ӨЗБЕКСТАН БАЙЛАНЫС ҲӘМ ИНФОРМАЦИЯЛАСТЫРЫЎ АГЕНТЛИГИ

ТАШКЕНТ ИНФОРМАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАРЫ УНИВЕРСИТЕТИ НӨКИС ФИЛИАЛЫ

Информатика ҳәм информациялық технологиялар кафедрасы

Информациялық технологиялар факультетиниң информатика ҳәм информациялық технологиялар бағдарының 4-курс студенти Султанов Ерполаттың

ПИТКЕРИЎ ҚӘНИЙГЕЛИК ЖУМЫСЫ

Темасы: **Объектке бағдарланған программаластырыў тилинде классларды жаратыў қураллары**

Илимий басшысы:_____доц.Бурханов Ш.А.

Кафедра баслығы:_____доц. Бурханов Ш.А.

НӨКИС - 2012 ж.

МАЗМУНЫ

КИРИСИЎ

Заманагөй программаластырыў тиллериниң көпшилиги тәрепинен қоллап қуўатланыўшы объектке бағдарланған программаластырыў ҳәзирги заман программаластырыў әмелиятында ең көп тарқалған ҳәм көп қолланылатуғын бағдар есапланады. Соның менен бирге объектке бағдарланған программаластырыўдың тийкарын дүзиўши концепция ҳәм механизмлердиң әмелге асырылыўлары ҳәр қыйлы тиллерде ҳәр түрли болыўы мүмкин. Сонлықтан ОБП ны үйрениў тек ғана конкрет программаластырыў тилин үйрениўден ибарат болып қалмастан, ал ОБП ның тийкарын қураўшы концепциялар ҳәм механизмлерди үйрениўден ибарат болыўы тийис. Профессионал программалаўшы инженер билиўи керек болған ең әҳмийетли затлардың бирине фундаментал концепциялар жыйыны жатады ҳәм ол оны биринши гезекте үйрениўи ҳәм қайта-қайта қолланыўы керек болады. Бир тәрептен, бундай программаластырыўды үйрениў усылы программалық тәмийнлеўди ислеп шығыўшылар алдында турған машқалаларды жақсы түсиниўге алып келсе, екинши тәрептен программаластырыўдың ҳәр қыйлы тиллер ҳәм технологияларының келип шығыў себеплерин ҳәм табыслы раўажланыўын жақсы түсиниўге мүмкиншилик береди.

Улыўма айтқанда, жаңа программаластырыў технологияларының пайда болыўы көбирек дәрежеде программалық тәмийнаттың жобаластырыў машқалаларының қурамалылығы өсиўи менен бирге өзгериске ушырап барыўына байланыслы деўге болады. Өз гезегинде, программалық тәмийнаттың қурамалылығының өсип барыўы адам хызметиниң ҳәр қыйлы тараўларының рәўажланыўлары ҳәм оларда заманагөй информациялық ҳәм компьютерлик технологиялардың қолланыўларынан келип шығады. Демек бундай жаңа мәселелерди шешиў программалық ҳәм аппаратлық тәмийнлеўдиң жаңа технологияларын ислеп шығыў, раўажландырыў ҳәм үйрениўди талап етеди.

Бакалавр питкериў қәнигелик жумысы объектке бағдарланған программаластырыў тилинде классларды жаратыў қуралларына арналған болып, ол кирисиў бөлиминен, 5 параграф, санлы мысал ҳәм оған қосымша, жуўмақлаў бөлиминен және пайдаланылған әдебиятлардан ибарат.

Питкериу қәнигелик жумысында объектке бағдарланған программаластырыў тилиниң принциплери ҳәм оларда проектлерди дүзиў этаплары сөз етиледи. Классларды жаратыўдың технологиялық қураллары ҳәм механизмлери қаралады.

Жумыстың кирисиў бөлиминде таңланған теманың актуаллығы, қурамы сөз етиледи.

Биринши параграфта объектке бағдарланған программаластырыўдың тийкарғы принциплери келтириледи.

Екинши параграфта классларды дүзиўдиң программалық қураллары ҳәм механизмлери қаралады.

Үшинши параграф BORLAND PASCAL 7.0 ҳәм Borland C++ 3.1 тиллеринде объектке бағдарланған программалар дүзиў технологияларына арналған.

Төртинши параграфта Delphi тилинде класслар ҳәм объектлер менен жумыс ислеў мүмкиншиликлери ҳәм өзгешеликлери келтирилген.

Бесинши параграф Delphi тилинде класслар дүзиўге мысаллар шешиўге арналған.

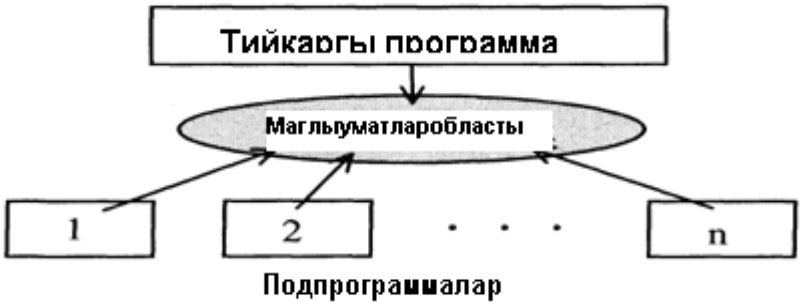
Жуўмақлаў бөлиминде питкериў қәнигелик жумысында алынған нәтийжелер ҳәм олардың әҳмийети ҳаққында сөз етиледи.

Қосымшада программа тексти келтирилген.

§1. Объектке бағдарланған программаластырыўдың тийкарғы принциплери

Көп жыллар даўамында программаластырыў әмелияты программаларды дүзиўдиң технологиялық тәсиллерин жетилистириўди талап етип келмекте ҳәм соның тийкарында программаластырыўдың сондай қуралларын жаратыў мүмкин болсын, нәтийжеде программаларды ислеп шығыў процессии әпиўайыластырылсын ҳәм қурамалырақ программалық системаларды жаратыў мүмкин болсын.

Дәслепки программалар жүдә әпиўайы болып, олар қайта исленетуғын мағлыўматлар ҳәм машина тилиндеги программалардан ибарат еди. Дәслеп ассамблер тилиниң оннан кейин жоқары дәрежедеги тиллердиң пайда болыўы деталластырыў дәрежесин пасейтиў есабынан программалардың қурамалылығын асырыў мүмкиншилигин пайда етти. Подпрограммалардың пайда болыўы менен программаларды ислеп шығыўдағы қыйыншылықлар азайды. Подпрограммаларды сақлаў ҳәм басқа программаларда пайдаланыў мүмкин болды. Нәтийжеде программалар тәрепинен шақырыў ҳәм пайдаланыў мүмкин болған есаплаў ҳәм хызмет етиўши подпрограммалардың оғада үлкен библиотекалары жаратылды. Ол ўақыттағы әдеттеги программа тийкарғы программа, глобал мағлыўматлар жыйыны ҳәм подпрограммалардан ибарат еди.



Бундай архитектураның ҳәлсиз тәрепи подпрограммалардың көбейиўи менен глобал мағлыўматлар жыйынының базыбир бөлегин подпрограммалардың биреўи тәрепинен бузыў итималлығының артыўына алып келди. Буған қосымша программаларды бирнеше программистлер

дүзгенде программа интерфейсин сәйкеслендириў машқаласы да қосылды. Усы кемшиликлерди жоғалтыў мақсетинде структуралы программаластырыў технологиясы жаратылды. Структуралы программаластырыўдың тийкарғы принциплери:

- Төменге қарап ислеп шығыў принципи;
- Структуралы программаластырыў принципи;
- Жалпы структуралық тексериў принципи;

Структуралық программаластырыў тийкарында қурамалы системаларды декомпозициялаў ҳәм оннан кейин сол бөлеклерди подпрограммалар көринисинде әмелге асырыў жатады. Структуралық жантасыўда мәселени әпиўайы структураға ийе үлес мәселелер иерархиясы көринисинде аңлатыў талап етилди ҳәм оның ушын адымба-адым деталластырыў усылы қолланылды. Бундай декомпозицияны процедуралық декомпозиция усылы деп атайды. Адымба-адым деталластырыў усылы төмендегилерден ибарат болды:

Программаның улыўма структурасы төмендеги 3 варианттан биреўи менен анықланды:

- Үлес мәселелер избе-излиги;
- Үлес мәселелер тармақлары;
- Үлес мәселелер тәкирарланыўлары

Өз гезегинде ҳәр бир үлес мәселе сол структуралар жәрдеминде және үлес мәселелерге бөлинеди ҳәм әпиўайы мәселеге шекем даўам еттириледи.

Мысал-1. Жазыў китапшасының процедуралық декомпозициясы.

Буннан кейин программалық тәмийнлеўдиң қурамаласыўы ҳәм көлеминиң артып барыўы ҳәм соның нәтийжесинде қәтелердиң де өсип барыўы жаңа усылларды ислеп шығыўды талап етти. Нәтийжеде модуллик программаластырыў технологиясы пайда болды. Модуллик программаластырыў технологиясы бирдей болған глобал мағлыўматлардан пайдаланатуғын подпрограммаларды компиляцияланыўшы бир модулге ажыратыўды талап етеди. Модуллер арасындағы байланыс арнаўлы интерфейс арқалы әмелге асырылады соның менен бирге модулдиң денесине(ондағы подпргаммалар) рухсат берилмейди.

Әмелият соны көрсетти модуллик принципке сүйенген структуралық программаластырыў усылы жәрдеминде 100 000 операторға ийе программаларды ислеп шығыўда пайдаланыў исенимли программаларды ислеп шығыўға мүмкиншилик береди екен. Егер программаның көлеми оннан асып кеткен жағдайда модуллер арасындағы байланыслардың тәсирин анықлаў мүмкин болмай қалады. Сол себептен жаңа технлогиялардан пайдаланыў зәрурлиги пайда болды ҳәм ол объектке бағдарланған программаластырыў технологиясына алып келди.

Программаластырыў технолгиясында ОБП ны қурамалы программалық тәмийнлеўди жаратыў технологиясы деп анықлайды ҳәм оның тийкарында программаларды ҳәр бири белгили типтеги класслардың экземпляры болған объектлер жыйыны деп қараў жатады.

ОБП ның тийкарғы артықмашлығы модуллик программаластырыўға салыстырғанда – модуллер арасындағы шақырыўлардың қысқарыўы ҳәм олар арасындағы информация көлеминиң азайыўы болып табылады. Буған мағлыўматларды локалластырыў ҳәм оларды подпрограммаларға интегралластырыў есабынан ерисиледи.

Соның менен бирге объектлик жантасыў программаларды ислеп шығыўдың жаңа технологиялық қуралларын усынады: нәсиллик, полиморфизм, композиция, толықтырыў усаған. Олар жәрдеминде әпиўайы объектлерден қурамалы объектлерди дүзиў мүмкин.

ОБП ның тийкарында төмендеги принциплер жатады: абстрактластырыў, рухсатты шеклеў, модуллик, иерархиялық, типлестириў, параллеллестириў, орнықлылық.

Абстракластырыў – бул предмет тараўы мәселесиндеги абстракцияларды ажыратыў процессии. Абстракция – базыбир объектти басқа объектлерден ажыратып турыўшы оның әҳмийетли характеристикаларының жыйыны. Предметтиң абстракциясы шешилетуғын мәселеден тиккелей ғәрезли болады: базыбир жағдайда бизди предмет формасы ал екинши жағдайда оның салмағы, үшинши жағдайда қандай материалдан жасалғанлығы ҳ..т.б қызықтырыўы мүмкин.

Рухсатты шеклеў - деп абстракцияның әҳмийетли характеристикаларына тәсир жасамайтуғын оны әмелге асырыў элементлериниң айырымларын жасырыўға айтамыз. Рухсатты шеклеў абстракцияны әмелге асырыўды еки бөлимнен ибарат етип тәрийплеўди талап етеди: интерфейс ҳәм әмелге асырыў бөлими.

Предметтиң барлық қәсийетлерин(оның аҳўалы ҳәм ис-ҳәрекетин қураўшы) бирдей абстракцияға бирлестириў ҳәм бул қәсийетлердиң әмелге асырылыўына рухсатты шеклеў **инкапсуляция** деп аталады.

Модуллик – программалық системаны ислеп шығыў принципи болып, ол оны бөлек бөлимлер түринде әмелге асырыўды көзде тутады.

Иерархиялық – бул абстракциялардың тәртиплескен ямаса ранжирленген системасы. ОБП да иерархияның еки түри қолланылады: **«пүтин/бөлек**» ҳәм **«улыўма/дара».**

«Улыўма/дара» иерархиясындағы ОБП ның ең әҳмийетли механизмлериниң бири – қәсийетлердиң мийраслыққа берилиўи. **Мийраслық** – бул абстракциялар арасындағы сондай қатнас болып, онда олардың бири екиншисиниң ямаса бирнешесиниң структуралық ямаса функционаллық бөлегин пайдаланады.

Типлестириў – деп объектлердиң қәсийетлерине қойылатуғын ҳәм ҳәр қыйлы типтеги абстракциялардың өз-ара алмасыўына жол қоймайтуғын шеклеўге айтылады.

Тип программалық объект пенен **статикалық** (объект типии компиляция ўақтында анықланады оны – **ерте байланыс** деймиз) ҳәм **динамикалық** (объект типии программа орынланыў ўақтында анықланады, оны – **кеш байланыс** деймиз) байланысқан болады.

Параллелестириў принципи – деп бирнеше абстракциялардың бир ўақытта актив аҳўалда болыў, яғный базыбир операцияларды орынлаў қәсийетине айтылады.

Орнықлылық – деп абстракцияның оны пайда етген программалық объект процессинен ўақыт бойынша ҳәм адреслик кеңисликтен кеңислик бойынша ғәрезсиз бар болыў қәсийетине айтамыз.

ОБП қолланып программалық системаларды ислеп шығыў этаплары.

ОБП қолланып программалық тәмийнлеўди ислеп шығыў процессии 4 этапты өз ишине алады: анализ, проектлестириў, эволюция, жетилистириў.

Анализ этабы- бунда мәселениң предмет тараўы анализи, ислеп шығылатуғын системаның объектлик декомпозициясы өткериледи ҳәм объектлердиң ис-ҳәрекетиниң ең әҳмийетли өзгешеликлери анықланады. Анализ нәтийжелери бойынша программалық өнимниң структуралық схемасы ислеп шығылады ҳәм онда тийкарғы объектлер ҳәм хабарландырыўлар көрсетиледи, абстракциялардың тәрийпи орынланады.

Проектлестириў этабы – ол еки бөлимнен турады: **логикалық** ҳәм **физикалық.**

Логикалық проектлестириўде класслар структурасы ислеп шығылады, яғный объектлер аҳўалын билдириўшилерди сақлаў ушын майданлар ҳәм объектлердиң ис-ҳәрекетин әмелге асырыўшы методлар алгоритмлери анықланады.

Физикалық проектлестириўде класслар тәрийплери модуллерге бирлестириледи, оларды иске қосыў схемасы таңланады, ускенелер, операциялық система ҳәм басқа программалық тәмийнлеў менен бирге ислесиў усыллары анықланады,

Система эволюциясы этабы – бул классларды этаплы әмелге асырыў ҳәм проектке қосыў процессии. Процесс тийкарғы программаны ямаса оның келешектеги проектин дүзиўден басланады.

Жетилистириў этабы – бул системаға жаңа функционал мүмкиншиликлерди қосыў ямаса бар қәсийетлерин өзгертиў процесси.

Объектлик декомпозиция. Объектлер ҳәм хабарландырыўлар.

ОБП технологиясын қолланыўда **мәселе шешими** қойылған мәселениң предмет тараўындағы болып атырған процесслерди имитациялаўшы базыбир системаны қураўшы функционал элементлердиң өз ара ҳәрекети нәтийжеси болып есапланады. Мәселени шешиў процессин **хабарландырыўлар** избе-излиги басқарады.

Жеке ис-ҳәрекетке ийе ҳәм параметрлери, ис ҳәрекети шешилетуғын мәселе шәртинен анықланыўшы системаның функционал элементлерине **объектлер** деп атаймыз.

Предмет тараўы мәселесин өз ара хабарландырыўлар менен алмасыўшы объектлердиң жыйыны ретинде аңлатыў процессии **объектлик декомпозиция** деп аталады. Мәселениң объектлик декомпозициясын орынлағанда ондағы объектлер жыйынын анықлаў ушын моделлестирилиўши процесстиң анализи өткериледи.

Мысал -1. Жазыў китапшасының объектлик декомпозициясы. Меню, Китапшаны ашыў, Жазыўларды киритиў, Жазыўларды излеў.

Солай етип, объектлик декомпозицияны өткериў бойынша төмендеги усынысларды келтирип өтсек болады:

1. Қурамалы системалар ушын объектлик декомпозиция этаппа этап орынланыўы тийис: биринши этапта ОД барлық система ушын, кейинги этапларда ОД үлес системалар ушын өткерилиўи тийис.

2. Системаны пүтини менен декомпозициялаўда объектлер сыпатында еки типтеги элементлер ажыратылыўы мүмкин:

- пайдаланыўшы интерфейси элементлери(меню айнасы, хабарландырыўлар айнасы, киритиў-шығарыў формасы айнасы ҳ.т.б.);
- сақлаў қураллары, мағлыўматларды шөлкемлестириў ҳәм түрлендириў (мағлыўматлар базасы, файллар, протоколлар, мағлыўматлар структурасы ҳ.т.б). Бунда ҳәр бир объект ушын қабыл қылынатуғын ҳәм жиберилетуғын хабарландырыўлар ҳәм тийкарғы характеристикалар көплиги анықланыўы тийис.

3. Декомпозиция процессии жетерли әпиўайы әмелге асырыў мүмкин болған объектлер алынғанша даўам етеди, яғный олар анық белгили структура ҳәм ис-ҳәрекетине ийе болады.

Объектлер ҳәм хабарландырыўлар

Өткен темада ОБП да объект деп мәселениң предмет тараўының бөлек әмелге асырылатуғын бөлимине айтылатуғыны көрсетилди. Солай етип ислеп шығылатуғын программа хабарландырыўларды жибериў арқалы өз ара байланыста болатуғын объектлерден турады екен. Ҳәрбир объект хабарландырыў алғаннан кейин оған жуўап бериўи керек болады, яғный хабар типине қарай жуўап ретинде белгили ҳәрекетти орынлаў керек. Хабарға реакция түри объекттиң аҳўалына байланыслы болады.

Объекттиң аҳўалы оның барлық мүмкин болған қәсийетлери дизиминиң базыбир конкрет мәнислер жыйыны менен характерленеди. Обектти басқа объектлерден ажыратып турыўшы қәсийетлери оның индвидуаллығын қурайды.

Объектлердиң ис-ҳәрекети қабыл еткен хабарландырыўға қайтарылатуғын реакциялардың белгили жыйынынан ибарат болады ҳәм ол жийи объект аҳўалынан ғәрезли болады.

Егер объект базыбир аҳўалға ийе болса, онда оның усы аҳўалы ҳаққында информация алыў мүмкиншилиги болады. Бундай информацияны алыў ушын объектке хабар-сораўнама жибериледи ҳәм жуўап ретинде ол

тийисли информацияны жибериўи керек. Бул жағдайда объект үстинде **селекция** операциясы орынланды делинеди.

Объектти оның толық аҳўалын ямаса оның бөлимлериниң аҳўалын өзгертиў ушын шақырыў **модификация** операциясының орынланыўына алып келеди.

Егер объект бирнеше бирдей типтеги компонентаплрға ийе болса, онда бул компонентаплрды избе-из қайта ислеўди талап етиўши операцияға **итерация** деп атаймыз.

Объектлер үстинде орынланатуғын операциялар дизимин келтиремиз:

Объектти дүзиў;

Объектти жоқ етиў;

модификация ;

селекция ;

итерация.

Солай етип объект тәрепинен қабыл етилген ҳәр бир хабар, жоқарыда көрсетилген операциялардың биреўи ямаса бирнешеўиниң орынланыўына себеп болады.

Объектлер үстинде операциялар типлери

§2. Классларды дүзиўдиң программалық қураллары

Декомпозиция нәтийжесинде алынған объектлерди әмелге асырыў принципинде қәлеген тилде, ҳәттеки ассамблер тилинде де орынлаў мүмкин. Бирақ арнаўлы қураллардың бар болыўы, программаластырыўды аңсатластырады, библиотекадағы таяр дүзилген класслар менен тәмийнлейди.

Объектлердиң абстракцияларын аңлатыў ушын программист тәрепинен мағлыўматлардың арнаўлы типии –**класс** қолланылады.

Класс – бул мағлыўматлардың структурланған типи болып, ол мағлыўматлар майданлары тәрийпин ҳәм усы майданлар менен жумыс ислеўши функция, процедураларды өз ишине алады.

Классларға қарата бул процедура ҳәм функцияларды методлар деп атаймыз. Мағлыўматлар ҳәм оларды қайта ислеў усылларын бирлестирип әмелге асырыў классларды реал объектлердиң моделлериниң аҳўалы ҳәм ис-ҳәректлерин сүўретлеўге қолай етеди. Бунда класстың майданлар жыйыны объекттиң аҳўаллары менен, ал методлар жыйыны объекттиң ис-ҳәрекетлер топламы менен анықланады.

Программаларда класс типиндеги өзгериўшилер пайдаланылады ҳәм оларды объектлер деп атаў қабыл етилген.

Мысал 1. Класстың тәрийпи(Айна классы). Мейли сондай классты ислеп шығыў керек болсын, оның өзгериўшилери экранда берилген рең ҳәм өлшемге ийе төртмүйешликти сызыўда қолланылсын. Проектленетуғын класстың майданы болыўы тийис ҳәм онда айнаның параметрлери: XI, Y1, X2, Y2 – айнаның шеп жоқарғы, оң төменги мүйешлериниң координаталары, Color – реңи сақланады. Соның менен бирге класста айнаны сызыўшы ҳәм параметрлердиң басланғыш мәнислерин киритиўши методлар болыўы тийис. Солай етип төмендеги структураға ийе классты аламыз:

Класс Окно:

поля X1,Y1,X2,Y2, Color

метод Инициализировать(aX 1 ,aY 1 ,aX2,aY2,aCo1or)

метод Изобразить

Конец описания.

Рухсатты шеклеў. Көпшилик объектке бағдарланған тиллер объекттиң базы майданлары ҳәм методларына рухсатты шеклеўге мүмкиншилик береди. Буның ушын арнаўлы қураллар жәрдеминде класстың интерфейси ҳәм әмелге асырыў бөлими ажыратылады. Тилден ғәрезсиз улыўма жағдайда класстың сүўретлениўи төмендегише болады:

Класс <имя класса>

интерфейс

<объявление полей и методов класса, к которым возможно обращение извне>

реализация

<объявление полей и методов класса, к которым невозможно обращение извне>

Конец описания.

Жоқарыда айтып кеткенимиздей мағлыўматлар майданын ҳәм олар менен жумыс ислеўши методларды бир пакетке бирлестириў ҳәм оларға рухсат етиўдиң арнаўлы қағыйдаларының бар болыўы инкапсуляция деп аталады.

Бир тәрептен интерфейстиң болыўы сырттан класстың объектлерин бузыўдан сақлайды. Екинши тәрептен объектлерге рухсат интерфейс тәрепинен регламентленетуғын болса, онда программада қәлеген объекттиң дүзилиўин, ядта сақланыўын ҳәм жумыс тамамланғаннан кейин жоқ етилиўин де есапқа алыўға туўра келеди. Объектлердиң дүзилиўи ҳәм жоқ етилиўи статикалық ямаса динамикалық түрде орынланады. Объектлердиң статикалық дүзилиўи программаны компиляциялаў ўақтында, ал оны жоқ етиў программа тамамланғаннан кейин программа менен бирге ядтан өшириледи.

Объектлердиң динамикалық дүзилиўи ҳәм жоқ етилиўи арнаўлы командалар жәрдеминде программаның жумыс ислеў процессинде

оынланады. Объектти дүзиў ҳәм оның майданларын инициализациялаў әмеллери объектти конструкторлаў деп аталса, объектти жоқ етиў әмели деструкторлаў деп аталады. Усы әмеллерди өз ишине алыўшы методларға сәйкес конструктор ўәм деструкторлар деп атайды.

Классларды ислеп шығыўдың тийкарғы қураллары

ОБП ны қоллаўшы тиллер нәсиллик, композиция, толықтырыў, полиморфизм механизмлерин әмелге асырыў есабынан жаңа классларды ислеп шығыўда жумысты әдеўир аңсатластырады.

Нәсиллик. ОБП да жаңа қурамалырақ классларды бар болған классларға майданларды қосыў ҳәм жаңа методларды анықлаў жолы менен дүзиў мүмкиншилиги жаратылған(иерархиялық принципи). Бунда базалық классқа ата-ана класс деп, ал оннан келип шыққан классқа әўлад класс деп атаймыз. Нәсилликтиң арнаўлы механизми класс әўладқа бир ямаса бирнеше ата-аналардың майданлары ҳәм методларынан пайдаланыўға мүмкиншилик береди. Егер класстың ата-анасы жалғыз болса, онда әпиўайы деп, ал бирнеше болса көп түрли деп аталады ҳәм олар иерархия дүзеди.

Нәсиллик механизминиң бар болыўы класста алдын тәрийпленген объекттиң қәсийетлери ҳәм параметрлерин қайтадан тәрийплемеўге мүмкиншилик береди, яғный олар жаңа классқа нәсилликке бериледи.

Мысал 1.8. Нәсиллик (реңин өзгертиўши класс Окно). Айна классының тийкарында оның әўладын дүземиз, оның хызмети экранда айнаның реңин өзгертетуғын болсын. Оның ушын ата-ана классқа Изменить_цвет деген методты қосыў жеткиликли:

Класс Окно_меняющее_цвет - родитель: класс Окно:

метод Изменить_цвет(aCo1or);

Конец описания.

Әпиўайы полиморфизм. Класслардың иерархиясын дүзгенде объектлердиң базыбир қәсийетлери атларын сақлағаны менен мазмуны өзгерген болыўы мүмкин. Бундай иерархияны әмелге асырыўда программаластырыў тилинде полиморфизм механизми болыўы тийис, яғный

ҳәр қыйлы иерархиялық дәрежедеги класслар ушын аты бирдей болған методтың ҳәр қыйлы әмелге асырыўларын тәмийнлеўши мүмкиншилик болыўы керек. ОБП да бундай полиморфизм әпиўайы деп аталады.

Полиморфизмди әмелге асырыўға байланыслы бирнеше терминлер пайдаланылады:

таза полиморфизм – бир функцияның коды оның аргументлери типлерине байланыслы ҳәр қыйлы түсиндирилиўи мүмкин, мәселен LISP ямаса SMALLTALK тиллеринде;

қайта жүклениўши (функциялардың полиморф атлары) – бирнеше функция бирдей ат пенен қолланылады ҳәм керекли функция аргументлери типлери ҳәм көриниў области менен анықланады. Егер аргумент типлери менен аықланса параметрли қайта жүклениў деп аталады;

қайта анықлаў (әпиўайы полиформизм) – ОБП да класслар иерархиясында методлардың ҳәр қыйлы анықламалары болғанда пайдаланылады ҳәм конкрет метод программаны компиляциялағанда оның типи менен анықланады(ерте байланыс). Бундай методлар статикалық полиморф деп аталады*;*

полиморф объектлер (қурамалы полиморфизм) - ОБП да класслар иерархиясында методлардың ҳәр қыйлы анықламалары болғанда пайдаланылады ҳәм конкрет метод программаны орынлағанда оның типи менен анықланады(кеш байланыс). Бундай методлар виртуал полиморф деп аталады*;*

улыўмаласқан функциялар ямаса шаблонлар - ОБП да параметрленген классларды әмелге асырыўда қолланылады (мәселен, C++ тилинде), бундай класстың параметрлери класстың методлары аргументлериниң типлеринен ибарат болады.

Мысал 1.9. Әпиўайы полиморфизм (класс Окно_с_текстом). Мейли Окно классы базасында Окно с текстом классын дүзиў керек болсын. Оның ушын Окно классының майданларына арнаўлы майдан қосыўымыз тийис, онда текстиң биринши ҳәрибиниң координатасы - Xt, Yt ҳәм Text тиң өзин

жазыў ушын майдан болсын. Және қосымша оны қайта ислеўши метод болыўы тийис.

Класс Окно_с_текстом - родитель: класс Окно:

поля Xt, Yt, Text

метод Инициализировать (aXl,aYl,aX2,aY2,aColor,aXt, aYt, aText)

метод Изобразить

Конец описания.

Қурамалы Полиморфизм ямаса полиморф объектлерди жаратыў.

Полиморф объектлер ямаса полиморф өзгериўшилер деп программаны орынлаў процессинде оның типинен басқа типтеги мәнисти бериў мүмкин болған өзгериўшилерге айтамыз.

Полиморф объекттиң типии программаны орынлаў этапында белгили болады, сонлықтан тилде объекттиң типин программаны орынлаў ўақтында анықлаўшы кеш байланыс механизми әмелге асырылыўы тийис.

Кеш байланыс әмелге асырылыўшы методларға виртуал деп атаймыз ҳәм оның ушын **virtual** хызмет сөзи қолланылады**.**

Кеш байланыс механизмин әмелге асырыў *таблицы виртуальных методов* (ТВМ) арнаўлы таблицалар жәрдеминде орынланады. ТВМ меншик ямаса нәсилликке ийе виртуал методлары бар болған ҳәр бир класс ушын дүзиледи ҳәм ол виртуал методлар адресинен турады.

Композиция. Объектлик декомпозиция нәтийжесинде бириниң ишине бири жайласқан объектлер алыныўы мүмкин. Бундай объектлерди әмелге асырыўшы класслар еки жол менен дүзилиўи мүмкин: нәсиллик ямаса композиция жолы. Композиция жолы менен классларды дүзиў класслардың дүзилиси уқсас болмағанда ямаса нәсилликти қолланыў мақсетке туўра келмегенде қолланылады.

Композиция деп класслар арасындағы бири екиншисиниң бөлеги болған қатнасқа айтылады. Конкрет жағдайда, композиция классқа басқа класстың объекти болған майдан киритиў жолы менен әмелге асырылады. Бундай майданларды объектлик майданлар деп атайды.

Толықтырыў. Базыбир классқа объектлерди қосыўды усы объектлердиң көрсеткишлеринен пайдаланып әмелге асырыўға да болады. Бул жолдың, классқа анық көрсетилген сандағы объектлерди қосыўды талап етиўши, объектлик майданларды қосыў жолынан айырмашылығы көрсеткишлерди пайдаланыў 0 ямаса көбирек объектлерди қосыўға мүмкиншилик береди. Толықтырыў механизми тийкарынан объектти ямаса объектлер структурасын оларды басқарыўшы базыбир классқа қосыў ушын пайдаланылады..

Классларды жаратыўдың қосымша қураллары

Алдыңғы лекцияда қаралған классларды ислеп шығыў қураллары тийкарғы есапланады. Олар ОБП принциплеринен келип шығады. Бирақ ҳәзирги ўақытқа келип ОБП ның қурамалырақ моделлери пайда болды ҳәм олар метакласслар, контейнер класслар, методларды ўәкилликке бериў, параметрленген класслар түсиниклерин өз ишине алады. Программаластырыў тиллеринде бундай қураллардың пайда болыўы эффективрек программаларды дүзиўге мүмкиншилик береди.

Метакласслар. Полиморф объектлер идеясының буннан былай рәўажланыўы жоқарырақ дәрежедеги абстракцияның – метакласслардың пайда болыўына алып келди. *Метакласс* - тип, болып оның мәниси класслар есапланады, яғный типке силтеме усаған. Метакласс типиндеги өзгериўшилерди классты ашық көрсетиў орнына пайдаланыўға болады.

Метакласслардың әмелге асырыў арнаўлы таблицаларды пайдаланыўға тийкарланған ҳәм оларда класс ҳаққында информация: класс аты, ата анна класс аты, методлар адреси сақланады. Бул таблицалар программаны орынланыў ўақтында пайдаланылатуғын болғанлықтан RTTI (Run Time Type Information - «орынланыў ўақыты типии информациясы») деп аталады.

Бул таблицалар төмендеги операцияларды әмелге асырыўда қолланылады:

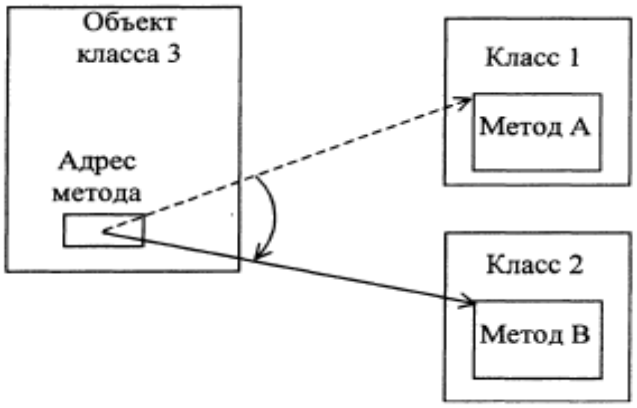
Объекттиң берилген классқа ямаса оның әўладларына тийислигин тексериў операциясы;

Объекттиң классын анықлаў операциясы – типти ашық қайта анықлаў операциясынан өзгешелиги қайта анықлаўдан алдын объекттиң берилген классқа жатыўы тексериледи;

Класстың қәсийетин тексериў ямаса анықлаў, жетилистириўдиң арнаўлы операциялары – бул операцияларды әмелге асырыў ушын программаластырыў тили класстың объектлери жоқ болғанда оған мүражат етиў(шақырыў) мүмкин болыўы ушын класстың майданлары ҳәм методларын тәрийплеў мүмкиншилигине ийе болыўы тийис.

Методларды ўәкилликке бериў. ***Ўәкилликке бериў*** – басқа класслар объектлериниң методларын пайдаланыўға айтамыз. Бул полиморф объектлерде қолланылатуғын методларды қайта анықлаўға альтернатив усыл болып есапланады. Қайта анықлаўдан өзгешелиги, ўәкилликке бериў бир классқа жатыўшы объектлердиң ҳәр қыйлы ис-ҳәрекетин анықлаўға мүмкиншилик береди. Бунда методларды алып пайдаланыў бир класс ямаса класслар иерархиясы шегарасында мүмкин болса, тап сондай ақ басқа иерархиядағы класслар объектлери ушында мүмкин болады.

Метод бул жағдайда көрсеткиш арқалы тиккелей болмаған түрде шақырылады. Ўәкилликке бериўди әмелге асырыў мүмкин болған тиллерде методларға көрсеткишлерди анықлаў мүмкинлиги тәмийнленген болыўы тийис. Методты тайынлаў ямаса аўмастырыў арнаўлы көрсеткишке сәйкес мәнис бериў менен әмелге асырылады.



Объектлерди ўәкилликке бериў

Ўәкилликке бериў *статикалық* ҳәм *динамикалық* деп ажыралады. Статикалық ўәкилликке бериўде сәйкес көрсеткиш программаны компиляциялаў процессинде инициализацияланады ҳәм программаны орынлаўда өзгермейди.

Динамикалық ўәкилликке бериўде көрсеткишке мәнис программаны орынлаў процессинде бериледи ҳәм жағдайға байланыслы өзгериўи мүмкин.

Статикалық ўәкилликке бериў еки жағдайда пайдаланылады:

1) егер объекттиң талап етилген ис-ҳәрекети басқа класс объектлери ушын алдын тәрийпленген болса онда ўәкилликке бериў программадағы қайталаныўлардан қутылыўға мүмкиншилик береди;

2) егер класс объектлериниң ис-ҳәрекети толық анықланбаған ҳалда жәрияланған болса (әдетте бундай етип базыбир билиотекалық класслар тәрийпленеди) ҳәм оның ис-ҳәрекети объектлердиң конкрет экземплярлары ушын анықластырылады.

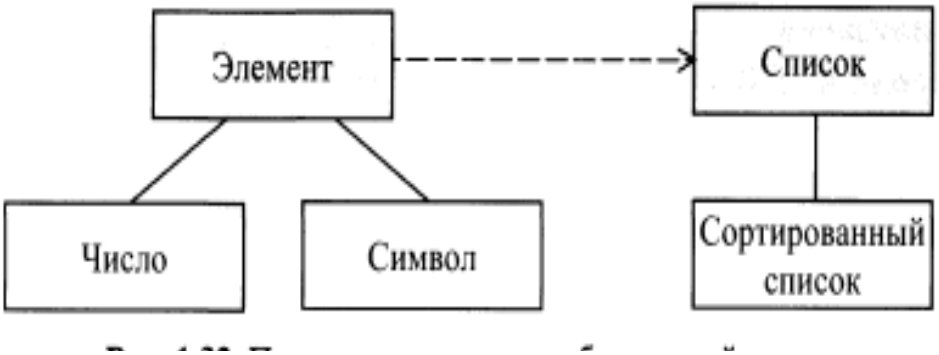
Динамикалық ўәкилликке бериў ис-ҳәрекети өзгерип турыўшы объектлерди дүзиўде қолланылады, яғный ис-ҳәрекеттиң конкрет варианты базыбир сыртқы жағдайлар менен анықланғанда.

Ўәкилликке бериўди пайдаланыўда соны гүзетиў зәрүр болады, яғный методтың басламасы методтың адресин сақлаўшы көрсеткиш типине сәйкес келиўи тийис.

Контейнер класслар. *Контейнерлер* - бул арнаўлы түрде шөлкемлестирилген объектлер болып, басқа класслардың объектлерин сақлаў ушын қолланылады.

Контейнерлерди әмелге асырыў ушын арнаўлы контейнер класслар ислеп шығылады. Контейнер класс әдетте бөлек объект ҳәм объектлер топары менен базыбир операцияларды орынлаўға мүмкиншилик беретуғын методлар жыйынын өз ишине алады. Контейнерлер көринисинде қағыйдаға муўапық қурамалы дүзилиске ийе мағлыўматлар әмелге асырылады (дизимлердиң ҳәр қыйлы түрлери, динамикалық массивлер ҳәм т.б.). Ислеп шығыўшы класс-элементтен нәсилликке класс алып, оған керекли

информациялық майданларды қосыў арқалы талап етилген структураға ийе болады. Зәрурлик болған жағдайларда ол контейнер класстан да нәсилликке класс алып, оған өзиниң методларын қосыўы мүмкин. (рис. 1.32).



Контейнер класс ҳәм элемент класс базасында класс дүзиў

§3. BORLAND PASCAL 7.0 ҳәм Borland C++ 3.1 тиллеринде объектке бағдарланған программаластырыў қураллары

BORLAND PASCAL 7.0 тилинде ең әпиўайы объектлик модел қолланылады. Бул моделде классларды әмелге асырыўды жасырыўдың арнаўлы қураллары ҳәм классларды ислеп шығыўдың қосымша қураллары жоқ. Сондай болсада, усы параграфта, усындай әпиўайыластырылған моделдиңде алдыңғы программаластырыў усылларынан артықмашлыққа ийе екенин көрсетиў тийкарғы мақсет есапланады.

Классты анықлаў.

Солай етип, алдын айтқанымыздай ОБП да класс дегенимиз – мағлыўматлардың структурланған типи болып, усы мағлыўматлар майданы ҳәм олар менен жумыс ислеўши методларды өз ишине алады. Borland Pascal 7.0 тилинде классты анықлаў еки этапта өткериледи. Биринши этапта класстың дүзилиси тәрийпленеди ҳәм онда класс аты, мағлыўматлар майданы ҳәм методлардың прототиплери көрсетиледи:

Type

<имя класса> = **object**

<имя поля данных 1> : <тип данных 1>;

<имя поля данных N> : < тип данных N>;

Procedure <имя метода 1>(<список параметров>);

Function <имя метода 2>(<список параметров>):<тип функции>;

Procedure <имя метода Ь>(<список параметров>);

Function <имя метода М>(<список параметров>):<тип функции>;

End;

Класстың мағлыўматлар майданы төмендеги типлерде болыўы мүмкин *{byte, shortint, word, integer, longint, real, single, double, extended, comp);* логикалықо *{boolean);* символлық *{char); қатарлық {string);* адреслик *(pointer);* диапазон типтеги; көплик; массив; файл *{text, file, file of..,);* класс; класс көрсеткиши ҳ.б.

Класстың методлары болып Borland Pascal 7.0 тилдиң функция ҳәм процедуралары хызмет етеди. Екинши этапта методлардың денелери тәрийпленеди ҳәм онда методтың аты алдынан точка қойылып оннан алдын класс аты жазылады ҳәм бунда параметрлер дизими, функция типии келтирилмейди.

Procedure < класс аты>.<метод аты>;

< локал өзгериўшилер, процедура ҳәм функциялар тәрийпи>

Begin <операторлар> **End;**

ямаса

Function < класс аты>.<метод аты>;

< локал өзгериўшилер, процедура ҳәм функциялар тәрийпи>

Begin <операторлар> **End;**

Мағлыўматлар ҳәм методлар бир класста тәрийпленеди ҳәм бул тәрийплеў мағлыўматларды олар менен жумыс ислеўши методлар менен бирлестиреди. Бунда класс ишинде тәрийпленген ҳәмме мағлыўматлар оның методларына қарағанда автомат глобал өзгериўшиге айланады, яғный класстың процедуралары ҳәм функциялары ушын улыўма рухсат етилген болады.

Жаңа тип болған - классты тәрийплеп усы класстың(типтиң) өзгериўшисин жәриялаў мүмкиншилигине ийе боламыз. Класс типиндеги өзгериўшини оның экзепляры ямаса объект деп атаў қабыл етилген. Объектлерди статикалық ҳәм динамикалық ядта да дүзиў мүмкин ҳәм керек болғанда объектлерден массивлер, дизимлер пайда етиў, оларды ямаса оларға көрсеткишлерди басқа классларға киритиў мүмкин.

Класстың майданлары ҳәм методларын шақырыў жазыўлардың майданларын шақырыўға уқсас болып, өзгериўши(объект) атын көрсетиў арқалы төмендегише орынланады:

- Қурамалы ат арқалы: <имя объекта>.<имя поля>

ямаса

<имя объекта>.<имя метода>(<список фактических параметров>);

- оператора присоединения with тиркеў операторы арқалы:

with <имя объекта> **do**

begin

... <имя поля>...

... <имя метода> (<список фактических параметров>)...

end;

Объектлер майданларын инициализациялаўда үш усылдан пайдаланыў мүмкин:

- тийкарғы программада майданларды инициализациялаўда мәнис бериў операторы жәрдеминде:

<объект аты>.<майдан аты> := <мәниси>

- инициализациялаўдың арнаўлы методын жаратыў ҳәм ол параметрлер дизимин алып олар арқалы майданларды инициализациялайды (бундай метод әдетте Init деп аталады);

- мына түрдеги типлестирилген турақлылардан пайдаланыў арқалы:

Const <имя объекта> : <имя класса> =

(<имя поля 1> : <значение поля 1>;

<имя поля 2> : <значение поля 2>;

<имя поля К>:<значение поля К>);

Бул усыллардан қайсысын таңлаў шешилетуғын мәселе өзгешелигине байланыслы өткериледи.

Мысал 1. Классты тәрийплеў (Окно классы). Мейли текстли режимдеги айнаны дүзиўши классты Win тәрийплеў талап етилсин. Бундай класстың тийкарғы методы болып айнаны сызыўшы процедура есапланады ҳәм ол Crt модулиниң стандарт процедураларын: Window, Textbackground ҳәм Clrscr шақырады. Буған қосымша айна өлшемлерин анықлаўшы еки метод қосқан мақул.

Нәсиллик.

Нәсиллик анықламасы алдыңғы лекцияларымызда анықланған болып, ол бойынша бир класс екинши класстың структуралық ямаса функционаллық бөлегинен пайдаланса әпиўайы нәсиллик деп, ал бирнеше класстың

структуралық ямаса функционаллық бөлегинен пайдаланса көп түрли нәсиллик деп атаған едик.

Borland Pascal 7.0 тилинде әпиўайы нәсиллик әмелге асырылған болып, ол бойынша класстың тек бир ғана ата-анасы болып, ал әўладлары қәлеген сандағы болыўы мүмкин. Қәлеген классты алдын тәрийпленген класстың әўлады деп жәриялаў мүмкин. Нәсиллик механизминен пайдаланыў нәтийжесинде ата-ана класстың майданлары ҳәм методларын қайтадан жазыўдан қутыламыз ҳәм оларда жоқ методлар ҳәм майданларды ғана тәрийплеп өтемиз.

Класстың ишинде оның ата-анасы методларын шақырыў өзиниң методларын шақырыўға уқсас болады. Класслар иерархиясында методты излеў төмендегише орынланады:

1. Биринши гезекте компилятор объект типин анықлайды.
2. Оннан кейин методты объекттиң класынан излейди егер тапса иске қосады.
3. Егер таппаса, онда метод класстың ата-анасынан изленеди, табылса шақырылады.
4. Егер ата-ана класстан табылмаса, онда оның ата-анасынан ҳәм онннан жоқарылардан изленеди, олардан да табылмаса компилятор 44 номерли қәтени көрсетеди.

Полиморфизм.

Borland Pascal тили әпиўайы ҳәм қурамалы полиморфизм механизмлерин әмелге асырады. Әпиўайы полиморфизм механизми ҳәр қыйлы дәрежедеги иерархиялы класслар ушын аты бирдей болған базыбир методтың ҳәр қыйлы әмелге асырылыўын тәмийнлейди. Бул жағдайда бирдей методлар статикалық полиморф деп аталады.

Мысал. Әпиўайы полиморфизм қолланылыўы. Мейли айнаға бирдей символлар избе-излигин шығарыўшы программа дүзиў талап етилсин. Керекли классты дүзиў ушын бар Win и Symb классларынан пайдаланамыз. Lot классын Symb классы әўлады деп жәриялаймыз ҳәм Symb классының

Print методын алмастырыўды қараймыз, себеби ол баспаға бир символды шығара алады. Ал, Lot классында бул Print методы N майданда көрсетилген сандағы символлар избе излигин баспаға шығарыўы тийис.

Қурамалы полиморфизм. Қурамалы полиморфизмде полиморф методтың конкрет жағдайы ол шақырылған объекттиң типи менен тек программа орынланыў ўақтында анықланады (Кеш байланыс). Кеш байланыс зәрурлиги келип шығыўы Borland Pascal тилинде ата-ана класс объектлерине көрсеткишке әўлад класс объектлериниң адреслерин мәнис етип бериў рухсат етилген. Кеш байланыстан пайдаланыў полиморф методтың керекли жағдайынан пайдаланыўға мүмкиншилик береди. Кеш байланыс қолланылыўшы полиморф методлар виртуаль деп аталады ҳәм оларды тәрийплеў ушын **virtual** хызмет сөзинен пайдаланылады**:**

Procedure <имя метода>(<список параметров>); **virtual;**

Function <имя метода> (<список параметров>):<тип функции>; **virtual;**

Виртуал методлардан пайдаланыўда төмендеги қағыйдалар орынланыўы тийис:

1. Егер базы бир класста метод виртуал деп жәрияланған болса, онда оның әўладларында да ол метод виртуал деп тәрийпленiўи тийис, виртуал методты статикалық метод пенен алмастырыўға болмайды.
2. Бирдей аттағы виртуал методларда формал параметрлердиң жайласыў тәртиби, саны ҳәм типлери өзгермеўи тийис.
3. Виртуал методларды өз ишине алыўшы класс арнаўлы статикалық метод – конструкторға ийе болыўы тийис. Оның ушын бул методта *Procedure* сөзи *Constructor* сөзи менен алмастырылыўы тийис.

Виртуал методлардың қолланылыўы үш жағдайда шәрт болады:

1. полиморф метод ата-ана класстың статикалық методынан шақырылса.
2. программада класс типиндеги параметрли функция ямаса процедура қолланылса ҳәм оның фактлик параметри ретинде әўлад класстың объекти берилиўи мүмкин болса ҳәм усы функция полиморф методты шақырған болса.

3. программада базалық классқа көрсеткиш пайдаланылып, оған әўлад класстың объекти адреси мәнис етип берилиўи мүмкин болса ҳәм полиморф метод усы көрсеткиш бойынша шақырылса.

BORLAND PASCAL 7.0 орталығында динамикалық объектлер ҳәм класслар библиотекасы

Алдын айтқанымыздай классты тәрийплеўден кейин биз бул класстың өзгериўшилерин жәриялаўымыз ҳәм оларды статикалық ҳәм динамикалық ядта жайластырыўымыз мүмкин. Динамикалық объектти жәриялаў ушын класс объектине көрсеткишлер пайдаланылады ҳәм төмендегише тәрийпленеди:

Var <имя указателя> : ^<имя класса>;

Объектти динамикалық ядта жайластырыў ушын New процедура ямаса функциясынан пайдаланылады:

Процедура **New** (<имя указателя>);

Функция <имя указателя>:=New(<тип «указатель на класс»>).

Borland Pascal 7.0 тилинде New процедура ямаса функциясының кеңейтилген вариантынан пайдаланыў мүмкиншилиги бар болып, бир операцияда объектке яд ажыратылады ҳәм конструктор шақырылады:

Процедура **New** (<имя указателя>, <имя конструктора>(<список параметров>));

Функция <имя указателя>: = New(<тип «указатель на класс»>,<имя конструктора>(<список параметров>))

New процедура ямаса функциясының бундай варианты мына тақлетте жумыс ислейди. Дәслеп объектке яд ажыратыў ҳәрекети әмелге асырылады ҳәм соңынан конструктор шақырылады.

Динамикалық объектлер ийелеген ядты босатыў ушын стандарт процедура қолланылады:

Dispose(<имя указателя>);

Егер объект классы виртуаль методларды өз ишине алса, онда объектти жоқ етиўде арнаўлы метод *–деструктор* шақырылыўы шәрт.

Класслар библиотекасын дүзиў.

Класслар библиотекасын дүзиўде классларды әмелге асырыўдың деталларын жасырыў мақсетке муўапық болады. Бул жағдайда класслардың тәрийпин модулдиң интерфейс бөлиминде орынлаў, ал методлар денелерин әмелге асырыў бөлиминде орынлаў мүмкин. Усындай жол менен әмелге асырыў бөлиминде толық анықланыўшы ишки классларды да тәрийплеў мүмкин.

Библиотека тек ғана классларды ғана емес,ал бул класслардың объектлеринде экспортқа шығарыўы мүмкин.Егер виртуал методлары бар объектлерди экспортлаў зәрур болса онда бундай өзгериўшилер ушын инициализация бөлиминде конструкторларды шақырыўды жайластырыў керек.

private стандарт директивасы жәрдеминде класстың майданлар, методлар бөлегин модул пайдаланыўшыларынан жасырын етип жәриялаў мүмкин, онда оларға класс жәрияланған модул ишинде рухсат берилген болады.

Класс тәрийпи ҳәм экспортқа шығарылыўшы объектлер бар программа модули структурасы мына түрде болады:

Unit <модул аты>;

Interface {интерфейс бөлими}

Type

< класс аты> = object

private <жасырын майданлар>;

public <ашық майданлар>;

private < объектлик методлар>;

public <ашық методлар>;

End;

Var <класстың экспортлық объектлерин жәриялаў>

Implementation {әмелге асырыў бөлими}

{методларды әмелге асырыў}

Begin {инициализация бөлими}

{ экспортлық объектлер конструкторын шақырыў}

End.

Композиция ҳәм толықтырыў

Бир бири менен ишине алыў қатнасында болған объектлерди әмелге асырыў ушын нәсиллик ямаса композиция механизми пайдаланылыўы мүмкин. Композиция деп класслар арасындағы сондай қатнасқа айтамыз егер олардың бири екиншисиниң бөлими болса. Композиция классқа объектлик майданларды киритиў жолы менен әмелге асырылады ҳәм нәсиллик мүмкин болмағанда қолланылады.

Borland Pascal 7.0 тили классларда объектлик майданлардан пайдаланыў мүмкиншилигин береди ҳәм оны мысалда көремиз.

Мысал. Объектлик майданлардан пайдаланыў. Шығарылатуғын қатар (OutS) Окно (Win) классының бөлими етип жәрияланған. Буны әмелге асырыў Win классы структурасына ObField объектлик майданды киритиў арқалы әмелге асырылған.

Borland C++ 3.1орталығында ОБП қураллары. Класслар, нәсиллик ҳәм полиморфизм.

BORLAND C++ 3.1 тилинде қолланылыўшы объектлик модел Borland Pascal 7.0 тилие салыстырғанда программалаўшыға көбирек мүмкиншиликлер береди, яғный бул тилде класстың ишки компоненталарына рухсатты шеклеўдиң, операцияларды қайта жүклеўдиң, шаблонлар жаратыўдың исенимлирек қураллары әмелге асырылған.

Класстың анықланыўы.

Басқа тиллердегидей BORLAND C++ 3.1 тилинде де класс структурланған тип болып, улыўма қәсийетке ҳәм ис-ҳәректке ийе болған предмет тараўының базыбир объектлер көплигин тәрийплеў ушын қолланылады. Ол төмендегише сүўретленеди:

class <имя класса>

{ private: <внутренние (недоступные) компоненты класса>;

protected: <защищенные компоненты класса>;

public: <общие (доступные) компоненты класса>;

};

Классты тәрийплеўде компонентлер ретинде объектлердиң параметрлерин сақлаўшы майданлар ҳәм олар менен жумыс ислеў қағыйдаларын тәрийплеўши функциялар қатнасады.

Private секциясында жәрияланған класстың компоненталары ишки деп аталады ҳәм олар усы класстың функциялары ҳәм оған дос болған класс функциялары ушын рухсат етилген болады.

Protected секциясында жәрияланған класстың компоненталары қорғалған болады ҳәм олар усы класстың функциялары ҳәм оған әўлад болған класс функциялары ушын рухсат етилген болады.

Public секциясында жәрияланған класстың компоненталары улыўмалық деп аталады ҳәм олар класстың сыртына программаның қәлеген жеринде рухсат етилген болады ҳәм тек усы секцияда класстың интерфейс бөлиминиң майданлары, методлары жәрияланады.

Класстың майданлары ҳәмме ўақытта класс ишинде тәрийпленетуғын болса, ал функциялары иште ҳәм сыртта да тәрийплениўи мүмкин. Сыртта тәрийпленсе төмендегише жазылады:

<тип функции> **<имя класса>::** <имя функции>(<список параметров>)

{<тело компонентной функции>}

C++ тили қағыйдасы бойынша компонентли функция денеси класс тәрийпи ишинде жайласса онда бул функция ишке қосылған (inline) деп аталады.

```
#include <stdio.h>
#include <conio.h>
 class X
  {private:     char c;
     public:     int x,y;
 /* определение встраиваемых компонентных функций внутри
описания класса */
```

```
void print{void)
 { clrscr();
Gotoxy(x,y);   printf("%c",c);
 x=x+10;  y=y+5;
 gotoxy(x,y);   printf("%c",c);
   }
    voidset_X(char ach,int ax,int ay){c=ach;   x=ax;  y=ay; }
   };
```

Мысал. Классты анықлаў.(Строка классы). Мейли символлар қатарын сақлаўшы ҳәм шығарыўшы классты тәрийплеў талап етилсин.

```
#include <iostream.h>
#include <string.h>
class String // начало описания класса
{ private:   char str[25];   //поле класса-строка из 25 символов
public:
      // прототипы компонентных функций (методов)
     void set_str (char *);   // инициализация строки
    void disploy_str(void); // вывод строки на экран
    char * return strfvoid); // получение содержимого строки
};
// описание компонентных функций вне класса
voidString::set_str(char * s)   {strcpyfstr^);}
void String: :display_str(void) {cout « str « endl;}
char * String::return_str(void) {return str;}
```

Класстың объекти анықланғаннан кейин, оның майданлары ҳәм функцияларын шақырыў толық атларын жазыў арқалы

<имя объекта>.<имя класса>: :<имя поля или функции>.

Мәселен:

a.String::str;

b->String::setjtr(stl);

c[i\.String: .displayjtr().

Ямаса қысқа атлар жәрдеминде әмелге асырылады

<имя объекта>.<имя поля или функции>;

<имя указателя на объект>-><имя поля или функщш>;

<имя объекта>[индекс]. <имя поля или функции>.

Мәселен:

a.str	*b->str*	*c[i].str ;*
a.display_str()	*b->display_str()*	*c[i].display_str().*

Объектлерди дүзиў ҳәм жоқ етиўде төмендеги қағыйдалар сақланады:

- глобал ҳәм локал статикалық объектлер **main** функциясын шақырыўдан алдын дүзиледи ҳәм программа тамамланғаннан кейин жоқ етиледи.
- Автомат объектлер оларды ҳәр сапары жәриялағанда дүзиледи ҳәм олар пайда болған функциядан шыққанда жоқ етиледи.

Яд динамикалық түрде ажыратылатуғын объект **new функциясы жәрдеминде дүзиледи ҳәм delete** функциясы жәрдеминде жоқ етиледи..

Объектлер майданларын инициализациялаў. Класстың майданларын жәриялаўда оларды инициализациялаў рухсат етилмейди, себеби оны тәрийплеў ўақтында оған яд ажыратылмаған болады. Ядты ажыратыў класс ушын емес ал оның объектлери ушын болғанлықтан майданларды инициализациялаў мүмкиншилиги конкрет класстың объекти жәрияланғаннан кейин болады. Программаны орынлаў ўақтында объекттиң майданына мәнислерди жазыў бирнеше усылда болады:

1. объект майданына мәнисти тиккелей бериў арқалы.
2. пайдаланып атырған класстың қәлеген функциясы ишинде.
3. С++ тилиниң улыўма қағыйдалары бойынша инициализациялаў операторы жәрдеминде.

This жабық параметри. Конкрет объект ушын компонентлик функция шақырылғанда бул функция ушын автомат ҳәм жабық түрде параметр ретинде шақырылып атырған объектке көрсеткиш бериледи. Бул көрсеткиш арнаўлы атқа **this** ийе болып класстың ҳәр бир функциясында константа ретинде анықланған:

<имя_класса> * const **this** = <адрес _объекта>.

This көрсеткиш функцияның қосымша жасырын параметри болып, конкрет объекттиң майданларына рухсат ушын пайдаланылады.

this->pole this->str this->fun().

Класстың статикалық компоненталары. Класс- бул тип, ал объект класстың программадағы конкрет ўәкили. Ҳәрбир объект ушын класстың майданлары копиясы бар болады. Егер бир типтеги барлық объектлер биргеликте базыбир мағлыўматларды пайдаланса онда бул мағлыўматларды жайластырыўда ҳәм класстың барлық объектлерине рухсат бериўде машқала келип шығыўы мүмкин. Буны статикалық компоненталар механизмин қолланып шешиўге болады.

Статикалық деп класстың **static** яд модификаторы жәрдеминде жәрияланған комонентаға айтамыз. Бундай компоненталар класстың бөлеги болып, бирақ класстың объектлерине кирмейди. Класстың статикалық майданларының тек бир копиясы бар болып, класстың барлық объектлерине улыўма болады.

Статикалық майданларының инициализациясы тек класстың сыртында болады, бирақ классқа тийислигин билдириўши тәриплеме менен көрсетиледи **<имя класса>::.**

Мәселен:

```
class point
{ intx.y;
Static int obj_count; /* статическое поле - (счетчик обращений) инициализация в этом месте невозможна */
public:
```

point (){x=0; y'=0; obj_count++;} // обращение к статическому полю

};

int point::obj_count=0; // инициализация статического поля

Класстың статикалық майданларынан қәлеген метод пайдаланыўы мүмкин болса да олар менен жумыс ислеўге рухсат берилген статикалық компонентлик функциялар бар болып, олар **this** параметрин алмайды, сонлықтан олар объекттиң көрсетпесиз класстың статикалық емес майданларын шақыра алмайды. Класстың статикалық компоненталарын қолланыў глобал өзгериўшилерге болған талапты азайтады.

Локал ҳәм ишпе-иш жайласқан класслар. Класс базыбир функция ишинде жәрияланса бундай классты С++ те локал деймиз. Локал класс жәрияланған функция локал класс компоненталарына тиккелей рухсатқа ийе емес, ол рухсатты ишке қосылған класстың объекттиниң атын көрсетиў арқалы алады. Локал класстың барлық компоненталық функциялары ишке қосылған болыўы тийис.

Айырым ўақытлары бир классты екинши класс ишинде жәриялаўға туўра келеди ҳәм оны ишке жайласқан деп атаймыз. Иштеги класстың компонентлик функциялары ҳәм статикалық компоненталары глобал класс сыртында тәрийпленииўи мүмкин.

Конструкторлар ҳәм деструкторлар. Конструкторлар ҳәм деструкторлар класстың арнаўлы методлары болып, класс объектлерин дүзиўде ҳәм жоқ етиўде автомат шақырылады. С++ тили қағыйдасы бойынша конструктор класс ийе болған атқа ийе болады, аргументлери болмаўы мүмкин, ҳеш қашан мәнис қайтарып бермейди ҳәм объектти дүзиўге зәрур операцияларды орынлайды.

Класстың майданларын инициализациялаўда конструктордан пайдаланыўды қараймыз бунда strl майданына рухсатты өзгертемиз. Конструктор sstr классының майданын инициализациялайды ҳәм сәйкеслик

болмаған жағдайда киритилетуғын қатардың узынлығын тексериў ҳәм қатарды дүзетиў бойынша ҳәрекетлерди әмелге асырады..

Деструкторлар. Деструктор аты алдында «~» - (префикс тильда) белгиси бар класс аты менен бир болады ҳәм ол объектти жоқ етиўге зәрур болған операцияларды орынлайды. Деструктор мәнис қайтармайды, параметрлерге ийе емес, нәсилликке берилмейди. Класс тек бир деструкторға ийе болады ямаса ийе болмайды ҳәм ол объект жоқ етилгенде автомат жабық түрде шақырылады.

Нәсиллик. C++ тилинде нәсииллик механизми қолланылады Класстың әўлады мына көринисте анықланады:

class <имя производного класса >:

<вид наследования >< имя базового класса>{<тело класса>};

Бунда нәсиллик түри **private, protected, public** гилт сөзлери менен менен анықланады. Базалық класстың майданлары ҳәм функцияларының әўлад класстан көриниўи нәсиллик түри менен анықланады ҳәм таблицада келтирилген

Таблица 1. Базалық класс компоненталарының әўладларда көриниси

Нәсиллик түри	Базалық классларда компоненталарды жәриялаў	Компоненталарының әўладларда көриниси
private	private	Рухсат жоқ
	protected	private
	public	private
protected	private	Рухсат жоқ
	protected	protected
	public	protected
public	private	Рухсат жоқ
	protected	protected
	public	public

Көп түрли нәсиллик. C++ тилинде көп түрли нәсиллик бар болып, төмендеги түрде жазылады:

class <имя производного класса>:

<вид наследования >< имя базового класса 1>,

<вид наследования >< имя базового класса 2>,

<вид наследования >< имя базового класса n>{...};

Нәсиллик түри базалық класс компоненталарына рухсат бериў режимин анықлайды.

Көп түрли нәсилликте әўлад классқа базалық класстың компоненталарын көп мәрте киритиўден сақланыў ушын виртуал нәсиллик қолланылады. Виртуал нәсилликте әўлад класс төмендегише тәрийпленеди:

c1ass<имя производного класса>:

virtual <вид наследования >< имя базового класса>{...};

Полиморфизм. C++ тилинде статикалық полиморфизм **(**ерте байланыс) функцияларды қайта анықлаў механизми жәрдеминде анықланады ҳәм бундай полиморф функциялар бир биринен қайтарылыўшы параметр типии менен ҳәм сигнатура (берилетуғын параметрлердиң типии ҳәм саны) менен ажралады.

Қурамалы полиморфизм Кеш байланыс механизми жәрдеминде әмелге асырылады ҳәм виртуал функцияларды тәрийплеўди талап етеди. Виртуал функция деп базалық класста *virtual сөзи* менен жәрияланыўшы ҳәм бир ямаса бирнеше әўлад классларда қайта анықланыўшы функцияға айтамыз. Бунда ҳәр қыйлы класстағы функция прототиплери тек ғана атлары ғана емес, ал қайтаратуғын нәтийжеси типи ҳәм сигнатурасы бойынша да бир бирине тең болыўы тийис. Егер функция типлери ҳәр қыйлы болса, олар ушын виртуаллық механизми киритилмейди. Виртуал функция базыбир класстың компонентасы болыўы шәрт. Ол басқа класс ушын дос деп жәрияланыўы мүмкин, бирақ статикалық деп жәрияланыўы мүмкин емес. Кем дегенде бир виртуал функциясы бар класс полиморф деп аталады.

§4. Delphi тилинде класслар ҳәм объектлер менен жумыс ислеў өзгешеликлери

Delphi программаластырыў тили Borland Object Pascal тили базасында жаратылған. Программалық компонентлердиң объектке – бағдарланған модели Delphi де бул модулде тийкарғы өзек болып кодтан максимал қайта пайдаланыў есапланады. Бул қолланбаларды алдыннан таярлаған объектлерден тез ислеп шығыўға ҳәм өзлериниң жаңа компонентлерин дүзиўге мүмкиншилик береди.

Ислеп шығарыўшылар ушын объекттиң типи бойынша ҳешқандай шеклеўлер орнатылмаған. Ҳақыйқатында да Delphi деги ҳәмме нәрсе Delphi тәрепинен жаратылған болып, ислеп шығарыўшылар программаластырыў орталығын дүзиўге пайдаланылған объектлерге ҳәм инструментлер менен ислеўге мүмкиншилик алады. Нәтийжеде Borland тәрепинен ямаса басқа фирмалар тәрепинен жеткерилген объектлер менен ислеп шығыўшылардың өзи жаратқан объектлери арасында ҳеш қандай айырма болмайды.

Delphi диң стандарт тәмийнлениўине 270 базалық класслар иеарархиясынан ибарат тийкарғы объектлер киреди.

Delphi де қолланбаларды корпаратив мағлыўматлар базасы ҳәм ойынлар ушын да бирдей жақсы программалар жазыў мүмкин. Көпшилик тәрептен буның себебин Windows орталығында пайдаланыўшы интерфейсин дүзиўди әмелге асырыў қуралы болып келгени менен түсиндириў мүсиндириў мүмкин. Уақыяларға байланысла модел Windows орталығында қурамалы ҳәм түсиниў қыйын еди. Керисинше Delphi де интерфейсти ислеп шығыў программист ушын аңсат мәселелрден болып қалды.

Усы себепли Delphi жәрдеминде дүзилген қолланбалар исенимли ҳәм орнықлы жумыс ислейди. Delphi бар болған объектлерди: DLL библиотекаларды, C ҳәм C++ те дүзилген объектлерди, OLE серверди, VBX объектлерди пайдаланыўды қоллап қуўатлады. Таяр компонентлерден жумыс ислеўши қолланбаны тез жыйнаў мүмкин ҳәм ислеп шығыўға кететуғын

шығынларды азайтыў ушын объектлерден қайта пайдаланыў мүмкин. Delphi ислеп шығыў командасы ушын да, жеке ислеп шығыўшылар ушын да анық архитектураны усыныс етеди, яғный компоненталарды қай жерде таярланғанына қарамастан қосып барыға мүмкиншилик береди. Олар CASE инструментлерин, код генераторларын ҳәм авторлық жәрдем программаларын қосып барыўы мүмкин.

Визуал компоненталар библиотекасы. Delphi де қолланбалар ислеп шығыўда пайдаланылатуғын компоненталар қолланбаларды ислеп шығыў орталығына киритилген болып, оларды дүзиўде фундамент ролинде пайдаланыўшы объектлер типлери жыйынан ибарат болады. Бул объектлер жыйыны Visual Component Library деп аталады.

VCL де басқарыўдың стандарт элементлери бар болып, олар редакторлаў жоллары, басқарыўдың аналитикалық элементлери, закладкалар, көп бетли жазыў китаплары компонентлери қосылады. Барлық объектлер хизмети бойынша бетлерге ажратылған ҳәм компоненталар палитрасына жайластырылған. VCL арнаўлы объектке ийе болып, ол Windows тиң графикалық интерфейслерин жеткерип береди ҳәм келип шығыўшыларға Windows орталығы деталларына кеўил болмастан сүўретлер салыўға мүмкиншилик береди.

Delphi диң тийкарғы өзгериўшилеринен бири тек ғана қолланбаларды дүзиўде визуал компонентлерден пайдаланыў ғана емес ал жаңа жаңа компоненталарды дүзиў ҳәм есапланады. Бундай мүмкиншилик ислеп шығыўшыларға басқа орталыққа өтпестен бир орталыққа жаңа инструментлер қосып барыўды тәмийнлейди.

Объектлер класслары иерархия көринисинде дүзилген болып, абстракт, аралық ҳәм таяр комплекслерден ибарат болады. Ислеп шығарыўшы таяр компоненталар пайдаланыўы, абстракт класслар тийкарында өзиниң объектлерин дүзиўи мүмкин.

Класс - бул арнаўлы түрлер болып, өзинде майдан, усыллар ҳәм қәсийетлерди бириктиреди.

Класс қурамалы структура болып, мағлыўматлар тәрийплеринен тысқары, процедура ҳәм функциялар тәрийплеринде өз ишине алады.

Әпиўайы класс тәрийпине мысал:

```
TPerson = class
private
fname: string[15];
faddress: string[35];
public
procedure Show;
end;
```

Класс мағлыўматлары майданлар, процедура ҳәм функциялар усыллар деб аталады.

Келтирилген мысалда TPerson – класс аты, fname ҳәм faddress – майданлар атлары, show - усыл аты.

Майдан - бул классқа бирлескен мағлыўматлар есапланады. Классқа қараслы майданлар әпиўайы жазыўлар майданындай болып, олардың парқы ҳәр қыйлы типте болыўы. Мәселен,

```
Type
  TchildClass=Class
      Fore: Integer;
      Ftwo: String;
      Fthree: Tobject;
End;
```

Майданларға мүрәжаат қылыў класс қәсийетлери ҳәм усыллары жәрдеминде әмелге асырылады. Майданға мүрәжаат қылыў ушын алдын класс аты жазылып, кейин ажыратыўшы точка қойылып майдан аты жазылады, Мәселен,

```
Var
MyObject: TchildClass;
Begin
```

MyObject.Fore:=16;

MyObject.Ftwo:=’қатор қиймати’;

End;

Майдан аты оған сәйкес қәсийет атының биринши ҳәриби “F” болыўы менен парықланады.

Delphi де қабыл қылынған келисим бойынша майданлар атлары f (field — майдан сөзинен) ҳәрибинен басланыўы тийис.

Усыллар

Классқа бириктирилген процедура ҳәм функцияларға усыллар дейиледи. Мәселен:

```
Type
  TchildClass=Class
      Fore: Integer;
      Ftwo: String;
      Fthree: Tobject;
      Function FirstFunc(x:Real):Real;
      Procedure SecondProc;
End;
```

Класс усыллары (класс тәрийпине киритилген процедура ҳәм функциялар) класс объектлери устинде әмел орынлайды. Усыл орынланыўы ушын объект аты ҳәм точкадан соң усыл аты көрсетилиўи тийис. Мәселен:

professor. Show;

Класс усылы тәрипленгенде класс аты ҳәм усы латы көрсетиледи. Мәселен:

```
// TPerson классы Show усылы
procedure TPerson.Show;
begin
Write ( 'Ном:' + fname + #13+ 'Адрес:' + faddress );
end;
```

Усыл денесинде объект майданларына мүрәжаат қылынғанда объект аты көрсетилмейди.

Усылға мүрәжаат қылыў программада оның атын көрсетиў менен орынланады. Мәселен:

```
Var
  MyObject: TchildClass;
  y: Real;
   Begin
  . . . . . .
  MyObject. SecondProc;
  y:=MyObject.FirstFunc(3.14);
   End;
```

Усыллар шақырылғанда шақырған объектге қолланба жеткериледи. Бул қолланбаға усыл ишинде self сөзи арқалы мүрәжаат қылыў мүмкин.

```
procedure TPerson.Tproc(Fore:Integer);
begin
self.Fore:=Fore;
end;
```

Объект

Объект - бул класстың реал нусқасы болып, мағлыўматлар ҳәм функциялардан турады. Ол программаның Var бөлиминде жәрияланады.

Мәселенан:

```
var
  student: TPerson; professor: TPerson;
```

Delphi де объект - бул динамикалық структура. Өзгериўши -объект мағлыўматларды емес, объект мағлыўматларына қолланбаны өз ишине алады. Соның ушын программалаўшы булл мағлыўматларға ядтан жай ажратыўды көзде тутыўы лазым..

Жай ажыратыў класс арнаўлы усылы - конструктор жәрдеминде әмелге асырылады. Бул усыл әдетте Create (жаратыў) атына ийе болады. Класс

тәрийпинде конструктор ушын procedure сөзи орнына constructor сөзи қолланылады.

Төменде қурамында конструктор қатнасқан TPerson классы тәрийпи келтирилген:

```
TPerson = class private
fname: string [ 15 ];
faddress: string[35];
constructor Cireate; // конструктор
public
procedure show; // усул
end;
```

Ядтан жай ажыратыў конструктор классқа қоллаў нәтийжесин мәнис сыпатында бериў арқалы әмелге асырылады. Мысал ушын

```
professor := TPerson.Create;
```

инструкциясы орынланыў нәтийжесинде professor объектине ядтан жай ажыратылады. Ядтан жай ажыратыўдан тысқары конструктор, әдетте объект майданларына басланғыш мәнислер бериў объект инициализациясы ўазыйпасын да орынлайды. Төмендеги TPerson объекти ушын конструктор мысалы келтирилген:

```
constructor TPerson.Create;
begin
fname := '';
faddress := '';
end;
```

Объект майданына мүрәжаат қылыў ушын объект аты ҳәм точкадан соң майдон аты көрсетиледи. Мәселен:

```
professor.fname
```

Объектке ажыратылған яд бөлегин босатыў ушын арнаўлы усыл деструктор Free. қолланылады. Мәселен,

```
professor.Free;
```

Инкапсуляция ҳәм объект қәсийетлери

Инкапсуляция дейилгенде объект майданларына туўрыдан туўры емес, тек класс усыллары арқалы мүрәжаат қылыўға айтылады.

Delphi тилинде объект майданларына мүрәжаат объект қәсийетлери арқалы әмелге асырылады. Объект қәсийетине мүрәжаат қылыў ушын еки усылдан пайдаланылады. Оқыў ушын ислетилетуғын функция аты Get болып, оған сәйкес қәсийет аты қосылып жазылады. Жазыў ушын қолланылатуғын усыл бир параметрли Set аты бөлек программа болып, оның атына да сәйкес қәсийет аты қосылып жазылады. Оқыў ҳәм жазыў усыллары ҳәм оның параметри де бирдей қәсийетке ийе болыўы тийис. Қәсийетти жәриялаў ушын Property, Read ҳәм Write сөзлери қолланылады. Read ҳәм Write усыл атлары болып, сәйкес түрде оқыў ҳәм жазыў ушын мөлжелленген.

Төменде еки Name ҳәм Address қәсийетлерин өз ишине алыўшы TPerson классы тәрийпи келтирилген:

```
type
TName = string[15]; TAddress = string[35];
TPerson = class
private
FName: TName;
FAddress: TAddress;
Constructor Create(Name:Tname);
Procedure Show;
Function GetName: TName;
Function GetAddress: TAddress;
Procedure SetAddress(NewAddress:TAddress);
public
Property Name: Tname
read GetName;
Property Address: TAddress
read GetAddress
```

write SetAddress;

end;

Программада student объектиниң, Address қәсийетине мәнис берилиўин төмендегише жазыў мүмкин

student.Address := 'Нөкис, Мамбетов 21, кв.3';

Сыртқы тәрептен программада қәсийетлерден пайдаланыў объект майданларынан пайдаланыўдан парық қылмайды. Бирақ қәсийетлер ҳәм объект қәсийетлери арасында принципиал паық бар: қәсийетке мәнис бериў ямаса қәсийет мәнисин оқыўда автомат түрде базыбир ўазыйпа орынлаўшы процедура шақырылады.

Программада қәсийет усылларына қосымша ўазыйпалар жүклеў мүмкин. Мәселен усыллар жәрдеминде қәсийетке берилетуғын мәнислер туўрылығын тексериў, жәрдемши процедураларды шақырыў.

Объект майданлары қәсийет сыпатында тәрийплеў майданларға мүрәжаатты шеклеўге имкан береди: мәселен тек оқыўға рухсат бериў мүмкин . Буның ушын қәсийет тәрийпинде тек оқыў усылын көрсетиў керек.

Жоқарыда келтирилген TPerson классы тәрийпинде Name қәсийетин тек оқыў мүмкин, Address — қәсийетин болса оқыў ҳәм жазыў ушын мөлшерленген.

Жазыўдан қорғалған қәсийетти объект инициализациясына орнатыў мүмкин. Төменде Tperson классы усыллары келтирилген. Булл усыллар TPerson классы объектин жаратыў ҳәм қәсийетлерине мүрәжаатты тәмийнлейди.

Constructor TPerson.Create(Name:TName);

begin

FName:=Name; end;

Function TPerson.GetName;

begin

Result:=FName; end;

function TPerson.GetAddress;

```
begin
Result:=FAddress; end;
Procedure TPerson.SetAddress(NewAddress:TAddress);
begin
if FAddress =' '
then FAddress := NewAddress;
end;
```

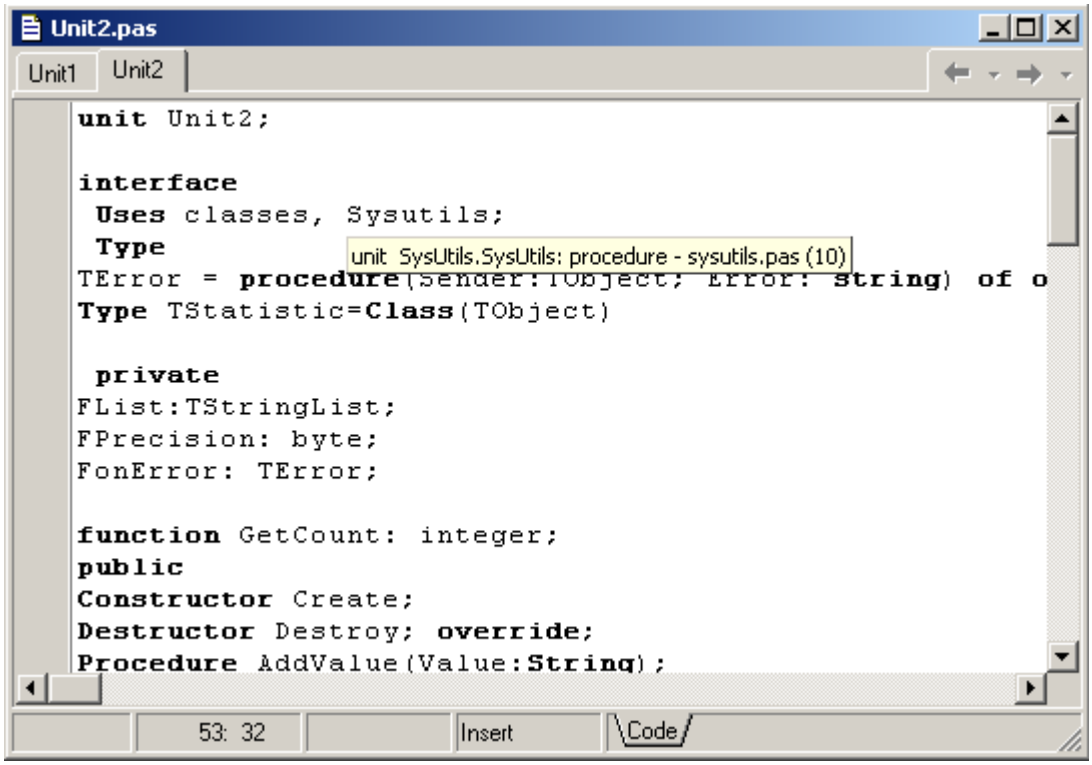
Класс объектин жаратыў ҳәм қәсийет мәнисин орнатыўға мысал:

```
student := TPerson.Create('Толепов');
student.Address := ''Нөкис, Мамбетов 21, кв.3';
```

§5. Delphi тилинде класслар дүзиўге мысаллар

Мысал. Delphi тилинде TStatistic классын дүзиў мысалын қараймыз. Бул класс киритилген санлардың квадратлары суммасын есаплайды. Бул мысалды классты қалай дүзиўди көрсетиў ушын қараймыз ҳәм ол классты дүзиўдиң усылларын көрсетеди.

Delphi ди иске қосып жаға проект айнасын ашамыз ҳәм оған File→New→Unit командасын орынлап және бир модул Unit2 ни қосамыз



Бул модулдиң ишине мына программа кодын жазамыз:

unit Unit2;

interface

Uses classes, Sysutils;

{Бизге меншикли ўақыяны жаратыў ушын процедуралық тип керек болады. Бул қандай да бир жағдайларда орынлаў ушын керек болған процедураның тәрийпи болады }

Type

TError = **procedure**(Sender:TObject; Error: string) **of object**;

{Класстың тәрийпин TObject классынан нәсилликке аламыз, себеби ата класстың ҳәзирше функционналлығы керек емес}
Type TStatistic=**Class**(TObject)
private {бул жерде тек ишки өзгериўшилер ҳәм процедуралар - "хызмет ушын пайдаланылады"}
{Майданлар тәрийпи, яғный өзгериўшилер тек класс ишинде жумыс ислейди, "сырттан" оларға рухсат жоқ.}
FList:TStringList;
FPrecision: byte;
{булда өзгериўши – ўақыяны анықлаў ушын}
FonError: TError;
{функция – класс ишинде пайдаланылады, "сырттан" тиккелей рухсат жоқ}
function GetCount: integer;
public {бунда тәрийпленгенлер класс пайдаланыўшылары ушын рухсат етилген болады}
{Конструктор – классты дүзиў усылы, оны тәрийплеў мәниске ийе егер ол арнаўлы жумыс ислесе – мәселен бизге FList өзгериўшисин дүзиў керек болады. Басқа жағдайда тәрийплеўди түсирип кетиўге болады – онда ата класс конструкторы жумыс ислейди}
Constructor Create;
{Деструктор – классты жоқ етиў усылы}
Destructor Destroy; **override**;
{Усыллар тәрийпи – усыллар процедуралардан парық қылмайды}
Procedure AddValue(Value:String);
Procedure Clear;
Function Solve:real;
{Қәсийетлер тәрийпи. Қәсийеттиң өзи ҳешқандай информацияны сақламайтуғынына дыққат аўдарамыз ол тек ишки структураға көрсеткиш. Мәселен Precision қәсийетин сақлаў ушын Precision өзгериўшиси

қолланылады. Ал Count қәсийетин оқыў ушын GetCount функциясы пайдаланылады}

Property Precision: byte read FPrecision write FPrecision;

Property Count: integer read GetCount;

{Ўақыяларды тәрийплеў. Ўақыялар дегенимиз не? – Бул процедураға көрсеткиш. Класстың өзи процедураның әмелге асырылыўын билмейди. Классқа процедураның тек басламасы белгили , биз программа кодында процедура әмелге асырылыўын жазамыз, ал класс тек керекли моменте оған басқарыўды береди, onError көрсеткишин қолланыў арқалы}

Property onError:TError read FonError write FonError;

end;

implementation

{ TStatistic }

constructor TStatistic.Create;

begin

inherited; {Дәслеп ата класс конструкторын шақырыў зәрур}

FList:=TStringList.create;{классымыз структурасын дүземиз}

end;

destructor TStatistic.Destroy;

begin

FList.Free;{ классымыз структурасын бузамыз }

inherited;{соңғы гезекте ата класс деструкторын шақырамыз}

end;

procedure TStatistic.AddValue(Value: String);

begin

FList.add(Value); {Усылды тап сондай етип әмелге асырамыз}

end;

procedure TStatistic.Clear;

```
begin FList.clear;
end;
function TStatistic.GetCount: integer;
begin Result:=FList.count+1;
end;
function TStatistic.Solve: real;
var i:integer;
begin
result:=0;
for i:=0 to FList.count-1 do
begin
try
 result:=result+(Sqr(strtofloat(FList[i])));
except
```

{мына конструкция. "on e:exception do" – жәрдеминде биз қәтелерди «e» өзгериўшиси ретинде анықлаймыз.

Бул өзгериўши пайдалы e.message қәсийетине ийе – ол қәтениң тәрийпин береди. Соңынан ўақыяны шақырыў келеди. Дәслеп пайдаланыўшы "if Assigned(FOnError) then" ўақыясын қолланып атырма сонны тексеремиз , егер пайдаланып атырған болса оның процедурасын шақырамыз: FOnError,оның параметрлери:

self – резервке қойылған өзгериўши – классымыздың экземплярына көрсеткиш, e.message – қәте тәрийпи}

```
on e:exception do
if Assigned(FOnError) then FOnError(Self, e.message);
end;
end;
end;
end.
```

Енди усы классты пайдаланыў мысалын қараймыз. Оның ушын Unit1 модулиниң ишине мына программалық кодты жазамыз:

```
unit Unit1;
interface
uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Button1: TButton;
procedure Button1Click(Sender: TObject);
private
procedure OnError(Sender:TObject; Error: string);
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
uses Unit2;
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
 var Statistic:TStatistic;
begin
Statistic:=TStatistic.create;
Statistic.onError:=onError;
Statistic.AddValue('123423');
Statistic.AddValue('123423');
showmessage(floattostr(Statistic.solve));
Statistic.Clear;
Statistic.AddValue('123423');
```

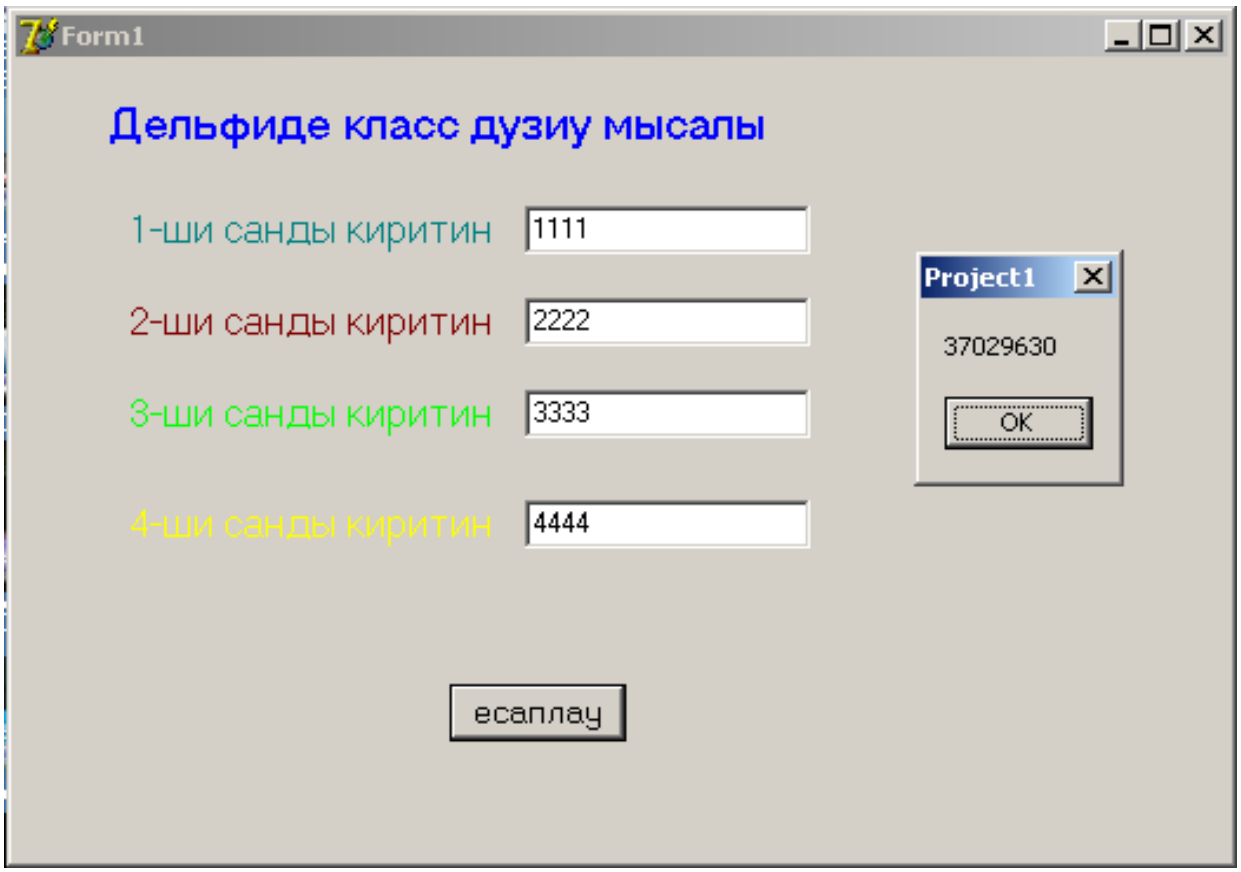
```
Statistic.AddValue('12ssss3');
showmessage(floattostr(Statistic.solve));
Statistic.Free;
end;
procedure TForm1.OnError(Sender: TObject; Error: string);
begin
showmessage('Error inside class:'+Sender.ClassName+#13#10+Error);
end;
end.
```

Бул модулде Button1 түймесиниң процедурасында биз дүзген Statistic классынан пайдаланып бирнеше санның квадратларының қосындысын есаплаў программасы коды жазылған. Санларды кириткенде қәте жиберилген жағдайда оны қайта ислеў ҳәм қәте түрин көрсетиў процедурасы да келтирилген. Ол қәте ҳаққында хабар береди ҳәм қәте киритилген санды қалдырып қалғанларының квадратлары қосындысын есаплайды.

Программаның интерфейсин жақсылаў мақсетинде оған бирнеше Edit бир жоллы редакторлары ҳәм Label меткалар қойылды. Редакторларға санлар киритиледи. Меткаларға тийисли түсиниклер жазылады. Нәтийже қәтелер айнасында шығады. Мәселени шешиўде Statistic классының мына методларынан пайдаланылды:

- Statistic.AddValue
- Statistic.Solve
- Statistic.Clear
- Statistic.GetCount.

Программаның айнасы көриниси төменде келтирилген


Form1
Дельфиде класс дузиу мысалы
1-ши санды киритин
1111
2-ши санды киритин
2222
3-ши санды киритин
3333
4-ши санды киритин
4444
есаплау
Project1
37029630
OK

ЖУЎМАҚЛАЎ

Әдетте системада көплеген объектлер жумыс ислейди. Олардың базылары уқсас ямаса бирдей типте болыўы да мүмкин. Мәселен банк системасында көплеген объект-есап бетлери, көплегенобъект-клиентлер болыўы мүмкин. Бирдей типтеги объектлер классларға бирлестириледи.

Класс түсиниги абстракт болғаны менен системаның программалық тәмийнлениўин ислеп шығыўды ҳәм әмелге асырылыўын әпиўайыластырады. Сонлықтан объектке бағдарланған программаластырыў тиллеринде классларды ислеп шығыў программалық тәмийнаттың ең әҳмийетли мәселелеринен есапланады.

Бул питкериў қәнигелик жумысында Delphi орталығында классларды ислеп шығыў технологиясы қаралған.

Питкериў қәнигелик жумысының тийкарғы нәтийжелери ретинде төмендегилерди көрсетиўге болады:

1. Объектке бағдарланған программаластырыў тиллериниң тийкарғы принциплери үйренилди.
2. Delphi орталығында классларды дүзиў қураллары көрип шығылды.
3. Delphi орталығында класслар менен жумыс ислеў өзгешеликлери үйренип шығылды.
4. Delphi орталығында классларды дүзиўге конкрет мысаллар шешилди.

Питкериў қәнийгелик жумысының нәтийжелери: көрип шығылған теориялық мағлыўматлар, шешилген мысаллар студентлер тәрепинен программаластырыў тиллерин үйрениўде ҳәм өз бетинше жумыслар орынлаўда методикалық қолланба бола алады деп есаплаймыз.

ӘДЕБИЯТЛАР

1. Фаронов В.В. Дельфи 5. Учебный курс. М. Нолидж, 2000г
2. Архангельский А.Я. Программирование в Delphi 5 – 2-е издание – М.: ЗАО «Издательство БИНОМ», 2000 г. – 1072 с.
3. Казаков Б.В. Основы объектно-ориентированного программирования в среде Delphi. Учебное пособие/Под ред. профессора Б.Г. Хмелевского. – Пенза: Изд-во Пенз. гос. ун-та, 2002. – 108 с.
4. Фленов М.В. Библия Delphi. БХВ-Петербург, 865 стр, 2004 г.
5. Галисеев Г.В.Компоненты в Delphi 7: Профессиональная работа. Диалектика, 619 стр, 2004 г.
6. Шпак Ю. А. Delphi 7 на примерах/Под ред. Ю. С. Ковтанюка — К.: Издательство Юниор, 2003. — 384 с., ил.
7. http://www.intuit.ru
8. http://algorithm.narod.ru
9. http://www.math.ru

Қ о с ы м ш а

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label5: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
    procedure OnError(Sender:TObject; Error: string);
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
 uses Unit2;
{$R *.dfm}
```

```
procedure TForm1.Button1Click(Sender: TObject);
 var Statistic:TStatistic;
 s,s1,s2,s3: string;
begin
s:=edit1.Text;
s1:=edit2.Text;
s2:=edit3.Text;
s3:=edit4.Text;
Statistic:=TStatistic.create;
Statistic.onError:=onError;
Statistic.AddValue(s);
Statistic.AddValue(s1);
Statistic.AddValue(s2);
Statistic.AddValue(s3);
showmessage(floattostr(Statistic.solve));
Statistic.Clear;
//Statistic.AddValue('123423');
//Statistic.AddValue('12ssss3');
//showmessage(floattostr(Statistic.solve));
Statistic.Free;
end;
procedure TForm1.OnError(Sender: TObject; Error: string);
begin
showmessage('Error inside class:'+Sender.ClassName+#13#10+Error);
end;
end.
```