Дәріс №9

Ерекшеліктерді өңдеу.

Көбінесе бағдарламаны орындау кезінде қателер орын алуы мүмкін. Олардың кейбіреулерін алдын-ала білу немесе болжау қиын, ал кейде мүлдем мүмкін емес. Мысалы, файлды жіберу кезінде желілік қосылыс кенеттен үзілуі мүмкін. Ұқсас жағдайлар ерекше жағдайлар деп аталады.

Java тілі осындай жағдайларды шешуге арналған арнайы құралдарды ұсынады. Осындай құралдардың бірі - try...catch...finally. Егер try блогында ерекше жағдай пайда болса, басқару осы ерекше жағдайды орындай алатын catch блогына өтеді. Егер мұндай блок табылмаса, пайдаланушыға жіберілмеген ерекшелік туралы хабарлама көрсетіледі және бағдарламаны одан әрі орындау тоқтатылады. Мұндай тоқтау орын алмайтындай етіп, try..catch блогын пайдалану керек. Мысалы:

```
int[] numbers = new int[3];
numbers[4]=45;
System.out.println(numbers[4]);
```

Біздің numbers массивінде тек 3 элемент болуы мүмкін, өйткені numbers [4] = 45 нұсқауларын орындаған кезде консоль ерекшелік шығарады және бағдарлама аяқталады. Енді осы ерекшелікті қарастырайық:

```
try{
    int[] numbers = new int[3];
    numbers[4]=45;
    System.out.println(numbers[4]);
}
catch(Exception ex){

    ex.printStackTrace();
}
System.out.println("Программа аяқталды");
```

try...catch блогын қолданған кезде, ең алдымен, try және catch нұсқалары арасындағы барлық нұсқаулар орындалады. Егер try блогында ерекше жағдай кенеттен орын алса, онда қалыпты орындау тәртібі тоқтап, catch нұсқауына өтеді. Сондықтан, бағдарламаның орындалуы numbers[4] = 45; жол нөмірлеріне жеткенде, бағдарлама тоқтап, catch блогына өтеді.

catch өрнегінде келесі синтаксис бар: catch (ерекше_типі айнымалы_аты). Бұл жағдайда ex айнымалысы жарияланады және ол Expression типіне жатады. Бірақ егер ерекшелік catch мәлімдемеде көрсетілген түрден ерекшеленбесе, онда ол өңделмейді, және бағдарлама қате туралы хабарламаны қарастырмайды немесе алып тастайды.

Exception типі барлық ерекшеліктер үшін негізгі класс болғандықтан, catch (Exception ex) өрнегі барлық ерекшеліктерді орындайды. Бұл жағдайда ерекше жағдай Exception класында анықталған printStackTrace() әдісін қолдана отырып, консольге қателіктер тізбегін жинауды азайтуға мүмкіндік береді.

catch орындау аяқталғаннан кейін, бағдарлама блоктаудан кейінгі барлық нұсқауларды орындау арқылы жұмысын жалғастырады.

try..catch конструкциясында finally блогы болуы мүмкін. Дегенмен, бұл блок міндетті емес және ерекше жағдайларды өңдеген кезде оларды қарастырмауға болады. finally блогы кез-келген жағдайда орындалады, егер try блогында ерекше жағдай орын алған болса да:

```
try{
    int[] numbers = new int[3];
    numbers[4]=45;
    System.out.println(numbers[4]);
}
catch(Exception ex){

    ex.printStackTrace();
}
finally{
    System.out.println("Блок finally");
}
System.out.println("Программа аяқталды");
```

Бірнеше ерекшеліктерді өңдеу. Java-да көптеген ерекше жағдайлар бар, және біз оларды қосымша блоктарды қосу арқылы ажырата аламыз:

```
int[] numbers = new int[3];
try{
    numbers[6]=45;
    numbers[6]=Integer.parseInt("gfd");
}
catch(ArrayIndexOutOfBoundsException ex){

    System.out.println("Выход за пределы массива");
}
catch(NumberFormatException ex){

    System.out.println("Ошибка преобразования из строки в число");
}
```

throw операторы. Бағдарламадағы ерекше жағдайлардың орындалуы туралы есеп беру үшін throw операторын пайдалануға болады. Яғни, осы оператордың көмегімен біз ерекше жағдай жасай аламыз және оны орындау процесінде атай аламыз. Мысалы, біздің бағдарламада нөмір енгізілген, және егер оның саны 30-дан асқан болса, ерекшелік бар:

```
package firstapp;
 import java.util.Scanner;
public class FirstApp {
    public static void main(String[] args) {
            try{ Scanner in = new Scanner(System.in);
        int x = in.nextInt();
        if(x>=30){
           throw new Exception("Число x должно быть меньше 30");
         }
      }
      catch(Exception ex){
                  System.out.println(ex.getMessage());       }
      System.out.println("Программа завершена");    }   }
```

Мұнда ерекшелік объектісін құру үшін ерекше жағдай туралы хабарлама жіберілетін Exception класының конструкторы қолданылады. Егер x саны 29-дан асатын болса, ерекшелік алынып тасталады және catch блогына өтеді.

catch блогында біз getMessage() әдісін қолдана отырып ерекше жағдай туралы хабарлама аламыз.

Ерекше кластар. Барлық ерекшеліктер үшін базалық класс - бұл Throwable класы болып табылады. Одан екі класс мұра болды: Error және Exception. Барлық қалған кластар осы екі кластан алынған.

Error класы Java жұмыс уақытындағы ішкі қателерді сипаттайды. Бағдарламалаушының мұндай қателіктерді өңдеудің мүмкіндігі шектеулі.

Ерекшеліктердің өзі Exception класынан мұра болып табылады. Осы ерекшеліктер арасында RuntimeException класын бөліп көрсету керек. RuntimeException - тексерілмеген ерекшеліктер деп аталатын топтың базалық класы - компилятор бұл ерекшеліктердің өңделмегенін тексермейді және әдіс декларациясындағы throws бас тартуға болады. Мұндай ерекшеліктер программистінің қателіктерінің нәтижесі, мысалы, түрлендірудің қате түрі немесе массив элементтер санынан асып кету.

Тексерілмейтін ерекше кластардың кейбірі:

- ArithmeticException: нөлге бөлу кезіндегі шығарылатын ерекшелік
- IndexOutOfBoundException: массив шекарасынан тыс индекс
- IllegalArgumentException: әдісті шақырған кезде дұрыс емес аргументті пайдалану
- NullPointerException: бос сілтемені пайдалану
- NumberFormatException: санды жолға түрлендіру қателігі

Exception класынан алынған барлық басқа кластар тексерілген ерекше жағдайлар (checked exceptions) деп аталады.

Тексерілген ерекше жағдайлардың кейбір кластары:

- CloneNotSupportedException: Clonable интерфейсі іске асырмайтын, объект клонданатын класс жағдайында
- InterruptedException: ағым басқа ағыммен тоқтатылған жағдайда
- ClassNotFoundException: класты табу мүмкін емес жағдайда.

Ұқсас ерекшеліктер try..catch құрылымын қолдану арқылы өңделеді. Немесе өңдеуді осы әдіс шақыратын әдіске жіберуге болады, ол throws операторынан кейін ерекше жағдайларды көрсете алады:

```
public Person clone() throws CloneNotSupportedException
{ Person p = (Person) super.clone();
    return p;
}
```

Нәтижесінде келесі ерекше жағдай иерархиясын аламыз:

Барлық ерекшелік кластары Exception класынан мұра болғандықтан, олардың барлығы ерекшеліктердің табиғаты туралы ақпарат беретін оның бірқатар әдістерін иеленеді. Осы әдістердің ішінде біз ең маңыздыларын атап өтеміз:

- GetMessage () әдісі ерекше жағдай туралы хабарламаны қайтарады
- GetStackTrace () әдісі ерекше стек тізбегі бар массивті қайтарады
- PrintStackTrace () әдісі стек тізбегін көрсетеді.

Мысалы:

```
try{
    int x = 6/0;
}
catch(Exception ex){
      ex.printStackTrace();
}
```

Өзіңіздің ерекше кластарыңызды құру. Стандартты Java класты кітапханасында бар ерекше жағдайлар кластары бағдарламаны орындау кезінде орын алатын ерекше жағдайлардың көпшілігін сипаттайды, бірақ кейде өзіңіздің жеке логикаңызбен өзіндік ерекшелік кластарын құру қажет.

Өзіңіздің ерекше жағдайлар класын құру үшін сіз оны Exception класынан иеленуіңіз керек. Мысалы, факториал есептейтін класс бар, ал егер әдіске жіберілгендер саны 1-ден аз болса, біз ерекше жағдайды тастауымыз керек:

```
class Factorial{

    public static int getFactorial(int num) throws FactorialException{

        int result=1;
        if(num<1) throw new FactorialException("The number is less than 1", num);

        for(int i=1; i<=num;i++){

            result*=i;
        }
        return result;
    }
}

class FactorialException extends Exception{

    private int number;
    public int getNumber(){return number;}
    public FactorialException(String message, int num){

         super(message);
         number=num;
     }
}
```

Мұнда факториалды есептеумен байланысты қатені анықтау үшін Exception енетін және есептеу туралы барлық ақпаратты қамтитын FactorialException класы анықталған. FactorialException конструкторында қате туралы хабарлама Exception базалық класының конструкторына жіберіледі: super(message). Сонымен қатар, сандарды сақтауға арналған жеке өріс бөлінеді.

Ерекшеліктерді факториалды есептеу әдісінде тастау үшін throw операторын қолдану арқылы шығарылады: throw new FactorialException("Число не может быть меньше 1", num). Сонымен қатар, бұл ерекшелік try..catch арқылы өңделмегендіктен, throws операторының көмегімен өңдеуді шақырамыз: public static int getFactorial(int num) throws FactorialException

```
1  public static void main(String[] args){
2
3      try{
4          int result = Factorial.getFactorial(6);
5          System.out.println(result);
6      }
7      catch(FactorialException ex){
8
9          System.out.println(ex.getMessage());
10         System.out.println(ex.getNumber());
11     }
12 }
```

Рахмет!