = 0;
y = 0;
}
double distance(int x, int y) {
int dx = this.x - x;
int dy = this.y - y;
```

```
return Math.sqrt(dx*dx + dy*dy);
}
double distance(Point p) {
return distance(p.x, p.y);
}
}
class PointDist {
public static void main(String args[]) {
Point p1 = new Point(0, 0);
Point p2 = new Point(30, 40);
System.out.println("p1 = " + pi.x + ", " + p1.y);
System.out.println("p2 = " + p2.x + ", " + p2.y);
System.out.println("p1.distance(p2) = " + p1.distance(p2));
System.out.println("p1.distance(60, 80) = " + p1.distance(60, 80));
}}
```

Ниже приведен результат работы этой программы:

```
C:\> java PointDist
p1 = 0, 0
p2 = 30, 40
p1.distance(p2) = 50.0
p1.distance(60, 80) = 100.0
```

finalize

Java тилида **finalize** номли усуллар киритиш имконияти мавжуд. Бу номли усуллар C++ тилидаги (калит белги **~)** ва Delphi тилидаги (калит сўз **destructor**) деструкторларга мосдир. Java бажариш мухити хар гал объектни ўчиришда шу услни чақиради.

Синфнинг статик компонеталари

Синф компанентаси ягона булиб ва хамма яратилган объектлар учун умумий булиши учун уни статик элемент сифатида таърифлаш яъни **static** атрибути оркали таърифлаш лозимдир. Объектларни яратишда синф статик маълумотлари такрорланмайди, яъни хар бир статик компонентлар бирдан-бир кўринишда мавжуд.

Статик усуллар фақат бошқа статик усулларга тўғридан тўғри мурожаат қилиши мумкин ва уларда this иловасидан фойдаланиш мумкин эмас. Ўзгарувчилар хам static типга эга бўлиши мумкин, бу холда уларга глобал ўзгарувчиларга ўхшаб дастур ихтиёрий қисмидан мурожаат қилиш мумкин. Статик усуллар ичида ностатик ўзгарувчиларга мурожаат қилиш мумкин эмас.

Синф статик маълумотларга фақатгина объект исми орқали мурожаат этиш мумкин. **<объект_номи>.<компонента_номи>**

Масалан:

```
complex a; a.count=5;
```

Лекин, статик компонетларга синф объекти аниқланмаган холда хам мурожаат этиш мумкин. Статистик компонентларга нафақат объект исми, балки синф исми орқали хам мурожаат этиш мумкин.

<синф_номи>. <компонента_номи>

Лекин шундай мурожаат фақатгина *public* компонентларга тегишли.

private статик компонентларга ташқаридан мурожаат этишда **функция – статик компонентлардан** фойдаланилади.

Бу функцияларни синф исми орқали чақириш мумкин.

<синф_номи> .: <статик_функция_номи>

Мисол.

```
class StaticClass {
static int a = 42;
static int b = 99;
static void callme() {
System.out.println("a = " + a);
} }
class StaticByName {
public static void main(String args[]) {
StaticClass.callme();
System.out.println("b = " + StaticClass.b);
} }
```

Дастур бажарилиши натижаси:

C:\> Java StaticByName

a = 42 b = 99

Назорат саволлари:

1. Синф қандай таърифланади?
2. Синф усулларини қўшимча юклаш мумкинми?
3. Конструкторлар вазифасини кўрсатинг.
4. Объектлар массиви яратилганда қандай конструкторлар чақирилади?
5. Статик компоненталар хусусий бўлиши мумкинми?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 8. МАВЗУ: СИНФЛАР ВА ОБЪЕКТЛАР БИЛАН ИШЛАШ ХУСУСИЯТЛАРИ

Режа: Runtime; Хотирани бошқариш; Бошқа дастурларни бажариш System синфи; Мухит хоссалари; Date синфи; Math синфи; Random синфи.

Runtime

Runtime синфи Java интерпретаторини инкапсуляция қилади. Бу синф объектини яратиш мумкин эмас, лекин статик усулидан фойдаланиб ишлаб турган объектга мурожаат қилиш мумкин. Одатда апплетлар ва бошқа дастурлар синф усулини чақирганда SecurityException истисноси хосил бўлади. Runtime объектини тўхтатиш учун exit(int code) усулини чақириш етарли.

Хотирани бошқариш

Java тилида хотира автоматик тозаланса хам дастур эффективлигини текшириш учун “уюм” хажмини аниқлаш ва эркин хотира хажмини хисоблаш мумкин. Бу маълумотни аниқлаш учун totalMemory ва freeMemory усулларидан фойдаланиш мумкин.

Зарур бўлганда хотира тозаловчини gc усулини чақириб автоматик ишга тушириш мумкин. Дастурга зарур хотирани хисоблаш учун аввал gc, сўнгра free-Memory усулларини чақириш керак. Шундан сўнг дастурни ишга тушириб freeMemory усули чақирилса, дастур қанча хотира ишлатишини аниқлаш мумкин.

Бошқа дастурларни бажариш

Хавфсиз мухитларда Java тилидан бошқа жараёнларни ишга тушириш учун фойдаланиш мумкин. Бунинг учун exec усулининг бир неча шаклларидан фойдаланиш мумкин. Усулга дастур номи ва бир неча параметрлар узатилади.

Қуйидаги мисолда Windows учун хос равишда exec усулидан фойдаланиб, notepad содда матн мухаррири жараёни ишга туширилади. Мисол тариқасида мухаррирга Java файлларидан бири узатилади. Этибор беринг — exec автоматик равишда "/" символларни Windowsга хос бўлган — "\" символларга алмаштиради.

```
class ExecDemo {
public static void main(String args[]) {
Runtime r = Runtime. getRuntime();
```

```
Process p = null;
String cmd[] = { "notepad", "/java/src/java/lang/Runtime.java" };
try {
p = r.exec(cmd);
} catch (Exception e) {
System.out.println("error executing " + cmd[0]);
}
} }
```

System

System синфида турли глобал функциялар ва ўзгарувчилар мавжуд. Масалан System.out.println() усули.

Бундан ташқари **currentTimeMillis** усули тизимли вақтни 1970 йил 1 январидан бўлиб ўтган миллисекундларда қайтаради.

Массивдан нусха олиш учун **arraycopy** усулидан фойдаланилади. Қуйида бир массивдан иккинчисига нусха олишга мисол келтирилган.

```
class ACDemo {
static byte a[] = { 65, 66, 67, 68, 69, 70, 71, 72, 73, 74 };
static byte b[] = { 77, 77, 77, 77, 77, 77, 77, 77, 77, 77 };
public static void main(
String args[]) {
System.out.println("a = " + new String(a, 0));
System.out.println("b = " + new String(b, 0));
System.arraycopy(a, 0, b, 0, a.length);
System.out.println("a = " + new String(a, 0));
System.out.println("b = " + new String(b, 0));
System.arraycopy(a, 0, a, 1, a.length - 1);
System.arraycopy(b, 1, b, 0, b.length - 1);
System.out.println("a = " + new String(a, 0));
System.out.println("b = " + new String(b, 0));
} }
```

Дастурдан кўриш мумкинки массивдан ўзига нусха олиш мумкин.

```
C:\> java ACDemo
a = ABCDEFGHIJ
b = MMMMMMMMMM
a = ABCDEFGHIJ
b = ABCDEFGHIJ
a = AABCDEFGHI
b = BCDEFGHIJJ
```

Мухит хоссалари

Java бажариш мухити Properties синфи объекти орқали мухит ўзгарувчиларига мурожаат қилишга имкон беради. Хоссалар тўлиқ рўйхатини олиш учун System.getProperties() усулини чақириш лозим.

Жадвал

Система стандарт хоссалари

Исм	Қиймат	Апплет учун рухсат
Java.version	Java интерпретатора версияси	ха
Java.vendor	Фойдаланувчи киритган идентификатор қатори	ха
java.vendor.url	Ишдаб чиқарувчи URLи	ха
java.class.version	Java API версияси	ха
java.class.path	CLASSPATH ўзгарувчиси қиймати	йўқ
java.home	Java мухити инсталляция қилинган каталог	йўқ
java.compiler	Компилятор JIT	йўқ
os.name	Операцион тизим номи	ха
os.arch	Дастур бажарилаётган компьютер архитектураси	ха
os.version	Web-тугун операцион тизими версияси	ха
file.separator	Платформага боғлиқ файл ажратувчилари (/ ёки \)	ха
path.separator	Платформага боғлиқ йўл ажратувчилари (: ёки ;)	ха
line.separator	Платформага боғлиқ сатр ажратувчилари (\n ёки \r\n)	ха
user.name	Жорий фойдаланувчи исми	йўқ
user.home	Фойдаланувчи каталоги	йўқ
user.dir	Жорий ишчи каталог	йўқ
user.language	2-символли махаллий тил коди	йўқ
user.region	2-символли мамлакат тил коди	йўқ
user.timezone	Кўзда тутилган вақт зонаси	йўқ
user.encoding	Кўзда тутилган бўйича белгилар коди	йўқ
user.encoding.pkg	Махаллий символларни Unicode кодига ўтказиш учун конверторлар пакети	йўқ

Date

Сана ва вақт билан ишлаш учун Date синфидан фойдаланилади. У орқали сана, йил , ой, хафта кунига, соат, минут, секундга мурожаат қилиш мумкин. Бу синф турли конструкторлари мавжуд. Энг соддаси — Date() — объектни жорий сана ва вақт билан инциализация қилади. Қолган уч конструктор қўшимча имкониятларга эга.

- Date(year, month, date) — кўрсатилган санани ўрнатади, вақт 00:00:00 (тунги) қийматга эга бўлади.
- Date(year, month, date, hours, minutes) — кўрсатилган сана ва вақтни ўрнатади, секунд 0 қийматни олади.
- Date(year, month, date, hours, minutes, seconds) — энг тўлиқ кўриниши, сана ва вақт хамда секундлар хам ўрнатилади.

get и set

Класс Date синфи объект атрибутларини ўрнатиш учун усулларга эга. Бу оилага кирувчи усулар — getYear, getMonth, getDate, getDay, getHours, getMinutes и getSeconds — бутун қиймат қайтаради.

Date синфи қийматини getTime усули long типидаги сон сифатида қайтаради. Бу сон 1970 йил 1 январидан ўтган миллисекундларга тенг.

Солиштириш

Date типидаги икки объектни солиштириш учун, санани миллисекундларга айлантириш лозим. Date синфи тўғридан тўғри солиштириш учун уч усулга эга: — before, after ва equals. Масалан

new Date(96, 2, 18).before(new Date(96, 2, 12)

true қайтаради, чунки ойнинг 12-куни 18-кунидан олдин келади.

Сатр ва сана

Объекты Date объектини турли форматдаги матнга конвертация қилиш мумкин. Аввалambор toString усули Date объектини қуйидагича сатрга алмаштиради “Thu Feb 15 22:42:04 1996”. Кейинги toLocaleString усули санани қисқароқ сатрга алмаштиради, масалан: “02/15/96 22:42:04”. Ва нихоят toGMTString усули санани Гринвич бўйича ўртача вақт форматига ўтказади: “16 Feb 1996 06:42:04 GMT”.

Math

Math синфи геометрия ва тригонометрияда ишлатиладиган сузувчи вергулли функцияларга эга. Бундан ташқари хисоблашларда ишлатиладиган икки константа мавжуд: — E (тахминан 2.72) ва PI (тахмина 3.14159).

Тригонометрик функциялар.

Қуйида келтирилган уч функция радианларда бурчакни ифодаловчи double типидаги параметрга эга бўлиб, мос тригонометрик функция параметрини қайтаради.

- sin(double a) радианда берилган a бурчак синусини қайтаради.
- cos(double a) радианда берилган a бурчак косинусини қайтаради.
- tan(double a) радианда берилган a бурчак тангенсини қайтаради.

Кейинги тўрт функция узатилган параметр қийматига мос бурчакни радианларда қайтаради.

- asin(double r) синуси r га тенг бурчакни қайтаради.
- acos(double r) косинуси r га тенг бурчакни қайтаради.
- atan(double r) тангенси r га тенг бурчакни қайтаради.
- atan2(double a, double b) тангенси a/b га тенг бурчакни қайтаради.

Даражага кўтариш, экспонента ва логарифм функциялари

- pow(double y, double x) x даражага кўтарилган y қайтаради. Масалан, pow(2.0, 3.0) тенг 8.0.
- exp(double x) e даражаси x қайтаради.
- log(double x) x натурал логарифмини қайтаради.
- sqrt(double x) x квадрат илдизини қайтаради.

Яхлитлаш

- ceil(double a) қиймати a дан катта ёки a га тенг бўлган энг кичик бутун сон қайтаради.
- floor(double a) қиймати a дан кичик ёки a га тенг бўлган энг катта бутун сон қайтаради.
- rint(double a) каср қисми олиб ташланган double типида a қийматини қайтаради.
- round(float a) энг яқин бутун сонга яхлитланган a қийматини қайтаради.
- round(double a) энг яқин узун бутун сонга яхлитланган a қийматини қайтаради.

Бундан ташқари Math синфида int, long, float ва double типлари билан ишловчи модул олиш, минимал ва максимал қийматни топиш усуллари полиморф версиялари мавжуд:

- abs(a) a модули (абсолют қиймати) ни қайтаради.
- max(a, b) ўз аргументлари энг каттасини қайтаради.
- min(a, b) ўз аргументлари энг кичигини қайтаради.

Random

Random — псевдотасодифий сонлар генератори бўлиб, унда ишлатилган алгоритм Дональд Кнут “Дастурлаш санъати ” китобининг 3.2.1 бўлимида келтирилган. Одатда бошланғич қиймат сифатида жорий вақт олинади, бу эса қайтарилувчи тасодифий сонлар олиниши эхтимоллигини камайтиради.

Random синфи объектидан 5 турдаги тасодифий сонларни олиш мумкин. Бу тип диапазони бўйича бир текисда тақсимланган бутун сонни олиш учун nextInt усулидан фойдаланилади. Шунга ўхшаш nextLong усули long типидаги тасодифий сонни қайтаради. Бундан ташқари nextFloat ва nextDouble усуллари мос равишда float ва double типидаги, 0.0..1.0 интервалда текис тақсимланган сонларни қайтаради. Ва нихоят nextGaussian усули ўрта қиймати 0.0 ва дисперсияси 1.0 бўлган нормал тақсимланган тасодифий сон қайтаради.

Назорат саволлари:

1.Бўш хотира хажмини қандай аниқлаш мумкин?
2.Мухит ўзгарувчиларига қандай мурожаат қилиш мумкин?
3.Вақт ва сана билан қандай синф ишлайди?
4.Математик синф усулларини кўрсатинг.
5. Қайси синф тасодифий сонлар генерацияси учун ишлатилади?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 9. МАВЗУ: СИНФЛАРДА ВОРИСЛИК

Режа: Ворис синф; Виртуал функциялар; Абстракт синфлар.

Ворис синф

Ворислик ўзининг барча аждодларининг хусусиятлари, маълумотлари, методлари ва воқеаларини мерос қилиб оладиган хосила синфини эълон қилиш имкониятини беради, шунингдек янги тавсифларни эълон қилиши хамда мерос сифатида олинаётган айрим функцияларни ортиқча юклаши мумкин. Базавий синфнинг кўрсатиб ўтилган тавсифларини мерос қилиб олиб, янги туғилган синфни ушбу тавсифларни кенгайтириш, торайтириш, ўзгартириш, йўқ қилиш ёки ўзгаришсиз қолдиришга мажбурлаш мумкин.

В JAVA тилида тўғридан тўғри аждод суперсинф деб аталади.

Хосила синфни эълон қилишнинг умумлашган синтаксиси:

class <синф номи>: [<кириш хуқуқини берувчи сецификатор >] <аждод синф номи> {…}

Конструктор ва деструкторларда ворислик

Конструкторлар мерос бўлмагани учун, хосила синфни яратишда ундан мерос бўлган маълумот – аъзолари асосий(базавий) синф конструктори орқали инициализацияланиши лозим. Асосий синф конструктори автоматик равишда чақирилади ва хосила синфни конструкторидан олдин бажарилади. Асосий (базавий) синфни конструкторининг параметрлари хосила синфни конструкторни аниқлашда кўрсатилади. Шундай қилиб аргументларни хосила синфни конструкторидан асосий (базавий) синфни конструкторига узатиш вазифаси бажарилади.

Асос синф конструктори параметри хосила синф конструктори таърифида **super** калит сўзи ёрдамида кўрсатилади.

Пример:

```
class Point3D extends Point { int z;
Point3D(int x, int y, int z) {
super(x, y); // Бу ерда суперсинф конструктори this.z=z чақирилади;
public static void main(String args[]) {
Point3D p = new Point3D(10, 20, 30);
System.out.println( " x = " + p.x + " y = " + p.y +
" z = " + p.z);
} }
```

Дастур бажарилиш натижаси:

C:\> java Point3D

x = 10 y = 20 z = 30

Синф объектлари пасдан тепага қараб конструкторланади: аввало асосий(базавий), кейин эса компонент – объектлар (агарда улар мавжуд бўлса), ундан кейин эса хосила синфнинг ўзи. Шундай қилиб, хосила синфнинг объекти қуйи объект сифатида асосий (базавий) синф объектини ўз ичига олади.

Объектлар тескари тартибда ўчирилади: аввало хосила, кейин унинг компонент – объектлари, ундан кейин эса асосий(базавий) объект.

Шундай қилиб объектни ўчириш тартиби унинг конструкторлаш тартибига нисбатан тескари бўлади.

Виртуал функциялар

Виртуал функциялар механизмига бирор компонент функциянинг хар бир хосилавий синфда алохида варианти мавжуд бщлиш лозим бщлганда мурожаат қилинади. Бундай функцияларга эга синфлар **полиморф** синфлар деб аталади ва объектли дастурлашда ахлохида ўринга эга.

Виртуал функциялар кечки ёки динамик боғланиш механизми асослангандир.

Кечки боғланишда эрта боғланишга ўхшаб адреслар статик равишда компиляция жарёнида эмас, балким динамик дастур бажарилиши жараёнида аниқланади. Боғлаш жараёни виртуал функцияларни адреслар билан алмаштиришдан иборат. Виртуал функциялар адреслар хақида маълумот сақланувчи жадвалдан фойдаланади.

Виртуаллик ворисликка ўтади. Функция виртуал деб эълон қилингандан сўнг хосила синфда қайта таърифи(шу прототип билан) бу синфда янги виртуал функцияни яратади.

JAVA тилида хама усуллар виртуалдир.

Мисол.

Мисолда синф / суперсинф сифатида боғланган икки синф кўрилган бўлиб, суперсинф ягона усули қайта таърифланган.

```
class A { void callme() {
System.out.println("Inside A's callrne method");
} }
class B extends A { void callme() {
System.out.println("Inside B's callme method");
}
 }
class Dispatch {
public static void main(String args[]) {
A a = new B();
a.callme();
}
}
```

Эътибор беринг main усули ичида А синфи ўзгарувчиси таърифланиб, В синфи объекти ёрдамида инициализация қилинган. Кейинги қаторда callme усули чақирилган. Транслятор А синфида callme усули мавжудлигини текширди, бажарувчи тизим ўзгарувчида В объекти сақланганлиги учун, А синфи эмас, В синфи callme усулини чақиради. Қуйида дастур бажарилиши натижаси келтирилган:

C:\> Java Dispatch

Inside B's calime method

Мисолда *усулларни динамик тайинлаш* (**dynamic method dispatch**) механизмидан фойдаланилган.

Усулларни қайта таърифлаш

Point синфининг янги остки синфи Point3D ўз суперсинфининг distance усулини мерос қилиб олади (мисол PointDist.java). Лекин Point синфида текисликда нуқталар орасида масофа қайтарувчи distance(mt x, int y) усули берилган. Биз бу усулни уч ўлчовли фазога мос келадиган қилиб, қайта таърифлашимиз (**override**) лозим. Кейинги мисолда distance усулини қўшимча юклаш (overloading), ва қайта таърифлаш (overriding) кўрсатилган.

```
class Point { int x, y;
Point(int x, int y) {
this.x = x;
this.y = y;
}
double distance(int x, int y) {
int dx = this.x - x;
int dy = this.y - y:
return Math,sqrt(dx*dx + dy*dy);
}
double distance(Point p) {
return distance(p.x, p.y);
}
}
class Point3D extends Point { int z;
Point3D(int x, int y, int z) {
super(x, y);
this.z = z;
(
double distance(int x, int y, int z) {
int dx = this.x - x;
int dy = this.y - y;
int dz = this.z - z;
return Math.sqrt(dx*dx + dy*dy + dz*dz);
}
double distance(Point3D other) {
return distance(other.x, other.y, other.z);
```

```
}
double distance(int x, int y) {
double dx = (this.x / z) - x;
double dy = (this.y / z) - y;
return Math.sqrt(dx*dx + dy*dy);
}
}
class Point3DDist {
public static void main(String args[]) {
Point3D p1 = new Point3D(30, 40, 10);
Point3D p2 = new Point3D(0, 0, 0);
Point p = new Point(4, 6);
System.out.println("p1 = " + p1.x + ", " + p1.y + ", " + p1.z);
System.out.println("p2 = " + p2.x + ", " + p2.y + ", " + p2.z);
System.out.println("p = " + p.x + ", " + p.y);
System.out.println("p1.distance(p2) = " + p1.distance(p2));
System.out.println("p1.distance(4, 6) = " + p1.distance(4, 6));
System.out.println("p1.distance(p) = " + p1.distance(p));
} }
```

Дастур бажарилиши натижаси:

C:\> Java Point3DDist

p1 = 30, 40, 10

p2 = 0, 0, 0

p = 4, 6

p1.distance(p2) = 50.9902

p1.distance(4, 6) = 2.23607

p1.distance(p) = 2.23607

Биз уч ўлчовли ва икки ўлчовли нуқталар орасида керакли масофани олдик. Мисолда *усулларни динамик тайинлаш* (**dynamic method dispatch**) механизмидан фойдаланилган.

final

Хамма усуллар ва ўзгарувчилар кўзда тутилган бўйича қайта таърифланиши мумкин. Агар ворис синфда бирор ўзгарувчи ёки усулни қайта таърифлашга хаққи йўқлиини кўрсатиш учун уларни final (Delphi / C++ тилида **virtual** сўзини ёзмаслик керак) деб таърифлаш лозим.

final int FILE_NEW = 1;

Қабул қилинган қоида бўйича final типидаги ўзгарувчиларни номлашда фақат юқори регистрдаги символлардан фойдаланилади. (C++ тилида препроцессор константалар). Баъзида final- усуллардан фойдаланиш код бажарилишини тезлаштиради — чунки, транслятор уларни жойлаштирилувчи (inline) код деб эълон қилади (байт-код тўғридан тўғри кодга жойлаштирилади).

Абстракт синфлар

Жуда бўлмаса битта абстракт усулга эга синф абстракт синф деб аталади. Абстракт усул деб қуйидаги кўринишга эга компонент функцияга айтилади:

abstract<тип> <исм > (< формал_параметрлар_рўйхати>);

Бу синф объектларини яратиш мумкин эмас. Абстракт синф фақат хосила синфлар учун асос синф сифатида ишлатилиши мумкин.

Хар қандай abstract усулга эга синф, abstract деб таърифланиши шарт. Бундай синфларда тўлиқ реализация мавжуд бўлмагани учун, new оператори ёрдамида вакилларини яратиш мумкин эмас. Бундан ташқари абстракт конструкторлар ва статик усуллар эълон қилиш мумкин эмас. Абстракт синф хар қандай вориси ёки суперсинф абстракт усулларини тўлиқ реализация қилиши керак, ёки ўзи абстракт деб элон қилиниши керак.

```
abstract class A {
abstract void callme();
void metoo() {
System.out.println("Inside A's metoo method");
} }
class B extends A {
void callme() {
System.out.println("Inside B's callme method");
} }
class Abstract {
public static void main(String args[]) {
A a = new B():
a.callme():
a.metoo():
} }
```

Бу мисолда хосила синфда реализация қилинган callme усули ва суперсинфда реализация қилинган metoo усулларини чақириш учун усулларни динамик тайинлаш усулидан фойдаланилган.

C:\> Java Abstract

Назорат саволлари:

1. Нима учун аввал аждод синф конструкторлари чақирилиб, сўнгра авлод синф конструктори чақирилади?
2. Усулларни динамик боғлаш деб нимага айтилади?
3. Ворисликда усуллар қандай қўшимча юкланади?
4. Синфлар библиотекасини қуришда ворисликдан қандай фойдаланилади?
5. Абстракт синфлар нима учун ишлатилади?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 10. МАВЗУ: ВОРИСЛИКДАН ФОЙДАЛАНИШ ХУСУСИЯТЛАРИ

Режа: Глобал суперсинф; equals ва toString усуллари; Умумлашган дастурлаш.

Object: глобал суперсинф

Object синфи хамма синфлар аждоди хисобланади. Java тилида хар бир синф Object синфини кенгайтиради. Лекин class Employee extends Oblects ёзиш шарт эмас. Агар суперсинф ошкор кўрсатилмаган бўлса Object суперсинф хисобланади. Java тилида хар бир синф Object синфини кенгайтиргани учун Object синфи имкониятларини билиш мухимдир.

Object типидаги ўзгарувчини ихтиёрий типдаги объектга илова сифатида ишлатиш мумкин:

Object obj = new Employee("Гарри Хакер", 35000);

Бу типдаги ўзгарувчидан фойдаланиш учун аввал бошланғич типни аниқлаб, типларни келтиришни амалга ошириш лозим:

Employee e = (Employee) obj;

equals ва toString усуллари

Object синфининг equals усули икки объект бир хиллигини текширади. Лекин equals усули Object синфига тегишли бўлгани учун, иккаласи бир хотира қисмига илова қилганлигини текширади. Икки объект эквивалентлигини текшириш учун equals усулини қўшимча юклаш лозим. Мукаммал equals усули яратиш қоидалари.

1. Ошкор otherObject параметрини чақириш — кейинчалик унинг типини other деб аталган бошқа ўзгарувчи типига келтириш лозим.

2. Текшириш, this ва otherObject иловалар бир хилми:

i f (this == otherObject) return true;

Одатда объектлар майдонини солиштиргандан кўра иловаларни солиштириш осондир.

3. Текшириш otherObject илова нульга (null) тенгми. Агар ха бўлса false қиймат қайтариш. Бу текширишни албатта амалга ошириш лозим.

if (otherObject == null) return false;

4. Текшириш this ва other объектлари битта синфга тегишлими.

Бу текшириш "симметриклик қоидасига " кўра мажбурийдир.

if (getClass() != otherObject.getClass()) return false;

5. Талаб қилинган синф ўзгарувчисига otherObject объектини ўзгартириш:

ClassName other = (ClassName)otherObject;

6. Хамма майдонларни солиштириш. Асосий типдаги майдонлар учун == оператори, объектли майдонлар учун —equals усули қўлланади. Агар икки объект хамма мйдонлари бир хил бўлса true қайтариш, акс холда false.

return fieldl == other.fieldl

&& field.2. equals (other . field2)

Масалан.

```
class Employee{
public boolean equals(Object otherObject) {
// Объектларни тез солиштириш,
if (this == otherObject) return true;
// Агар ошкор параметр — null, false қиймат қайтаради,
if (otherObject == null) return false;
// Агар синфлар устма уст тушмаса, улар эквивалент эмас.
if (getClass () != otherObject.getClassO) return false;
// Объект otherObject типи Employee ва у нульга тенг эмас.
Employee other = (Employee) otherObject;
// Объектлар майдонларини солиштириш,
return name.equals(other.name)
&& salary = other.salary
&& hireDay.equals(other.hireDay);
}
}
```

Объект типини getClass усули орқали аниқланади. Объектлар ўзаро тенг бўлиши учун бир синф объектлари бўлишлари керак.

Ворис ичида авввал сперсинф equals усулини чақириш лозим. Агар бу текшириш false қиймат қайтарса, демак объектлар тенг эмас. Агар текшириш муваффақиятли бажарилса остки синф майдонларини текширишга ўтиш мумкин.

Масалан қуйидагича.

```
class Manager extends Employee
{
public boolean.equals(Object otherObject)
{
if (!super.equals(otherObject)) return false;
Manager.other = (Manager)otherObject;
// Усул super.equals текширади
// this ва otherObject объектлари битта синфга тегишлими.
return bonus == other.bonus;
} }
```

Object синфининг яна бир мухим усули toString, бўлиб объектни сатр шаклида қайтаради. Бу усул деярли хамма синфларда қўшимча юкланади, ва объект холатини босмага чиқаришга мўлжалланган.

Кўп (хаммаси эмас) toString усуллари синф номи дан иборат бўлиб, квадрат қавсларда майдонлари қийматлари кўрсатилади. Қуйида Employee синфининг toString усули реализацияси кўрсатилган.

```
public String toString()
{
return "Employee[name" + name
+ ",salary =" + salary
+ ",hireDay =" + hireDay
}
```

Бу усулни такомиллаштириш мумкин. Синф номини toString усулига киритмасдан, getClass().getName() усулини чақирамиз ва синф номини ўз ичига олган сатрни оламиз.

```
public String toString()
{
return getClass().geName()
+ "[name=" + name
+ ",salary=" + salary
+ ",hireday=" + hireDay
}
```

Энди toString усули ворис синфлар билан хам ишлайди.

Албатта ворис синф яратган ластурчи ўз toString усулини яратиши ва ворис синф номини қўшиши лозим. Агар суперсинфда getClass ().getNamef() усули чақирилса, ворис синф super.ToString ( ) усулини чақиради. Manager синфида toString усулига мисол.

```
class manager extends Employee
{
public String toString()
{
return super.toString()+ "[bonus=" + bonus
```

Энди Manager синфи объекти холати қуйидаги шаклда чиқарилади:

Manager[name=...,salary=...,hireDay=...][bonus=...]

Агар объект сатр билан "+" амали ёрдамида конкатенация қилинса Java тили компилятори объект жорий холатини олиш учун автоматик равишда toString усулини чақиради.

Бирорр х — ихтиёрий объект учун дастурчи System.out.println(x) усулини чақирсин;

Бу холда println усули х. toString () усулини чақиради ва натижа сатрини чиқаради.

Object синфида аниқланган toString усули синф номи ва объект адресини чиқаради. Масалан

System.out.println(System.out);

чақириш натижасида қуйидаги сатр хосил бўлади

java.io.PrintStream@2f668 4

Бунинг сабаби шуки PrintStream синфида toString усули қўшимча юкланмаган.

Стандарт библиотекага тегишли кўп синфларда toString усули шундай аниқланганки, унинг ёрдамида дастурни созлаш учун керакли маълумот олиш мумкин. Баъзи созловчилар объектлар холатини экранда акслантинриш

учун toString усулини чақиришга имкон беради. Шунинг учун дастур трассировкасида, қуйидаги ифодалардан фойдаланиш мумкин

System.out.println{"Жорий холат = " + position);

Умумлашган дастурлаш

Object типидаги ўзгарувчиларда ихтиёрий синф ўзгарувчилари қиймати сақланиши мумкин, масалан String синфи:

Object obj = "Салом"; // Тўғри.

Лекин сонлар, символлар ва мантиқий ўзгарувчилар объектларга кирмайди.

obj = 5 ; // Нотўғри.

obj = false; // Нотўғри.

Бундан ташқари хамма типдаги массивлар, уларда объектлар ёки асосий типлардаги ўзгарувчилар сақланишига қарамай Object синфи вориси хисобланади.

Employee staff [] = new Employee[10];

Object arr = staff; // Тўғри.

arr = new int[10]; // Тўғри.

Бирор синфга тегишли объектлар массивини Object синфи объектлари массивига айлантириш мумкин. Масалан, Employee[] синфи массивини Object[ ] синфи массивини кутаётган усулга узатиш мумкин. Бу усул умумлашган дастурлаш учун фойдалидир (generic programming).

Қуйида умумлашган дастурлаш концепциясини кўрсатувчи мисол келтирилган. Бу мисолда массивда элемент индексини аниқлаш лозим.

```
static int find(Object[] a, Object key)
(int i;
For
 (i =0; i < a.length; i++)
if (a[i].equals(key)) return i;
return -1; // Индекс топилмаган.
}
Например,
Employee staff[] = new Employee[10];
Employee harry;
int n = find(staff, harry);
```

Шуни таъкидлаб ўтиш лозимки Object[ ] тпи массивини фақат бирор синф объектлари массивига айлантириш мумкин. Преобразовать массив типа int[ ] в массив типа Object[] типи массивига int[ ] типи массивини ўзгартириш мумкин эмас.

Агар бирор синф объектларидан массив, Object[ ] типидаги массивга айлантирилса, умумлашган массив бошланғич тип хақидаги маълумотни ўзида сақлаб қолади. Бу массивга бошқа типдаги элементни жойлаштириш мумкин эмас.

Назорат саволлари:

1. Хамма синфлар қайси синфнинг ворисларидир?
2. Глобал суперсинф усулларини кўрсатинг.
3. Фойдаланувчи солиштириш усули қандай яратилади?
4. Нима учун toString усули ишлатилади?
5. Умумлашган дастурлаш мохиятини тушунтиринг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 11. МАВЗУ: ИНТЕРФЕЙСЛАР

Режа: Интерфейс таърифи; Оператор implements; Интерфейсларда ўзгарувчилар.

Интерфейс таърифи

Интерфейс — бу усуллар жамламаси ошкор спецификацияси бўлиб, шу спецификацияни реализация қилаётган синфда бу усуллар таърифи албатта берилиши лозим. Интерфейсда бу усуллар реализацияси берилмайди. Абстракт синфлар каби интерфейслар кўплик ворисликда фойдаланиши мумкин. Конкрет синф фақат битта суперсинф вориси бўлиши мумкин лекин чекланмаган сондаги интерфейслар реализация қилиниши мумкин.

Интерфейслар

Java тилида интерфейслар усулларни дастур бажарилиши давомида динамик қўллашни (resolution) амалга ошириш учун ишлатилади. Интерфейслар синфларга ўхшайди, лекин усуллар реализацияси бўлмайди. Синф ихтиёрий сондаги интерфейсга эга бўлиши мумкин. Синфда интерфейлар усулари тўла тўпламини реализация қилиш лозим. Синф бу усуллари сигнатураси шу синф реализация қилаётган итерфейс усуллари сигнатураси билан бир хил бўлиши лозим. Интерфейслар синфлар ворисликка асосланган иерархияси билан кесишмайдиган ўз иерархиясига эга. Бу ворислик иерархияси билан боғланмаган турли синфларда битта интрфейсни реализация қилишга имкон беради. Интерфейслар кучи шундан иборат. Интерфейслар C++ тилида кўплик ворислик механизми аналогидир, лекин улардан фойдаланиш қулайроқ

Оператор interface

Интерфейс таърифи синф таърифига ўхшаш, фарқи шундаки интерфейсда маълумотлар эълони ва конструкторлар йўқдир. Интерфейс таърифи умумий кўриниши:

```
interface ном {
натижа_типи усул_номи1(параметрлар рўйхати);
тип final1_ўзгарувчи_исми = қиймат;
}
```

Интерфейсда эълон қилинган усуллар танаси йўқдир. Интерфейсда эълон қилинган ўзгарувчилар кўзда тутилган бўйича final – ўзгарувчилар хисобланади. Шунинг учун реализация қилувчи синфда уларнинг қийматини ўзгартириш мумкин эмас. Бундан ташқари ўзгарувчиларни интерфейсда таърифланганда константа қиймат билан инициализация қилиниши керак.

Қуйида callback номли усулга эга ва int типидаги параметрга эга интерфейс таърифи берилган.

```
interface Callback {
void callback(int param);
}
```

Оператор implements

Оператор implements — бирор интерфейс ёки интерфейсларни реализация қилувчи синф таърифига қўшимчадир

class синф_номи [extends суперсинф]
[implements интерфейс0 [, интерфейс1...]] { синф танаси }

Агар синфда бир нечта интерфейслар реализация қилинса, уларнинг номлари вергуль билан ажратилади. Қуйида интерфейсни реализация қилувчи синфга мисол берилган:

```
class Client implements Callback {
void callback(int p) {
System.out.println("callback called with " + p);
} }
```

Қуйидаги мисолда один таърифи берилган интерфейс callback усули, интерфейсга илова – ўзгарувчи орқали чақирилади:

```
class TestIface {
public static void main(String args[]) { Callback c = new client();
c.callback(42);
} }
```

Қуйида дастур бажарилиши натижаси беилган:

C:\> Java TestIface
callback called with 42

Интерфейсларда ўзгарувчилар

Интерфейслардан турли синфларга биргаликда фойдаланилувчи константаларни импорт қилиш учун фойдаланиш мумкин. Бу холда бирор синфда интерфейс реализация қилинса интерфейс ўзгарувчилари номлари бу синфда константа сифатида кўринади. Бу C ва C++ тилларида константларни #define директиваси ёки Pascal / Delphi тилларида const калит сўзи ёрдамида беришга мосдир.

Агар интерфейс ўз ичига усулларни олмаса, интерфейс реализацияси деб эълон қилинган синф хеч нарса реализация қилмайди. Константаларни синф номлар фазосига импорт қилиш учун final модификаторли ўзгарувчилардан фойдаланиш қулайдир.

```
import java.util.Random;
interface SharedConstants { int NO = 0;
int YES = 1;
int MAYBE = 2;
int LATER = 3;
```

```
int SOON = 4;
int NEVER = 5; }
class Question implements SharedConstants {
Random rand = new Random();
int ask() {
int prob = (int) (100 * rand.nextDouble());
if (prob < 30) return NO; // 30%
else if (prob < 60) return YES; // 30%
else if (prob < 75) return LATER; // 15%
else if (prob < 98) return SOON; // 13%
else return NEVER; // 2% } }
class AskMe implements SharedConstants {
static void answer(int result) {
switch(result) {
case NO:
System.out.println("No");
break;
case YES:
System.out.println("Yes");
break;
case MAYBE:
System.out.println("Maybe");
break;
case LATER:
System.out.println("Later");
break;
case SOON:
System.out.priniln("Soon");
break;
case NEVER:
System.out.println("Never");
break;
} }
public static void main(String args[]) {
Question q = new Question();
answer(q.ask());
answer(q.ask());
answer(q.askO);
answer(q.ask());
} }
```

Эътибор беринки дастур хар гал ишлатилганда хар хил натижа беради, чунки унда java.util пакетига тегишли Random тасодифий сонлар генератор ишлатилган.

C:\> Java AskMe

Later

Scon
No
Yes

Назорат саволлари:

1. Интерфейс таърифини келтиринг.
2. Интерфейс синфдан қандай фарқ қилади?
3. Нима учун оператор implements ишлатилади?
4. Интерфейс нима учун ишлатилади.
5. Биргаликда ишлатиладиган константаларни турли синфларга импорт қилиш учун интерфейсдан қандай фойдаланилади?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 12. МАВЗУ: ПАКЕТЛАР

Режа: Пакетлар; Оператор package; Оператор import; Мурожаатни чеклаш.

Пакет тушунчаси

Пакет (**package**) — бу синфлар номларини ажратиш учун ишлатиладиган контейнердир. Масалан сиз List синфини яратиб пакетга жойласангиз, кимдир яна List номли синф яратгани учун келиб чиқиши мумкин бўлган конфликтлар хақида ўйламасангиз хам бўлади.

Пакетлар

Мисолларда ишлатган хамма идентификаторлар битта номлар фазосида (name space) жойлашган эди. Шунинг учун конфликтларни олдини олиш учун, хар бир синф уникал номга эга бўлиши керак. Пакетлар — номлар фазоси билан ишлаш хамда кўринишни чеклаш учун мўлжалланган механизмдир. Хар бир .java файлда бир хил тўртта ички қисм мавжуд бўлиб, шу чоққача кўрилган мисолларда битта қисмдан фойдаланилган. Қуйида Java файл умумий кўриниши берилган.

ягона оператор package (шартли эмас)
ихтиёрий сондаги import операторлари(шартли эмас)
очиқ (public) синф ягона эълони
ихтиёрий сондаги пакет ёпиқ (private) синфлари (шартли эмас)

Оператор package

Берилган Java файлида учрайдиган биринчи оператор—package оператори бўлиб, берилган файлдаги синфлар қайси пакетда жойлашиши лозимлиги хақида трансляторга маълумот беради. Пакетлар синфлар номлари сақланувчи ажратилган номлар фазоларини аниқлайди. Агар сиз қуйидагича синфни маълум пакетда жойлашувчи деб эълон қилсангиз

package java.awt.image;

бу синф коди java/awt/image каталогда жойлашиши лозим.

ИЗОХ

Java транслятор пакетлар иерархияси илдизи деб қарайдиган каталогни CLASSPATH мухит ўзгарувчиси ёрдамида бериш мумкин. Бу ўзгарувчи ёрдамида пакетлар иерархияси учун бир нечта илдиз каталог бериш мумкин (оддий PATH даги каби ; орқали).

Пакетларда синфлар трансляцияси

Синфни пакетга жойлаштиришда каталоглар иерархияси пакетлар иерархиясига мос келиши лозим. Пакетни қайта номлаш учун унинг синфлари сақланаётган каталогни қайта номлаш лозим.

Масалан PackTest номли синф test пакетида яратилиши лозим. Бунинг учун test каталоги яратилиб, бу каталогга PackTest.Java файли жойлаштирилади ва трансляция қилинади. Хозирча хамма нарса жойида. Лекин ишга туширишга уринсангиз интерпретатор қуйидаги маълумотни чиқаради «can't find class PackTest» («PackTest синфини топа олмаяпман»). Янги синф test номли пакетда сақланади, шунинг учун хамма пакетлар иерархиясини пакетлар номларини нуқта билан ажратган холда кўрсатиш лозим - test.PackTest. Бундан ташқари каталоглар иерархиясида битта даражага кўтарилиб, яна «java test.PackTest» териш лозим, ёки CLASSPATH ўзгарувчига яратилаётган синфлар иерархиясидаги энг юқори каталогни киритиш лозим.

Оператор import

Ихтиёрий package операторидан сўнг, лекин берилган Java-файлдаги синфлар таърифидан олдин, import операторлари рўйхати келиши мумкин. Пакетлар синфларни бирбиридан ажратиш учун яхши механизмдир, шунинг учун Java тилида хамма стандарт синфлар пакетларда сақланади. Оператор **import** умумий кўриниши:

import пакет1 [.пакет2].(синфноми|*);

Бу ерда *пакет1* — юқори даражадаги пакет номи, *пакет2* — биринчи пакетга жойлаштирилган ва нуқта билан ажратилган, шартли бўлмаган пакет номи. Ва нихоят пакетлар иерархиясида йўл кўрсатилгандан сўнг, ёки синф номи кўрсатилади ёки метасимвол юлдузча. Юлдузча шуни билдирадики, агар Java-трансляторга пакети кўрсатилмаган синф керак бўлса, синф номи ўрнига юлдузчали пакетни кўриб чиқиши керак. Қуйида келтирилган кодда import операторидан фойдаланиш иккала шакли келтирилган:

import java.util.Date
import java.io.*;

Лекин шарт бўлмаса import оператори юлдузчали шаклидан фойдаланмаган мақул, чунки трансляция вақти узаяди (дастур хажми ва бажарилиш вақтига таъсир қилмайди).

Хамма Java олдиндан киритилган стандарт синфлари классы, java номли пакетда сақланади. Тил асосий функциялари java.lang пакетида сақланади. Бу пакет транслятор томонидан хамма дастурларга импорт қилинади. Бу хамма дастур бошига қуйидаги операторни жойлаштиришга эквивалент

import java.lang.*;

Агар юлдузчали import оператори ёрдамида уланаётган икки пакетда, бир хил номли синфлар мавжуд бўлса лекин улардан фойдаланилмаса транслятор маълумот бермайди. Лекин бундай синфдан фойдаланишга уринилса, дарров хато хақида маълумот чиқади ва import операторини қайси пакет синфи назарда тутилгани ошкор кўрсатилган холда қайта ёзиш зарур бўлади.

class MyDate extends Java.util.Date { }

Мурожаатни чеклаш

Java маълумотлар ва усуллар кўриниш сохасини созлаш мақсадида бир неча химоя даражаларига эга. Пакетлар сабабли Java кўриниш сохасига тегишли тўртта категория билан ишлай олиши лозим:

• Битта пакетдаги остки синфлар.
• Битта пакетдаги синфлар
• Турли пакетдаги остки синфлар.
• Остки синф бўлмаган ва битта пакетга тегишли бўлмаган синфлар.

Java тилида учта мурожаат хуқуқи мавжуд: private (ёпиқ), public (очиқ) ва protected (химояланган). Жадвал ячейкалари қиймати модификаторлар (устун) ва кўрсатилган жой (сатр) комбинацияси билан аниқланади.

	private	модификатор мавжуд эмас	private protected	protected	public
Ўша синф	ха	ха	ха	ха	ха
Ўша пакетдаги ворис синф	йўқ	ха	ха	ха	ха
Ўша пакетдаги мустақил синф	йўқ	ха	ха	ха	ха
Бошқа пакетдаги ворис синф	йўқ	йўқ	ха	ха	ха
Бошқа пакетдаги мустақил синф	йўқ	йўқ	йўқ	йўқ	ха

Бир кўринишда мураккаб туйилиши мумкин. лекин бир нечта содда қоидалар мавжуд. Элемент, public деб аниқланса, ихтиёрий жойдан мурожаат қилиш мумкин. Элемент private деб аниқланса, фақат синф ичида мурожаат қилиш мумкин. Агар элемент мурожаат даражаси модификатори кўрсатилмаган бўлса, бундай элемент пакет синфлари ва остки синфларида кўринади. Java тилида кўзда тутилган бўйича шу мурожаат даражасидан фойдаланилади. Агар элемент protected деб эълон қилинган бўлса, унга пакет ташқарисида хам мурожаат қилиш мумкин, фақат остки синфларда. Агар элемент фақат остки синфларда кўриниши шарт бўлса, бир пакетга тегишлими ёки йўқлигидан қатъий назар private protected комбинациясидан фойдаланиш лозим.

Қуйидаги узун мисолда мурожаат даражалари модификаторлари хамма комбинациялари берилган.Биринчи пакетда учта синф таъртфи берилган: Protection, Derived ва SamePackage. Бу синфлар биринчисида бешта ўзгарувчи таърифи берилган — хар бир мурожаат даражаси модификатори комбинацияси учун битта ўзгарувчи. Ўзгарувчи n кўзда тутилган мурожаат даражасига эга, n_pri —private даражасига, n_pro — protected, n_pripro — private protected ва n_pub — public. Қолган синфларда биринчи синф

ўзгарувчиларидан фойдаланилади. Трансляция жараёнида мурожаатни чеклаш натижасида хатоликка олиб келувчи код сатрлари бир сатрли изох кўринишида берилган (//) — хар бири олдида бу модификаторлар комбинацияси учун қаердан мурожаат мумкинлиги кўрсатилган. Иккинчи синф — Derived —Protection синфи остки синфи ва у хам p1 пакетда жойлашган. Шунинг учун у n_pri ўзгарувчидан ташқари хамма ўзгарувчиларга мурожаат қилиши мумкин. Учинчи синф, SamePackage, шу пакетда жойлашган, лекин Protection остки синфи эмас. Шунинг учун у n_pri ўзгарувчидан ташқари, n_pripro ўзгарувчига мурожаат қилолмайди, чунки бу ўзгарувчи мурожаат даражаси — private protected.

```
package p1;
public class Protection {
int n = 1;
private int n_pri = 2;
protected int n_pro = 3;
private protected int n_pripro = 4;
public int n_pub = 5;
public Protection() {
System.out.println("base constructor");
System.out.println("n = " + n);
System.out.println("n_pri = " + n_pri);
System.out.println("n_pro = " + n_pro);
System.out.println("n_pripro = " + n_pripro);
System.out.println("n_pub = " + n_pub);
} }
class Derived extends Protection {
Derived() {
System.out.println("derived constructor");
System.out.println("n = " + n);
// только в классе
// System.out.println("n_pri = " + n_pri);
System.out.println("n_pro = " + n_pro);
System.out.println("n_pripro = " + n_pripro);
System.out.println("n_pub = " + n_pub);
} }
class SamePackage {
SamePackage() {
Protection p = new Protection();
System.out.println("same package constructor");
System.out.println("n = " + p.n);
// фақат синфда
// System.out.println("n_pri = " + p.n_pri);
System.out.println("n_pro = " + p.n_pro);
// фақат синфда ва унинг ворисида
// System.out.println("n_pripro = " + p.n_pripro):
```

```
System.out.println("n_pub = " + p.n_pub);
} }
```

Назорат саволлари:

1. Пакетлар нима учун ишлатилади?
2. Пакет таърифини келтиринг.
3. Нима учун оператор import ишлатилади?
4. Синф элементлари орасида кўриниш тўртта категориясини кўрсатинг.
5. Нечта мурожаат даражаси мавжуд?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 13. МАВЗУ: ЖОЙЛАШТИРИЛГАН СИНФЛАР

Режа: Жойлаштирилган синфлар; Юқори даражадаги жойлаштирилган синфлар ва интерфейслар; Синф - аъзолар.

Жойлаштирилган синфлар

Java тили 1.1 версиясига киритилган энг мухим янгилик жойлаштирилган синфлардир. Жойлаштирилган синфлар бир синфни иккинчи синф аъзоси сифатида таърифлашга имкон берди.

Бир томондан жойлаштирилган синфлар Java тили синтаксисини тартибласа, иккинчи томондан бу синфлар киритилиши алохида холатлар ва қоидалар киритилишига сабаб бўлди. Шунга қарамай эхтиётлик билан ишлатилса жойлаштирилган синфлар ўта фойдали қўшимчадир. Айниқса AWT Java 1.1 да аниқланган ходисаларни қайта ишлаш янги модели билан бирга кўп ишлатилади.

Жойлаштирилган синфлар типлари

Java 1.0 синф ва интерфейсларни фақат юқори даражада пакетлар аъзоси сифатида киритишга имкон беради. Java 1.1 тилида юқори даражадаги синф ва интерфейслар янги типи ва қуйи даражадаги учта янги тип киритилган.

Юқори даражадаги жойлаштирилган синфлар ва интерфейслар

Юқори даражадаги жойлаштирилган синфлар ёки интерфейслар ўзлари тегишли бўлган синф ёки интерфейс статик элементлари сифатида аниқланади. Юқори даражадаги жойлаштирилган синф таърифланганда статик усуллар ёки ўзгарувчиларга ўхшаб static модификаторидан фойдаланилади. Жойлаштирилган интерфейслар кўзда тутилган бўйича хар доим статикдир

(лекин static модификатори ёрдамида таърифлаш мумкин) ва хар доим юқори даражада жойлашган. Жойлаштирилган синфлар ва интерфейслар, оддий синфлар ва интерфейсларга ўхшаш, яъни пакет аъзосидир. Фарқи шундан иборатки, жойлаштирилган синф ёки интерфейс номи шу синф ёки интерфейс жойлашган синф номини ўз таркибига олади. Масалан LinkedList синфи учун Linkable юқори даражадаги жойлаштирилган интерфейсни аниқлаш мумкин. Бу холда бу интерфейсга қуйидагича мурожаат қилиш лозим: LinkedList.Linkable. Юқори даражадаги синфлар ва интерфейслар ўзаро боғлиқ синфларни гурухлаш учун қулайдир.

Кейинги LinkedList.java мисолда юқори даражадаги жойлаштирилган интерфейс таърифи келтирилган. Бу интерфейс эълонида static калит сўзига эътибор беринг. Мисолдан бу интерфейс ўзи жойлашган синф ичида ва ташқи синфларда ишлатилишига эътибор бериш лозим.

public class LinkedList {

```
// Бу юқори даражадаги жойлаштирилган интерфейс
// статик аъзо сифатида аниқланади.
public interface Linkable {
public Linkable getNext();
public void setNext (Linkable node);
}
// Рўйхат сарлавхаси объект Linkable.
Linkable head;
// Усуллар таналари ташлаб юборилган.
public void insert (Linkable node) {...}
public remove (Linkable node) { ... }
}
// Бу синфда боғланган рўйхатда ишлаш кўзда тутилган
// тугун типи аниқланади. Жойлаштирилган,
// интерфейс номи қандай ёзилишига эътибор бериш лозим.
class LinkableInteger implements LinkedList.Linkable
{
// Қуйида тугун учун ўзгарувчилар ва конструктор келтирилган.
int i;
public LinkableInteger (int i) { this.i = i; }
// Қуйида интерфейс реализацияси учун зарур
// ўзгарувчилар ва усуллар келтирилган.
LinkedList.Linkable next;
public LinkedList.Linkable getNext() { return next; }
public void setNext (LinkedList.Linkable node) {
next=node;
} }
```

Агар *LinkedList.java* компиляция қилинса, икки синфлар файли хосил бўлади. Биринчи файл исми кутилганидек *LinkedList.class*. Шу билан бирга *LinkedList$Linkable.class* номли бошқа файл хам яратилади. Символ $ Java компилятори томонидан автоматик жойлаштирилади.

Java виртуал машинасига юқори даражадаги жойлаштирилган ёки ички синфлар ва интерфейслар хақида хеч нарса маълум эмас. Шунинг учун Java компилятори янги типларни стандарт жойлаштирилмаган синфлар файлларига келтириши керак токи Java интерпретатори уларни тушуна олсин. Бунинг учун дастур матни ўзгартирилиб, жойлаштирилган синфлар номлари олдига $ символи қўйилади.

Синф - аъзолар

Синф- аъзолар бирорта синф аъзоси сифатида таърифланади, лекин юқори даражадаги жойлаштирилган синфлардан фарқли, уларнинг таърифида static модификатори ишлатилмайди. Бу шуни билдирадики улар юқори даражадаги синфлар эмас, ички синфлардир. Худди шундай интерфейс-аъзо таърифлаш мумкин эмас, бундай тушунча йўқ (чунки интерфейс концепциясига зиддир). Кўп жихатдан синф- аъзо синф бошқа

аъзоларига – ўзгарувчилар ва усулларга ўхшашдир. Синф –аъзо шуниси қизиқки унинг коди ўзи тегишли синфнинг ихтиёрий ўзгарувчиси ёки усулига хатто private деб эълон қилинган ўзгарувчиси ва усулига мурожаат қилиши мумкин. Шу муносабат билан ўз ичига олувчи синф нусхаси учун Java тилига бир неча янги синтаксис хусусиятлар қўшилган.

Юқори даражадаги жойлаштирилган синфлар каби, синф – аъзолар ўз ичига олган синфлар учун керакли бўлган ёрдамчи синфлар сифатида ишлатилади. Синф – аъзолар юқори даражадаги жойлаштирилган синфлар ўрнига қуйидаги холларда ишлатилади: агар ўз ичига олган синф нусхаси ўзгарувчиларига мурожаат талаб қилинса ёки ёрдамчи синф хар бир нусхаси ўз ичига олган синф нусхаси билан боғлиқ бўлиши керак бўлса. Синф – аъзолардан фойдаланилганда бундай мослик автоматик бажарилади.

Юқорида кўрилган LinkedList синфи мисолига қайтамиз. Бу синфда боғланган рўйхат элементларига java.util.Enumeration интерфейси ёрдамида циклик мурожаат ташкил қилиш керак бўлсин. Бунга эришиш учун бу интерфейсни реализация қилувчи алохида синф таърифлаймиз ва LinkedList синфига Enumeration синфи алохида нусхасмини қайтарувчи усул қўшамиз. Қуйидаги мисолда Java 1.0 стилида бу йўналиш типик реализацияси келтирилган.

```
import java.util.*;
public class LinkedList {
// Юқори даражадаги жойлаштирилган интерфейс. Танаси ташлаб юборилган...
public interface Linkable {... }

// Рўйхат сарлавхаси.
Linkable head;

// Усуллар таналари ташлаб юборилган.
public void addToHead (Linkable node) {...}
public Linkable removeHead() {...}

// Бу усул Enumeration объектини қайтаради
// маълум LinkedList объекти учун.
public Enumeration enumerate() {
return new LinkedListEnumerator (this);
}
}

// Бу синфда Enumeration тип аниқланган, бўлиб
// LinkedList да элементлар рўйхатини олиш учун ишлатилади.
// LinkedListEnuroerator хар бир объект
// конструкторга узатилувчи LinkedList маълум объектига боғлиқ.
class LinkedListEnumerator implements Enumeration {
private LinkedList container;
```

```
private LinkedList.Linkable current;
public LinkedListEnumerator (LinkedList 1) {
container = 1;
current = container.head;
}
public boolean hasMoreElements() { return (current != null); }
public Object nextElement() {
if (current == nul) throw
new NoSuchElementException("LinkedList");
Object value = current;
Current = current.getNext();
Return value;
}
}
```

Шуни кўрсатиш лозимки LinkedListEnumerator синфи конструкторига LinkedList объектини ошкор равишда узатиш лозим. LinkedListEnumerator синфи юқори даражадаги алохида синф сифатида таърифланади, лекин LinkedList синфи аъзоси сифатида таърифлаш қулайроқ бўлар эди. Бу Java 1.1 тилида қуйидаги мисолда кўрсатилганидек синф – аъзо сифатида енгил амалга оширилади.

```
import java.util.*;
public class LinkedList
{
// Юқори даражадаги жойлаштирилган интерфейс. Танаси ташлаб юборилган …
public interface Linkable {...}
//Рўйхат сарлавхаси.
//Бу ўзгарувчини private деб эълон қилиш мумкин бўларди
//жойлаштирилган синфлар билан боғлиқ компилятор хатолари бўлмаганда.
/* private */ Linkable head;
// Усуллар таналари ташлаб юборилган.
public void addToHead(Linkable node) {...}
public Linkable removeHead() {...}
//Бу усул берилган объект учун Enumeration объектини қайтаради
//LinkedList. Эътибор беринг: LinkedList бирор объекти
// конструкторга ошкор узатилмайди.
public Enumeration enumerate() {return new Enumerator();}
//Бу ерда Enumeration интерфейси реализацияси келтирилган бўлиб,
// private модификаторли синф –аъзо сифатида аниқланади.
private class Enumerator implements Enumeration {
Linkable current;
// Эътибор беринг: конструктор ўз ичига олган синф неявно
//head ўзгарувчисига мурожаат қилмоқда.
```

```
public Enumerator (){ current = head; }
public boolean hasMoreElements() { return(current != null);}
public Object nextElement() {
if (current == null) throw
new NoSuchElementException("LinkedList") ;
Object value = current;
current = current.getNext();
return value;
}
}
}
```

Эътибор беринг қандай қилиб бу мисолда Enumerator синфи LinkedList синфига жойлаштирилади. Чиндан хам ёрдамчи синфни ўз ичига олувчи синф фойдаланадиган дастур қисмида таърифлаш қулайдир. Албатта мисолни компиляцияга бериб Enumerator синф-аъзо LinkedList$Enumerator.class файлга компиляция қилинишини кўриш мумкин. Дастур бирламчи матнида бир синф иккинчисига жойлаштирилган бўлса хам, компиляция қилинган дастур байт-кодида бундай эмас.

Эътибор беринг Enumerator синф – аъзо конструкторига ўз ичига олувчи синф LinkedList бирорта нусхаси узатилмайди. Синф – аъзо ўз ичига олган синф аъзоларига кўзда тутилган бўйича мурожаат қилиши мумкин. Ошкор иловаларга хожат йўқ. Яна шунга эътибор беринки Enumerator синфида ўз ичига олган синф *head* ўзгарувчисидан фойдаланилади, бу ўзгарувчи private сифатида таъриф этилганига қарамай. Умумий холда синф – аъзолар, хамда локал ва аноним синфлар, ўз ичига олган синф хусусий ўзгарувчилари ва усулларига (хатто синфларига!) мурожаат қилиши мумкин. Худди шундай ўз ичига олган синф жойлаштирилган синф ўзгарувчилари усуллари ва хатто синфларидан фойдаланиши мумкин. Худди шундай бир синфга жойлаштирилган икки синф бир бирининг private- аъзоларига мурожаат қилиши мумкин.

Назорат саволлари:

1. Жойлаштирилган синфлар деб қандай синфларга айтилади?
2. Жойлаштирилган синфлар типларини келтиринг.
3. Юқори даражадаги интерфейслар нима учун ишлатилади?
4. Синф – аъзолар хусусиятларини кўрсатинг.
5. Нима учун синф – аъзолар каби интерфейслар яратиб бўлмайди?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 14. МАВЗУ: ФАЙЛЛАР БИЛАН ИШЛАШ

Режа: Киритиш/Чиқариш; InputStream; OutputStream; FilelnputStream; FileOutputStream; ByteArraylnputStream;ByteArrayOutputStream; StringBufferlnputStream.

Киритиш/Чиқариш

Киритиш манбаи умумлашган тушунчаси бир неча маълумот олиш усулларига тегишли: дискли файлдан ўқиш, клавиатурадан киритиш,ёки тармоқ орқали ахборот узатиш қабул қилиш. Худди шундай чиқариш умумлашган манбаи сифатида дискли файллар, тармоқ орқали боғланиш ва хоказолар тушунилиши мумкин. Бу абстракциялар киритиш чиқариш (I/O) билан ишлаш учун қулай имконият яратади, чунки дастурдан, клавиатура ва тармоқни фарқлашни талаб этмайди. Java тилида бу абстракциялар оқим (stream) деб аталади ва java.io пакети бир неча синфларида реализация қилинган. Киритиш InputStream синфида инкапсуляция этилган, чиқариш — OutputStream синфида. В Java тилида бу абстракт синфларни бир неча специализациялари мавжуд бўлиб, дискли файллар, тармоқ билан боғланиш ва хотирадаги буферлар билан ишлаш хусусиятларини акс эттиради.

InputStream

InputStream —Java тилида кириш оқимлари моделини берувчи абстракт синфдир. Бу синф хамма усуллари хато юз берганда IOException истисно яратади. Қуйида InputStream синфи усуллари келтирилган.

- read() кириш оқимидаги навбатдаги символни бутун сон кўринишида қайтаради.
- read(byte b[])максимум b.length байтни кириш оқимидан b массивга ўиқишга харакат қилади. Оқимдан чиндан ўқилган байтлар сонини қайтаради.
- read(byte b[], int off, int len) максимум len байтни, расположив b массивга, off элементдан бошлаб ўқишга харакат қилади. Хақиқатан ўқилган байтлар сонини қайтаради.
- skip(long n) кириш оқимида n байтни ўтказишга харакат қилади. Хақиқатан ўтказилган байтлар сонини қайтаради.
- available() ўқиш мумкин бўлган байтлар сонини қайтаради.
- close() кириш манбаини беркитади. Бу оқимдан кейинги ўқишга харакат қилиш IOException истисносини яратади.
- mark(int readlimit) кириш оқими жорий позициясига белги қўяди. Бу белгидан оқимдан readlimit байт ўқилмагунча фойдаланиш мумкин бўлади.
- reset() оқим кўрсаткичини олдин қўйилган белгига қайтаради.
- markSupported() агар оқим mark/reset амалларини қўлласа true қайтаради.

OutputStream

InputStream каби OutputStream — абстракт синф. У Java чиқиш оқимлари моделини беради. Бу синф хамма усуллари void типига эга ва хатолик юз берганда IOException истисно яратади. Қуйида шу синф усуллари рўйхати берилган:

- write(int b) чиқиш оқимига бир байт ёзади. Бу усул аргументи int типига тегишли, шунинг учун write усулини ифода узатиб чақириш мумкин ва ифодани byte типига келтириш шарт эмас.
- write(byte b[]) чиқиш оқимига кўрсатилган байтлар массиви хаммасини ёзади.
- write(byte b[], int off, int len) оқимга массив len байтини b[off] элементдан бошлаб ёзади.
- flush() чиқариш амалини тугатиб, ихтиёрий чиқиш буферини тозалайди.
- close() чиқиш оқимини беркитади. Бу оқимга ёзишга кейинги хар қандай уриниш IOException яратади.

Файлли оқимлар

FileInputStream

Класс FileInputStream синфи маълумотларни файллардан киритиш учун ишлатилади. Қуйида келтирилган мисолда бу синфнинг битта дискли файлдан фойдаланувчи икки объекти яратилади.

```
InputStream f0 = new FileInputStream("/autoexec.bat");
File f = new File("/autoexec.bat"):
InputStream f1 = new FileInputStream(f);
```

FileInputStream синфи объекти яратилганда, у ўқиш учун очилади. FileInputStream синфи InputStream абстракт синф олти усулини қўшимча юклайди. Бу синф объектига mark ва reset усулларини қўллашга уриниш IOException истисно яратилишига олиб келади. Қуйида келтирилган мисолда қандай қилиб, алохида байт, байтлар массиви ва байтлар массиви қисмини ўқиш кўрсатилган. Бу мисолда яна available усули ёрдамида қанча ўқилмаган байтлар қолганлиги ва skip усули ёрдамида ўқиш керак бўлмаган байтларни ўтказиб юбориш кўрсатилган.

```
import java.io.*;
import java.util.*;
class FileInputStreamS {
public static void main(String args[]) throws Exception {
int size;
InputStream f1 = new FileInputStream("/wwwroot/default.htm");
size = f1.available();
System.out.println("Total Available Bytes: " + size);
System.out.println("First 1/4 of the file: read()");
for (int i=0; i < size/4; i++) {
```

```
System.out.print((char) f1.read());
}
System.out.println("Total Still Available: " + f1.available());
System.out.println("Reading the next 1/8: read(b[])");
byte b[] = new byte[size/8];
if (f1.read(b) != b.length) {
System.err.println("Something bad happened");
}
String tmpstr = new String(b, 0, 0, b.length);
System.out.println(tmpstr);
System.out.println("Still Available: " + f1.available());
System.out.println("Skipping another 1/4: skip()");
f1.skip(size/4);
System.out.println( "Still Available: " + f1.available());
System.out.println("Reading 1/16 into the end of array");
if (f1.read(b, b.length-size/16, size/16) != size/16) {
System.err.println("Something bad happened");
}
System.out.println("Still Available: " + f1.available());
f1.close();
}
}
```

FileOutputStream

FileOutputStream синфи FileInputStream синфи каби икки конструкторга эга. Лекин бу синф объектларини яратиш учун файл мавжуд бўлиши шарт эмас. FileOutputStream синфи объекти яратилганда чиқариш учун файл очишдан олдин яратилади.

Навбатдаги мисолда клавиатурадан киритилган символлар System.in оқимидан 12-байтли буфер тўлмагунча биттадан ўқилади. Шундан сўнг учта файл яратилади. Биринчи file1.txt файлга, буфердан символлар ёзилади, лекин хаммаси эмас битта ташлаб, яъни олдин нолинчи кейин иккинчи ва хоказо. Иккинчи file2.txt файлга буферга тушган хамма маълумот ёзилади. Ва нихоят учинчи файлга буфер ўртасида жойлашган ярми ёзилади, биринчи ва охирги чораклар ташлаб юборилади.

```
import java.io.*;
class FileOutputStreamS {
public static byte getlnput()[] throws Exception {
byte buffer[] = new byte[12];
for (int i=0; i<12; i++) {
buffer[i] = (byte) System.in.read();
}
return buffer;
}
```

```
public static void main(String args[]) throws Exception {
byte buf[] = getlnput();
OutputStream f0 = new FileOutputStream("file1.txt");
OutputStream f1 = new FileOutputStream("file2.txt");
OutputStream f2 = new FileOutputStream("file3.txt");
for (int i=0; i < 12; i += 2) {
f0.write(buf[i]);
}
f0.close();
f1.write(buf);
f1.close();
f2.write(buf, 12/4, 12/2);
f2.close();
} }
```

Хозирги пайтда FileOutputStream оқимни файл охирига қўшиш учун очиш мумкин эмас. Агар файл FileOutputStream конструктори ёрдамида очилса мавжуд маълумот йўқолади. Бу Java реализацияси камчилиги.

ByteArraylnputStream

ByteArrayInputStream - byte типидаги массив ишлатилувчи кириш оқими реализациясидир. Бу синфда икки конструктор бўлиб, хар бири биринчи параметр сифатида байтли массив талаб қилади. Қуйида келтирилган мисолда шу типдаги икки объект яратилади. Бу объектлар лотин алфавити символлари билан инициализация қилинади.

```
String tmp = "abcdefghijklmnopqrstuvwxyz";
byte b[] = new byte [tmp.length()];
tmp. getBytes(0, tmp.length(), b, 0);
ByteArrayInputStream input1 = new ByteArrayInputStream(b);
ByteArrayInputStream input2 = new ByteArreyInputStream(b,0,3);
```

ByteArrayOutputStream

ByteArrayOutputStream синфида — икки конструктор. Биринчи конструктор хажми 32 байтга тенг буфер яратади. Иккинчи конструктордан фойдаланилганда конструктор параметрида берилган хажмли буфер яратадиа (қуйида келтирилган мисолда — 1024 байта):

```
OutputStream out0 = new ByteArrayOutputStream();
OutputStream out1 = new ByteArrayOutputStream(1024);
```

Навбатдаги мисолда ByteArrayOutputStream объекти клавиатурадан киритилган символлар билан тўлдирилади, шундан сўнг турли манипуляциялар бажарилади.

```
import java.io.*;
import java.util.*;
class ByteArrayOutputStreamS {
public static void main(String args[]) throws Exception {
int i;
```

```
ByteArrayOutputStream f0 = new ByteArrayOutputStream(12);
System.out.println("Enter 10 characters and a return");
while (f0.size() != 10) {
f0.write( System.in.read());
}
System.out.println("Buffer as a string");
System.out.println(f0.toString());
System.out.println ("Into array");
byte b[] = f0.toByteArray();
for (i=0; i < b.length; i++) {
System.out.print((char) b[i]);
}
System.out.println();
System.out. println("To an OutputStream()");
OutputStream f2 = new File0utput8tream("test.txt");
f0.writeTo(f2);
System.out.println("Doing a reset");
f0. reset();
System.out.println("Enter 10 characters and a return");
while (f0.size() != 10) {
f0.write (System.in.read());
}
System.out.println("Done.");
} }
```

Бу мисолдаги файл test.txt, ичида:

```
C:\> type test.txt
0123456789
```

StringBufferlnputStream

StringBufferInputStream синфи ByteArrayInputStream синфи билан деярли бир хил. Фарқи шундаки бу синф объекти ички буфери байтли массив эмас String синфи нусхасидир. Бундан ташқари Java тилида унга мос StringBufferedOutputStream синфи йўқ. Бу синфда ягона конструктор:

StringBufferInputStream(String s)

Назорат саволлари:

1. Оқим деб нимага айтилади?
2. Кириш учун абстракт синфларни кўрсатинг.
3. Чиқиш учун абстракт синфларни кўрсатинг.
4. Файлдан ўқиш учун синфларни кўрсатинг.
5. Файлга ёзиш учун синфларни кўрсатинг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.

3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 15. МАВЗУ: ФАЙЛЛАР БИЛАН ИШЛАШ ХУСУИЯТЛАРИ

Режа: File; Каталоглар; Фильтрловчи оқимлар; Буферизацияланган оқимлар; PrintStream синфи.

File

File —**java.io** да тўғридан тўғри дискли файллар билан ишловчи ягона объектдир. Апплетларда файллардан фойдаланишга чегаралар қўйилган бўлса хам, файллар доимий сақлаш ва биргаликда фойдаланиш учун ягона ресурсдир. Каталог в Java тилида оддий файл сифатида қаралади, фақат қўшимча хоссали- файллар номлари рўйхатига эга бўлиб, бу рўйхатни list усули ёрдамида қараб чиқиш мумкин.

ИЗОХ

Java UNIX ва DOS тизимларида ишлатиладиган йўлда каталоглар номларини ажратувчи белгини тўғри қайта ишлайди. Агар UNIX стили — символ '/' ишлатилса, Windows системасида ишланганда Java автоматик равишда уларни '\' белгига алмаштиради. Унутманг агар сиз DOS да қабул қилинган ажратгичларга яъни '\' белгисига ўрганган бўлсангиз, уларни йўл сатрига улаш учун иккилаш керак, масалан \\java\\COPYRIGHT сатрдаги каби.

Объект стандарт хоссаларини аниқлаш учун File синфида турли усуллар мавжуд. Лекин File синфи носимметрик. Объект хоссаларини аниқлаш учун хоссалар кўп, лекин бу хоссаларни ўзгартириш усуллари мавжуд эмас. Қуйидаги мисолда файл турли характеристикаларини олиш учун турли усуллардан фойдаланилади:

```
import java.io.File;
class FileTest {
static void p(String s) {
System.out.println(s);
}
public static void main(String args[]) {
File f1 = new File("/java/COPYRIGHT");
p("File Name:" + f1 .getName());
p("Path:" + f1.getPath());
p("Abs Path:" + f1.getAbsolutePath());
p("Parent:" + f1.getParent());
p(f1.exists() ? "exists" : "does not exist");
p(f1.canWrite() ? "is writeable" : "is not writeable");
p(f1.canRead() ? "is readable" : "is not readable");
p("is " + (f1.isDirectory() ? " " : "not") + " a directory");
p(f1.isFile() ? "is normal file" : "might be a named pipe");
p(f1.isAbsolute() ? "is absolute" : "is not absolute");
```

```
p("File last modified:" + f1. lastModified());
p("File size:" + f1.length() + " Bytes");
} }
```

Дастур бажарилганда қуйидаги маълумот олиш мумкин:

File Name:COPYRIGHT (файл номи)
Path:/java/COPYRIGHT (йўл)
Abs Path:/Java/COPYRIGHT (илдиз каталогдан йўл)
Parent:/java (ота каталог)
exists (файл мавжуд)
is writeable (ёзишга рухсат берилган)
is readable (ўқишга рухсат берилган)
is not a directory (каталог эмас)
is normal file (оддий файл)
is absolute
File last modified:812465204000 (файл охирги модификацияси)
File size:695 Bytes (файл хажми)

Фақат файлларга қўллаш мумкин бўлган усуллар мавжуд (уларни каталогларга қўллаш мумкин эмас). Файл номини ўзгартириш учун renameTo(File dest) усулидан фойдаланилади (файлни бошқа каталокка кўчириш мумкин эмас). Дискдаги файлни delete усули ўчиради. Бу усул фақат файлни ўчиради, хатто бўш каталогни бу усул билан ўчириб бўлмайди.

Каталоглар

Каталоглар File синфи объектлари бўлиб, уларда бошқа файллар ва каталоглар рўйхати жойлашган. Агар File каталогга илова қилса, унинг isDirectory усули true қиймат қайтаради. Бу холда list усулини чақириб объектда жойлашган файллар ва каталоглар номларини чиқариш мумкин. Навбатдаги мисолда list усули ёрдамида каталогни кўриб чиқиш кўрсатилган.

```
import java.io.File;
class DirList {
public static void main(String args[]) {
String dirname = "/java"; // каталог номи
File f1 = new File(dirname);
if (f1.isDirectory()) { // f1 каталогми
System.out.println("Directory of ' + dirname);
String s[] = f1.list();
for ( int i=0; i < s.length; i++) {
File f = new File(dirname + "/" + s[i]);
if (f.isDirectory()) { // f каталогми System.out.println(s[i] + " is a directory"):
} else {
System.out.println(s[i] + " is a file");
} } } else {
System.out.println(dirname + " is not a directory");
} }
```

}

Ишлаш жараёнида бу дастур /java каталогидаги маълумотларни қуйидагича чиқариши мумкин:

C:\> java DirList
Directory of /java
bin is a directory
COPYRIGHT is a file
README is a file

FilenameFilter

Кўпинча list усули қайтараётган номлар сонини чеклаб, маълум шаблонга мос номларни чиқариш талаб этилади. Бунинг учунДля этого в пакет java.io пакетига FilenameFilter интерфейси киритилган. Объект, бу интерфейсни реализация қилиши учун, хар янги файл номи билан чақирилувчи accept() усулини таърифлаш лозим. Бу accept усули рўйхатга киритилиши лозим бўлган номлар учун true ва чиқариш лозим бўлган номлар учун false қайтариши лозим.

У класса File синфда каталоглар билан ишлашга мўлжалланган икки усул мавжуд. Остки каталог яратиш учун mkdir усули ишлатилади. Хали йўли кўрсатилмаган каталог яратиш учун mkdirs усулидан фойдаланиш лозим. У фақат кўрсатилган каталог эмас, хамма мавжуд бўлмаган аждод каталогларни яратади.

Фильтрловчи оқимлар

Чиқариш тизими параллел жараёнларга эга мухитда ишлаганда агар синхронизация мавжуд бўлмаса кутилмаган натижалар келиб чиқиши мумкин. Бунинг сабаби бир неча қуйи жараёнларнинг битта оқимга мурожаат қилишидир. Бу синфда мавжуд хамма конструкторлар ва усуллар InputStream ва OutputStream синфларидаги конструкторлар ва усуллар билан бир хил. Фильтрловчи оқимларнинг фарқи шундаки уларнинг усуллари синхронизация қилингандир.

Буферизацияланган оқимлар

Буферизация қилинган оқимлар фильтрланувчи оқимлар кенгайтмаси бўлиб, уларда киритиш- чиқариш оқимларига хотирада буфер уланади. Бу буфер икки функция бажаради:

- У java бажарувчи мухитига бир неча байтни киритиш- чиқаришга имкон беради. Бунинг натижасида мухит унумдорлиги ошади.
- Оқимни буфери мавжуд бўлгани учун маълумотларни ўтказиш, белгиларни ўрнатиш ва буферни тозалаш тамалларини бажариш мумкин бўлади.

BufferedInputStream

Киритиш- чиқаришни буферизациялаш — бундай амалларни оптималлаш кенг тарқалган усулидир. BufferedlnputStream синфи Java тилида InputStream ихтиёрий объектини буферизация қилинган оқим билан “ўраб

олишга” имкон беради ва шу билан унумдорликни оширади. Бу синфда икки конструктор мавжуд бўлиб, биринчиси

BufferedInputStream(InputStream in)

хажми 32 байтли буфердан фойдаланиб, буферизацияланган оқим яратади. Иккинчисида

BufferedInputStream(InputStream in, int size)

оқим буфери хажми конструктор иккинчи параметри орқали берилади. Умумий холда буфер оптимал хажми операцион тизимга, оператив хотира хажмига ва компьютер конфигурациясига боғлиқ.

BufferedOutputStream

BufferedOutputStream объектига чиқариш ихтиёрий OutputStream объектига чиқариш билан деярли бир хил, шу фарқ биланки янги остки синф қўшимча flush усулига эга. Бу усул буферни мажбуран тозалаш ва ундаги маълумотни ташқи қурилмага физик чиқариш учун ишлатилади. Бу синф конструктори биринчи шакли:

BufferedOutputStream(OutputStream out)

хажми 32 байтли буферга эга оқим яратади. Иккинчи шакли:

BufferedOutputStream(OutputStream out, int size)

Керакли буфер хажмини киритишга имкон беради.

PushbacklnputStream

Одно из необычных применений буферизации — реализация операции pushback (вернуть назад). Pushback символни ўқигандан сўнг оқимга қайтариш учун InputStream га қўлланади. Лекин PushbacklnputStream имкониятлари чекланган – оқимга биттадан ортиқ символни қайтаришга уриниш IOException истисно яратилишига олиб келади. Бу синфда — ягона конструктор

PushbackInputStream(InputStream in)

InputStream усулларидан ташқари, PushbacklnputStream аргументида берилган ch символни киритиш оқимига қайтарувчи unread(int ch) усулига эга.

SequencelnputStream

SequencelnputStream синфи бир неча кирувчи оқимларни битта оқимга қўшиш янги имкониятига эга. SequenceInputStream синфи конструкторида параметр сифатида InputStream икки объекти, ёки InputStream объектлари коллекциясини ўз ичига олган сановчи ишлатилади:

SequenceInputStream(Enumeration e)
SequenceInputStream(InputStream s0, InputStream s1)

Иш жараёнида синф келиб тушган сўровларни бажариб, биринчи оқимдан то тугамагунча маълумотларни ўқийди, сўнгра иккинчисига ўтади ва хоказо.

PrintStream

PrintStream синфи System пакети чиқаришда файлли дескрипторлари орқали фойдаланиб келинган хамма форматлаш утилиталаридан фойдаланишга имкон беради. Шу пайтгача “System.out.println”, ёзилганда чиқарилаётган маълумотларни форматловчи синфларга эътибор берилмаётган эди. У класса PrintStream синфида икки конструктор: **PrintStream(OutputStream out)** и **PrintStream(OutputStream out, boolean autoflush)**. Иккинчисининг autoflush параметри Java бажарувчи мухити чиқариш оқим устида буферни тозалаш амали автоматик бажариш керак ёки керакмаслигини кўрсатади.

В Java тилида - PrintStream объектларида print ва println усуллари мавжуд бўлиб, улар ихтиёрий объектлар хатто Object объектлари билан ишлай олади. Агар бу усуллар аргуметлари сифатида примитив типлардан бири ишлатилмаса, Object синфининг toString усулини чақиради ва шундан сўнг маълумотни чақиради.

Назорат саволлари:

1. Дискли файллар билан ишлаш учун қайси объектдан фойдаланилади?
2. Каталог бу нима?
3. Каталогни кўриб чиқиш усулини кўрсатинг.
4. Оқимни буферизациялаш қандай амалга оширилади?
5. Буферизацияланган оқим деб нимага айтилади?

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 16. МАВЗУ: ИСТИСНОЛАРНИ БОШКАРИШ

Режа: Истисно холатлар; Асослар; Истиснолар типлари; Ушлаб олинмаган истиснолар; try ва catch; Бир неча catch бўлимлари; Жойлаштирилган try операторлари**.**

Истисно холатлар

Java тилида истисно — дастур коди бирор қисмида хосил бўлган истисно холатни тасвирловчи объектдир. Истисно холат юзага келганда Exception синфи объекти яратилади. Бу объект шу типдаги истисно холатни қайта ишловчи усулга узатилади. Истиснолар ғайри оддий холатлар хақида маълумот бериш учун «қўлда» яратилиши мумкин.

Асослар

Java тилида истисинолар қайта ишлаш механизми бешта калит сўзлардан фойдаланади: — **try**, **catch**, **throw**, **throws** и **finally**. Бу механизм ишлаш схемаси қуйидагича. Сиз код блокини бажаришга (try) уринасиз, ва хатолик юзага келса, тизим истисно (throw) генерация қилади. Бу истисно типига қараб ушлаб олиш (catch) ёки кўзда тутилган қайта ишловчига (finally) узатишингиз мумкин.

Қуйида истиснолар қайта ишлаш блоки умумий кўриниши келтирилган.

```
try {
// код блоги }
catch (ТипИстисно1 e) {
// ТипИстисно1  туридаги истиснони қайта ишловчи}
catch (ТипИсключения2 e) {
// ТипИстисно2  туридаги истиснони қайта ишловчи
throw(e) // истиснони қайта яратиш }
finally {
}
```

Истиснолар типлари

Истиснолар иерархияси юқорисида Throwable синфи жойлашган. Хар бир типдаги истисно Throwable синф остки синфидир. Throwable синфи икки вориси истиснолар остки синфлар иерархиясини икки шохга ажратади. Биринчиси —Exception синфи— фойдаланувчи дастур коди томонидан ушлаб олиниши лозим бўлган истисноларни таърифлаш учун ишлатилади. Иккинчичи —Error - синфи фойдаланувчи дастур коди томонидан ушлаб олинмаслиги лозим бўлган истисноларни таърифлаш учун ишлатилади.

Ушлаб олинмаган истиснолар

Объект-истиснолар маълум истисно холатлар юз келиши натижасида Java бажарувчи мухити томонидан автоматик яратилади. Масалан навбатдаги дастур бажарилиши натижасида нольга бўлиш юзага келадиган ифодани ўз ичига олади.

```
class Exc0 {
public static void main(string args[]) {
int d = 0;
int a = 42 / d;
} }
```

Мисол бажарилиши натижаси.

```
C:\> java Exc0
java.lang.ArithmeticException: / by zero
at Exc0.main(Exc0.java:4)
```

Эътибор беринг яратилган истиснолар тури Exception хам Throwable хам эмас. Бу Exception вориси, яни: дастур бажарилиш натижасида қандай хато юз бергани хақида маълумот берувчи ArithmeticException. Қуйидаги шу синф версиясида худди шу истисно яратилади, лекин main усули кодида эмас.

```
class Exc1 {
static void subroutine() {
int d = 0;
int a = 10 / d;
}
public static void main(String args[]) {
Exc1.subroutine();
} }
```

Бу дастурда Java бажариш мухити чақириқлар стекидаги хамма маълумотни чиқаради.

```
C:\> java Exc1
java.lang.ArithmeticException: / by zero
at Exc1.subroutine(Exc1.java:4)
at Exc1.main(Exc1.java:7)
```

try ва catch

Истиснолардан химоя қилиш лозим бўлган дастур кодини бериш учун try калит сўзи ишлатилади. Дархол try-блокдан сўнг қайта ишлаш лозим бўлган истисно типини берувчи catch блок жойлашади.

```
class Exc2 {
public static void main(String args[]) {
try {
int d = 0;
int a = 42 / d;
}
catch (ArithmeticException e) {
System.out.println("division by zero");
}
} }
```

Яхши лойихалаштирилган кўпгина catch-бўлимлар мақсади яратилган истиснони қайта ишлаш ва дастур ўзгарувчиларини маълум холатга келтириш, токи дастурни хеч қандай хатолик юз бермагандек давом эттириш

мумкин бўлсин (мисолда қуйидаги огохлантириш чиқарилади – division by zero).

Бир неча catch бўлимлари

Баъзи холларда битта дастур коди турли типдаги истиснолар яратиши мумкин. Бундай холатларни қайта ишлаш учун, Java try-блок учун ихтиёрий catch- блоклар киритишга имкон беради. Энг махсус истиснолар синфлари биринчи келиши керак, чунки бирор ворис синф ишлатилмайди агар суперсинфдан кейин келса. Қуйидаги дастурда икки типдаги истисно ушланади, бу икки махсус қайта ишловчилардан кейин Throwable синфи хамма ворисларини ушлаб олувчи catch бўлими келади.

```
class MultiCatch {
public static void main(String args[]) {
try {
int a = args.length;
System.out.println("a = " + a);
int b = 42 / a;
int c[] = { 1 };
c[42] = 99;
}
catch (ArithmeticException e) {
System.out.println("div by 0: " + e);
}
catch(ArrayIndexOutOfBoundsException e) {
System.out.println("array index oob: " + e);
}
} }
```

Бу мисол параметрсиз ишга туширилса нольга бўлиш истисносини келтириб чиқаради. Агар команда қаторида бир ёки бир неча параметр бериб a қиймати нольдан катта ўрнатилса, бўлиш оператори бажарилади лекин кейинги операторда индекс массив чегарасидан чиқиш ArrayIndexOutOf Bounds истисноси яратилади. Қуйида иккала усулда ишга туширилган дастур натижалари келтирилган.

```
C:\> java MultiCatch
a = 0
div by 0: java.lang.ArithmeticException: / by zero
C:\> java MultiCatch 1
a = 1
array index oob: java.lang.ArrayIndexOutOfBoundsException: 42
```

Жойлаштирилган try операторлари

Худди ўзгарувчилар кўриниш сохалари каби try операторларини бир бирига жойлаштириш мумкин. Агар қуйи даражадаги try операторида яратилган истиснога мос catch бўлими мавжуд бўлмаса, ташқи try оператори

catch бўлимлари текширилади. Масалан икки try оператори усулни чақириш ёрдамида ўзаро жойлаштирилган мисол.

```
class MultiNest {
static void procedure() {
try {
int c[] = { 1 };
c[42] = 99;
}
catch(ArrayIndexOutOfBoundsException e) {
System.out.println("array index oob: " + e);
} }
public static void main(String args[]) {
try {
int a = args.length();
System.out.println("a = " + a);
int b = 42 / a;
procedure();
}
catch (ArithmeticException e) {
System.out.println("div by 0: " + e);
}
} }
```

Назорат саволлари:

1. Истисно таърифини келтиринг.
2. Истисноларни қайта ишловчи синфлар иерархиясини кўрсатинг.
3. Истиснони қайта ишлаш синтаксисини келтиринг.
4. Истиснолар билан қандай операторлар боғлиқ?
5. Истиснони генерация қилувчи функция синтаксисини келтиринг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 17. МАВЗУ: ИСТИСНОЛАР БИЛАН ИШЛАШ ХУСУСИЯТЛАРИ

Режа: throw; throws; finally; Exception ворислари.

throw

Оператор throw истисноларни «қўлда» генерация қилиш учун ишлатилади. Буни амалга ошириш учун Throwable остки синф объектига эга бўлиш керакдир. Бу объектни catch оператори параметри сифатида қабул қилиш, ёки new оператори ёрдамида яратиш мумкин. Қуйида throw оператори умумий кўриниши келтирилган.

throw ОбъектТипаThrowable;

Бу оператор бажарилганда код нормал бажарилиши тўхтатилади ва ундан кейинги оператор бажарилмайди. Энг яқин try мос catch қайта ишловчи мавжудлигигна текширилади. Агар у мавжуд бўлса бошқариш унга узатилади. Агар мавжуд бўлмаса кейинги ички жойлашган try блок текширилади,токи мос catch бўлим топилмагунча ёки, Java тизими истиснолар қайта ишловчиси дастурни тўхтатмагунча. Қуйидаги мисолда аввал объект – истисно яратилади, сўнгра оператор throw истисно холат яратади, шундан сўнг қайта яратилади — бу гал биринчи ушлаб олган catch бўлими коди томонидан.

```
class ThrowDemo {
static void demoproc() {
try {
throw new NullPointerException("demo");
}
catch (NullPointerException e) {
System.out.println("caught inside demoproc");
throw e;
} }
public static void main(String args[]) {
try {
demoproc();
}
catch(NulPointerException e) {
System.out.println("recaught: " + e);
}
} }
```

Бу мисолда истиснони қайта ишлаш икки босқичда амалга оширилади. Бош main усули истисно учун контекст яратади ва demoproc усулини чақиради. Чақирилган demoproc усули хам истиснони қайта ишлаш учун контекст яратади NullPointerException синфи янги объектини яратади ва throw оператори ёрдамида бу истиснони яратади. Истисно demoproc усули ичидаги кейинги сатрда ушлаб олинади, объект-итисно қайта ишловчи кодга

параметр е орқали узатилади. Истиснони қайта ишлаш коди истисно хақида маълумот чиқаради, сўнгра истиснони қайтадан throw оператори ёрдамида яратади, натижада у main усулидаги истиснони қайта ишловчига узатилади. Қуйида бу мисол бажарилиши келтирилган.

C:\> java ThrowDemo
caught inside demoproc
recaught: java.lang.NullPointerException: demo

throws

Агар усул ўзи қайта ишламайдиган иситиснолар яратиши мумкин бўлса, бу усулга мурожаат қилувчи усуллар ўзини химоя қилиши учун эълон қилиш лозим. Усул яратиши мумкин бўлган истиснолар рўйхатини бериш учун **throws** калит сўзи ишлатилади. Агар усул ошкор равишда мос синфга тегишли истисно яратса (яъни throw оператори ёрдасмида) истисно типи усул эълонидаги throws операторида кўрсатилиши керак. Шуни хисобга олган холда усул таърифи синтаксиси қуйидагича кенгайтирилиши мумкин:

тип имя_метода(аргументлар рўйхати) throws истиснолар_рўйхати {}

Қуйидаги дастурда procedure усули истисно яратади, лекин уни ушлаб олиш кодига эга эмас хамда, усул таърифида эълон қилмайди. Бу дастур коди трансляция қилинмайди.

```
class ThrowsDemo1 {
static void procedure() {
System.out.println("inside procedure");
throw new IllegalAccessException("demo");
}
public static void main(String args[]) {
procedure();
} }
```

Бу мисолни трансляция қилиш учун трансляторга, procedure усули IllegalAccessException типидаги истисно яратиши мумкинлиги хақида маълумот бериш керак ва main усули шу истиснони қайта ишлаш учун добавить код қўшиш керак:

```
class ThrowsDemo {
static void procedure() throws IllegalAccessException {
System.out.println(" inside procedure");
throw new IllegalAccessException("demo");
}
public static void main(String args[]) {
try {
procedure();
}
catch (IllegalAccessException e) {
System.out.println("caught " + e);
```

```
}
} }
```

Қуйида шу дастур бажарилиш натижаси келтирилган.

```
C:\> java ThrowsDemo
inside procedure
caught java.lang.IllegalAccessException: demo
```

finally

Баъзида код маълум қисми қапндай истиснолар яратилгани ва ушланганига қарамасдан бажарилиши талаб этилади. Бундай код қисмларини яратиш учун finally калит сўзи ишлатилади. Хатто усулда яратилган истиснога мос catch блоги мавжуд бўлмаса хам , блок finally бошқариш try блогидан кейинги операторларга ўтгунча албатта бажарилади. Хар бир try блоги учун мос жуда бўлмаса битта catch бўлими ёки или блок finally блоги бўлиши шарт. Блок finally усул бажарилиши бошланишида вақтинчалик фойдаланиш учун олинга файлларни ва бошқа ихтиёрий ресурсларни бўшатиш учун қулайдир. Қуйидаги мисолда икки усулга эга синф берилган бўлиб, бу усуллар бажарилиши турли сабабларга кўра тўхтатилади, лекин иккаласида хам чиқишдан олдин finally бўлими коди бажарилади.

```
class FinallyDemo {
static void procA() {
try {
System.out.println("inside procA");
throw new RuntimeException("demo");
}
finally {
System.out.println("procA's finally");
} }
static void procB() {
try {
System.out.println("inside procB");
return;
}
finally {
System.out.println("procB's finally");
} }
public static void main(String args[]) {
try {
procA();
}
catch (Exception e) {}
procB();
} }
```

Бу мисолда истисно яратилгани учун procA усулида муддатдан олдин try блогидан чиқилади, лекин йўл йўлакай «ташқарида» finally бўлими

бажарилади. Бошқа усул procB try-блокдаги return операторини бажариб ишни тўхтатади, лекин усулдан чиқишдан олдин finally блоги дастур коди бажарилади. Қуйида шу дастур бажарилиши натижаси келтирилган.

C:\> java FinallyDemo
inside procA
procA's finally
inside procB
procB's finally

Exception ворислари

Фақат Throwable синфи ворислари яратилиши ёки ушлаб олиниши мумкин. Содда типлар — int, char ва хоказо, хамда Throwable вориси бўлмаган синфлар масалан, например, String ва Object, истисно сифатида ишлатилиши мумкин эмас. Истиснолардан фойдаланиш энг умумий йўли Exception синфи ворисларини яратиш. Қуйидаги дастурда Exception синфи вориси киритилган.

```
class MyException extends Exception {
private int detail;
MyException(int a) {
detail = a:
}
public String toString() {
return "MyException[" + detail + "]";
}
}
class ExceptionDemo {
static void compute(int a) throws MyException {
System.out.println("called computer + a + ").");
if (a > 10)
throw new MyException(a);
System.out.println("normal exit.");
}
public static void main(String args[]) {
try {
compute(1);
compute(20);
}
catch (MyException e) {
System.out.println("caught" + e);
}
} }
```

Бу мисол анча мураккабдир. Унда MyException синфи класса Exception синфи вориси сифатида таърифланган. Бу ворис синфда объект ўзгарувчисига бутун қиймат ёзувчи махсус конструктор ва объект – истиснода сақланувчи қийматни чиқарувчи toString усули мавжуд. ExceptionDemo синфида compute усули таърифланган бўлиб, бу усул

MyExcepton типидаги истисно яратади агар усул параметри қиймати 10 дан катта бўлса. Химояланган блокдаги main усули аввал compute усулини мумкин бўлган қиймат билан чақиради, сўнгра мумкин бўлмаган (10 дан катта) қиймат билан. Қуйида дастур бажарилиши натижаси келтирилган.

C:\> java ExceptionDemo
called compute(1).
normal exit.
called compute(20).
caught MyException[20]

Назорат саволлари:

1. Истисноларни генерация қилиш синтаксисини келтиринг.
2. Истисно генерация қилинганда қандай амаллар бажарилади?
3. Истиснолар рўйхатини бериш учун қандай оператордан фойдаланилади?
4. Блок finally нима учун ишлатилади?
5. Истиснолар қайта ишловчи ворис синфларни кўрсатинг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 18. МАВЗУ: СТАНДАРТ ТИПЛАР УЧУН СИНФЛАР

Режа: Типлар учун содда устқурмалар; Number; Double ва Float; Чексизлик ва NaN; Integer ва Long; Character; Boolean; Катта сонлар.

Типлар учун содда устқурмалар

Java юқори унумдорликни таъминлаш учун примитив типлардан масалан, int ва char типларидан фойдаланади. Бу типлар Java синфлар иерархиясига тегишли эмас. Улар усулларга қиймат бўйича узатилади, илова бўйича узатиш мумкин эмас. Шунинг учун хар бир примитив тип учун Java тилида махсус синф киритилган.

Number

Абстракт синф Number хамма стандарт скаляр иплар билан ишлаш учун интерфейс тақдим этади: — long, int, float ва double.

Бу синфда объект қийматини олишга имкон берувчи усул бўлиб, бу усул объект қийматини примитив тип қиймати сифатида қайтаришга имкон беради (балким яхлитланган холда):

• doubleValue() объект қийматини double типидаги қиймат сифатида қайтаради.

• floatValue() объект қийматини float типидаги қиймат сифатида қайтаради.

• intValue() объект қийматини int типидаги қиймат сифатида қайтаради.

• longValue()объект қийматини long типидаги қиймат сифатида қайтаради.

Double ва Float

Double ва Float синфлари —Number синфи ворислари. Суперсинфда эълон қилинган тўртта мурожаат усулларига қўшимча бу синфлар double ва float қийматлар билан ишлашни енгиллаштирувчи усулларга эга. Хар бир синфда объектларни double ва float типидаги қийматлар билан инициализация қилишга имкон берувчи конструкторлар мавжуд. Бундан ташқари фойдаланувчи қулайлиги учун бу объектларни хақиқий сон сатрли кўринишини тасвирловчи String объекти билан инициализация қилиш мумкин. Қуйидаги мисолда Double синфи объектларини иккала конструкторлар ёрдамида яратиш кўрсатилган.

```
class DoubleDemo {
public static void main(String args[]) {
Double d1 = new Double(3.14159);
Double d2 = new Double("314159E-5");
System.out.println(d1 + " = " + d2 + " -> " + d1.equals(d2));
} }
```

Дастур натижаси шуни кўрсатадики, equals усули true қайтаради, бу шуни билдирадики иккала ишлатилган конструкторлар бир хил Double синфи объектини яратади.

C:\> java DoubleDemo
3.14159 = 3.14159 -> true

Чексизлик ва NaN

IEEE спецификациясида хақиқий нуқтали сонлар икки double типидаги қиймат мавжуд бўлиб, махсус маънога эга: чексизлик ва NaN (Not a Number — ноаниқлик). Double синфида бу шартларни текширувчи икки шаклдаги тестлар мавжуд —double қиймати параметр сифатида узатилувчи (статик) усуллар шаклида ва Double синфи объектида сақланувчи сонни текширувчи усуллар шаклида.

• isInfinite(d) агар кўрсатилган double типидаги сон абсолют қиймати чексиз катта бўлса true қайтаради.

• isInfinite() агар Double объектида сақланаётган сон абсолют қиймати чексиз катта бўлса, true қайтаради.

• isNaN(d) агар double типидаги сон қимати ноаниқ бўлса true қайтаради.

• isNaN() агар Double объектида сақланаётган сон қиймати ноаниқ бўлса, true қайтаради.

Навбатдаги мисолда бири чексиз иккинчиси ноаниқ қийматли икки Double объекти яратилади.

```
class InfNaN {
public static void main(String args[]) {
Double d1 = new Double(1/0.);
Double d2 = new Double(0/0.);
System.out.println(d1 + ": " + d1.isInfinite() + ", " + d1.isNaN());
System.out.println(d2 + ": " + d2.isInfinite() + ", " + d2.isNaN());
} }
```

Қуйида дастур бажарилиши натижаси келтирилган:

C:\> java InfNaN
Infinity: true, false
NaN: false, true

Integer ва Long

Integer синфи —int, short ва byte типлар учун устқурма синф, Long синфи эса —long типи учун. Number суперсинфидан ворисликка ўтган усуллардан ташқари, Integer ва Long синфлари сонларни матнли кўриниши билан ишлаш ва сонларни матнли кўринишда тасвирлаш усулларига эга. Бу усуллар турли вариантлари ўзгартиришда ишлатиладиган саноқ тизими асосини кўрсатишга имкон беради. Одатда иккилик, саккизлик ва ўн олтилик саноқ тизимлари ишлатилади.

- parseInt(String) String ўзгарувчида сақланувчи бутун сон матнли кўринишини , int типидаги қийматга ўзгартиради. Агар бутун сатрли

кўриниши мумкин бўлган форматда бўлмаса NumberFormatException истисно яратилади.

- parseInt(String, radix) олдинги усул вазифасини бажаради, фақат иккинчи параметр ёрдамида 10 дан фарқли саноқ тизими асосини кўрсатиш мумкин.
- toString(int) параметр сифатида узатилган бутун сонни ўнлик саноқ тизимидаги матнли кўринишга келтиради.
- toString(int, radix) биринчи параметр сифатида узатилган бутун сонни иккинчи параметрда кўрсатилган саноқ тизимидаги матнли кўринишга келтиради.

Character

Character —char типидаги оддий синф - устқурма. Унда символ устида текширишлар ва ўзгартишлар бажарувчи бир неча статик усуллар мавжуд.

- isLowerCase(char ch) агар символ-параметр қуйи регистрга тегишли бўлса true қайтаради (фақат a-z диапазон эмас, ISO-Latin-1 кодировкадан фарқли қуйи регистрдаги символлар назарда тутилади).
- isUpperCase(char ch) юқори регистрдаги символлар учун худди шу вазифани бажаради.
- isDigit(char ch) ва isSpace(char ch) рақамлар ва бўшлик белгилари учун true қайтаради.
- toLowerCase(char ch) ва toupperCase(char ch) символларни юқори регистрдан қуйи оегистрга ва аксини бажаради.

Boolean

Boolean синфи — мантиқий қийматлар атрофида нозик устқурмадир, у boolean типи қиймат бўйича эмас илова бўйича узатиш керак бўлганда ишлатилади.

Катта сонлар

Агар мавжуд бутун сонли типлар аниқлиги етарли Java .math пакетидаги BigInteger ва BigDecimal номли синфлардан фойдаланиш мумкин. Бу синфлар рақамлар сони ихтиёрий бўлган бутун сонлар билан ишлашга мўлжалланган. Biglnteger ва BigDecimal синфлари бутун ва хақиқий сонлар учун ихтиёрий аниқликдаги арифметик амалларни жорий этади.

Оддий сонни катта сонга ўзгартириш учун қуйидаги статик усулдан фойдаланилади

valueOf:

Biglnteger a = Biglnteger.valueof(100);

Афсуски катта сонларга + ва * математик операторларни қўллаб бўлмайди. Бунинг ўрнига методы add ва multiply усулларидан фойдаланиш лозим.

Biglnteger c = a.add(b); // c = a + b

Biglnteger d = c.multiply(b.add(BigInteger.value.Of(2)));

// d - c * (b + 2)

Қуйидаги дастурда олдин келтирилган лотереяда ютиш имкониятларини хисобловчи дастур янги варианти келтирилган. Энди бу дастур катта сонлар билан ишлай олади. Агар лотерея ўйинида Например, если вам предложили сыграть в лотерее, в которой нужно угадать 60 сонни чисел из 490 мумкин сонлардан топиш лозим бўлса дастур маълумот берадики ютиш имконияти 1 тақсим

716395843461995557415116222540092933411717612789263493493351013459481104668848.

Омад!

Олдин келтирилган дастур қуйидаги операторни хисоблар эди:

lotteryOdds = lottery * (n - i + 1) / i;

Катта сонлар билан ишланганда эквивалент оператор қуйидаги кўринишга эга.

lotteryOdds = lotteryOdds.multiply(Biglnteger.valueOf(n-i+1)
.divide{Biglnteger.valueOf(i));

```
import javax.swing.*;
import Java.math.*;
public class BiglntegerTest
{
public static void main(String[] args)
{
String input = JOptionPane.showlnputDialog
("Қанча номер топиш лозим?");
int k = Integer.parselnt(input);
input = JOptionPane.showlnputDialog
("Энг катта мумкин бўлган номер нечага тенг?");
int n = Integer.parselnt(input);
Биномиал коэффициентларни хисоблаш
n * (n - 1) * (n - 2) • * . . . * (n - к + 1)
Biglnteger lotteryOdds = Biglnteger.valueOf(1);
for (int i = 1; i <= k;
lotteryOdds = lotteryOdds
.multiply(Biglnteger.valueOf(n - i + 1))
.divide(Biglnteger.valueOf(i));
System.out.println("Сизни имкониятингиз 1 тақсим " + lotteryOdds
+ ". Омад!");
System.exit(0) ;
```

Назорат саволлари:

1. Типлар учун махсус синфлар нима учун ишлатилади?
2. Хамма стандарт скаляр типлар билан ишлашга мўлжалланган абстракт синфни кўрсатинг.

3. Типлар билан ишловчи синфларни кўрсатинг.
4. Катта сонларбилан ишлаш учун қандай синфлардан фойдаланилади?
5. Катта сонларни қўшиш ва кўпайтириш усулларини кўрсатинг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

МАЪРУЗА 19. МАВЗУ: АБСТРАКТ ТИПЛАР УЧУН СТАНДАРТ СИНФЛАР

Режа: Вектор синфи; Абстракт луғат синфи; Хеш жадвал синфи.

Vector

Vector — элементлар сонини ошириш имкониятига эга бўлган объектларга иловалар массивидир. Ўз ичида Vector динамик кенгайиш стратегиясини реализация қилиб, фойдаланилаётган хотира ва хотира ажратишга кетадиган амаллар сонини минималлашга имкон беради. Объектларни Vector объекти охирига addElement усули ёрдамида қўшиш мумкин ёки индекс билан кўрсатилган позицияга, insertElementAt усули ёрдамида жойлаш мумкин. Vector синфига объектлар массивини ёзиш мумкин, бунинг учун copyInto усулидан фойдаланиш мумкин. Vector синфига объектлар коллекцияси ёзилгандан сўн, индивидуал элементларни Contains, indexOf ва lastIndexOf усуллари ёрдамида топиш мумкин. Бундан ташқари elementAt, firstElement ва lastElement усуллари Vector объектидан керакли холатдаги объектларни олишга имкон беради.

Stack

Stack синфи—Vector синфи вориси бўлиб, содда “биринчи кирган — биринчи чиқади " (FIFO) механизмини реализация қилади. Ўз аждодининг стандарт усулларига қўшимча Stack синфи элементни стек бошига қўшиш учун push усули ва энг юқори элементни чиқариб олиш учун pop усулини қўшади. Энг юқори элементни стекдан чиқармасдан ўқиш учун peek усулидан фойдаланиш мумкин. вы можете получить верхний элемент, не удаляя его из стека. Стекни бўшлигини empty усули билан текшириш мумкин. Аар бўш бўлса бу усул true қайтаради. Элементни излаш учун search усули ишлатилади. Бу усул элементни стек бошига келтириш учун керак pop амаллари сонини қайтаради. Агар берилган элемент стекда мавжуд бўлмаса бу усуд -1 қайтаради.

Қуйидаги дастурда аввал стек яратилади, унга бир неча Integer типидаги объект қўшилади ва стекдан чиқарилади.

```
import java.util.Stack;
import java.util.EmptyStackException;
class StackDemo {
static void showpush(Stack st, int a) {
st.push(new Integer(a));
System.out.println("push(" + a + ")");
System.out.println("stack: " + st);
}
static void showpop(Stack st) {
```

```
System.out.print("pop -> ");
Integer a = (Integer) st.pop();
System.out.println(a);
System.out.println("stack: " + st);
}
public static void main(String args[]) {
Stack st = new Stack();
System.out.println("stack: " + st);
showpush(st, 42);
showpush(st, 66);
showpush(st, 99);
showpop(st);
showpop(st);
showpop(st);
try {
showpop(st);
}
catch (EmptyStackException e) {
System.out.println("empty stack");
} }
}
```

Қуйида дастур бажарилиши натижаси келтирилган. Истисноларни қайта ишловчи бўш стекдан маълумот олишга уринилганда қандай ишлашига эътибор беринг.

```
C:\> java StackDemo
stack: []
push(42)
stack: [42]
push(66)
stack: [42, 66]
push(99)
stack: [42, 66, 99]
pop -> 99
stack: [42, 66]
pop -> 66
```

stack: [42]

pop -> 42

stack: []

pop -> empty stack

Dictionary

Dictionary (луғат) — абстракт синф бўлиб, ўзида “калит-қиймат” типидаги маълумотларни сақлайди. Калит — исм бўлиб, қийматга мурожаат қилиш учун ишлатилади. Калит ва унга мос қийматни луғатга put(key, value) усули ёрдамида киритиш мумкин. Калит бўйича қиймат олиш учун get(key) усулидан фойдаланилади. Калитлар ва қийматларни сановчи шаклида (объект Enumeration) методами keys ва elements усуллари ёрдамида олиш мумкин. Луғатдаги “калит-қиймат” жуфтликлари сонини size усули қайтаради, isEmpty усули возвращает true қайтаради, агар луғат бўш бўлса. Калит ва унга мос қийматни ўчириш учун remove(key) усули ишлатилади.

HashTable

HashTable —Dictionary вориси бўлиб, луғат конкрет реализациясидир. HashTable объектида ихтиёрий турдаги объектлар сақланиши мумкин. Бу коллекцияда индексация учун хам ихтиёрий объектлардан фойдаланиш мумкин. Кўп холларда HashTable, калит сифатида старлар (яъни String типидаги объектлар) хизмат қилувчи объектларни сақлаш учун ишлатилади. Навбатдаги мисолда HashTable китоб хақида маълумот сақлаш учун ишлатилади.

```
import java.util.Dictionary;
import java.util.Hashtable;
class HTDemo {
public static void main(String args[]) {
Hashtable ht = new Hashtable();
ht.put("title", "The Java Handbook");
ht.put("author", "Patrick Naugnton");
ht.put("email", "naughton@starwave.com");
ht.put(“age", new Integer(30));
show(ht);
}
static void show(Dictionary d) {
System.out.println("Title: " + d.get("title"));
System.out.println("Author: " + d.get("author"));
```

System.out.println("Email: " + d.get("email"));

System.out.println("Age: " + d.get("age"));

} }

Бу дастур бажарилиш натижаси шуни кўрсатадики параметри Dictionary абстракт синф бўлган show усули, main ичида ht га киритилган хамма маълумотларни чиқара олади.

C:\> java HTDemo

Title: The Java Handbook

Author: Patrick Naughton

Email: naughton@starwave.com

Age: 30

Properties

Properties —HashTable синфи вориси бўлиб, жадвалда балким аниқланмаган қийматларни олишга имкон берувчи бир неча усуллар қўшилган. Масалан getProperty усулида ном билан бирга кўзда тутилган қиймат кўрсатиш мумкин:

getProperty("ном","кўзда_тутилган_қиймат");

Агар жадвалда “ном” мавжуд бўлмаса усул “кўзда тутилган қиймат ” қайтаради. Бундан ташқари янги объект яратилганда синф конструкторига параметр сифатида бошқа Properties объектини узатиш мумкин, бу холда у янги объект яратишда кўзда тутилган қиймат сифатида ишлатилиши мумкин. Объект Properties ихтиёрий пайтда оқим — объект Stream дан ўқилиши мумкин. Қуйидаги мисолда бир неча хоссалар яратилиб, сўнгра ўқилади:

import java.util.Properties;

class PropDemo {

static Properties prop = new Properties();

public static void main(String args[]) {

prop.put("Title", "put title here");

prop.put("Author", "put name here");

prop.put("isbn", "isbn not set");

Properties book = new Properties(prop);

book.put("Title", "The Java Handbook");

book.put("Author", "Patrick Naughton");

System.out.println("Title: " +

book.getProperty("Title"));

System.out.println("Author: " +

book.getProperty("Author"));

System.out.println("isbn: " +

book.getProperty("isbn"));

System.out.println("ean: " +

book.getProperty("ean", "???"));

} }

Бу мисолда Properties синфининг, Title, Author ва isbn майдонлари учун кўзда тутилган бўйича уч қийматга эга prop объекти яратилди. Шундан сўнг book номли яна бир Properties объектини яратиб, Title ва Author майдонлар учун реал қийматлар берилди. Кейинги уч қаторда мавжуд уч калит учун getProperty усули қайтарган қиймат чиқарилди. Тўртинчи қаторда getProperty чақириғида мавжуд бўлмаган “ean” калит турган эди. Бу калит book объектда ва кўзда тутилган prop объектда мавжуд бўлмагани учун, getProperty усули чақириғида кўрсатилган кўзда тутилган қийматни, яъни “???” қайтарди:

C:\> java PropDemo

Title: The Java Handbook

Author: Patrick Naughton

isbn: isbn not set

ean: ???

Назорат саволлари:

1. Нима учун вектор синфи ишлатилади?
2. Вектор синфи усулларини кўрсатинг.
3. Стек қандай механизмдан фойдаланади?
4. Луғат учун абстракт синфни кўрсатинг.
5. Луғат синфи ворисларини кўрсатинг.

Адабиёт:

1. Смирнов Н.И. Java -2. Учебное пособие. М.:«Три Л», 2000. -320 с.
2. Арнольд К. и др. Язык программирования JAVA/ Вильямс. 2001. 624 с.
3. Брюс Эккель. Философия Java . Библиотека программиста. 2003.

«ОБЪЕКТГА ЙЎНАЛТИРИЛГАН ДАСТУРЛАШ ТИЛЛАРИ» ФАНИДАН АМАЛИЁТ МАШҒУЛОТЛАРИ

Амалий иш №1. Операторлар ва функциялар

I. Масалани қўйилиши

1. Цикл операторидан фойдаланиб вариантда кўрсатилган элементлар йиғиндиси топилсин.

2. Цикл операторидан фойдаланиб вариантда кўрсатилган масала ечилсин.

II. Вариантлар

1	$a_n=1/n$	Рақамлар суммасини хисоблаш
2	$a_n=1/n^2$	Ўрта рақамни хисоблаш
3	$a_n=1/n+1/n^2$	Максимал рақамни хисоблаш
4	$a_n=(n+1)/n^2$	Минмал рақамни хисоблаш
5	$a_n=(n+1)/n^3$	Бош рақамни хисоблаш
6	$a_n=(n+1)*(1/n^3+1/n^2)$	Рақамлар сонини хисоблаш
7	$a_n=1/(n+1)$	Хамма нолларни ўчириш
8	$a_n=1/(n+1)^2$	Хамма бирларни ўчириш
9	$a_n=n/(n+1)^2$	Бош рақамни ўчириш
10	$a_n=n/(n+1)^3$	Жуфт рақамларни ўчириш
11	$a_n=1/(n+1)+1/(n+1)^2$	Тоқ рақамларни ўчириш
12	$a_n=1/(n+1)^2+1/(n+1)^3$	Иккиликка ўтказиш
13	$a_n=1/2^n$	Саккизликка ўтказиш
14	$a_n=1/3^n$	Рақамлар суммасини хисоблаш
15	$a_n=1/2^n+1/3^n$	Ўрта рақамни хисоблаш
16	$a_n=n/2^n$	Максимал рақамни хисоблаш
17	$a_n=n/3^n$	Минимал рақамни хисоблаш
18	$a_n=n*(1/2^n+1/3^n)$	Бош рақамни хисоблаш
19	$a_n=(n+1)/2^n$	Рақамлар сонини хисоблаш
20	$a_n=(n+1)*(1/2^n+1/3^n)$	Хамма нолларни ўчириш
21	$a_n=(n+1)/3^n$	Хамма бирларни ўчириш
22	$a_n=(n+1)/2^n+n/3^n$	Бош рақамни ўчириш
23	$a_n=1/(2*n+1)$	Жуфт рақамларни ўчириш
24	$a_n=1/(2*n+1)^2$	Тоқ рақамларни ўчириш
25	$a_n=1/(2*n+1)^3$	Иккиликка ўтказиш
26	$a_n=n/(2*n+1)^2$	Саккизликка ўтказиш
27	$a_n=n/(2*n+1)^3$	Сонни айлантириш

III. Ечимга мисоллар

1.1. Мусбат жуфт бутун сонлар йиғиндиси топилсин.

```
public class met {
  static int sum (int n){
        int i, s=0;
        for( i=2; i<n; i+=2 ) s+=i;
        return s;
        }
  public static void main(String[] args) {
        int k=sum(6);
        System.out.println("summa="+k);
      }
      }
```

1.2. Қатор йиғиндиси ε=10-4 аниқлик билан топилсин. Умумий хади: $a_n=1/2^n$

```
public class met {
static double sum (double eps){
        double a=0.5;double s=0;
        while( a>eps){
        s+=a;
        a*=0.5;
        }
        return s;
        }
public static void main(String[] args) {
        double eps=0.0001;
        System.out.println(sum(eps));
}
}
```

2. Хамма рақамларни чиқариш:

```
public class met {
static void print (int n){
        int i;
        do{
        i=n%10;
        System.out.print(i);
        n/=10;
        }
        while(n>0);
        }
public static void main(String[] args) {
print(121);
}
}
```

Амалий иш №2. Массивлар

I. Масалани қўйилиши

1. Сонли массивлар билан ишловчи функция ёки усул яратинг ва дастурда фойдаланинг.

2. Қатор элементлари берилган шартга мос келишини текширувчи функция ёки усул яратинг ва дастурда фойдаланинг.

II. Вариантлар

1	**max_element**	Хамма элементлар берилганга тенг
2	**min_element**	Хамма элементлар берилганга тенг эмас
3	**count**	Хамма элементлар берилгандан катта
4	**find**	Хамма элементлар берилгандан кичик
5	**sort**	Хамма элементлар ўсувчи
6	**stable_sort**	Хамма элементлар ўсувчи эмас
7	**partial_sort**	Хамма элементлар камаювчи
8	**fill**	Хамма элементлар камаювчи эмас
9	**replace**	Хамма элементлар нольга тенг эмас
10	**reverse**	Хамма элементлар нольга тенг
11	**rotate**	Жуда бўлмаса битта элемент берилгандан катта
12	**random_shuffle**	Жуда бўлмаса битта элемент берилгандан кичик
13	**remove**	Хамма элементлар манфий
14	**unique**	Хамма элементлар ўзаро тенг
15	**copy**	Хамма элементлар уникал
16	**copy_bacwards**	Хар бир жуфт элемент икки қўшнисидан катта
17	**replace_copy**	Хар бир жуфт элемент икки қўшнисидан кичик
18	**remove_copy**	Хар бир тоқ элемент икки қўшнисидан катта
19	**unique_copy**	Хар бир тоқ элемент икки қўшнисидан кичик
20	**rotate_copy**	Жуда бўлмаса битта элемент манфий
21	**search**	Жуда бўлмаса битта элемент нольга тенг
22	**find_end**	Жуда бўлмаса битта элемент нольга тенг эмас
23	**equal**	Жуда бўлмаса битта элемент берилганга тенг
24	**mismatch**	Жуда бўлмаса битта элемент берилганга тенг эмас
25	**transform**	Биринчи элемент максимал
26	**swap_ranges**	Биринчи элемент минимал
27	**accumulate**	Охирги элемент минимал

III. Ечимга мисоллар

1. Массивда мусбат элементлар сони топилсин

```
public class met {
  static int pozitivcount (int a[] ){
  int i,  s=0;
  for( i=0; i<a.length; i++ ) if (a[i]>0) s++;
  return s;
  }
public static void main(String[] args) {
        int []a={ 3, -5, 7, 9, 11, -13, 15 };
        int s = pozitivcount ( a );
        System.out.println("s="+s);
  }
  }
```

2. Хамма элементлари мусбат

```
public class met {
static boolean pozitiv (int  a[])
{
for( int i=0; i<a.length; i++ ) if (a[i]<0) return false;
return true;
}

public static void main(String[] args) {
 int coll[]={8,1,2,3,9};
 if (pozitiv(coll))
 System.out.println("Хамма элементлар мусбат ");
else System.out.println("Хамма элементлар мусбат эмас ");
}
}
```

Амалий иш №3. Сатрлар

I. Масалани қўйилиши

1. Сатрлар билан ишловчи усул яратинг ва дастурда фойдаланинг.
2. Сатрни берилган сонга мослигини текширувчи функция ёки усул яратинг ва дастурда фойдаланинг.

II. Вариантлар

1	Strcpy	Ишорасиз бутун учлик сон
2	Strncpy	Ишорасиз бутун саккизлик сон
3	Strcat	Ишорасиз бутун ўнлик сон
4	Strncat	Ишорасиз бутун ўнолтилик сон
5	Strset	Ишорасиз бутун берилган асосли сон
6	Strnset	Бутун иккилик сон
7	Strcmp	Бутун саккизлик сон
8	Strncmp	Бутун ўнлик сон
9	Strdup	Бутун ўнолтилик сон
10	Strlwr	Бутун берилган асосли сон
11	Strupr	Ишорасиз хақиқий иккилик сон
12	Strchr	Ишорасиз хақиқий саккизлик сон
13	Strrchr	Ишорасиз хақиқий ўнлик сон
14	Strbrk	Ишорасиз хақиқий ўнолтилик сон
15	Strrev	Ишорасиз хақиқий берилган асосли сон
16	Strcpy	Ишорасиз бутун иккилик жуфт сон
17	Strncpy	Ишорасиз бутун саккизлик жуфт сон
18	Strcat	Ишорасиз бутун ўнлик жуфт сон
19	Strncat	Ишорасиз бутун ўнолтилик жуфт сон
20	Strset	Ишорасиз бутун берилган асосли жуфт сон
21	Strnset	Бутун иккилик жуфт сон
22	Strcmp	Бутун саккизлик жуфт сон
23	Strncmp	Бутун ўнлик жуфт сон
24	Strdup	Бутун ўнолтилик жуфт сон
25	Strlwr	Бутун берилган асосли жуфт сон
26	Strupr	Хақиқий иккилик
27	Strchr	Хақиқий ўнлик

III. Ечимга мисоллар

1. Берилган сондаги символлар конкатенацияси

```
public class met {
static String strncat(String s1, String s2,int koll)
{
return s1.concat(s2.substring(0,koll));
}
public static void main(String[] args) {
System.out.println(strncat("Hello","World",3));
}
}
```

2. Ишорасиз иккилик сон

```
public class met {
    static boolean check ( String a ){
        int i;
        for( i=0; i<a.length(); i++ )
        if ((a.charAt(i)<'0')||(a.charAt(i)>'1')) return false;
        return true;
        }
    public static void main(String[] args) {
        String a=new String("100");
        System.out.println("result="+check(a));

    }
}
```

Амалий иш №4. Синфлар ва объектлар

I. Масалани қўйилиши

1. Фойдаланувчи синфи яратилсин. Синфда параметрли конструктор, статик майдон ва статик усул яратилсин.

2. Фойдаланувчи синфи яратилсин. Синфда параметрли ва қийматлар текширилувчи конструктор, деструктор, статик майдон ва статик усул яратилсин. Конструкторда хотира динамик ажратиш ва деструкторда ўчириш ташкил этилсин.

II. Вариантлар

1	ТЕЛЕФОН (абонент исми, рақами, манзили, тури)	ШОФЕР
2	КОРАБЛЬ (исми, сувсилжиши, тури, ёши)	ЎҚИТУВЧИ
3	ДИСК (номи, хажми, нархи, мамлакат)	ХАРБИЙ
4	ОВҚАТ(номи, тури, калориялиги, нархи)	ИШЧИ
5	ТАСВИР (номи, тасвирчи, йили, галерея)	АКТЁР
6	АВТОМОБИЛЬ (номер, ном, мамлакат, нарх)	ФУТБОЛИСТ
7	САМОЛЁТ (ном, мамлакат, йил, жойлар сони)	БОКСЕР
8	БУЮРТМА (номер, махсулот, сумма, сана)	ЎҚУВЧИ
9	КОСТЮМ (размер, мамлакат, нарх, ранг)	АБИТУРИЕНТ
10	МАХСУЛОТ (исм, сон, қиймат, мамлакат)	ЎҚУВЧИ
11	ОБИДА (номи, архитектор, шахар, йил)	ТАЛАБА
12	СУБСТАНЦИЯ (номи, рақам, атом оғирлиги, заряд)	ХИЗМАТЧИ
13	КИТОБ (номи, муаллиф, нашриёт, сахифалар сони)	КВИТАНЦИЯ
14	ФУТБОЛИСТ (исм, команда, амплуа, ёш)	ЦЕХ
15	ХИЗМАТЧИ (исм, ёш, бўлим, стаж)	ЁДГОРЛИК
16	РАДИОДЕТАЛ (исми, маркаси, узел, санаси)	МАШҲУЛОТ
17	ТУЛПОР (лақаб, насл, вазн, ёш)	СПЕКТАКЛ
18	АВТОБУС(рақам, тури, пункт боши, охири)	ЁДГОРЛИК
19	КУРАТОР (исми, кафедра, стаж, гуруҳ)	КАССЕТА
20	ЧЎҚҚИ (номи, баландлиги, тоғлар, давлат)	БОКСЧИ
21	ГУРУҲ(исми, факултет, курс, талабалар сони)	КАФЕДРА
22	ПОЕЗД (исми, вагонлар сони, тури, ҳайдовчи)	АХБОРОТ
23	ЮЛДУЗ (номи, массаси, радиус, юлдуз туркуми)	ЖУРНАЛ
24	РЕКОРД(масофа, вақт, сана, спортчи)	МАХСУЛОТ
25	ДАВЛАТ (исми, ҳукумат, юза,аҳоли)	СУБСТАНЦИЯ
26	КОМПЬЮТЕР(номи, процессор, хотира, қиймати)	САМОЛЁТ
27	ХОККЕИСТ(исми, ёши, ким, команда)	КОМАНДА

III. Ечимга мисоллар

1. Фойдаланувчи синфи яратилсин. Синфда параметрли конструктор, деструктор, статик майдон ва статик усул яратилсин.

1. Фойдаланувчи синфи яратилсин. Синфда параметрли конструктор, статик майдон ва статик усул яратилсин.

```
class employee
{
  private String name;
  private long employee_id;
  private double salary;
  private static int count=0;
public employee(String name, long employee_id, double salary)
{
  count++;
  this.name=name;
  this.employee_id = employee_id;
  this.salary = salary;
 };
public void show_employee()
{
 System.out.println("Исм: "+ name);
 System.out.println("Номер: "+ employee_id);
 System.out.println("Маош: " + salary);
}
public static int getcount(){return count;};
}

public class met {
public static void main(String[] args) {
employee worker=new employee("Happy Jamsa",101,1.1);
worker.show_employee();
System.out.println("count="+employee.getcount());
}
}
```

2. Фойдаланувчи синфи яратилсин. Синфда параметрли ва қийматлар текширилувчи конструктор, деструктор, статик майдон ва статик усул яратилсин. Конструкторда хотира динамик ажратиш ва деструкторда ўчириш ташкил этилсин.

```
public class employee
{
```

```
private String name;
private long employee_id;
private float salary;
private static int count=0;
public employee(String name, long employee_id, float salary) {
count++;
this.name=new String(name);
this.employee_id = employee_id;
if (salary < 50000.0)
this.salary = salary;
else // Нотўғри маош
this.salary = 0.0f;
}
public void show_employee()
{
  System.out.println("Исм: "+ name);
  System.out.println("Номер: "+employee_id);
  System.out.println("Маош: " + salary);
}
public static int getcount(){return count;};
}

public class met {
public static void main(String[] args) {
 employee worker=new employee("Happy Jamsa", 101, 10101.0f);
 worker.show_employee();
 System.out.println("count="+employee.getcount());
}
}
```

Амалий иш №5. Синфлар орасидаги муносабатлар

I. Масалани қўйилиши

1. Биринчи синф объекти иккинчи синф майдони сифатида аниқланг.

2.. Фойдаланувчи синфи яратинг. Берилган критерийга мос келувчи объектларни чиқарувчи уч усулга эга синф яратинг. Иккинчи синфни биринчи синфга иложи бўлса дўстона синф сифатида аниқланг. Дастурда объектлар массивини киритиш ва берилган шартга мос келувчи объектларни чиқариш ташкил этилсин.

II. Вариантлар

1	хизматчи, инженер	телефон
2	кафедра ходими, домла	китоб
3	цирк артисти, ўргатувчи	тест
4	радиодетал, конденсатор	вилоят
5	қурилма, монитор	қўғирчоқ
6	китоб, дарслик	квитанция
7	имтиҳон, битирув имтиҳони	автомобиль
8	шаҳар, мегаполис	вертолёт
9	маҳсулот, сутли маҳсулот	монархия
10	банк хужжати, чек	тигр
11	поезд, экспресс поезд	пароход
12	самолёт, одам ташувчи лайнер	хоккеист
13	давлат,республика	аскар
14	ўрмон ҳайвони, айиқ	идора
15	елканли кема, корвет	рассом
16	футболчи, дарвозобон	телеграф
17	офицер, рота командири	волейболист
18	муассаса**,** клиника	медсестра
19	режиссер, театр режиссери	колледж
20	телефон, мобил телефон	хайкалтарош
21	хоккеист, вратарь	телеграф
22	врач, хирург	завод директори
23	мактаб, лицей	район
24	режиссер, кинорежиссер	ишчи
25	поезд, экспресс	декан
26	хайдовчи, машинист	трагик
27	медик, хирург	механизм

III. Ечимга мисоллар

1. Объектлар иерархиясини аниқланг

```
public class book {
   private String title;
   private String author;
   private int pages;
public book(String title, String author, int pages)
{
   this. title=title;
   this.author=author;
   this. pages = pages;
};
public void show_book()
{
   System.out.println( "Номи: " +title);
   System.out.println("Муаллиф: "+author);
   System.out.println("Сахифалар: " +pages);
}
}

public class library_card {
 private String catalog;
 private book  member;
public library_card(String title, String author, int pages,
String catalog)
{
   member=new book(title, author, pages);
   this.catalog=catalog;
}
public void show_card()
{
   member.show_book() ;
   System.out.println("Каталог: " +catalog);
 }
}
public class met {
public static void main(String[] args) {
library_card card=new library_card(" C++","Jamsa", 272, "101CPP");
card.show_card();
}
}
```

2. Биринчи синфни иккинчисига дўстона сифатида таърифлаш.

```
class book {
String title;
```

```
String author;
int pages;
public book(String title,String author, int pages)
{
this.title=title;
this.author=author;
this.pages=pages;
}
public void show_book(){
System.out.println("Номи: "+ title);
System.out.println( "Муаллиф: " + author);
System.out.println( "Сахифалар: " + pages);
}
}

class filtr_book{
public static void filtr_title (book a[], String title){
for(int i=0;i<a.length;i++)  if(a[i].title==title) a[i].show_book();
}
public static void filtr_author (book a[], String author){
for(int i=0;i<a.length;i++) if(a[i].author==author) a[i].show_book();
}
public static void filtr_pages (book a[], int pages){
for(int i=0;i<a.length;i++) if(a[i].pages>=pages) a[i].show_book();
}
}

public class met {
public static void main(String[] args) {
book a[]=new book[2];
a[0]=new book("Jamsa","C++",272);
a[1]=new book("Ekkel","JAVA",565);
filtr_book.filtr_title(a,"Jamsa");
filtr_book.filtr_author(a,"C++");
filtr_book.filtr_pages(a,565);
}
}
```

Амалий иш №6. Ворислик

I. Масалани қўйилиши

1. Содда ворислик асосида синфлар иерархиясини аниқланг.
2. Абстракт синф асосида синфлар иерархиясини аниқланг.

II. Вариантлар

1	хизматчи, инженер	қурилма, телефон
2	кафедра ходими, домла	нашр, китоб
3	цирк артисти, ўргатувчи	синов, тест
4	радиодетал, конденсатор	жой, вилоят
5	қурилма, монитор	мол, қўғирчоқ
6	китоб, дарслик	хужжат, квитанция
7	имтиҳон, битирув имтиҳони	траснпорт воситаси, автомобиль
8	шаҳар, мегаполис	учувчи аппарат, вертолёт
9	маҳсулот, сутли маҳсулот	давлат, монархия
10	банк хужжати, чек	ҳайвон, тигр
11	поезд, экспресс поезд	корабль, пароход
12	самолёт, одам ташувчи лайнер	ўйинчи, хоккеист
13	давлат,республика	ҳарбий жавобгар, аскар
14	ўрмон ҳайвони, айиқ	муасса, идора
15	елканли кема, корвет	маданият ходими, рассом
16	футболчи, дарвозобон	алоқа воситаси, телеграф
17	офицер, рота командири	ўйинчи, волейболист
18	муассаса**,** клиника	медик, медсестра
19	режиссер, театр режиссери	ташкилот, колледж
20	телефон, мобил телефон	санъаткор, хайкалтарош
21	хоккеист, вратарь	алоқа воситаси, телеграф
22	врач, хирург	рахбар, завод директори
23	мактаб, лицей	жой, район
24	режиссер, кинорежиссер	шахс, ишчи
25	поезд, экспресс	ходим, декан
26	хайдовчи, машинист	артист, трагик
27	медик, хирург	ускуна, механизм

III. Ечимга мисоллар

1. Содда ворислик асосида синфлар иерархиясини яратиш

```
public class book
{
  protected  String title;
  protected String author;
  protected  int pages;
public book(String title, String author, int pages)
{
   this. title=title;
   this.author=author;
   this. pages = pages;
};
public void show_book()
{
   System.out.println( "Номи: " +title);
   System.out.println("Муаллиф: "+author);
   System.out.println("Сахифалар: " +pages);
}
}
public class library_card extends book {

   private String catalog;

public library_card(String title, String author, int pages,
String catalog)
{
  super(title, author, pages);
   this.catalog=catalog;
}
public void show_card()
{
   show_book() ;
   System.out.println("Каталог: " +catalog);
 }
}
public class met {
public static void main(String[] args) {
library_card card=new library_card(" C++","Jamsa", 272, "101CPP");
card.show_card();
}
}
```

2. Абстракт синф асосида синфлар иерархиясини яратиш

```
public abstract class book
{
  protected  String title;
  protected String author;
  protected  int pages;
public book(String title, String author, int pages)
{
   this. title=title;
   this.author=author;
   this. pages = pages;
};
public abstract void show();
}

public class library_card extends book {

   private String catalog;

public library_card(String title, String author, int pages,
String catalog)
{
  super(title, author, pages);
   this.catalog=catalog;
}
public void show()
{
  System.out.println( "Номи: " +title);
  System.out.println("Муаллиф: "+author);
  System.out.println("Сахифалар: " +pages);
   System.out.println("Каталог: " +catalog);
 }
}
public class met {
public static void main(String[] args) {
library_card card=new library_card(" C++","Jamsa", 272, "101CPP");
card.show();
}
}
```

Амалий иш №7. Универсал функциялар ва усуллар

I. Масалани қўйилиши

1. Универсал усулдан фойдаланиб дастур яратинг.

2. Кетма кетлик элементлари берилган шартга мос келишини текширувчи универсал усулга эга синф яратинг. Тестлашни бажаринг.

II. Вариантлар

1	**max_element**	Хамма элементлар берилганга тенг
2	**min_element**	Хамма элементлар берилганга тенг эмас
3	**Count**	Хамма элементлар берилгандан катта
4	**Find**	Хамма элементлар берилгандан кичик
5	**Sort**	Хамма элементлар ўсувчи
6	**stable_sort**	Хамма элементлар ўсувчи эмас
7	**partial_sort**	Хамма элементлар камаювчи
8	**Fill**	Хамма элементлар камаювчи эмас
9	**replace**	Хамма элементлар нольга тенг эмас
10	**reverse**	Хамма элементлар нольга тенг
11	**Rotate**	Жуда бўлмаса битта элемент берилгандан катта
12	**random_shuffle**	Жуда бўлмаса битта элемент берилгандан кичик
13	**remove**	Хамма элементлар манфий
14	**unique**	Хамма элементлар ўзаро тенг
15	**Copy**	Хамма элементлар уникал
16	**copy_bacwards**	Хар бир жуфт элемент икки қўшнисидан катта
17	**replace_copy**	Хар бир жуфт элемент икки қўшнисидан кичик
18	**remove_copy**	Хар бир тоқ элемент икки қўшнисидан катта
19	**unique_copy**	Хар бир тоқ элемент икки қўшнисидан кичик
20	**rotate_copy**	Жуда бўлмаса битта элемент манфий
21	**Search**	Жуда бўлмаса битта элемент нольга тенг
22	**find_end**	Жуда бўлмаса битта элемент нольга тенг эмас
23	**Equal**	Жуда бўлмаса битта элемент берилганга тенг
24	**mismatch**	Жуда бўлмаса битта элемент берилганга тенг эмас
25	**transform**	Биринчи элемент максимал
26	**swap_ranges**	Биринчи элемент минимал
27	**accumulate**	Охирги элемент минимал

III. Ечимга мисоллар

1. Массивда ноль элементлар сони топилсин

```
public class alg {
public  static long count(long n, Object b, Object a[])
{
        int i; int s=0;
        for(i=0; i<n; i++) if (a[i].equals(b)) s++;
        return s;

}
}
public class met {
public static void main(String[] args) {
Integer a[]=new Integer[7];
Integer b=new Integer(0);
int i;
for(i=0;i<7;i++) a[i]=new Integer(i);
a[3]=new Integer(0);
long s=alg.count(7, b, a);
System.out.println("result="+s);
}
}
```

2. Хамма элементлари мусбат

```
public abstract class Obj {
abstract boolean pozitiv();
}
public class alg {
public static boolean pozitiv (Obj a[]) {
  for( int i=0; i<a.length; i++ ) if (!(a[i].pozitiv())) return false;
  return true;
  }
}
public class Int extends Obj{
  private int a;
  public Int( int a1) {a=a1;}
  public boolean pozitiv(){if (a>=0) return true; else return false;}
}
public class met {
public static void main(String[] args) {
Int a[]=new Int[5];
for(int i=0;i<5;i++) a[i]=new Int(i);
System.out.println("result="+alg.pozitiv(a));
```

```
}
}
```

Амалий иш №8. Оқимли синфлар ва файллар билан ишлаш

I. Масалани қўйилиши

1. Синф яратилсин. Синф киритиш усули яратилсин. Дастурда синф типидаги массив киритилиб файлга ёзилсин.

2. Синф яратилсин. Синф чиқариш функцияси яратилсин Файлга ёзилган маълумотлар массивга ўқилсин. Массив чиқарилсин.

II. Вариантлар

1	ШОФЕР (исм, номери, телефон, техпаспорт номери)	ТЕЛЕФОН
2	ЎҚИТУВЧИ (исм, кафедра, стаж, юклама)	КОРАБЛЬ
3	ХАРБИЙ (исм, унвон, қўшин тури, хизмат муддати)	ДИСК
4	ИШЧИ (исм, цех номери, завод, стаж)	ОВҚАТ
5	АКТЁР (исм, театр, унвон, роллар сони)	ТАСВИР
6	ФУТБОЛИСТ (исм, команда, амплуа, ёш)	АВТОМОБИЛЬ
7	БОКСЕР (исм, оғирлик, ёш, ғалабалар сони)	САМОЛЁТ
8	ЎҚУВЧИ (исм, ёш, мактаб, синф)	БУЮРТМА
9	АБИТУРИЕНТ (исм, сана, балл, ўртача балл)	КОСТЮМ
10	ЎҚУВЧИ (исм, синф, телефон, ёш)	МАХСУЛОТ
11	ТАЛАБА (исм, гурух, ёш, вилоят)	ОБИДА
12	ХИЗМАТЧИ (исм, ёш, бўлим, стаж)	СУБСТАНЦИЯ
13	КВИТАНЦИЯ (рақами, сана, манзили, эгаси)	КИТОБ
14	ЦЕХ (номер, исми, ишчи сони, завод)	ФУТБОЛИСТ
15	ЁДГОРЛИК (номи, архитектор, давлат, йил)	ХИЗМАТЧИ
16	МАШҲУЛОТ (ном, гуруҳ, аудитория,ўқитувчи)	РАДИОДЕТАЛ
17	СПЕКТАКЛ (номи, автор, режиссер, театр)	ТУЛПОР
18	ЁДГОРЛИК (номи, архитектор, давлат, йили)	АВТОБУС
19	КАССЕТА (фильм номи, режиссер, вақт, нархи)	КУРАТОР
20	БОКСЧИ (исми, вазн, учрашувлар сони, ютуқлар)	ЧЎҚҚИ
21	КАФЕДРА (номи, мудири, факультет, телефон)	ГУРУҲ
22	АХБОРОТ (ташувчи, хажми, номи, муаллифи)	ПОЕЗД
23	ЖУРНАЛ (исми, даврийлик, нашриёт, тираж)	ЮЛДУЗ
24	МАХСУЛОТ (исми, сони, нархи, фирма)	РЕКОРД(
25	СУБСТАНЦИЯ (номи, рақам, ядроли ўлчов, заряд)	ДАВЛАТ
26	САМОЛЁТ (маркаси, сиғими, компанияси, давлат).	КОМПЬЮТЕР
27	КОМАНДА (номи, шахар, ўйинчилар, очколар)	ХОККЕИСТ

III. Ечимга мисоллар

1. Хизматчи хақидаги маълумотни файлга чиқариш

```
import java.io.*;
public class person {
public String name;
public int year;
public person(String name,int year){
  this.name=name;
  this.year=year;
}
 }

public class files {
public static void  main(String[] args) throws Exception{
  person a=new person("Smit",1125);
  OutputStream f0 = new FileOutputStream("file1.txt");
  byte buffer[] = new byte[25];
  String t=a.name+"\n"+String.valueOf(a.year);
  buffer=t.getBytes();
  f0.write(buffer);
  f0.close();
}
}
```

2. Хизматчи хақидаги маълумотни файлдан ўқиш

```
import java.io.*;
public class files1 {
public static void  main(String[] args) throws Exception{
  InputStream f1 = new FileInputStream("file1.txt");
  int size = f1.available();
  for(int i=0;i<size;i++)
  System.out.print((char)f1.read());
  f1.close();
 }
}
```

Амалий иш №9. Истисноларни бошқариш

I. Масалани қўйилиши

1. Файл F1 яратилсин ва унга 20 дан кам бўлмаган сонлар ёзилсин. Файл F2 га вариантда кўрсатилган қийматлар нусха олинсин. Дастурда истиснолар қайта ишлансин.

2. Истисно холатларни хисобга олган холда ифодани хисобловчи функция ёки усул яратинг ва дастурда фойдаланинг.

II. Вариантлар

1	Жуфт сонлар	a=1/x
2	Жуфт номерли сатрлар	a=1/(1+x)
3	Тоқ сонлар	a=1/(1-x)
4	Тоқ номерли сатрлар	$a=1/x^2$
5	Биринчи рақами 1 сонлар	$a=1/(1-x)^2$
6	Биринчи харфи A сатрлар	$a=1/(1+x)^2$
7	Охирги рақами 1 сонлар	$a=x^{1/2}$
8	Охирги харфи A сатрлар	$a=(1-x)^{1/2}$
9	Сонлар 4 дан бошлаб	$a=(1+x)^{1/2}$
10	Сатрлар 4 номердан бошлаб	$a=(x-1)^{1/2}$
11	Сонлар 4 дан катта эмас	$a=1/x^{1/2}$
12	Сатрлар номери 4 дан катта эмас	$a=1/(1-x)^{1/2}$
13	Охирги рақами 1 сонлар	$a=1/(1+x)^{1/2}$
14	Охирги харфи A сатрлар	$a=1/(x-1)^{1/2}$
15	Биринчи харфи B сатрлар	a=1/sin(x)
16	Охирги рақами 0 сонлар	a=1/cos(x)
17	Сонлар K дан бошлаб	a=1/(1-sin(x))
18	Сатрлар K номердан бошлаб	a=1/(1-cos(x))
19	Сонлар K дан катта эмас	a=1/(1+sin(x))
20	Сатрлар номери K дан катта эмас	a=1/(1+cos(x))
21	Жуфт сонлар K дан бошлаб	a=tg(x)
22	Жуфт сатрлар K номердан бошлаб	a=ctg(x)
23	Тоқ сонлар K дан бошлаб	a=1/tg(x)
24	Тоқ сатрлар K номердан бошлаб	a=1/ctg(x)
25	Жуфт сонлар K дан катта эмас	a=ln(x)
26	Жуфт сатрлар номери K дан катта эмас	a=1/ln(x)
27	Тоқ сонлар K дан катта эмас	$a=1/(1-e^x)$

III. Ечимга мисоллар

1.1. Файл F2 га F1 файлдан 1 рақамга тугаган сонлар нусха олинсин.

```
import java.io.*;
import java.util.*;
public class met {

public static void file_copy(String source, String target) throws Exception
  {
  int k;
  File f=new File(source);
  if (!f.exists()) throw new file_open_error();
  InputStream input_file=new FileInputStream(source);
  OutputStream output_file=new FileOutputStream(target);

  int size = input_file.available();
  byte b[] = new byte[4];
  for (int i=0; i < size/4; i++) {
  if (input_file.read(b)!=b.length) throw new file_read_error();

  if ((char)b[3]=='1')
  output_file.write(b);
  }
input_file.close();
output_file.close();
  }
public static void main(String[] args) throws Exception {
  try {
        file_copy("SOURCE.TXT", "TARGET.TXT") ;
        }
catch (file_open_error e){
        System.out.println( "file_open_error ");
System.exit(1);
        }
catch (file_read_error e){
  System.out.println ("file_read_error ");
        System.exit(1);
        }
  }
}
```

1.2. Файл F2 га F1 файлдан «A» харфидан бошланган сатрлар нусха олинсин.

```
import java.io.*;
import java.util.*;
public class met {

public static void file_copy(String source, String target) throws Exception
  {
  int k;
  File f=new File(source);
  if (!f.exists()) throw new file_open_error();
  InputStream input_file=new FileInputStream(source);
  OutputStream output_file=new FileOutputStream(target);

  int size = input_file.available();
  byte b[] = new byte[4];
  for (int i=0; i < size/4; i++) {
  if (input_file.read(b)!=b.length) throw new file_read_error();
  if ((char)b[0]=='A')
  output_file.write(b);
  }
input_file.close();
output_file.close();
}
public static void main(String[] args) throws Exception {

  try {
       file_copy("SOURCE.TXT", "TARGET.TXT") ;
       }
catch (file_open_error e){
       System.out.println( "file_open_error ");
System.exit(1);
       }
catch (file_read_error e){
  System.out.println ("file_read_error ");
       System.exit(1);
       }
  }
}
```

2. Истиснолардан фойдаланиб икки сон бўлинмасини хисоблаш.

```
public class Cath {
static int divide(int a, int b)
{
return a/b;
};
public static void main(String[] args) {
try {
int b = divide(42,0);
}
catch (ArithmeticException e) {
System.out.println("div by 0: " + e);
}
}
}
```

Амалий иш №3 учун илова

«Сатрлар билан ишлаш учун функциялар»

	Функ-ция	Прототип ва функция қисқа таърифи
1	Strcpy	char *strncpy(char *s1, const char *s2); s1 сатр символларини s2 сатрга нусха олади.
2	Strncpy	char *strncpy(char *s1, const char *s2, int kol); s1 сатр kol символини s2 сатрга нусха олади.
3	Strcat	char *strcat(char *s1, const char *s2); s1 сатр охирига s2 сатр символларини улайди.
4	Strncat	char *strncat(char *s1, const char *s2, int kol); s1 сатр охирига s2 сатр kol символларини улайди.
5	Strset	char *strnset(char *str, int c, int kol); s1 сатрни c символ билан алмаштиради.
6	Strnset	char *strnset(char *str, int c, int kol); s1 сатр биринчи kol символини c символ билан алмаштиради.
7	Strcmp	int strcmp(const char *s1, const char*s2); s1 ва s2 сатрларни солиштиради натижа < 0 агар s1 < s2 натижа == 0 агар s1 == s2 натижа > 0 агар s1 > s2
8	Strncmp	int strncmp (const char *s1, const char *s2, size_t maxlen); maxlen дан кўп бўлмаган s1ва s2 сатр символларини солиштиради.
9	Strdup	char *strdup (const char *str); хотира ажратади ва унга str сатр нусхасини кўчиради.
10	Strlwr	char *strlwr(char *s1); s1 сатрдаги катта харфларни мос кичик харфларга айлантиради.
11	Strupr	char *strupr(char *s1); s1 сатрдаги кичик харфларни мос катта харфларга айлантиради.
12	Strchr	char *strchr(char *s1, int c); s1 сатрда c символ биринчи киришини қидиради.
13	Strrchr	char *strrchr(char *s1, int c); s1 сатрда c символ охирги киришини қидиради.
14	Strbrk	char *strbrk(char *s1, char *s2); s1 сатрда s2 сатр ихтиёрий символининг биринчи киришини қидиради.
15	Strrev	char *strrev(char *s); Сатрни айлантиради

Амалий иш №2 ва №7 учун илова

«Универсал функциялар ва усуллар»

1	**max_element**	энг катта қиймат
2	**min_element**	энг кичик қиймат
3	**count**	қийматни киришлари сонини хисоблайди
4	**find**	қийматни биринчи киришини хисоблайди
5	**sort**	Тартиблайди
6	**stable_sort**	тенг элементлар тартибини сақлаб тартиблайди
7	**partial_sort**	берилган қисмни тартиблайди
8	**fill**	хамма элементларни берилган қиймат билан алмаштиради
9	**replace**	кўрсатилган қийматли элементларни алмаштиради
10	**reverse**	элементлар тартибини тескарига алмаштиради
11	**rotate**	элементларни циклик силжитади
12	**random_shuffle**	элементлар ўрнини тасодифий алмаштиради
13	**remove**	берилган қийматли элементларни ўчиради
14	**unique**	тенг қўшни элементларни бирини ўчиради
15	**copy**	биринчи элементдан бошлаб нусха олади
16	**copy_bacwards**	охирги элементдан бошлаб нусха олади
17	**replace_copy**	берилган қийматли элементларни алмаштириб, нусха олади
18	**remove_copy**	берилган қийматли элементларни ўчириб, нусха олади
19	**unique_copy**	тенг қўшни элементларни бирини ўчириб нусха олади
20	**rotate_copy**	элементларни циклик силжитиб, нусха олади
21	**search**	кетма кетлик биринчи киришини топади
22	**find_end**	кетма кетлик охирги киришини топади
23	**equal**	икки кетма кетлик тенглигини текширади
24	**mismatch**	икки кетма кетликда фарқ қилувчи биринчи элементни топади
25	**transform**	икки кетма кетликни улайди
26	**swap_ranges**	икки кетма кетлик ўрнини алмаштиради
27	**accumulate**	суммани хисоблайди