Дәріс 1

Программалау технологиясының эволюциясы. Құрылымдық және объектіге бағытталған программалау қағидалары. Программалау терминдерінің және әдістемелерінің сипаттамасы. Абстракция, инкапсуляция, жаратылысынан ие болу және полиморфизм түсінігі. Компонентік технология.

Лекция жоспары:

1. Программалау технологиялары
2. Объекті-бағытталған программалау
3. Абстракция, инкапсуляция
4. Компонеттік технология

Лекция мазмұны

Қазіргі кезде 3500-ден жоғары әр түрлі программалау тілдері бар және осылардың ішінен шектелген саны ғана программалық бөлімді құруға жаппай қолданылады. Программалау тілдерінің саны көп болуына байланысты олардың біршене классификациясы бар. Оның ішінде негізгі 2 түрге: фукционалдық мәніне (қызметі) және қолданылатын программалау технологиясына байланысты бөлінеді. Программалау тілдері фукнционалдық мәніне байланысты 4 үлкен классқа бөлінеді:

1) программалауға үйретуші;
2) жалпы мәнді;
3) проблемалық-бағытталған; 4) параллель программалаушы.

Қазіргі жоғары деңгейлі тілдерді осы классикацияға сәйкес қарастырамыз.

Программалауға үйретуші тілдер қатарына жататын жоғарғы деңгейлі тілдердің негізгілері Logo, Basic және Pascal. Logo тілі 60-шы жылдардың аяғында С.Пейперттің басшылығымен құрылды және ЭЕМ жаңа қолданушыларға, балаларға программалаудың негізін үйретуге бағытталған. Үйретуге арналған жоғары деңгейлі тілдердің ішіндегі кең тарағаны 1965 жылы Д.Кемени мен Т.Курц құрған Basic тілі. Дербес компьютерлерге бірінші қолданылған жоғарғы деңгейлі тіл және операциялық ортасы – Basic тілі. 60-жылдардың ортасынан бастап мамандар арасында құрылымдық программалау мәселесі көтеріле бастады. 1971 жылы Н.Вирт құрылымдық технологияға үйретуші Pascal тілін құрды.

Жалпы мәнді жоғары деңгейлі тілдер қатарына әр түрлі класс есептерін тиімді программалауға бағытталған тілдер жатады. Бұл классқа жататын тілдердің негізгілері С, Modula, Ada тілдері. Бұл тілдердің негізі 1966 жылы құрылған, ғылыми, пәндік облыстағы есептерді программалауға мүмкіндік беретін PL/1 тілі. PL/1 тілі ЖС ЭЕМ сериялы модельдерде кеңінен қолданылды. Си тілін 1972 жылы Д.Ритчи құрды және Unix операциялық жүйесі осы тілде жазылды. Си тілі ассемблер тілінің де және жоғары деңгейлі тілдің де мүмкіндіктерін қамтамасыз ететін болғандықтан орта деңгейлі тіл деп атайды. Pascal тілінің идеологиясын тарату мақсатымен Н.Вирт 1980 жылы Modula–2 тілін құрды. Бұл программалау тілінің негізі – программа бір-біріне тәуелсіз модульдерден тұрады.

1978 жылы қазіргі АҚШ-та әскери қаруды басқаруға қолданылатын қосымшаларды программалуға арналған Ada тілі құрылды. Ada тілі құрылымды программалау тілі болып табылады және параллель программалау мүмкіндіктерін қамтамасыз етеді.

Проблемалық–бағытталған жоғары деңгейлі тілдер деп нақты пәндік облыс есептерінің мәселесін түгел қамтитын тілдерді атаймыз. Бұл классқа жататын бірінші жоғары деңгейлі тіл – Fortran тілі. Fortran-I тілін 1956 жылы IBM фирмасы құрды және ғылыми-техникалық есептерді шешуге арналған. Жасанды интеллект символдық информацияларын және тізімдерін өңдеуге арналған есептерді программалауға Lisp, Prolog тілдері қолданылады. Lisp тілін 50-жылдары Д.Макартни құрды. 70-жылдары Lisp тілінің негізінде құрылған Prolog тілі логикалық программалау тілі болып табылады және 5 буынның ЭЕМ жапон проектісінде негізгі тіл болып таңдалған. Әр Prolog программа жәй тұжырымдар немесе импликациялардан тұратын сөйлемдерден тұрады, инструкция қолданылмайды.

Дәстүрлі неймандық архитектурадан ЭЕМ дәстүрлі емес параллель архитектуралы есептеу машиналарына көшуіне байланысты параллель алгоритмдерді сипаттаушы инстументальды құрылғылар пайда бола бастады. Параллель программалау тілдерінің негізі – параллель есептеулерді программалау процесін ықшамдау, параллель архитектуралы есептеу машиналарына арналған параллель программалық жүйелердің тиімдісін алу. Параллель программалауды жабдықтаудың қиындығы тиімді параллель жөндеушілерді құру. Қазіргі кезде қолданылып жүрген интерактивті параллель жөндеушілер IPSC (Intel фирмасының дербес компьютерлеріне арналған) және PDBX (мультипроцессорлы ЭЕМ арналған). Жиі қолданылатын векторлы матрицалық есептерге параллель программалар кітапханасы құрылған, BLAS- сызықтық алгебра программаларының кітапханасы, NAG – сандық алгоритмдер кітапханасы. Бұл есептерге белгілі матрицалы параллельдеушілер ProSolvar (Intel фирмасы) және жалпы мәнді параллельдеуші Parafrace (Д.Кук құрған) қолданылады.

1. Программалау технологиялары

Программалау процесін жақсартатын және кең қолданылатын әдістердің бірі – құрылымдық программалау. Құрылымдық программалаудың 3 бөлігі (құраушысы) бар:

1. Модульдік программалау
2. Құрылымдық кодтау
3. Жоғарыдан төменге қарай жобалау

Модульдік программалау дегеніміз – программаны логикалық бөліктерге бөлу процесі. Программа бірнеше модульдерге бөлінеді және мына 2 мақсат орындалуы тиіс:

1) Модулдің дұрыс болуы және оның контекстерден тәуелсіз болуы қажет;
2) Модулдің ішкі жұмыстарын білмей тұра әр түрлі модулдерден программа құру мүмкіндігінің болуы қажет.

Мысал ретінде стандарт математикалық функциялардың есептелу программасын қарастыруға болады. Программист sin(x) функциясын программаның кез-келген жерінде қолдана алады және оған функцияның есептелуіне қай әдістің қолданып тұрғанын білудің қажеттілігі жоқ. Модуль өлшемі 60 жолдан аспауы керек және модульдер өзара тәуелсіз болуы керек. Байланысқан элементтерді бір модульге, байланыспаған элементтерді әр түрлі модульге жинау керек. Модульдерді қолдана отырып программа күрделілігін төмендетуге болады.

Pascal тілінде модуль процедуралар мен функциялардың көмегімен құрылады, Си тілінде функциялардың көмегімен құрылады.

Құрылымдық кодтау деп программада басқарушы конструкциялардың– шартты операторлардың, циклдің (параметрлі, цикл-әзір, цикл-дейін) қолданылуын айтады. Шартсыз көшу операторы программада сирек қолданылуы керек немесе шартты оператордың, циклдің көмегімен өзгертілуі керек.

Программаны жоғарыдан төмен қарай жобалаудың өз иерархиялық құрылымы бар және қысқа есеп қойылымынан басталады. Одан кейін есеп бірнеше ұсақ ішкі есептерге бөлінеді. Ішкі есептердің өзі де ішкі есептерге бөлінуі мүмкін. Әр қадамда ішкі есептің орындайтын негізгі функциялары анықталуы керек. Бөлу процесі әр ішкі есеп қарапайым болғанға дейін, яғни әр ішкі есепке бір модуль сәйкес келгенше созылады.

2. Объекті-бағытталған программалау

Объекті-бағытталған программалау тілдерінің негізгі концепциясы– құрылатын қосымша өзара байланысқан негізгі объектілерден тұрады. Объекті-бағытталған технологияда қолданушы үш базалық элементпен: объектілер, хабар және класстармен жұмыс істейді.

Объектілер дегеніміз бірнеше рет қолданылатын программалық модулдерден, яғни байланысқан мәліметтер мен процедуралардан тұрады. Объект құрылымы екі бөліктен тұрады: айнымалылар және әдістер. Әдістер объект функциясының алгоритмін анықтайтын процедуралар мен функциялар жиынынан тұрады. Объектілі айнымалылар жәй мәліметтерден (сан, массив, текст) және күрделі құрылымды информациялардан (график, дыбыс т.б.) тұрады.

Объектілердің өзара байланысуына хабарлар қолданылады және үш бөлімнен тұрады: объект идентификаторы, ағымдағы объектіде қолданылатын әдіс аттары және таңдалған әдіс режимін қалпына келтіретін қосымша информациялар. Күрделі программалар бірнеше біртипті объектілерді қолдануы мүмкін. Бұл жағдайда әр объект үшін әдістер мен айнымалылар туралы информацияны жазу тиімсіз. Бұл мақсатқа объектілер класы деген түсінік енгізілген. *Класс* дегеніміз біртипті объектілерге арналған шаблон және объектілі айнымалылар типтері мен әдістерін анықтайтын информациялардан тұрады. Объекті-бағытталған технологияға негізделген программалау тілдері: SmallTalk/v, Object Pascal, ACT++, C++, Simula, Actor, Classic–Aga және т.б.

Объекті-бағытталған программалаудың негізгі үш принципі бар: инкапсуляция, тұқымқуалау, полиморфизм.

Логикалық программалау

Логикалық программалау тілдері PROLOG және LISP жасанды интеллект проблемаларының есептерін шешуге арналған. LISP тілін 50-інші жылы Д.Макартни символдық информацияларды өңдеуге арнап құрды. LISP тілінің мәліменттерінің негізгі құрылымы тізімдер, тізімнің элементтері атомдар. Lisp тілінің бір ерекшелігі динамикалық жаңа объектілерді құру мүмкіндігі, объект есебінде программаның өзі де қатыса алады.

LISP тілі және оның модификациялары символды өңдеуге арналған бағытталған программалық бөлімді құруға кең қолданады және қазіргі кезде көптеген тиімді компиляторлары бар.

70-жылдары Lisp тілінің негізінде құрылған Prolog тілі логикалық программалау тілі болып табылады. Prolog программасының негізгі элементі атом болып табылады және жеке объектілер арасындағы қарапайым қатынастарды көрсетеді, басқа программалау тілдеріне қарағанда атом түсінігінің мағыналық мәні басқа. Тіл тек сипаттамадан тұрады және инструкциялары жоқ, яғни процедуралы емес. Әр Prolog программа сөйлемдер жиынынан тұрады, яғни жәй тұжырымдар немесе импликациялар. Prolog тілінің базасында эксперттік жүйелер, білімді көрсететін жүйелер, білім базасы және жаратылыс тілдерін өңдейтін жүйелер құрылады. Prolog тілінің негізіне математикалық логика элементтері қолданылады. Программа объектілер арасындағы қатынас терминдері арқылы сипатталады. Логикалық программалау тілдерінің жетістігі параллель программалау принципі қолданылады. Prolog тілінің көптеген танымал модификациялары бар, оның ішінде ең көп тарағаны – Borland фирмасының Turbo Prolog программалау жүйесі. Жасанды интеллект проблемасына арналған жаңа логикалық және функционалды программалау тілдері құрылуда, мысалы, DURAL, VALID тілдері.

3. Абстракция, инкапсуляция

Объектіге бағдарланған ұстанымның концептуалдық негіздеріне объектіге бағдарланған ұстанымның моделі жатады. Объектілік модельдеудің негізгі элементтері: абстаркциялау, инкапсуляция, модульділік және иерархия. Қосымша элементтері: типтелу, паралеллизм және тұрақтылық.

Абстракциялау – бұл қандай да болмасын объектіні, өзге объектілерден ажырататын белгілері, сипаттамалары және т.б. арқылы бөліп алу, жалпы абстракциялау объектінің сыртқы ерекшеліктеріне негізделеді. Объектіге бағдарланған ұстанымда, берілген объектінің дұрыс абстракциялануы, жобалаудың негізгі міндеттерінің бірі болып саналады.

Инкапсуляция – бұл объектінің, өзінің ішкі элементерінің, бір бірінен ажыратылу процесі. Бұл процесс кезінде объектінің ішкі құрылымдары мен оқиғалары бір- бірінен дұрыс ажыратылады. Инкапсуляция объектінің интерфейсін қорғау үшін қолданылады немесе объектілік ұстанымда класстың ресурстарын, тек оның өзінің ғана пайдалануын қолдайды. Абстракциялау мен инкапсуляция бірін бірі толықтырады.

Модульділік – бұл программалық жабдықтың декомпозициялану кезінде өзара байланысқан, бірақ өте әлсіз байланысқан модульдерге бөліну қасиеттері. Инкапсуляция мен модульділік қасиеттері абстракцияларды бір- бірінен ажыратады.

Иерархия – бұл жүйедегі абстракцияланудың бір- біріне бағынышты түрде реттеліп орналасуын тағайындайды. Бұл күрделі жүйедегі класстардың құрылымы (иерархиясы). Мысалы, жай және көп қабылдаушылықты айтуға болады.

Инкапсуляция

Біз объекттер әлемінде өмір сүреміз. Стол, автомобиль, ручка, тақта – бұлардың бәрі объекттер. Осындай физикалық объекттермен қатар абстрактілі объекттер де бар. Мысалы: сандар. Демек, объект – бұл кезкелген физикалық не абстрактілі айқындалған барлық(сущность). Объект – бұл жалпы философиялық ұғым. Оны ұзақ уақыттан бері философтар зерттеп келеді.

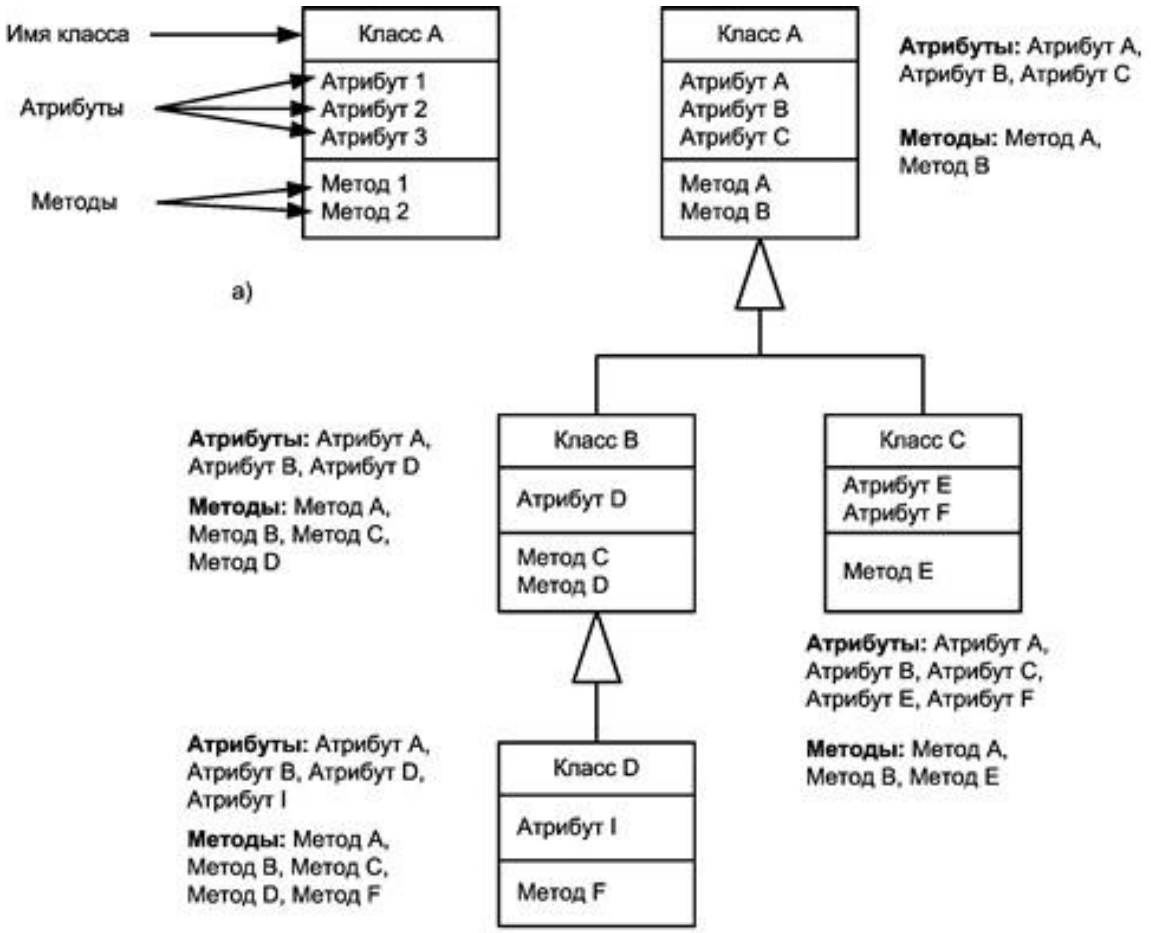
Объекттер атрибуттарымен сипатталады. Мысалы, автомобильдің атрибуттары – ең жоғары жылдамдығы, двигателінің қуаты, кузов түсі т.б. Атрибуттардан бөлек объекттер қайсы бір функционалды мүмкіндіктерге ие болады. Бұл мүмкіндік объектті-бағытталған программалауда (ОБП) **амал** не **тәсіл** делінеді. Мысалы, автомобиль жүре

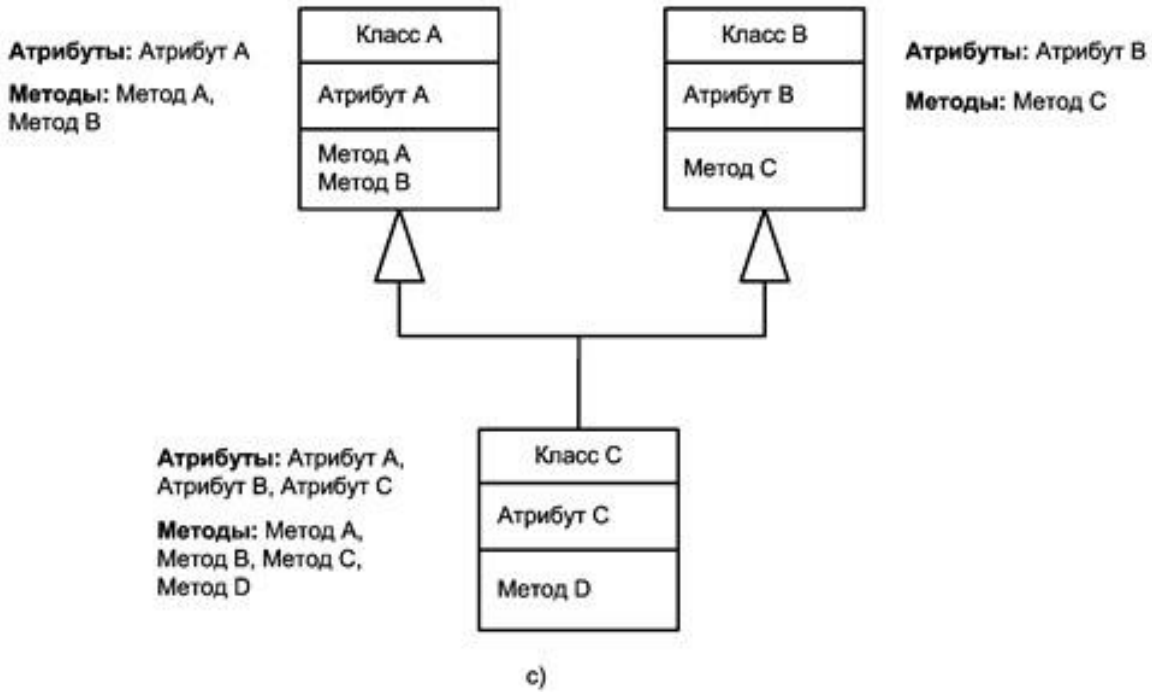
алады, бұл оың функционалды мүмкіндігі, яғни, орындай алатын амалы(тәсілі), корабль – жүзе алады, компьютер – есептеулер жүргізеді. Осылайша, объект атрибуттар мен тәсілдерді инкапсуляция жасайды, яғни басқа объекттерден (өзара әсерететін) өз тәсілінің қалай құрылғандығын жасырады. Мысалы, теледидар каналын ауыстыру үшін пульттегі керекті батырманы басу жеткілікті, күрделі механизм іске түсіп, нәтижеде керекті каналға ауыстырылады. Бізге телевизор мен пультте қандай механизмдер іске түсіп, қалай орындалатындығын білу міндетті емес, телевизордың мұндай мүмкіндігі(тәсілі) бар екендігін және оны қалай іске түсірудібілу жеткілікті. Инкапсуляция, яғни құрылымын жасыру қасиеті ОБП-ң базалық қасиеті болып табылады. Ол талап етілген тәсілдерге ие пайдаланушы объекттерін құруға мүмкіндік береді. Пайдаланушы бұл объекттердің құрылымын білмей-ақ, олармен жұмыс істей беруіне болады.

Мұрагерлік. Полиморфизм

Әрбір объект қайсыбір объектер класының экземпляры болып табылады. Мысалы, Audi 6 автомобилі осы модель автомобильдері классының экземпляры болып табылады, ал, мысық сүтқоректілер классының экземпляры. Осылайша, класс – бұл абстрактілі ұғым. Класстар бір-бірімен әртүрлі қатынаста болады. Осындай қатынастардың негізгілерінің бірі: **клас-ішкі класс** қатынасы. Бұл объектті-программалауда мұрагерлік қатынасы ретінде белгілі. Мысалы, Audi 6 автомобильдер классы жеңіл автомобильдер классына ішкі класс болып табылады. Жеңіл автомобильдер классы көліктер классының ішкі классы. Көліктер өз кезегінде транспорт құралдары классының ішкі классы. Транспорт құралдарына көліктермен қатар самолеттер, поезд, корабль т.б. жатады.

Мұрагерлік қатынасында берлық атрибуттар мен тәсілдер аталық класстанн ұрпақ-классқа беріледі. Мұрагерлік көпдеңгейлі болуы мүмкін. Бұл жағдайда, иерархияның төменгі деңгейіндегі класстар тікелей не жанама аталық класстардың барлық қасиеттерін мұраға алады.


Имя класса
Атрибуты
Методы
Класс A
Атрибут 1
Атрибут 2
Атрибут 3
Метод 1
Метод 2
а)
Класс A
Атрибут A
Атрибут B
Атрибут C
Метод A
Метод B
Атрибуты: Атрибут A, Атрибут B, Атрибут C
Методы: Метод A, Метод B
Класс B
Атрибут D
Метод C
Метод D
Атрибуты: Атрибут A, Атрибут B, Атрибут D
Методы: Метод A, Метод B, Метод C, Метод D
Класс C
Атрибут E
Атрибут F
Метод E
Атрибуты: Атрибут A, Атрибут B, Атрибут C, Атрибут E, Атрибут F
Методы: Метод A, Метод B, Метод E
Класс D
Атрибут I
Метод F
Атрибуты: Атрибут A, Атрибут B, Атрибут D, Атрибут I
Методы: Метод A, Метод B, Метод C, Метод D, Метод F
б)


Класс A
Атрибут A
Метод A
Метод B
Атрибуты: Атрибут A
Методы: Метод A, Метод B
Класс B
Атрибут B
Метод C
Атрибуты: Атрибут B
Методы: Метод C
Класс C
Атрибут C
Метод D
Атрибуты: Атрибут A, Атрибут B, Атрибут C
Методы: Метод A, Метод B, Метод C, Метод D
с)

Сурет 1 - Әдіс және қасиеттердің өзара байланысы

Бірегей мұрагерліктен бөлек көпмұрагерлік те бар. Бұл жағдайда класс бірден бірнеше класстардың мұрагері болып табылады, және оларың барық қасиеттерін мұраға алады. Көпмұрагерлік қатынасын қолдану барысында мынаған көңіл аудару керек: ұрпақ класс бірдей атаулы бірақ, мазмұны әртүрлі қасиеттерге ие болып қалуы мүмкін.

Мұрагерлікте бір тәсілдің орнын басқа бір тәсіл алмастыруы мүмкін. Мысалы, транспорт құралдары классы қозғалу қасиетіне ие(жалпыға бірдей). Бұл әдіс ұрпақ-класстарда нақтыланады: автомобиль – жүреді, самолет – ұшады, корабль – жүзеді. Тәсіл семантикасының осылайша өзгеруі полиморфизм делінеді. Полиморфизм – бұл сол бір атаулы тәсілдің контекстке тәуелді(дербес жағдайда, қайсы классқа тиістілігіне байланысты) әртүрлі әрекеттер орындауы.

Объекттер полиморфизмі

Объектігі бағытталған программалауды қолдану барысында объекттердің полиморфизмі қамтамасыз етіледі. Бұл термин нені білдіреді?

Объекттердің полиморфизмі мынаны білдіреді: **әртүрлі объекттерге жіберілген** *бір және сол бір хабарлама* **программаның орындалу этапында нақты қайсы объект осы хабарламаның қабылдаушысы болуына байланысты** *әртүрлі әрекеттердің* **орындалуына** (әртүрлі тәсілдердің шақырылуы)**әкелуі мүмкін.**

Басқаша айтқанда: қасиеті, қызметі, ішкі құрылымы бойынша әртүрлі объекттер мағынасы бойынша бірдей(програмисттің көзқарасы тұрғысынан) әрекеттерді табиғаты мен ішкі құрылымына байланысты әртүрлі орындауы мүмкін.

Егер хабарламаны жіберу **тәсілдің процедура сияқты шақырылуына** алмастырылған болса(Си, Паскаль тілдерінде), онда **полиморфизм** мынаны білдіреді: тәсіл шақыруы сақталған программа кодының сол және тек сол бір бөлігі кодтың орындалу кезінде қайсы класстың экземпляры әрекетті орындаушы боп табылуына байланысты әртүрлі тәсілдердің шақырылуына әкеледі.

Мысал.

Қазірше нақты бір тілдің синтаксисін қолданып жатпаймыз. Айталық, **P** – объект көрсеткіші болсын. **P** көрсеткіші програмаманың орындалу кезінде түрлі класстарға нұсқауы мүмкін. Айталық, ол параметр ретінде қайсыбір **ppp** процедураға берілсін. Хабарлама жіберу тілдің құралдарымен былайша жазылатын болсын P.Show; **ppp** процедурасы "паскалдік" стилде былайша сипатталсын

```
procedure ppp(P);
begin     P.Show;
end;
```

ppp процедурасын шақыру кезінде, **P** ретінде түрлі класстардың көрсеткіші берілуі мүмкін. Полиморфизм әсері мынадан көрінеді: сол және сол бір код (**P.Show**) **Show** хабарламасына жауап ретінде процедура параметрі ретінде берілген P көрсеткіші қайсы классты нұсқауына байланысты түрлі тәсілдің шақырылуына әкеледі ***Объектілі-бағдарлы программалау тәсілі жөнінде***

Бейсик, Паскаль сияқты дәстүрлі программалау тілдерінде күрделі, үлкен программаларды дайындаудың кемшілігі – ол үшін программалаушылар тобының көп күш жұмсауы қажет болатын.

Программалауды жеңілдету үшін 80-жылдары **объект, класс (object, class)** ұғымдары енгізіліп, **объектілі – бағдарлы программалау (ОБП)** тәсілі негізге алынды. **ОБП –** Паскальда қолданылатын процедуралық және құрылымдық программалаудың және модуль құрудың дамытылған түрі. Ол берілгендерді белгілі бір абстракциялық деңгейде көрсетіп, **модульдік** программалауды пайдаланады. Мысалы, Visual Basic программалау жүйесі объектілі программалау тәсілін пайдаланып, Qbasic тілі негізінде визуальды түрде құрылған. Паскальдың **ОБП** тәсілін пайдаланып құрылған жаңа нұсқасы Object Pascal (Объектілі Паскаль) деп аталады. **Delphi** осы тілдің негізінде дайындалған. Объектілі программалаудың ыңғайсыздығы – онда дәстүрлі программалау тәсілдері пайдалана берілмейді, бірақ ондағы көп қиындықтар арнайы тәсілдерді пайдалану арқылы тез шешілген.

ОБП тілінің дәстүрлі программалау тілдерінің өзгешелігі – онда, өрістерге қоса, мынадай ұғымдар негізге алынған: **класс, объект, өңдеу тәсілі, объект қасиеті және оқиға**.

Өрістер – Турбо Паскальдағы жазу (record) типінің өрістері сияқты;

Қасиеттер – объект сипаттамалары (параметрлері);

Тәсілдер – кластың өрістері мен қасиеттерін өңдеу процедуралар

мен функциялар;

объект – түрлі мәндердің қасиеттері тәсілдерінің жиынтығы; **оқиға –** объект жағдайының өзгеруі.

Класс - өрістер, қасиеттер және тәсілдерінің бірлігінен тұратын тип, не, жалпы түрде, өңдеу тәсілдері не қасиеттері ортақ түрде сипатталатын объектілер (нысандар) жиынтығы.

Жүйе объект үшін жадтың динамикалық облысынан арнайы орын қалдырады. Яғни, шын мәнінде, объект жай айнымалы емес, ол – жадтың динамикалық облысының кездейсоқ адресін сақтайтын көрсеткіш. Бірақ программада оған Турбо Паскальда пайданылатын көрсеткіш белгісі (^) енгізілмей жазылады.

Объект құрылған кезде автоматты түрде **конструктор** (constructor) деп аталатын тәсіл шақырылып, ол объектіні динамикалық облыста орналастырады, динамикалық жадтан объектіні арнайы **деструктор**

(destructor) тәсілі жояды.

Сонымен, **ОБП –**да пайдаланылатын объект сөзінің екі мағынасы бар: нақты объект (мысалы, геометриялық дене, формада орнатылған компонент, т.б.) және айнымалы (абстрактты, дерексіз объект). Delphi-де екінші объект нақты **класс данасын** анықтайды.

Класқа **иерархиялық** (бағынышты, жоғарыдан төмен) кластардың енуі де мүмкін. Мысалы, геометриялық фигуралар класы жазық фигуралар және кеңістік фигуралары болатын екі ішкі кластық фигураларға бөлінеді. Ал жазық фигуралар класы төбелері бар (үшбұрыш, көпбұрыш) және төбелерсіз (шеңбер, эллипс) болып екі ішкі класқа бөлінеді. Объектілі программалауда барлық кластар иерархиялық түрде негізгі **Tobject** класынан тарайды **(Tobject → Tpersistent → Tcomponent → …).** Әдетте негізгі класты аталық, бағыныңқы кластарды ұрпақ не сәбилік кластар деп атайды (**Tobject** – барлық кластардың арғы атасы, **Tcomponent –** барлық иерархиялық компоненттердің жоғарғысы (**компонент** деп Tcomponent класының мұрагері болатын класс данасын атайды)). Әр ұрпақ өзінің аталық класының мүмкіндіктерін (өрістерінің, қасиеттерінің, тәсілдерінің сипаттамаларын) қабылдайды. Мысалы, Tobject класының **Create**

(бастапқы мән меншіктеу (инициалдау)) тәсілі барлық ұрпақ кластарға тиісті (Tobject класында конструктор Create деп аталады). Әр ұрпақ өзіне қосымша жаңа мүмкіндіктер (сипаттамалар) қосып, келесі ұрпақтарға беруі мүмкін.

Delphi –де класс атауын **T** әрпінен бастау келісілген. Формаға енгізілген компонент данасын сандық индекс қосылған класс атауымен аталады. Ұрпақ кластың сипатталу түрі:

type

<ұрпақ класс атауы**> =** class (<негізгі класс атауы>); **end;**
var

< объект > : < класс атауы>;

мұндағы класс мүшелері - өрістер, қасиеттер және тәсілдер. Мысалы,

type

Tform1= **class** (TForm)

Button1 : TButton;

Label1 : TLabel;

Procedure Button1 Click (Sender : TObject); **end;**

var Form1 : Tform1;

TForm1 = class (TForm) жазуы Tform класының үлгісімен құрылатын жаңа (ұрпақ) TForm1 айнымалысы – класс данасы.

Программаны дайындау барысында Delphi модульдің **Interface** бөліміне формаға енгізілген компоненттер мен оқиғасы таңдалған тәсілдерді сипаттауды автоматты түрде енгізеді, тәсілдерді іске асыру нұсқаулары модульдің **inplementation** бөліміне қолдан жазылады.

Жалпы, **Delphi –де класс** деп өзіне ұқсас даналарды дайындау үшін үлгі түрінде алынатын, толық түрде дайындалған программа үзіндісін де атайды. Программалаушы бұрын дайындалған класс данасының көшірмесін түрлі программаларға не бір программаның түрлі жерлеріне енгізіп, программалауды және оның көлемін көп жеңілдетуіне болады.

Delphi құрамына жүздеген дайын кластар енгізілген. Әр компонент қатаң түрде тек бір класқа ғана тиісті. Мысалы, Delphi – де форма (TForm) компоненті - **Forma** класына тиісті, осы типті айнымалы (Form1) – класс объектісі. Класс бөлігі ретінде сипатталып, модульдің

Inplementation (іске асыру) бөлігінде мәтіні жазылатын сәйкес процедура - тәсіл, мысалы, **procedure** TForm1.Button1Click.

Форма ішіне орналастырылған TEdit компонентінің Edit1 данасын (объектісін) екі рет шерткен кезде пайда болатын оқиғаны өңдеу тәсіліне (процедурасына) мысал:

procedure TForm1.Edit1Click(Sender : TObject); **begin**

Edit1.Text:=‘Сіз редакциялау өрісін екі рет шертіңіз‘; e**nd;**

программаны іске қосу командасы берілген соң компилятор автоматты түрде негізгі модульге енгізілген төмендегі екі тәсілді орындайды:

```
Application.CreateForm(TForm1, Form1);
Application.Run;
```

Бұлардың біріншісі жадта Form1 объектісін құрады (объектіні инициалдайды), екіншісі Application объектісіне қосымшаның негізгі формасын іске қосуға нұсқау береді (Run). Жоғарыда жазылған процедураның орындалу нәтижесінде форманың Edit1 өрісіне оның Text қасиетіне меншіктелген мәтін жазылады.

Ескерту. Белгісіз жағдайда программада тәсілді не қасиет орнату командасын иерархиялық түрде жазу қажет, мысалы:

```
TForm1.Button1.FontSize:= 14;
```

Мұндағы иерархияның бірінші объектісі – **форма (Form1),** екінші объект – формада орнатылған **Button1** түймесі, үшінші объект – түйменің **шрифт қасиеті;** команда – форманың Button1 объектісіне енгізілген **FontSize** (шрифт өлшемі) айнымалысына 14 мәнін меншіктеу. Яғни, объектілер арасына қойылатын **нүкте** сәйкес иерархиялық объектіге сілтеме.

ОБП –да класс үшін *мұрагерлік, инкапсуляция* және *полиморфизм* принцптері енгізілген.

Мұрагерлік-программада негізгі класс типінің ұрпақ иерархиялық кластарда да сақталуы. Мысалы.көлік, жеңіл жүк машинасы т.б.. болып бөлінеді. Олардың жұмыс істеу жағдайына байланысты, сипаттамаларына ортақ және бөлек өрістер, қасиеттер мен тәсілдер енгізілуі мүмкін. Әр қайсысында тип элементтерін толық сипаттау тиесілі емес. Бірінің типін аталық (негізгі) етіп қабылдап, екіншісіне қосымша қасиеттерді енгізу жеткілікті.Delphi-дің барлық кластары *TObject* класына мұрагер.

Инкапсуляция (ішінде, біртұтас).Класс өрістер, қасиеттер және тәсілдердің бірлігінен тұрады.Олардың бір тұтатас түрінде қарастыру *инкапсуляция* деп аталады. Әр класс аяқталған толық бір іс-әрекетті бойында сақтайды. Кластың мұндай принципі *өзінде инкапсульдайды делінеді.*

Класта инкапсульданған процедуралар – функциялар, тәсілдер.

Полиморфизм (көп түрлілік)-мұрагер кластардың аталық класқа енгізілген тәсіл атауы бойынша басқа қасиеттер енгізілген тәсіл атауы бойынша басқа қасиеттер енгізілген ұқсас есепті шешуіне болатындығы, яғни түрлі кластарда түрлі әрекеттерді орындайтын тәсілдерге *бірдей атау беру мүмкіндігі.*

4. Компонеттік технология

DeIphi-дің негізгі ерекшелігі — онда қосымша құруда компоненттік және объектілік тәсілдер пайдаланылды (Windows ортасында пайдаланатындықтан, Delphi-де программаны көбінесе қосымша деп айтады). Бұл программалау технологиясында нағыз революция жасады деуге болады.

Компоненттік тәсілдің мәнісі жеңіл: әр қосымша кітапханасы программалау ортасында дайындалып, арнайы іс-әрекеттерді орындайтын компоненттер элементтерінен жинақталады. Олар жеткіліксіз болса, объектіні өңдеуге арналған үстеме программа құрылады. **Delphi-де** қолданылатын негізгі кітапхананы визуальды компоненттер кітапханасы **(VCL, Visual Component Library)** деп атайды.

Компоненттер панелінде топтобымен жинақталған, жүздеген кластарға тиісті, стандартты компоненттер бар. Пайдаланушы жаңа компонент дайындап, оны осы панельге қосуына да болады.

Бақылау сұрақтары

1. Программалау технологиясының эволюциясы?
2. Құрылымдық және объектіге бағытталған программалау қағидалары?
3. Программалау терминдерінің және әдістемелерінің сипаттамасы?
4. Абстракция, инкапсуляция, жаратылысынан ие болу және полиморфизм түсінігі?
5. Компоненттік технология?

Ұсынылатын әдебиеттер

1. Бадд Т. Объектно-ориентированное программирование в действии. Питер. 1997.
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++, 2-е изд./Пер. сс англ. –М.: «Издательство Бином», Спб.: «Невский диалект», 2001. 3. Бьярн Страуструп. Язык программирование С++. Киев: Диасофт, 1993. 1,2 часть.