Дәріс-4. Объектіге бағытталған программалау негіздері

Жоспар:

1 Мәліметтерді енгізу операциялары
2 Объектіге бағытталған программалау әдістемесі.
3 Объектілер.
4 Кластар
5 Конструкторлар
6 Инкапсуляция, мұралау және полиморфизм.

1 **Мәліметтерді енгізу операциялары**

Пернетақтадан айнымалылар мәнін енгізу үшін System.in.read (byte [ ] bt) әдісі қолданылады. Бұл әдіс енгізілген цифрларды bt байттар жиымына жазады, мұнда әрбір байтқа бір цифрдан жазылады. Мұнан кейін байттар жиымын Double(new String(bt)).doubleValue0 әдісімен нақты сандарға түрлендіру қажет. Оған қоса туындауы мүмкін ерекше жағдайларды да өңдеу керек болады.

Төмендегі мысалда консольмен байланысқан кіріс ағымына сөз тіркестерін енгізу қарастырылады.

Тіркес енгізу мысалы:

InputStr.java

import java.io.*; **public class** InputStr {
public static void main(String[] args) {

/* System.in байттық кіріс ағымы InputStreamReader класс объектісін құру кезінде конструкторға беріледі */

InputStreamReader is = **new** InputStreamReader(System.in); /* мәліметтерді буферлеу жүзеге асады */

BufferedReader bis = **new** BufferedReader(is); **try** {

System.out.println("Өз атыңызды енгізіңіз және <Enter> пернесін басыңыз:"); /*Буферден қатарды оқу; readLine() әдісі мүмкін болатын қателерді өңдейді */

String name = bis.readLine(); System.out.println("Привет, " + name); } **catch** (IOException e) { System.out.print("Енгізу қатесі " + e);

} } }

Азат деген тіркесті енгізсеңіз, онда келесі ақпарат шығады:

Привет, Азат!

Бұл мысалда java.io кітапханасынан алынған InputStreamReader және BufferedReader класс әдістері мен конструкторлары, жеке жағдайда System.in кіріс ағымымен байланысты readLine() буферден қатарды оқу әдісі де қолданылған.

2 Объектіге бағытталған программалау әдістемесі

Мәліметтерді өңдеу программа объектіге бағытталған программалау (ОБП) — программалық кодтар жасаудың қазіргі тәсілі, ол құрылымдық программалаудан кейін келді. ОБП құрылымдық программалауды ауыстырған жоқ, ол оны ары қарай логикалық жетілдіру кезеңінен өткізді.

Құрылымдық программалау негізі — есепті шешудің сатылы бұтақ тәріздес құрылымын жасау және оның программалық кодын жекелеп жазып шығу болып табылады. Құрылымдық программалаушылар бір логикалық бірлікке қатысты процедуралар мен мәліметтерді жеке файлға құрылым (Си-де struct) арқылы жинақтай отырып жасап шығады.

Дәстүрлі процедуралы программалаудың әдістері күрделі программалар құруға жауап бере алмайды. Сондықтан объектіге бағытталған программалау (ОБП) пайда болды.

Оның ерекшеліктері:

- Программалық жабдықтамалардың күрделілік дәрежесін төмендету;
- Программалық жабдықтамалардың сенімділігін жоғарылату;
- Программалық жабдықтамалардың кейбір компоненттерін қайтадан қолдану мүмкіндігін қамтамасыз ету.

ОБП қолданбалы программалауда жедел дамушы бағыттардың бірі болып табылады.

Объектілер

ОБП-да объектілер негізгі элемент болып табылады. Объект — бұл мәліметтер мен функциялар жиынынан тұратын бірыңғай конструкция. Әрбір объект үшін оларға қолданылатын нақты операциялар жиынтығы бар. Мысалы, дербес компьютердің операциялық жүйесіндегі файлдарға қолданылатын операциялар:

- құру;
- ашу;
- файлдан оқу;
- файлға жазу;
- жабу;
- жою.

Объект интерфейсі – бұлар пайдалануға болатын объектінің айнымалылары мен функциялары.

Объектінің ішкі құрылымы — программалау тілінде объектінің ішкі айнымалыларымен функцияларын сипаттау.

Кластар

Кластарда мәліметтер және программалық кодтар болады. Ондағы кодтар класс тәсілдері ішінде орналасады. Басты бір тәсілмен – main() тәсілімен таныстыңдар. Енді толығырақ программалар жазу үшін, объектілік көзқарасты мысал арқылы қарастырайық.

Класс — объект типінің спецификациясы, ал объект — кластың бір нақты экземпляры (данасы).

Негізінде, класс дегеніміз — абстрактылық ұғым, ол ешбір мәліметтерді өңдемейді. Солардың негізінде объектілер жасай отырып, солар арқылы керекті мәліметтермен жұмыс істеуге болады.

Класта сипатталған мәліметтер (айнымалылар мен константалар) класс мүшелері, ал функциялар — класс тәсілдері деп аталады. Класс данасында сақталатын мәліметтер объект айнымалылары (instancevariables) деп аталады.

Кластардан объектілер жасау синтаксисін қарастыралық. Мысалдарда келесі синтаксистік конструкция қолданылған:

Vehiclecar = newVehicle();

Бұл өрнек екі әрекет атқарады: мұндаVehicle типіндегі car айнымалысы жарияланады және Vehicle класының осы данасын жедел жадыға орналастыру үшін орын бөлінеді де, car айнымалысына сол жады адресімен шіктеледі.

Сонымен, car айнымалысында объектінің өзі емес, соған сілтейтін нұсқауыш сақталады. Сол объектіні жасауды екі командамен былай да жазуға болар еді:

Vehiclecar; // айнымалыны жариялау — // объектіге нұсқауыш

car = newVehicle(); /* объект құру, айнымалыға оның адресін меншіктеу */

car айнымалысын жариялау нәтижесінде ол мән қабылдамай, тек сілтеме ғана алады. Объект new командасы арқылы жасалып, оның адресі car айнымалысына меншіктелуі тиіс. Сонан соң ғана ол объектімен байланысады.

Конструкторлар

Алдыңғы мысалдарда Vehicle типті әрбір объектінің ішкі айнымалылары мәндерін код ішінде бердік. Бірақ мұндай әрекет кәдімгі программалауда жиі қолданылмайды. Мұнда бір мәнді меншіктеуді ұмытып, қате жіберіп алуға болады.

Объектіге бастапқы мән берудің жақсы бір мүмкіндігі - бұл конструкторды пайдалану. Конструктор — бұл объектіні жасау кезінде шақырылатын арнайы тәсіл. Бұл тәсілдің аты класс атымен бірдей болып келеді. Ол ешқандай мән қайтармайды да, тек объект айнымалыларына бастапқы мән беру үшін қолданылады.

Java тілінде әрбір класта кем дегенде бір конструктор болады. Ол айқын түрде берілмеуі де мүмкін, мұндайда компилятор класқа конструкторды келісім бойынша қосады да, объект айнымалыларына мән меншіктейді. Белгілі бір класс үшін өз конструкторыңызды жарияласаңыз, автоматты түрде іске қосылатын конструкторға қол жеткізуге болмайды.

Инкапсуляция

Инкапсуляция (encapsulation) термині «қара жәшік» ретінде қарастырылатын объектінің ішкі құрылымын жасыру болып табылады. Инкапсуляция - кластың ішкі құрылымын жасырып, сыртқы құрылымнан бөлу деген сөз. Объектінің қасиеттері белгілі болып саналады, яғни олар сырттан

қол жеткізуге болатын айнымалылар. Бірақ бұл функциялар қалай құрылған, олар қандай алгоритммен жұмыс істейді, ол туралы программалаушыға айтылмайды. Программалаушы немесе объектіні тұтынушы адам объектінің қосымша ішкі функциялары мен айнымалылары бар ма, олар қол жеткізуге болатын қасиеттер мен тәсілдермен қалай байланысқан, ол жағын білмейді.

Мұралау

Мұралау (inheritance) - бұл кластар ара қатынасы, мұнда бір класс екінші бір кластың құрылымын немесе тәсілдерін пайдаланады . Мұралау "жалпы /жалқы" иерархиясын енгізеді, мұнда ішкі класс сыртқы кластың қасиеттерін өзіне мұра ретінде қабылдап алады. Ішкі кластар әдетте мұралап алған құрылымды немесе әдісті толықтырады немесе басқаша анықтайды.

Полиморфизм

Полиморфизм (polymorphism) ОБП-ның іргелі ұғымы. "Полиморфизм" сөзі грек тілінен аударғанда "көптеген формасы бар" дегенді білдіреді. Мысалы, біз векторлық графика редакторын жасамақшы болдық делік, оның құрамындағы қарапайым графикалық фигураларды (притивтерді) бірнеше кластар - Point, Line, Circle, Box и т.б. түрінде сипаттап алуымыз керек. Олардың әрқайсысында экранға белгілі бір фигура шығаратын draw әдісін анықтаймыз. Бізге осы примитивтерді қарастыра отырып, ішіндегі керектісін таңдайтын draw тәсілін анықтайтын код жазуымыз керек. Полиморфизмді білмейтін маман осылардың әрқайсысы үшін жиым (массив) жазар еді (әр фигура үшін), ал полиморфизмдегі draw тәсілі мұны әжептеуір жеңілдетеді.

Бақылау сұрақтар:

1. ОБП деген не?
2. ОБП–ның ерекшеліктері қандай?
3. Класс деген не?
4. Java тілінде әрбір класта кем дегенде неше конструктор болады?
5. Инкапсуляция деген не?
6. Мұралау деген не?
7. Полиморфизм деген не?