O’ZBEKISTON RESPUBLIKASI OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI

BUXORO DAVLAT UNIVERSITETI

Fizika – matematika fakulteti

“Amaliy matematika va axborot texnologiyalari” kafedrasi

Cho’lliyev Shohruh Ibadullayevich

JAVA DASTURLASH TILIDA SATR VA FAYL OQIMLARI BILAN ISHLASH BO’YICHA USLUBIY QO’LLANMA

5480100-“Amaliy matematika va informatika” ta’lim yo’nalishi bo’yicha bakalavr darajasini olish uchun

BITIRUV MALAKAVIY ISHI

“Ish ko`rildi va himoya qilishga ruxsat berildi”
Kafedra mudiri.
_____ dots.T.B.Boltayev
«_____» _____2015 y.

Ilmiy rahbar _____ dots.O.I. Jalolov
“_____” _____ 2014y.
Taqrizchi _____ katta o’qit. T. Sohibov
“_____” _____ 2015 y.

“Himoyaga ruxsat berildi”
Fakultet dekani _____prof.Sh.M.Mirzayev
“_____” _____ 2015-y.

Buxoro – 2015 yil

MUNDARIJA

KIRISH

I.BOB. JAVA DASTURLASH TILINING SINTAKSISI VA ASOSIY OPERATORLAR.

1.1. Java dasturlash tilining sintaksisi. Ma’lumotlar tiplari……..

1.2. Java dasturlash tilida asosiy operatorlar va standart funksiyalar……

1.3. Java dasturlash tilida boshqarish operatorlari…….

II.BOB. JAVA DASTURLASH TILIDA SATR VA FAYLLAR OQIMI.

2.1. Java dasturlash tilida satr oqimi va ular ustida amallar……

2.2. Java dasturlash tilida fayl oqimi va ular ustida amallar.....

2.3. Ma’lumotlarni faylda yozish va o’qish…….

XOTIMA……

ADABIYOTLAR RO’YXATI……...

KIRISH

O’zbekiston Respublikasi mustaqillik odimlarini dadil qo’yayotgan hozirgi davrda axborotlashgan jamiyat qurish masalasi mamlakatimiz uchun naqadar katta ahamiyat kasb etayotgani hech kimga sir emas.

Vatanimizning kelajagi xalqimizning ertangi kuni, mamlakatimizning jahon hamjamiyatidagi obro’-e’tibori avvalambor farzandlarimizning unib-o’sib, ulg’ayib, qanday inson bo’lib hayotga kirib borishiga bog’liqdir. Biz bunday o’tkir haqiqatni hech qachon unutmasligimiz kerak.

Respublikamiz mustaqillikka erishgach, ta’lim tizimida tub o’zgarishlar sodir bo’ldi. Jamiyat taraqqiyotini yoshlar hayotida ishlab chiqarish milliy hamda umuminsoniy munosabatlarda zamonaviy qarash imkonyatlari kengaydi. Yoshlar ta’lim tarbiyasi bilan mashg’ul bo’ladigan ijtimoiy institutlar son va sifat jihatdan o’sdi. Davlat standartlari tanlanib, pedagogik amaliyotda tadbiq etila boshlandi. Mustaqillik yillarida axborot texnologiyalari sohasida juda katta o’zgarishlar yuz bermoqda, shu bilan bir qator bu yoshlarga keng imkonyatlar yaratilmoqda. Ijtimoiy hayotning barcha sohalarida axbarot texnologiyalari har yerda jadallik bilan kirib kelmoqda va bilimlarni oshishi jamiyat a’zolarini o’z ustlarida ko’proq ishlashga yangi-yangi o’zgarishlar bilan undashga olib kelmoqda.

Mustaqil respublikamizda yuz berayotgan siyosiy, iqtisodiy, ilmiy-texnikaviy va madaniy o’zgarishlar Oliy ta’lim tizimida ham o’z aksini topmoqda.

Hozirgi kunda kompyuter texnologiyalari kirib bormagan soha deyarli uchramaydi. Kompyuter texnologiyalardan nafaqat hisoblash ishlarini olib borish uchun balki, hayotga tadbiq qilinadigan dasturiy tizimlar, rasm va video tasmalarni qayta ishlovchi, katta hajmli ma’lumotlarni o’zida saqlovchi dasturlar yaratish, elektron tijorat uchun ham foydalaniladi. So’nggi yillarda kompyuter va uning dasturiy ta’minotiga bo’lgan talab va qiziqishlar ortib bormoqda. Bu esa o’z navbatida dasturchidan katta izlanish va mahoratni talab qiladi.

Bu muammoga hukumatimiz tomonidan alohida e’tibor berilmoqda. Yurtboshimiz I.A.Karimovning 2001 - yil may oyida Oliy Majlisning 5-sessiyasida so’zlagan nutqida axborot texnologiyalari va kompyuterlarni jamiyat hayotiga,

kishilarning turmush tarziga, maktab va oliy o'quv yurtlariga jadallik bilan olib kirish g'oyasi ilgari surilgan edi. Prezidentimiz tashabbusi bilan yurtimizda "kompyuter va axborot texnologiyalarini rivojlantirish", shuningdek, "Internetning xalqaro axborot tizimlariga keng kirib borishini ta'minlash" dasturini ishlab chiqishni tashkil etish chora-tadbirlari to'g'risidagi qarorlari qabul qilindi. 2002 yil 30 mayda O'zbekiston Respublikasi Prezidentining "Kompyuterlashtirishni yanada rivojlantirish va axborot- kommunikatsiya texnologiyalarini joriy etish to'g'risida"gi farmoni va uning ijrosini amalga oshirish yuzasidan va axborot-kommunikatsiya texnologiyalari sohasida strategik ustuvorlikni amalga oshirishga doir amaliy chora-tadbirlarni ta'minlash maqsadida Vazirlar Mahkamasining 2002 yil 6 iyundagi "2002-2010 yillarda kompyuterlashtirish axborot- kommunikatsiya texnologiyalarini rivojlantirish dasturi " to'g'risidagi qarori e'lon qilindi.

Yangi axborot – kommunikatsion texnologiyalari hozirgi vaqtda eng dolzarb mavzulardan biri bo'lib kelmoqda, sababi har bir sohani o'rganish, izlanish va tajriba orttirish uchun turli usullardan foydalanish kerak bo'ladi. Shuning uchun yangi axborot – kommunikatsion texnologiyalardan foydalanish maqsadga muvofiqdir.

Hozirgi zamon mutaxassislari, faoliyat doiralari qanday bo'lishidan qat'iy nazar informatika bo'yicha keng ko'lamdagi bilimlarga, zamonaviy hisoblash texnikasi, informatsion aloqa va kommunikatsiya tizimlari, orgtexnika vositalari va ulardan foydalanish borasida yetarli malakalarga ega bo'lishi, hamda yangi information texnika va texnologiya asoslarini uning ertangi kuni, rivoji to'g'risidagi bilimlarni o'zida mujassamlashtirgan bo'lishi kerak. Zamonviy hisoblash texnikasi va information texnologiyalarning kun sayin rivojlanib, jamiyatning esa tobora informatsiyalashib borishi sababli, uzluksiz ta'lim tizimining o'rta va yuqori bosqichlariga informatika, ishlab chiqarish va boshqarish jarayonlarini kompyuterlashtirish bo'yicha bir qator o'quv fanlari kiritilgan.

O`zbekiston Respublikasi demakratik, huquqiy va fuqarolik jamiyatini qurish yo`lidan borayotgan bir paytda ta`lim sohasida amalgam oshirilayotgan

islohotlarning bosh maqsadi va harakatga keltiruvchi kuchi har tomonlama rivojlangan barkamol insonni tarbiyalashdir.

Axborot jamiyatni rivojlantiruvchi va uning taraqqiyotiga asos bo'luvchi muhim vosita hisoblanadi. Shu kabi axborot insoniyat tarixida eng muhim iqtisodiy ko'rsatkichlardan biri bo'lsa, jamiyatni kompyuterlashtirish esa iqtisodiyotni tarkibiy jihatdan qayta qurishga asosiy harakatlantiruvchi kuchdir. Oxirgi davrda axborotli muhitda katta o'zgarishlar bo'lib bormoqda. Ana shu o'zgarishlar qog'ozsiz texnologiya zarurligini keltirib chiqardi. Bu esa o'z navbatida EHM ning yanada keng rivojlanishiga sabab bo'ldi. Kompyuterlashtirish texnik muammolarni o'z ichiga olsa, axborotlashtirish kompleks jarayon bo'lib, u jamiyat hayotining barcha jabhalarini qamrab oladi. Kompyuterlar uning texnik asosinigina tashkil etadi xolos.

Respublikamizda viloyatlar, shaharlar, tumanlarga qarashli korxonalar, tashkilotlar va muassasalar zamonaviy kompyuter texnikalari bilan jihozlanib, ular maxsus qurilmalar (telefon tarmog'i, modem va boshqalar) yordamida axborotlarni uzatish va qabul qilish imkoniyatiga ega bo'lmoqdalar.

Xalqning boy intellektual merosi va umumbashariy qadriyatlar asosida, zamonaviy madaniyat, iqtisodiyot, fan, texnika va texnologiyalarning yutuqlari asosida kadrlar tayyorlashning mukammal tizimini shakllantirish O'zbekiston taraqqiyotining muhim shartidir.

XX asr o'rtalariga kelib tezkor mashina mexanizmlaridan foydalana boshlandi. Murakkab texnika va texnologiyalar o'ylab topildi. Ko'pgina masalalarni hal qilish jarayonida axborot hajmi behisob bir majmuaga aylandi, hamda bu axborotlarni yig'ish va uzatish vositalarini yaratish, ularni vaqtida qayta ishlab, boshqarish uchun zarur bo'lgan choralarni belgilab chiqish kerak bo'ladi. Ko'pchilik vazifalarni bajarishda boshqarish jarayonini takomillashtirish, axborot tizimini joriy etish, mutaxasislarni kompyuterda ishlashga o`rgatish muhim ahamiyatga ega. Undan tashqari biznes va tadbirkorlikni rivojlantirish sohasida juda katta ishlar olib borish lozim.

Mavzuning dolzarbligi, nazariy va amaliy ahamiyati. JAVA dasturlash tili obektga mo'ljallangan dasturlash tili hisoblanadi. Juda ko'p murakkab masalalarni hal qilishda albatta fayl oqimlaridan foydalanishga to'g'ri keladi. Ushbu ishda fayllar ustida bajariladigan barcha amallar keltirilgan. Bu ishning dolzarb ekanligini ko'rsatadi. JAVA dasturlash tilining ahamiyati shundaki bu til sistema tanlamaydi, windowsda ishlayotgan dastur boshqa sistemalarda ham ishlay oladi.Ish uslubiy qo'llanma ko'rinishda tayyorlanadi. Bu juda katta nazariy va amaliy ahamiyatga ega.

Tadqiqot ob'ekti va predmeti. Jetbrains IntelliJ IDEA muhitida, dasturlash tili. Zamonaviy kompyuter.

Ishning maqsadi va vazifasi. JAVA dasturlash tili yordamida fayllar bilan ishlash, ular ustida bajariladigan amallarni misollar yordamida yoritib berish va o'zbek tilida uslubiy qo'llanma tayyorlash.

Tadqiqot uslubi va uslubiyoti. Mavjud adabiyotlarni o'rgangan holda kerakli natijalarga erishish va ularga asoslangan holda o'zbek tilida uslubiy qo'llanma tayyorlash.

Olingan asosiy natijalar. Talablar yoki foydalanuvchilar mustaqqil o'rganishlari uchun o'zbek tilida JAVA dasturlash tilida asosiy operatorlar,satrlar va fayllar oqimi bilan ishlash bo'yicha uslubiy qo'llanma tayyorlandi.

Natijalarning ilmiy yangiligi va amaliy ahamiyati. O'zbek tilida JAVA dasturlash tilida asosiy operatorlar,satrlar va fayllar oqimi va ular ustida bajariladigan amallar bo'yicha uslubiy qo'llanma qilinmagan.

Tadbiq etish darajasi va iqtisodiy samaradorligi. Qo'llanish sohasi. Xulosa va takliflar. Ushbu qo'llanmadan dasturlash asoslari fanidan o'quv darslarida foydalanish mumkin. Bundan tashqari JAVA dastrulash tilini mustaqil o'rganuvchilar ham ushbu uslubiy qo'llanmadan foydalanishlari mumkin.

Ishning hajmi va tuzilishi. Bitiruv malakaviy ishi kirish qismi, 2 ta bob, har bir bobning qisqacha xulosasi, adabiyotlar ro'yxati, xotima va ilovalardan iborat bo'lib, 60 betda bayon qilingan. Kirish qismi 5 betdan iborat, tushuntirish qismi

35 betdan iborat unda 3 ta rasmdan foydalanildi. Bitiruv malakavi ishida 7 ta adabiyotdan foydalanildi.

I-BOB

JAVA DASTURLASH TILINING SINTAKSISI VA ASOSIY OPERATORLAR.

1.1. JAVA DASTURLASH TILINING SINTAKSISI. MA'LUMOTLAR TIPLARI

JAVA tilida tuziladigan dastur *.java faylida yoziladi – bu shunday faylki u bitta yoki bir nechta sinflardan tashkil topgan bo'ladi. JAVA translyatori shunday taxmin qiladiki: programmaning saqlanish matni kengaytmasi *.java faylda saqlanadi. Tranlyasiya jarayoni natijasida har bir sinfning kodi alohida faylida yozib qo'yiladi. Birinchi o'rinda biz bu bobda "Hello World" programmasini ishga tushiramiz .Bundan keyin esa mantiqiy elementlarni ko'rib chiqamiz. Bob oxirida esa siz JAVA dasturlash tilida mustaqil ishlash haqida yetarlicha ma'lumot olasiz.

Hello World

Mana sizning birinchi JAVA programmangiz:

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

Java tili barcha programma kodi sinflar ichida qayd etilishini talab qiladi. Yuqorida aytib o'tilgan matn misollarni HelloWorld.java da yozib qo'yish lozim. Fayl nomidagi harflarni uning tarkibidagi sinf nomi bilan bir xilligini tekshirish zarur agar sinf nomi public deb e'lon qilingan bo'lsa.

Translyator HelloWorld.class faylini mustaqil misol dagi bayt-kodli prosessordan taqdim etiladi.Olingan kodni ishga tushirish uchun Java tilida bajariladigan vaqtda ega bo'lmoq zarur.Unga bajariladigan yangi sinflarni yuklamoq kerak.Shuni aytib o'tish kerakki bu yerda sinf nomi haqida so'z bormoqda fayl nomi haqida emas.

```
C: > java HelloWorld
```

Hello World

Albatta Hello World misoli oddiy,lekin shunday oddiy misol ham Java tilini endi o'rganmoqchi bo'lganlar uchun qo'rqinchli bo'lishi mumkin,aksincha u sizni ko'plab ma'lumotlar bilan tanishtiradi.Kelinglar diqqat bilan birinchi misolni qatorma-qator ko'rib chiqamiz.

public class HelloWorld {

Bu qatorda class so'zi ishlatilgan u translyatorga yangi sinfni ta'riflashga tayyorligini bildiradi.

Sinfning to'liq ta'rifini birinchi qatordagi figurali qavs va beshinchi qatordagi yopiladigan qavs oralig'ida joylashgan.Figurali qavs Java tilida xuddi C va C++ tilidagidek qo'llaniladi.

public static void main (String args []) {

Bir ko'rishda bu misol murakkab tuyulishi mumkin.Gap shundaki Java tilida global funksiyalar mavjud emas.Bunday misol kitobning birinchi qismining ko'plab betlarida uchratish mumkin, kelinglar ikkinchi qatorning har bir elementini ko'rib chiqamiz.

Public

Qatorni elementlarga ajratayotganda public kalit so'ziga duch kelamiz.Bu kirish modifikatori bo'lib u bilan programist xoxlagan metodini qo'llashi mumkin.Bunday paytda public kirish modifikatori main metodi barcha sinflar uchun qulay.

Static

Keyingi kalit so'z –static. Bu so'z yordamida metodlar e'lon qilinadi.

Static so'z ishlatilgan e'lonlar metodi faqatgina local va statistic misollar bilan ishlashi mumkin.

Void

Siz bu yerda ko'pincha metodlarga muxtojlik sezishingiz mumkin.Bu holatda ekranga faqat qatorni chiqaramiz, main metodini natija qaytarishi shart emas.

Aynan shuning uchun ham void modifikatori ishlatiladi.

Main

Bu metodning nomi . Sinfni ishlatish uchun albatta main metodi bo'lishi kerak.

System. out. prlntln("Hello World!");

Bu qatorda out ob'ektining println metodi bajariladi.Out ob'ekti OutputStream sinfida e'lon qilingan.Figurali qavs to'rtinchi qatordagi main metodi e'lonidan so'ng yopiladi,xuddi shunday qavs beshinchi qatorning Hello World sinfi e'lonidan so'ng qo'llaniladi.

Leksik negiz.

Java tilini minimal darajada o'rganib chiqdik,endi esa orqaga qaytib tilning umumiy strukturasini ko'rib chiqamiz.

Java programmasi – bu probellar,kalit so'zlar,operatorlar,identifikatorlar va bo'luvchilar to'planmasi.

Probellar

Java – bu matn programmalarini formatlashga ruxsat beruvchi til.Programma yaxshi ishlashi uchun hech qanday xususiylik yo'q.Masalan Hello World sinfini ikki qatorda yoki boshqa xoxlagan uisulda yozib olish mumkin.

Komentariyalar

Komentariyalar programma kodi bilan hech qanday aloqasi bo'lmasada u ko'ribchiqiladigan matnning bir bo'lagi hisoblanadi. Komentariylar uch xil ko'rinishda bo'ladi.

1. Bir qatorda komentariy.
2. Bir necha qatorda komentariy.
3. Dokumentlashtirish komentariysi.

Bir qatorni egallovchi komentariylar

// belgisi bilan boshlanib qator oxirida tugaydi.Bunday kommentariylar ko'rinishi alohida qatordagi kodlar uchun joylashtirishgaqulay hisoblanadi.

a = 42; // a o'zgaruvchiga 42 ni ozlashtirish

Aniqroq ko'rib chiqish uchun bir necha qatorda joylashtirilgan komentariylardan foydalanishimiz mumkin.Komentariylar matnini /* belgisi bilan boshlab */ bilan tugatamiz.

```
/*
 Bu kod ozgina o'ylantiruvchi
Tushuntirishga harakat qilaman
  …..
*/
```

Uchinchi asosiy komentariylar formasi ko'proq javadoc programmasiga moslashtirilgan.Komentariylarning bu turi quidagicha:e'londan avval ochiq sinfda joylashtirish uchun /** belgisidan boshlaymiz.Bu komentariy oddiy komentariy singari */ belgisi bilan tugallanadi.

Javadoc programmasi @ belgisi bilan boshlanadigan dokumentlashtiradigan komentariylarni ajrata oladi.

Masalan :

```
/**
*  Bu sinf ajoyib ishlarni qilishga qodir.
*  Maslaxatimiz shundaki bu sinfni ochiqchasiga ko'rib chiqmoqchi bo'lsangiz uni baza sifatida olishingiz mumkin.
*  @see Java. applet. Applet
*  ©author Patrick Naughton
*  @version 1. 2
*/
```

Kalit so'z

Kalit so'z bu – maxsus idenfikator bo'lib, u java tilida turlarni,modnfikatorlarni terish va programmani boshqarishda foydalaniladi.Bu so'zlardan faqat tayinlangandagina foydalanish mumkin.Ulardan indifikatorlar,sinflar va metodlar sifatida foydalanib bo'lmaydi.

1.1-jadval. Kalit so'zlar.

Abstract	boolean	Break	Byte	byvalue
case	cast	Catch	Char	class
const	continue	default	Do	double

else	Extends	false	final	finally
float	For	Future	generic	goto
if	implements	import	inner	instanceof
Int	interface	Long	native	new
null	operator	outer	package	private
protected	Public	Rest	return	short
static	super	switch	syncronized	this
throw	throws	transient	true	try
var	void	volatile	while	
clone	equals	finalize	getClass	hashCode
notify	notifyAll	toString	wait	

Identifikatorlar

Identifikatorlar metodlarni,sinflarni,obetklarni nomlashda foydalaniladi. Identifikator sifatida harflar,sonlar va belgilardan foydalaniladi.

Butun literallar

Butun sonlar – bu ko'pincha oddiy programmalarda uchraydi.Xoxlagan butun soningiz masalan 1,2,3,42 bu butun literallar.Bu misolda o'nlik sonlar ishlatilgan.O'nlik sonlardan tashqari sakkizlik o'n oltilik sonlar ham literallar sifatida qo'llaniladi.O'rtacha o'nlik sonlar noldan boshlana olmaydi,shuning uchun programmalarda nol to'qqiz (09) qo'llaniladi.

Nuqtali literallar.

Nuqtali literallar kasrli o'nlik sonlarni anglatadi.Ularni oddiy formatda yozish mumkin.Oddiy formatda raqam bir qancha o'nlik sonlardan iborat.Sonlardan so'ng o'nlik nuqtasi keyin esa kasr ko'rinishdagi o'nlik sonlar. Masalan 2.0, 3.14159 va 6667.Bu standart formatda yozilgan o'nlik sonlar.

Mantiqiy literallar.

Mantiqiy literalning ikki ma'nosi mavjud:-true (haqiqiy) va false (yolg'on).True va False ning mantiqiy ma'nosi hech qanday raqamli

ko’rinishda namoyon bo’lmaydi. JAVA da true kalit so’zi birga teng emas, false ham 0 ga teng emas.

Belgili literallar.

Javada belgilar - bu UNICODE belgilar jadvalining indeksi hisoblanadi.

Amal	Tavsif
\ddd	Simvolning oldingi nomi (ddd)
\uxxxx	
\’	Ajratish belgisi
\”	Qo’shtirnoq
\\	
\r	
\n	Satrni ko’chirish(line feed, new line)
\f	Sahifani ko’chirish (form feed)
\t	
\b	Orqaga qaytish (backspace)

TIPLAR.

Bu bobda biz Java tilining barcha muhim turlari bilan tanishib olamiz. Qanday ularni o’zlashtirish,turlar aralashmasining ifodalanishini o’rganib chiqamiz.

ODDIY TIPLAR.

Javada sakkizta oddiy tip mavjud – byte,short,int,long,char, float,double va boolean.Ularni to’rt turga ajratish mumkin.

1. Butun.Ularga byte, short, int, long turlarni kiradi.Bular belgili butun sonlar uchun mo’ljallangan.
2. Nuqtali turlarga – float va double tegushli.Ular kasr sonlarni ifodalovchi sonlarni taqdim etadi.
3. Char turi.Bu tur belgilar jadvali elementlarini ifodalovchi tur hisoblanadi.
4. Boolean turi.Bu mantiqiy kattaliklarni tasvirlovchi maxsus tur hisoblanadi.

BUTUN SONLAR.

Java tilida belgisiz sonlar mavjud emas.Bu tilning barcha sonli turlari belgili.byte turi o'n oltili songa teng bo'lsa 0*80 son -1 ga teng bo'ladi.

BYTE

Byte – turi sakkiz bitli belgili tur hisoblanadi.Uning chegarasi -128 dan 127 gacha.U ko'proq setdan yoki fayldan olingan bitlar to'plamini saqlaydi.

```
byte b;
byte c = 0x55;
```

SHORT

Short – bu o'n olti bitli belgili tur.Uning chegarasi -32768 dan 32767 gacha.Bu tur Javada juda kam uchraydi.

```
short s;
short t = Ox55aa;
```

INT

Int turi – bu 32 – bitli belgili butun sonni tasvirlaydi.Bu turning chegarasi -2147483648 dan 2147483647 gacha.Ko'pincha bu tur butun sonlarni asrash uchun foydalaniladi.Bu tur hisoblagichlarni obrabotka qilish jarayoniga juda mos keladi.

Yaqin yillar orasida bu 32 bit 64 bitga to'laqonli javob beradi.

```
int i;
int j = 0x55aa0000;
```

LONG

Long turi 64 bitli butun sonlarni tasvirlaydi.Uning chegarasi yetarlicha ulkan.

```
long m;
long n = Ox55aa000055aa0000;
```

pastda butun sonlarning turlari,ularning chegarasi,razryadi jadval shaklida ko'rsatilgan.

Nomi	Razryadi	Chegarasi

Long	64	-9, 223, 372, 036, 854, 775, 808… 9, 223, 372, 036, 854, 775, 807
Int	32	-2, 147, 483, 648… 2, 147, 483, 647
Short	16	-32, 768...32, 767
Byte	8	-128…127

Nuqtali sonlar kasr sonlar ishlatiladigan misollar uchun kerak bo'ladi. Java da float va double operatorlari bilan ishlash uchun standart ko'rinishida jadval tuzilgan.

Nomi	Razryadi	Chegarasi
double	64	7e-308.. 1. 7e+ 308
float	32	4e-038.. 3. 4e+ 038

Float

Birdaniga aniqlik kiritish uchun float kalit so'zidan foydalaniladi.Ma'no-mazmunini saqlash uchun 32 bit zarur bo'ladi.

```
float f;
float f2 = 3. 14F; // barcha literallar double
```

Double

Double ma'no-mazmunini saqlash uchun 64 bit zarur bo'ladi.

```
double d;
double pi = 3. 14159265358979323846;
```

Tipni keltirish.

Tipni keltirish(type casting) – C++ ning noqulay xossalaridan biridir,biroq bu xossa Java tilida ham saqlanib qolgan.Ba'zan shunaqa vaziyatlar yuzaga keladiki,sizda biror tipdagi kattalik mavjud,biroq uni boshqa tipdagi o'zgaruvchiga ta'minlashingiz kerak bo'ladi.Ba'zi tiplar uchun buni tipni keltirmasdan ham bajarish mumkin bo'lib,bunday hollarda tiplarni avtomatik

almashtirish haqida gapiriladi.Javada avtomatik almashtirish faqat shu holda mumkinki,agar aylantirish natijasida hosil bo'luvchi o'zgaruvchining sonlarni tasvirlash aniqligi boshlang'ich qiymatni saqlash uchun yetarli bo'lsa.Bunday almashtirish M: byte yoki short tipidagi o'xgaruvchi qiymatini yoki o'zgarmasni int tipiga o'tkazishda yuz beradi.Bu kengayish (widening) yoki oshirish(promotion) deb nomlanadi,sababi kichikroq razryaddagi tip kattaroq sig'imli tipgacha kengayadi. Int tipining o'lchami byte tipi diapazonidagi sonlarni sig'dirish uchun doim yetarlidir.Kattalikni muayyan tipga keltirish uchun shu kattalikdan oldin qavs ichiga olingan tip nomini yozish kerak.Quyida keltirilgan kod parchasida manba tipi (int tipidagi o'zgaruvchi)ni voris tipiga(byte tipidagi o'zgaruvchiga) keltirish namoyish etiladi.Agar bu amal chog'ida butun qiymat byte uchun mumkin bo'lgan diapazon chegarasidan oshib ketganda edi,u byte diapazoniga modul bo'yicha bo'lish orqali kuchaytirilardi.(modul bo'yicha songa bo'lish natijasi – bu shu songa bo'lishda olingan qoldiq)

```
int a = 100;
byte b = (byte) a;
```

Ifodalarda tipning avtomatik almashtirilishi

Ifodaning qiymatini hisoblayotgan chog'ingizda oraliq natijalarni saqlash uchun talab etiladigan aniqlik yakuniy natijani ifodalash uchun talab etiluvchi aniqlikdan katta bo'lishi kerak.

```
byte a = 40;
byte b = 50;
byte c = 100;
int d = a* b / c;
```

(a*b) oraliq ifoda qiymati byte uchun ruxsat berilgan diapazondan oshib ketishi mumkin.Shu sababli Java avtomatik ravishda ifodaning har ikki tarafining tipini int tipigacha shunday oshirdiki,(a*b) qiymati uchun joy yetarli bo'ladi.

Avtomatik tip almashtirish ba'zan kutilmagan xatolar haqida transyatorning xabar berishiga olib keladi. M: Quyida keltirilgan kod korrekt

ko’rinsada,translyatsiya fazasida xatolik haqida xabar beradi.Unda biz 50*2 qiymatni byte ozgaruvchiga joylashtirmoqchimiz 100 – byte tipida juda yaxshi joylashadi.Biroq natija tipining int ga avtomatik almashtirilishi natijasida translyatordan xatolik haqida xabarga ega bo’lamiz – sababi int ni byte ga ta’minlash chog’ida aniqlik yo’qotilishi mumkin.

```
byte b = 50;
b = b* 2;
```

^ Incompatible type for =. Explicit cast needed to convert int to byte. (= uchun mos kelmaydigan tiplik int ni byte ga oshkor keltirish zarur)

To’g’rilangan matn :

```
byte b = 50;
 b = (byte) (b* 2);
```

natijasida b ga 100 qiymat to’g’ri ta’minlanishiga olib keladi.

Agar ifodada byte,short va int tipidagi o’zgaruvchilar ishlatilsa,u holda to’lib ketmaslik uchun butun ifoda tipi avtomatik int tipiga o’tkaziladi.Agar ifodada hech bo’lmasa bitta o’zgaruvchii tipi long bo’lsa,butun ifoda tipi ham longgacha oshiriladi.Oxirida L (yoki l) belgisi bo’lmagan butun literallar (o’zgaruvchilar) ega bo’ladi.Agar ifodada float tipidagi o’zgaruvchilar qatnashsa,ifoda tipi floatga o’tkaziladi.Agar ifodada hech bo’lmasa bitta oprant double tipida bo’lsa,butun ifoda doublega keltiriladi .Jimlik holatida Java barcha qo’zg’aluvchan nuqtali literallarni double tipiga ega deb qabul qilinadi.Quyida keltirilgan dastur ifodadagi har bir kattalikning tipi har birar operatorning ikkinchi operandi bilan mos bo’lishi uchun qanday qilib oshirilishi ko’rsatilgan.

```
 class Promote {
public static void main (String args []) {
byte b = 42;
char c = 'a';
short s = 1024;
int i = 50000;
```

```
float f = 5.67f;
double d =.1234;
double result = (f* b) + (i/ c) - (d* s);
System. out. println ((f* b)+ "+ "+ (i / c)+ " - " + (d* s));
System. out. println ("result = "+ result);
}}
```

Belgilar.

Char tipi uchun ikki bayt xotira ajratiladi va u ixtiyoriy simvolni saqlaydi. Boshqacha qilib aytganda shu simvolning Unicode dagi kodini saqlaydi.Ammo bu qiymat bilan ixtiyoriy arifmetik operasiyalarni bajarish mumkin.Agar operasiyaning natijasi yana char tipida saqlansa u Unicode dagi qanqaydir kod deb tasavvur qilinadi.

Javada belgilarning tasviri uchun qatorda Unicode kodi ishlatilsa char tipining razryadi bu tilda 16-bitga teng.Unda o'n minglab belgilarni saqlash mumkin.Char turining chegarasi 0…65535.

Unicode bu o'nlab belgilar jamlanmasi.Uning tarkibiga – lotin,grek,arab alfaviti,kril alifbosi va yana ko'plab belgilar to'plami kiradi.

```
char c;
char c2 = Oxf132;
 char c3 = ' a';
 char c4 = '\n';
```

Pastda keltirilgan kodda biz bazali belgilarga butun sonlarni qo'shamiz.

```
int three = 3;
char one = '1';
 char four = (char) (three+ one);
```

Boolean tipi

Java tilida Boolean nomli mantiqiy ma'nolarni anglatuvchi oddiy tur mavjud.Bu tur ikki ma'noni anglatadi.

True (Haqiqat) va False (Yolg'on)

```
boolean done = false;
```

Oddiy tip deb shunday tipga aytiladiki uning qiymati bo'laklarga bo'linmaydi(ya'ni bo'lish ma'nosiz) bunday tiplar boshqa tiplarga asoslanmaydi.

Biz oddiy tiplarni barchasi bilan tanishib chiqdik,jumladan:butun sonlar, belgilar va mantiqiy ma'nolar.Yaxshisi barcha ma'lumotlarni bir yerga to'playmiz.

```
class SimpleTypes {
public static void main(String args []) {
 byte b = 0x55;
short s = 0x55ff;
 int i = 1000000;
long l = 0xffffffffL;
 char c = ' a' ;
 float f = .25f; double d = .00001234;
boolean bool = true;
System.out.println("byte b = " + b);
 System.out.println("short s = " +s);
 System.out.println("int i = " + i);
System.out.println("long l = " + l);
System.out.println("char c = " + c);
System.out.println("float f = " + f);
System.out.println("double d = " + d);
System.out.println("boolean bool = " + bool);
 }}
```

Bu programmani kiritgandan so'ng quyidagi natijani olamiz:

```
C: \> java SimpleTypes
 byte b = 85
 short s = 22015
int i = 1000000
 long l = 4294967295
```

char c = a

float f = 0.25 double

d = 1.234e-005

boolean bool = true

E’tibor bersangiz ayrim butun sonlar o’nlik sanoq tizimida namoyon bo’ladi,lekin biz ayrim sonlarni o’ oltilik sanoq tizimida kiritganmiz.

Massivlar.

Massiv tipi deb shunday murakkab tipga aytiladiki uning quymati bir nechta boshqa tipdagi qiymatlardan tashkil topgan ularning soni chekli hammasi bitta tipga tegishli va tartiblashtirilgan (nomerlangan).Bunday qiymatlar massivning elementlari deyiladi.Har bir element o’zining tartib nomeriga ega va bu nomer elementning massivdagi indeksi deyiladi.

Massiv turlarini e’lon qilishda to’rtburchak qavslardan foydalaniladi. Pastda ko’rsatilgan misolda month_days e’loni va uning int turning butun sonlar massivi turi keltirilgan.

int month_days [ ];

Demak month_days bu butun o’n ikkilik sanoq sistemasi sonlar jo’natmasi.

month_days = new int [12];

Pastda keltirilgan misolda yil oylarining tarkibidagi sonlarning elementlari massivi tuzilgan.

```
class Array { public static void main (String args []) {
 int month_days[];
month_days = new int[12];
 month_days[0] = 31;
 month_days[1] = 28;
month_days[2] = 31;
 month_days[3] = 30;
 month_days[4] = 31;
 month_days[5] = 30;
```

```
month_days[6] = 31;
 month_days[7] = 31;
 month_days[8] = 30;
 month_days[9] = 31;
 month_days[10] = 30;
 month_days[11] = 31;
System.out.println("April has " + month_days[3] + " days.");
} }
```

Zapuskdan so'ng programma aprel kunlari pastda ko'rsatilgan holda namoyon bo'ladi.

```
C: \> java Array
 April has 30 days.
```

Massivning inisiamuatori vergullar bilan ajratilgan iboralar ro'yxatini tashkil etadi va ifoda oxirida figurali qavs qo'yiladi.Vergullar massiv elementlarini bir – biridan ajratib turadi.Bunday paytda massiv qancha ma'no saqlashi kerak bo'lsa shuncha element ham saqlashi zarur bo'ladi.

```
class AutoArray {
 public static void main(String args[]) {
 int month_days[] = {31,28,31,30,31,30,31,31,30,31,30,31};
System.out.println("April has " + month_days[3] + " days.");
} }
```

Ko'p o'lchovli massiv.

Haqiqatdan ham Javada Haqiqiy ko'p misolli massiv mavjud emas.Lekin massivlarning massivi bor.Pastda ko'rsatib o'tilgan kodda double turi an'anaviy o'n olti elementdan tuzilgan matrissani tasvirlaydi.Bu matrissaning ichki realizasiyasi – double massivlarining massivi.

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

Keyingi kodning ko'rinishi ham xuddi shunday xotirani inisiallashtiradi.Bu tasvir matrissa haqiqatdan ham o'z ichiga massivlarni olganligini bildiradi.

```
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 };
```

Keyingi misolda double turi elementli to’rtga to’rt hajmli matrissa tuziladi,uning diagonal elementi birliklar bilan to’ldiriladi qolgan elementlar esa nolga teng bo’ladi.

```
class Matrix {
 public static void main(String args[]) { double m[][];
m = new double[4][4];
m[0][0] = 1;
 m[1][1] = 1;
 m[2][2] = 1;
m[3][3] = 1;
System.out.println(m[0][0] +" "+ m[0][1] +" "+ m[0][2] +" "+ m[0][3]);
System.out.println(m[1][0] +" "+ m[1][1] +" "+ m[1][2] +" "+ m[1][3]);
System.out.println(m[2][0] +" "+ m[2][1] +" "+ m[2][2] +" "+ m[2][3]);
System.out.println(m[3][0] +" "+ m[3][1] +" "+ m[3][2] +" "+ m[3][3]);
 }
}
```

Bu programmani kiritib quyidagi natijani olamiz.

```
C : \> Java Matrix
 1 0 0 0
 0 1 0 0
 0 0 1 0
 0 0 0 1
```

E’tibor bering – agar siz element nolga teng bo’lishini xoxlasangiz,uni inisiallashtirishingiz shart emas,bu avtomatik tarzda sodir bo’ladi.

Pastda ko’rsatilgan misolda ishlangan qator nomeri va ustun nomeri ifoda etilgan matrissa elementlari ko’rsatilgan.E’tiboringizni qarating massiv inisiali ichida nafaqat literallarni balki ifodalarni ham ishlatish mumkin.

```
class AutoMatrix {
public static void main(String args[]) { double m[][] = {
{ 0*0, 1*0, 2*0, 3*0 }, { 0*1, 1*1, 2*1, 3*1 },
 { 0*2. 1*2, 2*2, 3*2 }, { 0*3, 1*3. 2*3, 3*3 } }:
 System.out.println(m[0][0] +" "+ m[0][1] +" "+ m[0][2] +" "+ m[0][3]);
System.out.println(m[1][0] +" "+m[1][1] +" "+ m[1][2] +" "+ m[1][3]);
System.out.println(m[2][0] +" "+m[2][1] +" "+ m[2][2] +" "+ m[2][3]);
System.out.println(m[3][0] +" "+m[3][1] +" "+ m[3][2] +" "+ m[3][3]); } }
```

Bu programmani joylashtirib quyidagi natijani olasiz.

```
C: \> Java AutoMatrix
0 0 0 0
0 1 2 3
0 2 4 6
0 3 6 9
```

Endi siz Java tilidagi sakkiz oddiy tur bilan qanday ishlashni bilib oldingiz. Siz bu turlarning ob'ektlarini tuzishni va ularning har birining razryadini ko'rib chiqdingiz. Siz ularning qanday arifmetik ayirmalarga to'g'ri kelishini bilib oldingiz.

1.2. JAVA DASTURLASH TILIDA ASOSIY OPERATORLAR VA STANDART FUNKSIYALAR

Operatorlar bu – bir yoki ikkita argument ustidagi harakat demakdir. Sintaksik operatorlar indifikator va literallar orasida ko'proq joylashtiriladi.Ularning ro'yxati uchinchi jadvalda ko'rsatilgan.

+	+=	-	-=
*	*=	/	/=
|	|=	^	^=
&	&=	%	%=
>	>=	<	<=
!	!=	++	--

>>	>>=	<<	<<=
>>>	>>>=	&&	||
==	=	~	?:
	Instanceof	[ ]	

Bo’luvchilar (Ajratuvchilar)

Faqat bir nechta belgilar guruhi Java programmasida nomsiz qolgan.Bu programma kodiga tashqi tomondan ta’sir qiluvchi oddiy bo’luvchilar.

Belgi	**Nomi**	**Qayerlarda ishlatiladi**
( )	Aylana qavs	Metod chaqirishda va e’lonlardagi parametrlar ro’yxatini ajratib turadi.Shu jumladan operasiya topshiriqlarida ishlatiladi.
{ }	Figurali qavs	Sinflardagi kod blokini chegaralab turadi.Metod local miqyosida qo’llaniladi
[ ]	Kvadrat qavs	E’londan massivlarda ko’proq qo’llaniladi.Aloxida elementlar massivi ham shu jumladandir.
;	Nuqtali vergul	Operatorlarni bo’lmaydi. (ajratadi)
,	Vergul	Indikatorlarni ajratib operatorlarni bog’lash vazifasini o’taydi
.	nuqta	Sinf nomlarini va paket nomlarini ajratadi.

Operatorlar.

Java tilidagi operatorlar – bu maxsus belgilar bo’lib,translyatorga sizning ayrim operasiya o’tkazuvchilar bilan operasiya o’tkazmoqchi ekanligingiz haqida ma’lumot beradi.Ayrim operatorlar bitta operasiya o’tkazuvchi bo’lishini talab etadi.

Ular unarli deb nomlanadi.

Operatorlar operasiya qiluvchilar oldida qo’yilsa ular prefiksli deb nomlanadi,keyin qo’yilsa esa postfiksli operatorlar deyiladi.Ko’pchilik operatorlarni ikki operasiya o’tkazuvchilar orasiga qo’yadi,ular infiksli operatorlar deb nomlanadi.

Javada 44 ta operatorlar mavjud.Ularni 4 sinfga ajratish mumkin. Arifmetik,bitli,tenglashtiruvchi operatorlar va mantiqiy operatorlar.

Arifmetik operatorlar.

Arifmetik operatorlar xuddi algebradagidek ayrim vazifasini bajaradi. Kirish huquqiga ega bo’lgan operasiya o’tkazuvchilar raqamli turlarga ega bo’lmoqlari lozim.Masalan bu operatorlar mantiqiy tiplar bilan ishlay olmaydi,char tipi bilan esa ishlay oladi.

OPERATOR	NATIJA	OPERATOR	NATIJA
+	Qo’shish	+ =	Qoshish bilan o’zlashtirish
-	Ayirish	-=	Ayirish bilan o’zlashtirish
*	Ko’paytirish	*=	Ko’paytirish bilan o’zlashtirish
/	Bo’lish	/=	Bo’lish bilan o’zlashtirish
%	Modul bo’yicha bo’lish	%=	Modul bo’yicha bo’lish bilan o’zlashtirish
++	Inkrement	--	Dekrement

TO’RT ARIFMETIK HARAKAT.

Pastda opereatorlarning ustida ish olib boorish oddiy programmasi keltirilgan.

```
class BasicMath  { public static void int a = 1 + 1;
 int  b = a *  3;
 main(String args[]) {
 int c = b / 4;
int d = b - a;
 int e = -d;
System.out.println("a = "  +  a);
System.out.println("b = "  +  b);
 System.out.println("c = "  +  c);
System.out.println("d = "  +  d);
System.out.println("e = "  +  e);
 } }
```

Bu programmani ishga tushirgandan so'ng siz quyidagi natijani olishingiz kerak.

```
C: \> java BasicMath
a = 2    b = 6    c = 1
d = 4    e = -4
```

MODUL BO'YICHA AJRATUVCHI OPERATORLAR.

Modul bo'yicha ajratuvchi operatorlar yoki mod operatori % belgisi bilan ajratiladi.Bu operator birinchi operatsiyada ajratishdan qolgan qoldiqni ikkinchi operatsiyaga qaytarish vazifasini bajaradi.C++ funksiyasidan farqli o'laroq bu programma faqat butun sonlar bilan cheklanib qolmaydi.Pastda ko'rsatilgan programma shu operatorni tasvirlaydi.

```
class Modulus {
 public static void main (String args []) {
 int x = 42;
double y = 42.3;
```

```
System.out.println("x mod 10 = " + x % 10);
System.out.println("y mod 10 = " + y % 10);} }
```

Bu programmani ishga tushirgandan so’ng siz quyidagi natijani olishingiz kerak.

```
C:\> Modulus
 x mod 10 = 2
 y mod 10 = 2.3
```

O’ZLASHTIRISH ARIFMETIK OPERATORLARI.

Har bir arifmetik operatorning operatsiya yaratish jarayonida bir vaqtda o’zlashtirish formasi mavjud.

Pastda shu operatorning tasvirlash jarayoni keltirib o’tilgan.

```
class OpEquals {
public static void main(String args[]) {
 int a = 1; int b = 2;
 int c = 3;
a += 5;
b *= 4;
c += a * b;
c %= 6;
System.out.println("a = " + a);
System.out.println("b = " + b);
 System.out.println("c = " + c);
} }
```

Mana bu esa shu programmani natijasi hisoblanadi.

```
C:> Java OpEquals
a = 6
b = 8
c = 3
```

INKREMENT VA DEKREMENT.

Increment va decrement nomli ikki operator turi mavjud.Bular operatsiya birligidagi qo’shuv va ayiruvning qisqartirilgan varianti hisoblanadi.Bu operatorlarning qulay tomoni shundaki prifiks va postfiks formasida ham qo’llab bo’ladi.Keyingi misolda increment va decrement operatorlarining qo’llanishi ko’rsatilgan.

```
class IncDec {
 public static void main(String args[]) {
 int a = 1;
 int b = 2;
int c = ++b;
int d = a++;
c++;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
} }
```

Programmadan olingan natija:

```
C:\ java IncDec
a = 2
b = 3
c = 4
d = 1
```

BUTUN SONLI BUL OPERATORLARI.

long, int, short, char va byte - butun sonlar turlari uchun qo’shimcha operatorlar turi ajratilgan.Ular bilan alohida bitlarning holatini tekshirib turish mumkin.Jadvalda shunday operatorlar ko’rsatilgan.Arifmetik bitlar operatori har bir bit bilan mustaqil kattalikda ishlashi mumkin.

OPERATOR	NATIJA	OPERATOR	NATIJA

~	Bitli inkor qilish (NOT)		
&	Bitli va (AND)	&=	Bitli va (AND) o'zlashtirish bilan
|	Bitli yoki (OR)	|=	Bitli yoki (OR) o'zlashtirish bilan
^	Bitli o'chirish yoki (XOR)	^=	Bitli o'chirish yoki (XOR) o'zlashtirish bilan
>>	O'ngga yurmoq	>>=	O'ngga yurmoq o'zlashtirish bilan
>>>	Nollar bilan to'ldirilgan o'ngga yurish	>>>=	Nollar bilan to'ldirilgan o'ngga yurish o'zlashtirish bilan
<<	Chapga yurish	<<=	Chapga yurish o'zlashtirish bilan

MANIPULYAR BITLAR BILAN PROGRAMMALARGA MISOLLAR.

Pastda ko'rsatilgan jadvalda har bir bitdagi operatorning o'z operandlari kombinatsiyasida sodir bo'lishi mumkin bo'lgan ishtiroki ko'rsatilgan. Jadvaldan keyin keltirilgan misol bu operatorning Java tilidagi dasturda ishlatilishini ko'rsatadi.

A	B	OR	AND	XOR	NOT A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

```
class Bitlogic {
```

```
public static void main(String args []) {
String binary[] = { "OOOO","0001","0010","0011","0100","0101",
 "0110","0111",   "1000","1001","1010","1011","1100","1101","1110","1111"};
int a = 3;  // 0+2+1 или двоичное 0011
int b = 6;  // 4+2+0 или двоичное 0110
int c = a | b;
int d = a & b;
int e = a ^ b;
int f = (~a & b) | (a & ~b);
int g = ~a & 0x0f;
System.out.println(" a = " + binary[a]);
System.out.println(" b = " + binary[b]);
System.out.println(" ab = " + binary[c]);
System.out.println(" a&b = " + binary[d]);
System.out.println(" a^b = " + binary[e]);
System.out.println("~a&b|a^~Ь = " + binary[f]);
System.out.println(" ~a =  " + binary[g]);
} }
```

Mana bu esa shu programmani natijasi hisoblanadi:

```
C: \> Java BitLogic
a = 0011
b = 0110
a | b = 0111
a & b = 0010
a ^ b = 0101
 ~a & b | a & ~b = 0101
~a = 1100
```

O’NGGA VA CHAPGA YO’NALISH.

<< operatori o’z bitlaridachapga yurishni ta’minlaydi.Bu holda chap razryaddagi bitlar bir qismi chegaradan chiqadi va yo’qoladi.Shundan keyin o’ng pozisiyalar nollar bilan to’ldiriladi.

>> operatori Java tilida o’ngga yo’nalishni bildiradi.U chap operandning bitlarini o’ng tomonga joylashtiradi.Chap operandning bitlari o’ngdagi so’zlari oxirigacha borsa o’z-o’zidan yo’qoladi.Siljish paytida o’ngdagi razriyadlar o’rnini avvalgi belgilar razriyadi egallaydi.Buni belgilar razriyadining kengaytirilishi deb aytiladi.

```
class HexByte {
static public void main(String args[]) {
charhex[]= {'0','1,'2','3','4','5','6','7','8','9','a','b','c','d','e','f};
byte b = (byte) 0xf1;
System.out.println(“b = 0x” + hex[(b >> 4) & 0x0f] +  hex[b & 0x0f]); } }
```

Pastda shu programmani natijasi ko’rsatilgan.

```
C:\> java HexByte
b = 0xf1
```

BELGISIZ O’NGGA SILJISH.

Ko’pincha o’ngga siljishda belgilar razriyadining kengaytirilishiga yo’l qo’ymasdan chapdagi razriyadlar bo’shatilib o’rni nollar bilan to’ldirilishi talab etiladi.

```
class ByteUShift  {
static public void main(String args[]) {
char hex[] ={'0','1’,'2','3','4','5','6','7','8','9','a','b','c','d','e','f’};
byte b = (byte) 0xf1;
byte c = (byte) (b >> 4);
byte d = (byte) (b >> 4);
byte e = (byte) ((b & 0xff) >> 4);
System.out.println(" b = 0x" + hex(b >> 4) & 0x0f] + hex[b & 0x0f]);
System.out.println(“ b >> 4 = 0x" + hex[(c >> 4) & 0x0f] + hex[c & 0x0f]);
System.out.println(“b >>> 4 = 0x" + hex[(d >> 4) & 0x0f] + hex[d & 0x0f]);
```

System.out.println(“(b&0xff)>>4=0x" + hex[(e >> 4) & 0x0f] + hex[e & 0x0f]);

} }

Bu misol uchun b o’zgaruvchili erkin musbat sonni ko’rsatish mumkin,bunda biz 0xfl o’n oltilik sondan foydalandik.O’zgaruvchi natijada b belgili harakat bilan o’zlashtiriladi to’rt razryad o’ngga. Kutilganidek belgili razryadning kengayishi 0xfl ni 0xFF ga oylantirilishiga olib keladi.So’ngra d o’zgaruvchi belgisiz b harakatini to’rt razryad o’ngga suradi.d natijasida 0xof natijasida yana 0xff ni olishimizni ham kiritish mumkin.Bu – belgili razryadni kengayishi natijasi b o’zgarganda ,avtomatik bajarilganda int o’ngga harakatlanish oldidan bajariladi. Natijada e o’zgaruvchisi uchun kerakli natijaga erishishimiz mumkin – ahamiyat 0xof.Buning uchun o’ngga harakatlanishdan oldin b o’zgaruvchini og’zaki ko’paytirish 0xff maskasiga kerak.

Bunday yo’l bilan tozalash katta razryadlarni avtomatik ko’paytirish tipi bajariladi.

Buning natijasida belgisiz o’ngga harakat qilishga zarurat qolmaydi,chunki biz belgili bitning AND operasiyasidan keyingi holatni bilamiz.

C: \> java ByteUShift

---

b = 0xf1

b >> 4 = 0xff

b >>> 4 = 0xff

b & 0xff) >> 4 = 0x0f

O’ZLASHTIRISHNING BULLI OPERATORLARI.

Arifmetik operatorlar kabi barcha bitli operatorlarga yaqin formalari mavjud.U avtomatik tarzda chap operandning operasiya natijalarini o’zlashtiradi.

Keyingi misolda bir nechta butun sonlar ko’rsatilgan,ular bilan operatorlar yordamida har xil operasiyalar o’tkazish mumkin.

class OpBitEquals {

public static void main(String args[]) {

```
int a = 1;
 int b = 2;
int c = 3;
a |= 4;
b >>= 1;
c <<= 1;
a ^= c;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
} }
```

Programma natijasi quyidagicha:

```
C:\> Java OpBitEquals
a = 3
b = 1
c = 6
```

MUNOSABAT OPERATORLARI.

Ikki mohiyatni tenglashtirish uchun Java da munosabat va tenglikni tasvirlovchi operatorlar to'plami mavjud.Bu operatorlar ro'yxati jadvalda ko'rsatilgan.

OPERATORLAR	NATIJA
= =	Tenglik
! =	Teng emas
>	Katta
<	Kichik
> =	Katta yoki teng
< =	Kichik yoki teng

Istalgan tipning ahamiyati shundaki,butun va kasr sonlar,belgilar,silkalarni tekshirish operatorni qo'llab tenglikka (= =) va tengsizlikka (!=) tenglashtirish mumkin.

Java tilida C va C++ dagidek tenglikni tekshirish (= =) ketma ketligida tushunilishiga diqqat qiling.Bitta (=) belgisi bu o’zlashtirish operatori.

MANTIQIY OPERATORLAR.

Mantiqiy operatorlar pastda keltirilgan jadvalda ular haqidagi hisobotlar ,Boolean tipidagi operandlar bilangina kesishadi.

Operator	Natija	Operator	Natija
&		&=	
|		=	
^		^=	
||		= =	
&&		!=	
!		?:	

Mantiqiy operatorlar harakati natijasida natijasida har xil kombinatsiya natijasida operandlar ahamiyati jadvalda ko’rsatilgan.

A	B	OR	AND	XOR	NOT A
false	False	False	False	false	true
True	False	True	False	true	false
false	True	True	False	true	true
True	True	True	True	false	false

Pastda ko’rsatilgan programma bizga tanish bo’lgan BitLogik misolini amaliyotda to’liq takrorlaydi.Faqat bus afar biz logic bulevalar ahamiyati bilan ishlaymiz.

```
class BoolLogic {
public static void main(String args[]) {
boolean a = true;
boolean b = false;
boolean c = a | b;
boolean d = a & b;
boolean e = a ^ b;
boolean f = (!a & b) | (a & !b);
```

```
boolean g = !a;
System.out.println(" a = " + a);
System.out.println(" b = " +  b);
System.out.println(" a|b = " + c);
System.out.println(" a&b = " + d);
System.out.println(" a^b = " + e);
System.out.println("!a&b|a&!b = " + f);
System.out.println(" !a = " + g);
} }
C: \> Java BoolLogic
a = true
b = false
a|b = true
a&b = false
a^b = true
!a&b|a&!b = true
!a = false
```

MANTIQIY IFODANI TEZKOR BAHOLASH OPERATORLARI.

(short circuit logical operators)

Mantiqiy operatorlar to'plamiga ikki qiziqarli qo'shimcha mavjud.Bu mantiqiy ifodaning tezkor baholashida xizmat qiluvchi AND va OR operatorlarining al'ternativ taklifi hisoblanadi.Biz bilamizki agar birinchi OR operatorining operandi true ahamiyatiga ega bo'lsa,unda ikkinchi operatordan mustaqil ravishda natijasida true kattaligida bo'ladi.AND operatorida birinchi operand false bo'lsa u ikkinchi operandning natijasiga hech qanday ta'sir o'tkazmaydi,u doim false ga teng bo'ladi.

Agar siz && va // operatorlari o'rniga & va / formalarini qo'llasangiz, unda Java da o'ng operandning mantiqiy ifodasining bahosini ishlab chiqarmaydi.

TERNARLI OPERATORLAR. if-then-else

if-then-else operatorning umumiy ko’rinishi.

Ifoda 1: Ifoda 2: Ifoda 3:

Birinchi operand sifatida – ifoda 1 – bunda xoxlagan ifoda qo’llanilishi mumkin,uning natijasi Boolean turi hisoblanadi.Agar natija true ga teng bo’lsa unda ikkinchi operand tuzgan operator bajariladi bu esa ikkinchi ifoda deb aytiladi.Agar birinchi,operand false gat eng bo’lsa unda uchinchi operand – uchinchi ifoda – bajariladi.

Ikkinchi va uchinchi ya’ni ikkinchi va uchinchi ifodalar bir turning ahamiyatini qaytarishlari va void turiga ega bo’lmasliklari kerak.

```
class Ternary {
public static void main(String args[]) {
int a = 42;
int b = 2;
int c = 99;
int d = 0;
int e = (b == 0) ? 0 : (a / b);
int f = (d == 0) ? 0 : (c / d);
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
System.out.println("a / b = " + e);
System.out.println("c / d = " + f);
} }
```

Bu programmani ishlatib quyidagi natijani olasiz:

```
C: \>java Ternary
a = 42          b = 2
c = 99          d = 0
a / b = 21      c / d  = 0
```

OPERATORLAR PRIORITETI.

Java da aniq tartib yoki operatsiya prioriteti harakat qiladi.Elementar algebrada bizni ko’paytirish va bo’lish,qo’shuv va ayiruvdan prioriteti yuqori deb o’rgatishgan.Programmalashtirishda ham operasiya prioritetini kuzatib bormoq lozim.Jadvalda Java tili operasiyasidagi barcha prioritetlar tartibi ko’rsatib o’tilgan.

( )	[ ]	.	
~	!		
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	Op=		

Jadvalning birinchi qatorida biz hali uchratmagan uchta nooddiy operatorlarga duch keldik.Aylana qavslar () prioritetlarni yaqqol tasvirlashda qo’llaniladi.Avvalgi boblarda ko’rib chiqqanimizdek to’rtburchak qavslar massivlarni indeksirolavashda ishlatiladi.

ANIQ PRIORITETLAR.

Yuqori prioritetlar aylana qavslarga ega.Siz ifodaga bir nechta juft qavslar qo’shishingiz mumkin.

a >> b + 3

Bu misollarning qay biri birinchi qator bilan o’xshash ?

a >> (b + 3) va (a >> b) + 3.

Qo’shuv operatorining prioriteti,siljish operatoridan yuqori,shuning uchun to’g’ri javob a>> (b + a)

Demak biz Java tili operatorlarining barcha turini ko’rib chiqdik.Endi siz berilgan turli xil tiplarni tasavvur qilishingiz mumkin.Navbatdagi bo’limda vetvleniya konstruksiyasi bilan tanishamiz,programmalarni bajarilishini boshqarishni o’rganamiz.

1.3. JAVA DASTURLASH TILIDA BOSHQARISH OPERATORLARI

Ular sizga tanish,lekin ularni Javada ko’rib chiqmoq lozim.

if-else

Umimiylashtirilgan formada operator quyidagicha yoziladi.

if (mantiqiy ifoda) оператор1; [else оператор2;]

else ni ajratish shart emas.Operatorlar o’rnida qo’shimcha operatorlar turishi mumkin.Ular figurali qavs bilan tugallanadi.Mantiqiy ifoda bu xoxlagan ifoda bo’lib Boolean turi mohiyatini qaytaradi.

```
int bytesAvailable;
// ...
if (bytesAvailable > 0) {
ProcessData();
bytesAvailable -= n;
 }  else
waitForMoreData();
```

Bu programmada esa if – else operatori qaysi yilning qaysi oyida qo’llanilganini anglatadi.

```
class IfElse {
public static void main(String args[]) {
int month = 4;
String season;
if (month == 12 || month == 1 || month == 2) {
season = "Winter";
} else if (month ==3 || month == 4 || month == 5) {
```

```
season = "Spring";
} else if (month == 6 || month == 7 || month == 8) {
season = "Summer";
} else if (month == 9 || month == 10 || month == 11) {
season = "Autumn";
} else {
season = "Bogus Month";
}
System.out.println( "April is in the " + season + ".");
} }
```

Bu programmani bajarilgandan so'ng quyidagi natijani olasiz:

```
C: \> java IfElse
April is in the Spring.
```

break

Java tilida goto operatori mavjud emas.Shuning uchun ham ayrim hollarda gotoni,break operatori o'rinbosarlik vazifasini bajaradi.Bu operator blokni ishlashi to'xtatilib boshqarishni operatorga berishi haqida ma'lumot beradi.Bloklarni nomlash uchun Javada belgilar mavjud. Break operatori sikllar bilan ishlash jarayonida va switch operatorlarida belgisiz ishlasa ham bo'ladi.

Masalan:bu programmaga uchta blok kritilgan,har birini o'zining belgisi mavjud.Ichkari blokda turgan break operatori blokidan so'ng turgan operatorni o'tishini chaqiradi.Bunda ikkita println operatori tushurib qoldiriladi.

```
class Break {
public static void main(String args[]) {
boolean t = true;
a:      {
b:        {
c:            {
```

```
System.out.println("Before the break"); // Перед break if (t) break b;
System.out.println("This won't execute"); //Не будет выполнено }
System.out.println("This won't execute"); // Не будет выполнено }
System.out.println("This is after b"); //После b
} } }
```

Programmadan so’ng siz quyidagi natijani olasiz:

```
C:\> Java Break
Before the break
This is after b
```

switch

switch operatori programma kodining har xil qismlariorasidagi o’tish tasvirlanadi.Bu operatotning umumiy ko’rinishi quyidagicha:

```
switch ( ifoda ) {
case mohiyat1:
break;
case mohiyat 2:
break;
case mohiyat bilan:
break;
default:
}
```

Ifodani ayirish natijasi biror bir oddiy turning mohiyati bo’lishi mumkin,bundan tashqari har bir mohiyat case operatorida ko’rsatib o’tilgan.Buu barcha mohiyatlar literallar bo’lishi kerak.Agarda siz ikkita case operatorida bir xil mohiyatni kiritsangiz,translyator sizga,xato qilganingiz haqida xabar beradiyoki ifodaning mohiyati biror bir case operatori bilan o’xshamasa,boshqaruv kodga topshiriladi,kod esa default kalit so’zidan so’ng joylashgan.Shuni aytib o’tish joizki default operatori unchalik ham shart emas.Agar biror bir case operatori ifoda mohiyati bilan o’xshamasa va switch ga default operatori bo’lmasa unda programma switch operatoridan keyingi operatorga topshiriladi.Switch operatori

ichidagi belgisiz break boshqaruvni kodga berishga olib keladi.Agar break bo’lmasa case bo’limidan keyingisi bajariladi.Gohida case bo’limida bir nechta switch operatoriga ega bo’lish qulay.

```
class SwitchSeason {
public static void main(String args[]) {
int month = 4;
String season;
switch (month) {
case 12: // FALLSTHROUGH
case 1: // FALLSTHROUGH
case 2:
season = "Winter";
break;
case 3: // FALLSTHROUGH
case 4: // FALLSTHROUGH
case 5:
season = "Spring";
break;
case 6: // FALLSTHROUGH
case 7: // FALLSTHROUGH
case 8:
season = "Summer";
break; case 9: // FALLSTHROUGH
case 10: // FALLSTHROUGH
case 11:
season = "Autumn";
break;
default:
season = "Bogus Month";
}
```

```
System.out.println("April is in the " + season  +  ".");
} }
```

Pastda esa yanada foydali misol keltirilgan.Bu erda switch operatorining kirish qatorida har xil belgili kodlar boshqarishni olib boradi.Programma satrlar soni,belgilarni hisoblaydi.

```
class WordCount {
static String text = "Now is the tifne\n" +
  "for all good men\n" +
     "to come to the aid\n" +
"of their country\n"+
           "and pay their due taxes\n";
static int len = text.length();
public static void main(String args[]) {
boolean inWord = false;
int numChars = 0;
int numWords = 0;
 int numLines = 0;
for (int i=0; i < len; i++) {
char c = text.charAt(i);
numChars++;
   switch (c) {
case '\n': numLines++; // FALLSTHROUGH
 case '\t': // FALLSTHROUGH
case ' ' : if (inWord) {
 numWords++;
inWord = false;
 }
break;
default: inWord = true;
}
```

```
}
System.out.println("\t"+numLines+"\t" + numWords + "\t" + numChars); } }
```

Bu programmada so’zlarni hisoblash uchun satrlarni qayta ishlash uchun bir nechta konsepsiya ishlatilgan.

SIKLLAR.

Hamma sikllarni ham to’rt qismga bo’lish mumkin – inisializasiya, tana,iterasiya va yakunlash sharti. Javada uchta siklli konstruksiya mavjud: while,do while,for.

While

Bu erda <shart> mantikiy ifoda. <shart> tekshiriladi agar shart chin natija bersa <operator > bajariladi va yana <shart> tekshiriladi. Agar <shart> yana chin natija bersa <operator> yana bajariladi va yana <shart> tekshiriladi va xokazo toki <shart> dan yolgon natija chikguncha <shart> dan yolgon natija chikdiki <operator> bajarilmasdan while operatori uz ishini tugatadi.

Pastda while operatorining umumiy ko’rinishi ko’rsatib o’tilgan.

```
[ inisializasiya; ]
while ( yakunlash )  {
tana;
[iterasiya;] }
```

Inisializasiya va iterasiya shart emas.Pastda while sikli pechati uchun o’n satrli “tick” ko’rsatib o’tilgan.

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

do-while

Operator bajariladi va undan keyin <shart> bajariladi. Agar <shart>ning natijasi chin bulsa yana <operator> bajariladi va yana <shart> tekshiriladi va xokazo toki <shart>dan yolgon natija chikguncha. <shart>dan yolgon natija chikdiki do-while operatori uz ishini tugatadi deb xisoblanadi.

Izox: Bu erda <operator> xech bulmaganda bir marta ishlaydi.

gohida sikl tanasini bo'lmaganda bir martta bajarishga ehtiyoj seziladi,hattoki mantiqiy ifoda boshdan false mohiyatini qabul qilayotgan bo'lsada,bunday paytda Javada do-while siklli konstruksiya qo'llaniladi.Uning umumiy ko'rinishi quyidagicha.

[inisializasiya;] do { tana; [iterasiya;] } while (yakunlash);

Keyingi misolda sikl tanasi yakunlash shartini birinchi tekshirilishigacha bajaradi.Bu iterasiya nkodini yakunlash sharti bilan joylashtirishni ta'minlaydi.

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

FOR

Bu operatorda to'rttala sikl uchun hamjoy ajratib ko'rsatilgan.Pastda for operatorining umumiy ko'rinishi keltirilgan.

for (inisializasiya; yakunlash; iterasiya) tana;

Har qanday for operatori yordamida yozilgan sikl while ko'rinishida yoki teskari yozish mumkin.

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

```
} }
```

Keyingi misolda – programma variantini,teskaridan sanab borish ko’rsatilgan:

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

Misol:

```
class Months {
static String months[] = {
"January", "February", “March", "April", "May", "June", "July", "August",
"September", "October", "November", "December" };
static int monthdays[]= { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
static String spring = "spring";
static String summer = "summer";
static String autumn = "autumn";
static String winter = "winter";
static String seasons[]={winter,winter,spring,spring, spring, summer,
                    summer, summer, autumn, autumn, autumn, winter };
public static void main(String args[]) {
for (int month = 0; month < 12; month++) { System.out.println(months[month] + "
is a "  + seasons[month] + " month with " + monthdays[month] + " days.");
}
} }
```

Dastur natijasi:

```
C:\> Java Months
January is a winter month with 31 days.
February is a winter month with 28 days.
March is a spring month with 31 days.
April is a spring month with 30 days.
```

May is a spring month with 31 days.

June is a summer month with 30 days.

uly is a summer month with 31 days.

August is a summer month with 31 days.

September is a autumn month with 30 days.

October is a autumn month with 31 days.

November is a autumn month with 30 days.

December a winter month with 31 days.

VERGUL OPERATORI.

Shunday hajmlar ham bo'ladiki for siklining inisializasiya va interasiya bo'limlari bir necha operatorlarni talab qiladi. For siklining sarlavhasidagi figurali qavsning qo'shma operatorida qo'yish mumkin emas,Java alternative yo'l taqdim etadi.Vergul bir necha operatorlarni bo'lish(ajratish) uchun faqatgina for n operatorining aylana qavslari uchun ruxsat etiladi.

Pastda for siklliga misol keltirilgan.Bu erda inisializasiya bir nechtadan operator mavjud.

```
class Comma {
public static void main(String args[]) {
int a, b;
for (a = 1, b = 4; a < b; a++, b--) {
System.out.println("a = " + a);
System.out.println("b = " + b);
}
} }
```

Xulosa qilib shuni aytib o'tish kerakki,sikl bor yo'g'i ikki martta bajariladi.

```
C: \> java Comma
a = 1
b = 4
```

a = 2

b = 3

continue

ayrim variantlarda keyingi iterasiyaga tezda o’tish talab etiladi.Bu maqsadda Javada continue operatori yaratilgan.

Keyingi misolda continue operatori har bir qatorda ikkitadan raqam ishlatilishi uchun qo’llaniladi.

```
class ContinueDemo {
public static void main(String args[]) {
for (int i=0; i < 10; i++) {
System.out.print(i + " ");
if (i % 2 == 0) continue;
System.out.println("");
} }}
```

Agar indeks aniq bo’lsa,unda sikl yangi qatorda xulosasiz davom etadi.Bu programmaning natijasi quidagicha ifodalanadi:

```
C: \> java ContinueDemo
0 1
2 3
4 5
5 7
8 9
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));
}          } }}
```

Xuddi break operatoridagidek continue operatorida ham belgi yuklash nmumkin. Continue operatori bu programmada hisobi bilan ichki siklni yakunlashiga va keyingi hisobli tashqi sikl iterasiyasiga olib keladi.Ish jarayonida programma quyidagi satrlarni taqdim etadi:

```
C:\> Java ContinueLabel
0
0 1
0 2 4
    1.  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
```

I bob yuzasidan qisqacha xulosa. Mening bitiruv malakaviy ishimning 1bobi JAVA dasturlash tilining sintaksisi va asosiy operatorlari bag'ishlangan bo'lib bu bob 3ta bo'limdan iborat. Birinchi bo'limda java dasturlash tilining sintaksisi va ma'lumotlar tiplar, java dasturlash tilida qanday e'lon qilinishi va ifodalanishi to'g'risida malumotlar keltirilgan.ikkinchi bo'limda java dasturlash tilining asosiy operatorlari va standart funksiyalari haqida chuqur ma'lumotlar va misollar keltirilgan.Uchinchi bo'limda boshqarish operatorlari haqida ya'ni tarmoqlanuvchi, takrorlanuvchi va variant operatorlari misollar yordamida tushuntirilgan. Bu jarayonlarni boshqarishda break va continue operatorlari haqida ham ma'lumotlar keltirilgan,massivlar, ko'p o'lchovli massivlar, massivlar ustida bajariladigan amallar misollar orqali tushuntirib o'tilgan. Sinf tushunchasi, sinflarni e'lon qilish tuzilmasi va sinflar ustida bajariladigan amallar ham misollar orqali tushuntirib berilgan. Kirish qismida Java dasturlash tili haqida qisqacha malumotlarni berib

o’tdim. Qolgan har bir rejada reja yuzasidan malumotlarni aytib o’tishga va har bir malumotni misollar bilan boyitib borishga harakat qildim.

II.BOB

JAVA DASTURLASH TILIDA SATR VA FAYLLAR OQIMI.

2.1. JAVA DASTURLASH TILIDA SATR OQIMI VA ULAR USTIDA AMALLAR.

SATR YARATISH.

Java o’zida shu operasiya uchun standart qisqartmani tashkil qiladi.Yozuv literal sifatida ko’rinishga ega bo’lib juft qo’shtirnoqlar bilan yakunlanadi.Quyidagi fragment kodi avvalgi fragmentlarning ekvivalentiga teng,unda satr char turning massivi bilan inisiallashtiradi.

```
String s = "abc";
System.out.println(s);
```

String ob’ekti bilan umumiy metodlardan birining qo’llanilishi – length metodi bo’lib u satrdagi belgilar sonini qaytaradi.Keyingi fragment uch sonini tasvirlaydi,chunki qo’llaniladigan satrda uchta belgi mavjud.

```
String s = "abc";
System.out.println(s.length);
```

Javada shunisi qiziqarliki har bir literalli satr uchun String sinfli taqdimot yaratiladi,endi siz bu sinfning metodini literalli – satr bilan chaqirishingiz mumkin.Keyingi misol ham uch sonini tasvirlaydi.

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

SATRLARNI QO’SHISH.

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

Satrida + operatori yordamida uch satr bir satrga umumiylashtirilgan. Metodlarning ekvivalentini topgandan ko’ra uni o’qish va tushinish ancha oson.

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

Aniqlashtirish bo'yicha String sinfining har bir ob'ekti o'zgarishi mumkin emas.Qatordagi belgilarni almashtirish va yangi belgilar qo'yish mumkin emas. Bir satrning oxiriga yana birini qo'yish mumkin ham emas.

OPERATORLAR BAJARILISHINING KETMA – KETLIGI.

Yana oxirgi misolimizga murojaat etamiz.

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

age string bo'lmagan holda,peremen int turiga mansub bo'lsa bu satrning kodi translyator magiyasidan ko'proq yakunlanadi.
Keyingi misolni ko'rib chiqamiz:

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

Birinchi o'rinda butun sonlarni qo'yilishini xoxlasangiz unda qavslardan foydalanish zarur:

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

SATRLARNI O'ZGARTIRISH.

String sinfining to String metodi yoki shaxsiy realizasiyasi mavjud.

```
class Point {
int x, y;
Point(int x, int y) {
this.x = x;
this.y = y;
}
 public String toString() {
return "Point[" + x + ", " + y + "]";
} }
class toStringDemo {
public static void main(String args[]) {
Point p = new Point(10, 20);
System.out.println("p = " + p);
} }
```

Misoldan olingan natija**:**

C:\> Java toStringDemo
p = Point[10, 20]

TENGLASHTIRISH.

Agar ikki satrning bir xilligini bilmoqchi bo'lsangiz unda siz String sinfining equals metodidan foydalanishingiz mumkin.Bu metodning alternative formasi **equalsIgnoreCase** deb nomlanadi**.**
Keyingi misolda ikki metodning qo'llanilishi illyustrasiyasi keltirilgan.

```
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));
} }
```

Natija:

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

Nusxalashda satrlarning modifikatsiyasi(o'zgarishi)

String sinfini o'zgartirib bo'lmasligi sababli,har safar satrni modifikatsiyalamoqchi bo'lsangiz uni yoki StringBuffer nusxalashingiz,yoki string sinfining tavsiflanadigan,satrga o'zgartirish kiritib,uni yangi nusxasini yaratadigan m-didan birini ishlatishingiz lozim.

Substring.

Subtring m-di yordamida String `idan qism satrni ajratib olishingiz mumkin.Bu m-d original (asil) satrdan chaqiruv chog'ida,ko'rsatilgan indekslar diapazonidagi belgilarning yangi nusxasini yaratadi,kerakli qism satrning birinchi – simvoli indekslarini ko'rsatish mumkin.Bunda yangi satrga birinchi ko'rsatilgan belgidan

boshlab to oxirgi indeks bilan ko’rsatilgan simvolgacha(lekin uning o’zi emas) bo’lgan barcha simvollar(ya’ni belgilar) nusxalanadi.

"Hello World".substring(6) -> "World"
"Hello World".substring(3,8) -> "lo Wo"

Concat

Satrlarning qo’shilishi yoki konkatenatsiyasi concat m-di yordamida bajariladi.Bu m-d String sinfining yangi `ini yaratadi,unga boshlang’ich satrini butunlay nusxalaydi va oxiridan m-d parametric sifatida ko’rsatilgan satrini qo’shadi.

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

Replace

Replace m-diga parametr sifatida ikkita belgi sifatida uzatiladi.Birinchi belgi b/n mos tushuvchi barcha belgilar satrning yangi nusxasida ikkinchi belgi b/n almashtiriladi.

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

toLowerCase va toUpperCase

Bu m-dlar juftligi mos ravishda boshlang’ich satrning barcha belgilarini kichik yoki katta registrga(kichik yoki bosh harflar) almashtiradi.

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

Trim

Trim m-di boshlang’ich satrning boshi va oxirida kelgan bo’sh joy belgilarni yo’qotadi.

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

valueOf

Agar siz biror berilganlar tipi bilan ishlasangiz va shu tipning qiymatini o’qilishi qulay shaklga keltirmoqchi bo’lsangiz,dastlab undagi qiymatini satr ko’rinishiga o’tkazishingiz kerak.Buning uchun valueOf m-di mavjud.Bunday static m-d Javada mavjud bo’lgan ixtiyoriy berilganlar tipi uchun aniqlangan(barcha shunday

m-dlar o’zaro moslashgan,ya’ni bitta nomdan foydalanadilar.) Shu sababli ixtiyoriy tip qiymatini satrga aylantirish qiyinchilik tug’dirmaydi.

StringBuffer

StringBuffer – String sinfining egizagi bo’lib, satrlar bilan ishlashda talab etiladigan narsalardan ko’pini taqdim etadi. String sinflari tayinlangan (fiksirlangan) uzunlikdagi belgilar ketma-ketligi bo’lib, ularni o’zgartirib bo’lmaydi. StringBuffer tipidagilar esa shunday belgilar ketma-ketligi, ularni kengaytirish yoki modifikatsiyalash mumkin. Javada ikkala sinf ham keng qo’llaniladi, biroq ko’pchilik dasturchilar faqat “String tip”lari bilan, “+” operatorini qo’llagan holda ishlashni ma’qul ko’radilar. Bunda Java StringBuffer bilan barcha kerakli amallarni o’zi “sahna ortidan” bajaradi.

Konstruktorlar.

StringBuffer `ni paranametrlarsiz hosil qilish mumkin, bunda unda o’n oltita belgini saqlash uchun joy ajratiladi, biroq satr uzunligini o’zgartirib bo’lmaydi. Siz yana Konstruktorga butun son uzatishingiz va buferning talab etilgan o’lchami oshkor holda berishingiz mumkin ,va nihoyat, **Konstruktorga** satr uzatishigiz mumkin, bunda unga nusxalanadi, qo’shimcha ravishda yana o’n oltita belgi uchun joy ajratiladi. **StringBufferning** joriy uzunligini length metodini chaqirib aniqlash mumkin.: **StringBuffer** `ida satr uchun ajratib qo’yilgan barcha joyni aniqlash uchun esa capacity metodini ishlatish kerak. Quyidagi misolni bu holda izohlab beradi:

```
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());
} }
```

Bu dasturning natijasidan ko’rinib turibdiki **StringBuffer** `ida satrlar bilan ishlash uchun qo’shimcha joy ajratilgan.

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

ensureCapacity

Agar siz **StringBuffer** `ini yaratib bo'lganingizdan so'ng unda ma'lum miqdordagi belgilar uchun joy ajratib qo'ymoqchi bo'lsangiz, buffer o'lchamini o'rnatish uchun **ensureCapacity** m-didan foydalanishingiz kerak. Ayniqsa, buferga ko'pgina kichikqroq satrlarni qo'shishga to'g'ri kelishini oldindan bilsangiz, bu metodni qo'llash juda foydalidir.

setLength

Agar siz nogahon bufferdagi satr uzunligini oshkor holda o'rnatishingizga to'g'ri kelsa, setLength m-dini ishlatishingiz mumkin.Agar siz `dagi satr uzunligining kattaroq sonni uzatsangiz bu m-d yangi kengaygan satr oxirini nol(0) kodli belgi bilan to'ldiradi sal keyinroq keltiriladigan setCharDemo dasturida setLength m-di buferni qisqartirish uchun ishlatiladi.

charAt va setCharAt

StringBuffer `idan bitta belgini charAt m-di yordamida ajratib olinadi.Boshqa m-d setCharAt satrning berilgan o'rniga(pozitsiyasiga) kerakli belgini yozishga imkon beradi.Bu m-dlarning ishlatilishi misol bilan keltirilgan:

```
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));
} }
```

Ushbu dastur ishga tushirilganda olinadigan natija:

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

append

StringBuffer sinfining **append** m-di odatda satrli ifodalarda "+" operatori qo'llanilganda oshkormas holda chaqiriladi.Har bir parametr uchun **String.valueOf** m-di chaqiriladi va uning natijasi joriy StringBuffer `iga qo'shiladi.Buning ustiga har safar append m-di chaqirilganda u o'zi bilan birga

chaqirilgan StringBuffer `iga ko’rsatgichni qaytaradi.Bu esa m-dni ketma-ket chaqirishlar zanjirini hosil qilishga imkon beradi.Quyidagi misolda shu xossasi ko’rsatilgan.

```
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);
} }
```

Misol natijasi:

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

Insert

Insert m-di append m-di bilan shu jihatlar bir xilki,har bir tip uchun ushbu m-dni qo’llash mumkin. Biroq,appenddan farqli ravishda u String.valueOf m-di tomonidan qaytariladigan belgilarni **StringBuffer** `i oxiriga qo’shmaydi balki uni birinchi parametr bilan beriladigan buferning muayyan joyiga qo’yish mumkin.Navbatdagi misolda "there" satri "hello" va "world!" orasiga qo’yiladi.

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

Ushbu dastur ishga tushirilsa quyidagi satr chop etiladi:

```
C:\> java insertDemo
hello there world!
```

Biz satrlar oqimi bilan ishlaganda satrning ixtiyoriy hajmdagi so’zlardan iboratligi ma’lum,bizga yana satr bilan ishlashning qulay tomoni sonlardan ham foydalanib ularning yig’indisini topishimiz mumkin.Bu dastur satrga ixtiyoriy sondagi sonlarni kiriyamiz dastur esa ularning yig’indisini topib beradi.

```
import java.io.*;
import java.util.*;
public class str {
        public static void main(String [] args) throws IOException{
```

```
        String s;
         int k,l,j=0;
         Scanner sc=new Scanner(System.in);
          System.out.print("satrni kirit..s=");
          s=sc.nextLine();
         Scanner sc1=new Scanner(s);
         while (sc1.hasNext()){
            l=sc1.nextInt();
             j=j+l;
         }

         System.out.println("bu sonlar yig;indisi  "+j+" ga teng");
    }
}
```

Satrlar oqimi bilan ishlaganda biror satrda ixtiyoriy miqdorda so’zlar kiritilgan bo’lsa bu satrda nechta so’z borligini aniqlab beradi

```
import java.io.*;
import java.util.*;

public class leo {

      public static void main(String [] args) throws IOException{

         String s,s1;
         int k=0,l,j=0,i=0;
         Scanner sc=new Scanner(System.in);
         System.out.print("satrni kirit..s=");
         s=sc.nextLine(); i++;
         Scanner sc1=new Scanner(s);
         k=0;
         while (sc1.hasNext()){
            s1=sc1.next();
            k++;
         }

        System.out.println("satrda  " +k+ " ta  so'z bor ");
      }
   }
```

Endi shu dasturning kengroq ko’rinishi biror A fayl yaratamiz va bu faylga ixtiyoriy matn kiritsak bu matnning har bir satrida nechtadan so’z borligini aniqlab beradi

```
import java.io.*;
import java.util.*;
public class gg {
      public static void main(String[] args) throws IOException{
         Scanner sc=new Scanner(new File("a.txt"));

              String s=" ";
         int i=0,k=0;
         while (sc.hasNext()){
            s=sc.nextLine();k=0; i++;
            Scanner sc1=new Scanner(s);
            while(sc1.hasNext()){
            s=sc1.next();
             k++;
            }
            System.out.println(i+" - satrda "+k+" ta so`z bor");
         }


         }
      }
```

2.2. JAVA DASTURLASH TILIDA FAYLLAR OQIMI USTIDA AMALLAR.....

Fayllar va satrlar bilan ishlaganda ishlatiladigan ba’zi funksiyalar

Statistik qatorlar **String klassi** **yo**rdamida tuziladi literali yordamida avtomatik tuziladi va yana ” +” operatsiyasi qatorlarni birlashtirish uchun ishlatiladi.

Agar operandlardan bittasi qator bo’lmasa u avtomatik qatorga o’zgartiriladi. Shu obyektlarni ishlatish uchun **toString**() metodi oraqali amalga oshiriladi

String obyektining bazi metodlari

length() **–** satr uzunligini ko’rsatadi

compareTo*(String anotherString)* – qatorninng leksigrafik tengligi;

concat*(String str)* – ikki qatorni birlashtiradi;

endsWith*(String suffix)* – ko'rsatilgan sufiks bilan tugashini ko'rsatadi

startsWith*(String prefix)* – qatorning ko'rsatilgan prifiks orqali boshlanishini ko'rsatadi
startsWith*(String prefix, int toffset)* – qatordagi ko'rsatilgan prefiksni joylashishini ko'rsatadi
equals*(Object anObject)* – ko'rsatilgan qatornig idintichnaligini tekshiradi
getBytes*()* – qatorning baytlardagi ko'rinishi
hashCode*()* – qatorning XESH kodi
indexOf*(int ch)* – qatorga kiritlgan birinchi simvolni qidirish
indexOf*(int ch, int fromIndex)* – ko'rsatilgan qatordagi birinchi kiritilgan simvolni qidirish;
indexOf*(String str) birinchi kiritililgan satr ostini qidirish*;
indexOf*(String str, int fromIndex)* –birinchi ko'rsatilgan satr ostini belgilangan joyda qidirish;
lastIndexOf*(int ch)* – oxirgi kiritilgan simvolni qidirish
lastIndexOf*(int ch, int fromIndex)* – oxirgi kiritilgan simvolni ko'rsatilgan joyda qidirish
lastIndexOf*(String str)* – oxirgi kiritlgan satrni qidirish;
lastIndexOf*(String str, int fromIndex)* – ko'rsatilgan joyda oxirgi kirilgan qatorni qidirish;
replace*(char oldChar, char newChar)* – bir joyda bor bo'lgan qatorni boshqasiga almashtirish
replace*(CharSequence target, CharSequence replacement)* –bir satr ostini boshqasiga o'zgartirish.
substring*(int beginIndex, int endIndex)* – satr ostini satr ko'rinishida qaytarish
toLowerCase*()* –qatorni pastki registrga joylashtirish;
toUpperCase*()* – qatorni yuqori registrga joylashtirish
trim*()* – qatorning oxiridagi bo'sh simvollarni qirqib olish
valueOf*(a)* – har xil tiplarni qatorga o'zgartiradigan statidtik meto.

read() – joriy simvolni kiritish oqimiga butun ko'rinishda ko'rsatadi;

read(byte b[]) - b.length baytida kiritilgan oqimni massivga o'qiy olgan oqimni ko'rsatadi;

read(byte b[], int off, int len) – maxsimum len baytini o'qishga harakat qiladi,Off elementidan boshalanadigan b massiviga joylashtiradi real o'qigan baytlarning miqdorini qaytradi

skip(long n) – kiritish oqimiga n baytlarni kiritishga harakat qiladi o'tkazilgan baytlar miqdorini ko'rsatadi;

available() – joriy paytda o'qilayotgan baytlar miqdorini ko'rsatadi;

close() – kiritish manbasini yopadi ,qayta ochish chog'ida IOException.ni faollashtiradi

write(int b) kiritish oqimiga bir baytni yozadi,etibor bering metodning argumenti int tipiga mansub u write yordamida jumla kiritiladi jumlani baytlarda kiritish shart emas.

write(byte b[]) –ko'rsatilgan massivdagi hamma baytlarni kiritish oqimiga yozadi

write(byte b[], int off, int len) –len bayti massivida oqimning bir qismini yozadi b[off] boshlab.

flush() –kiritish jarayonini tugatadi hamma buferni tozalaydi.

Endi faylar oqimiga oid misollar keltiraman

```
import java.io.File;


   class DirList {
      public static void main(String args[]) {
         int k=0;
         String dirname = "/SALOM"; // papka nomi
         File fl = new File(dirname);
         if (fl.isDirectory()) {

            System.out.println("Directory of" + dirname);
                  String s[]=fl.list();
            for (int i=0; i < s.length; i++) {
               File f = new File(dirname + "/SALOM" + s[i]);
               k++;
               if(f.isDirectory())
               {

                  System.out.println(s[i] +" bu ma'lumot");
               }
               else {
                  System.out.println(s[i] + " bu file");
               }
            }
         }
```

```
        else {

            System.out.println(dirname + " BU MA'LUMOT EMAS");

         }
          System.out.println("Fayllar soni "+k );
      }
   }
```

Bu dastur bizga biror papkani kiritsak uni ichida nimalar joylashganini va u ma’lumot yoki fayl va ular jami nechtaligini aniqlab beradi.

NATIJA:

Directory of/SALOM
VisualStudio2012 bu file
VM_TEST bu file
VS2012_ULT_rus bu file
Активатор bu file
Кобилов учун bu file
Чуллиев учун bu file
Fayllar soni 6

```
import java.io.File;


class FileTest {
      static void p(String s) {
         System.out.println(s);
      }
      public static void main(String args[]) {
         File fl = new File("/Golden boy D/Siyosat.docx");
         p("fayl nomi:" + fl .getName());
         p("yo'l:" + fl.getPath());
         p("to'liq yo'l:" + fl.getAbsolutePath());
         p("bosh katalog:" + fl.getParent());
         p(fl.exists() ? "mavjud" : "mavjud emas");
         p(fl.canWrite() ? "ko'chirish mumkin" : "ko'chirish mumkin emas");
         p(fl.canRead() ? "o'qish mumkin" : "o'qish mumkin emas");
         p( fl.isDirectory() ? " yo'q" : "ha");
         p(fl.isFile() ? "oddiy fayl" : "oddiy fayl emas");
         p("oxirgi модификациялangan fayl :" + fl. lastModified());
```

```
        p("fayl o'lchami:" + fl.length() + " baytlarda");
    }
}
```

Bu dastur bizga papkamizdagi faylning qayerda joylashganini,to'liq yo'lini,tipinini vao'lchami qanaqaligini aniqlab beradi

NATIJA:

```
fayl nomi:Siyosat.docx
yo'l:\Golden boy D\Siyosat.docx
to'liq yo'l:E:\Golden boy D\Siyosat.docx
bosh katalog:\Golden boy D
mavjud
ko'chirish mumkin
o'qish mumkin
ha
oddiy fayl
oxirgi модификацияlangan fayl :1429211232054
fayl o'lchami:225904 baytlarda
```

2.3. MA'LUMOTLARNI FAYLDA YOZISH VA O'QISH

Biz JAVA dasturlash tilida ma'lumotlarni yozish va o'qishda funksiyalardan foydalanamiz,biz yuqorida kiritishda

```
StringBuffer sb = new StringBuffer("hello world !");
```

Funksiyasidan foydalandik endi esa xuddi shunday vazifani bajarishda

```
Scanner sc=new Scanner(new File("a.txt"));
```

foydalansak ham bo'ladi.Demak biz faylga yozish va o'qish uchun quyidagi kodlarni kiritamiz

```
Scanner sc=new Scanner(new File("a.txt"));
    PrintWriter pw=new PrintWriter(new File("b.txt"));
```

Biz bu yerda yangi fayl yani "a.txt" degan fayl yaratdik.
PrintWriter funksiyasidan foydalansak natijani " b.txt " faylga yozadi…
Endi quyidagicha misollar bilan ko'ramiz

Biz bu dasturda satr uzunligini fayllar bilan ishlash orqali topamiz
Yani “ a.txt ” fayliga satrni kiritsak “b.txt ”faylida uning uzunligini ko’ramiz.

```
import javax.swing.*;
import java.util.*;
import java.io.*;
public class my6 {
   public static void main(String[] args) throws IOException{
      Scanner sc=new Scanner(new File("a.txt"));
      PrintWriter pw=new PrintWriter(new File("b.txt"));

      String s=sc.nextLine();

      int k=0; int j,i;int m=0;
      for (i=1;i<=s.length();i++)
                  k++;
         pw.println("satr uzunligi k="+k);
      sc.close();pw.close();


   }
}
```

Natija :a.txt → salom
b.txt → 5

Biz fayllar bilan ishlaganda faqat satr yoki matn bilan ishlashimiz shart emas xoxlagan turdagi dasturni tuza olamiz,bu dastur n-chi tub sonni topib beradi.

```
import javax.swing.*;
import java.io.*;
import java.util.*;
public class my7 {
   public static void main(String[] args) throws IOException{
   Scanner sc=new Scanner(new File("a.txt"));
    PrintWriter pw=new PrintWriter(new File("b.txt"));

      int n=sc.nextInt();
      int j,k,t,m=0;
      int i;

         for (i=2;true;i++){ k=0;
        for (j=2;j<=Math.sqrt(i);j++)
           if (i%j==0) k++;
            if (k==0)m++;
```

```
                if (m==n){
               pw.println(i);break;} }


            sc.close();pw.close();
   }
}
```

Natija :"a.txt" → 5
"b.txt"→11

Men JAVA dasturlash tilida fayllar bilan ishlash jarayonida IntelliJ IDEA muhitida ishladim "kirilchadan lotinchaga va lotindan kirilchaga" o'giradigan dasturni tuzdim bunda fayllarning o'rni juda muhim edi bu dastur kodi quyidagicha

Lotin → Kiril

```
import java.util.*;
import java.io.*;
public class KL {
      public static void main(String[] args) throws IOException{
         int i,j;
         Scanner sc=new Scanner(new File("a.txt"));
         PrintWriter pw=new PrintWriter(new File("b.txt"));
    while (sc.hasNext())
      {


       String k=sc.nextLine(),
x="ABSYUKEHGOZXFQVPRLDJEMITabsyukengozxfqvprldjemith",
y="АБСЙУКЕҲГОЗХФҚВПРЛДЖЭМИТабсйукенгозхфқвпрлджэмитҳ"
;
   k=k.replace("yo","ё");
   k=k.replace("ch","ч");
   k=k.replace("sh","ш");
   k=k.replace("yu","ю");
   k=k.replace("ya","я");
   k=k.replace("Yo","Ё");
   k=k.replace("Ch","Ч");
   k=k.replace("Yu","Ю") ;
   k=k.replace("Ya","Я");
   k=k.replace("Sh","Ш");
   k=k.replace("O'","Ў");
   k=k.replace("o'","ў");
   k=k.replace("g'","ғ");
```

```
   k=k.replace("G'","Ғ");
   k=k.replace("ye","e");

       StringBuffer K=new StringBuffer(k);
   for ( i=0;i<k.length();i++){
      for(j=0;j<=48;j++)
         if (K.charAt(i)==x.charAt(j))
         K.setCharAt(i,y.charAt(j));
   }

            pw.println(K);   }
         sc.close();pw.close();


      }}
```

Bu dastur **lotindan kirilga** o'giradigan bunda siz **"hasNext"** funksiyasidan foydalanayapmiz bu fuksiya faylni oxirigacha o'qish uchun kerak bo'ladi.

Natija :" a.txt"→ Assalom O'zbekiston
"b.txt"→Ассалом Ўзбекистон

Kiril → lotin

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

      public static void main(String[] args) throws IOException{
         int i,j;
         Scanner sc=new Scanner(new File("a.txt"));
         PrintWriter pw=new PrintWriter(new File("b.txt"));
     while (sc.hasNext())

       {

String k=sc.nextLine(),
x="АБСЙУКЕНГЎЗХФҚВПРОЛДЖЭМИТҲаҳбсйукенгўзхфқвпролджэ
мит",
y="ABSYUKEHGOZXFQVPROLDJEMITHahbsyukengozxfqvproldjemit";
            k=k.replace("ё","yo");
            k=k.replace("ч","ch");
            k=k.replace("ш","sh");
```

```
  k=k.replace("ю","yu");
  k=k.replace("я","ya");
  k=k.replace("Ё","Yo");
  k=k.replace("Ч","Ch");
  k=k.replace("Ш","Sh");
  k=k.replace("Ю","Yu");
  k=k.replace("Я","Ya");
  k=k.replace("ў","o'") ;
  k=k.replace("Ў","O'");
  k=k.replace("Ғ","G'");
  k=k.replace("ғ","g'");
  k=k.replace("Ц","S");
  k=k.replace("ц","s");

  StringBuffer K=new StringBuffer(k);
  for ( i=0;i<k.length();i++)
 {
    for(j=0;j<=51;j++)
       if (K.charAt(i)==x.charAt(j)) K.setCharAt(i,y.charAt(j));


  }

  pw.println(K);
                    }
sc.close();pw.close();

              }
               }
```

Natija :" a.txt "→Салом жава
"b.txt" →Salom java

Bu dasturda ko'plab funksiya va protseduralar dan foydalandik.... **replace** ni ham ishlash jarayonini ko'rdingiz.
String va StringBufferni taqqoslab Stringda yo'q funksiyalarni StringBufferdan oldik va dasturni to'liq ko'rinishga keltirdik….

2 bob yuzasidan qisqacha xulosa. Mening bitiruv malakaviy ishimning 2-bobi **JAVA dasturlash tilida satr va fayllar oqimiga** bag'ishlangan bo'lib bu bob 3ta bo'limdan iborat. Birinchi bo'limda Java dasturlash tilida satr oqimi va ular ustida amallar, yani satrlarlar bilan ishlash ularga oid bir nechta murakkab dasturlar keltirilgan.

Ikkinchi bo'limda Java dasturlash tilida fayl oqimi va ular ustida amallar deb nomlanadi bu bo'limda men fayllar oqimi haqida keng ma'lumotlar berdim,va ular ustida ishlash jarayonida bir nechta murakkab masalalarni dasturini namoyish qildim,bu bo'limda sizga fayllar

oqimi haqida chuqur ma'lumotlar va dasturlar keltirilgan.Uchinchi bo'limda ma'lumotlarni faylda yozish va o'qish, bu bo'limda men fayllar bilan ishlashda yani biror a.txt fayl yaratib natijani bu yaratgan faylga faylga biror "salom "degan so'z kiritsam b.txt faylga biz kiritgan "salom" so'zini chiqarib beradi.bunga o'xshash va bundan murakkab bir nechta dasturlar tuzdim. Kirish qismida Java dasturlash tili haqida qisqacha malumotlarni berib o'tdim. Qolgan har bir rejada reja yuzasidan malumotlarni aytib o'tishga va har bir ma'lumotni misollar bilan boyitib borishga harakat qildim.

Xotima

I bob yuzasidan qisqacha xulosa. Mening bitiruv malakaviy ishimning 1bobi JAVA dasturlash tilining sintaksisi va asosiy operatorlari bag'ishlangan bo'lib bu bob 3ta bo'limdan iborat. Birinchi bo'limda java dasturlash tilining sintaksisi va ma'lumotlar tiplar, java dasturlash tilida qanday e'lon qilinishi va ifodalanishi to'g'risida malumotlar keltirilgan.ikkinchi bo'limda java dasturlash tilining asosiy operatorlari va standart funksiyalari haqida chuqur ma'lumotlar va misollar keltirilgan.Uchinchi bo'limda boshqarish operatorlari haqida ya'ni tarmoqlanuvchi, takrorlanuvchi va variant operatorlari misollar yordamida tushuntirilgan. Bu jarayonlarni boshqarishda break va continue operatorlari haqida ham ma'lumotlar keltirilgan,massivlar, ko'p o'lchovli massivlar, massivlar ustida bajariladigan amallar misollar orqali tushuntirib o'tilgan. Sinf tushunchasi, sinflarni e'lon qilish tuzilmasi va sinflar ustida bajariladigan amallar ham misollar orqali tushuntirib berilgan. Kirish qismida Java dasturlash tili haqida qisqacha malumotlarni berib o'tdim. Qolgan har bir rejada reja yuzasidan malumotlarni aytib o'tishga va har bir malumotni misollar bilan boyitib borishga harakat qildim.

2 bob yuzasidan qisqacha xulosa. Mening bitiruv malakaviy ishimning 2-bobi **JAVA dasturlash tilida satr va fayllar oqimiga** bag'ishlangan bo'lib bu bob 3ta bo'limdan iborat. Birinchi bo'limda Java dasturlash tilida satr oqimi va ular ustida amallar, yani satrlarlar bilan ishlash ularga oid bir nechta murakkab dasturlar keltirilgan.

Ikkinchi bo'limda Java dasturlash tilida fayl oqimi va ular ustida amallar deb nomlanadi bu bo'limda men fayllar oqimi haqida keng ma'lumotlar berdim,va ular ustida ishlash jarayonida bir nechta murakkab masalalarni dasturini namoyish qildim,bu bo'limda sizga fayllar

oqimi haqida chuqur ma'lumotlar va dasturlar keltirilgan.Uchinchi bo'limda ma'lumotlarni faylda yozish va o'qish, bu bo'limda men fayllar bilan ishlashda yani biror a.txt fayl yaratib natijani bu yaratgan faylga faylga biror "salom "degan so'z kiritsam b.txt faylga biz kiritgan "salom" so'zini chiqarib beradi.bunga o'xshash va bundan murakkab bir nechta dasturlar tuzdim. Kirish qismida Java

dasturlash tili haqida qisqacha malumotlarni berib o’tdim. Qolgan har bir rejada reja yuzasidan malumotlarni aytib o’tishga va har bir ma’lumotni misollar bilan boyitib borishga harakat qildim.