Java Exceptions - Try...Catch

ПРЕПОДАВАТЕЛ: ИНЖ. В. МАРИНОВА

Java Exceptions

Когато се изпълнява код на Java, се откриват различни грешки: грешки при писането на код от програмиста, грешки от неправилни входни данни или други случайни събития.

Когато се установи грешка Java ще прекъсне работата на кода и ще генерира съобщение за грешка.

Терминологията за това събитие е:
Java will throw an exception (throw an error).

За да бъдат прихванати различните видове грешки се използват операторите try... catch...

Try….Catch…. Синтаксис:

```
try {
        //  Block of code to try
}
catch(Exception e) {
        //  Block of code to handle errors
}
```

В частта на **try** оператора се дефинира блокът от код, който ще се тества за грешки при изпълнението си.
В частта на **catch** оператора се дефинира блок от програмен код, който ще се изпълни, ако бъде прихваната грешка в частта на try блока.

Операторите try и catch се използват заедно.

Пример, при който ще възникне грешка:

```
public class MyClass {
    public static void main(String[ ] args) {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[10]); // error!
  }
}
```

Съобщението за грешка се генерира от системата и ще изглежда по подобен начин:

```
Exception in thread "main,, java.lang.ArrayIndexOutOfBoundsException: 10
        at MyClass.main(MyClass.java:4)
```

Тъй като в програмният код няма елемент с номер 10 в деклариранят масив.

За да прихванем този вид грешка, ще променим кода с операторите try…catch… по следният начин:

```
public class MyClass {
    public static void main(String[ ] args) {
        try {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[10]);
        } catch (Exception e) {
            System.out.println("Something went wrong.");
        }
  }
}
```

Съобщението на екрана ще е следното:

Something went wrong.

Оператор finally:
Този оператор се изпълнява след като кодът е прихванат от try...catch, независимо от резултата:

Пример:

```
public class MyClass {
    public static void main(String[] args) {
        try {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[10]);
        } catch (Exception e) {
            System.out.println("Something went wrong.");
        } finally {
            System.out.println("The 'try catch' is finished.");
        }
  }
}
```

Съобщението на екрана ще е следното:

```
Something went wrong.
The 'try catch' is finished.
```

Пример за друг вид грешка : при аритметична операция. Използване на повече от един оператор catch.

```
class Example1 {
    public static void main(String args[]) {
        int num1, num2;
        try {
            num1 = 0;
            num2 = 62 / num1;
            System.out.println(num2);
            System.out.println("Hey I'm at the end of try block");
        }
        catch (ArithmeticException e) {
            System.out.println("You should not divide a number by zero");  }
        catch (Exception e) {
            System.out.println("Exception occurred");  }
        System.out.println("I'm out of try-catch block in Java.");
  }
}
```

Съобщението на екрана :

```
You should not divide a number by zero
I'm out of try-catch block in Java.
```

Пример за прихващане на няколко вида грешки (няколко catch оператора) :

```
class Example2{
    public static void main(String args[]){
        try{
            int a[]=new int[7];
            a[4]=30/0;
            System.out.println("First print statement in try block");  }
        catch(ArithmeticException e){
            System.out.println("Warning: ArithmeticException");   }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Warning: ArrayIndexOutOfBoundsException");   }
        catch(Exception e){
            System.out.println("Warning: Some Other exception"); }
        System.out.println("Out of try-catch block...");
  }
}
```

Съобщение на екрана :

```
Warning: ArithmeticException
Out of try-catch block...
```

Вложени оператори Try…Catch…:

```
....
//Main try block
try {
        оператори 1;
        try {   //try-catch block inside another try block
                оператори 2;
                try {   //try-catch block inside nested try block
                        оператори 3;
                 }catch(Exception e2) {
                        //Exception Message
                }
        }catch(Exception e1) {
                //Exception Message
        }
}catch(Exception e3) {   //Catch of Main(parent) try block
      //Exception Message
}
....
```

```
class NestingDemo{
    public static void main(String args[]){
        try{ //main try-block
            try{ //try-block2
                try{ //try-block3
                    int arr[]= {1,2,3,4};
                    System.out.println(arr[10]);
                }catch(ArithmeticException e){
                    System.out.print("Arithmetic Exception");
                    System.out.println(" handled in try-block3");   }
            }catch(ArithmeticException e){
                    System.out.print("Arithmetic Exception");
                    System.out.println(" handled in try-block2");   }
        }
        catch(ArithmeticException e3){
            System.out.print("Arithmetic Exception");
            System.out.println(" handled in main try-block");   }
        catch(ArrayIndexOutOfBoundsException e4){
            System.out.print("ArrayIndexOutOfBoundsException");
            System.out.println(" handled in main try-block");   }
        catch(Exception e5){
            System.out.print("Exception");
            System.out.println(" handled in main try-block");   }
}}
```

Примери за вложени оператори

```
class Nest{
   public static void main(String args[]){
        try{   //Parent try block
              try{   //Child try block1
                     System.out.println("Inside block1");
                    int b =45/0;
                     System.out.println(b);
              }catch(ArithmeticException e1){
                     System.out.println("Exception: e1");  }
             try{  //Child try block2
                     System.out.println("Inside block2");
                      int b =45/0;
                     System.out.println(b);
              }catch(ArrayIndexOutOfBoundsException e2){
                     System.out.println("Exception: e2");   }
             System.out.println("Just other statement");
       }
       catch(ArithmeticException e3){
             System.out.println("Arithmetic Exception");
             System.out.println("Inside parent try catch block");  }
       catch(ArrayIndexOutOfBoundsException e4){
             System.out.println("ArrayIndexOutOfBoundsException");
             System.out.println("Inside parent try catch block");  }
       catch(Exception e5){
             System.out.println("Exception");
             System.out.println("Inside parent try catch block");  }
       System.out.println("Next statement..");
}}
```

Примери за вложени оператори

Резултат:

```
Inside block1
Exception: e1
Inside block2
Arithmetic Exception
Inside parent try catch block
Next statement..
```

Пояснения :

Имаме 2 try-catch блока в тялото на главния try . Отбелязани като block 1 и block 2 в примера.

Block1: деление на нула т.е. ArithmeticException, "Exception: e1" ще се отпечата.

Block2: Тук има ArithmeticException, но в този блок тя не се прихваща (защото ArrayIndexOutOfBoundsException е в този блок търсената грешка) и контролът се предава на главния try-catch(parent), който хваща грешката като ArithmeticException в родителският catch blocks. Това ще генерира съобщението “Inside parent try catch block”.

Parent Try Catch block: Тук няма открита грешка и за това ще се отпечата съобщението “Next statement..”.

Важно: Ако вложеният child catch block не прихване някоя грешка, то тя ще се прихване от външният parent catch block, ако има проверка за този вид грешка. Ако такава проверка не е сложена, системата ще генерира автоматично съобщение за грешката и нейният вид.

Благодаря за вниманието!