EXPLORE DESIGN PERFECTION

Introduction to Java

Carlos Kavka

ESTECO SpA

EXPLORE DESIGN PERFECTION

Introduction to Java

Part IV - Exceptions

esteco.com

Exceptions

The usual behavior on runtime errors is to abort the execution

```
public class TestExceptions1 {
  public static void main(String[] args) {

    String s = "Hello";
    System.out.print(s.charAt(10));
  }
}
```

For example, here there is an error in the charAt() call

```
$ java TestExceptions1
Exception in thread "main"
java.lang.StringIndexOutOfBoundsException:
String index out of range: 10
at java.lang.String.charAt(String.java:499)
at TestExceptions1.main(TestExceptions1.java:11)
```

Exceptions

```
public class TestExceptions2 {
  public static void main(String[] args) {

    String s = "Hello";
    try {
      System.out.print(s.charAt(10));
    } catch (Exception e) {
      System.out.println("No such position");
  }
}
```

The exception can be trapped by using a try-catch block

```
$ java TestExceptions2
No such position
```

Handling exceptions

It is possible to specify interest on a particular exception

```
public class TestExceptions4 {
  public static void main(String[] args) {

    String s = "Hello";
    try {
      System.out.print(s.charAt(10));
    } catch (StringIndexOutOfBoundsException e) {
      System.out.println("No such position");
      System.out.println(e.toString());
  }
}
```

And also send messages to an exception object

```
$ java TestExceptions4
No such position
java.lang.StringIndexOutOfBoundsException:
String index out of range: 10
```

Handling multiple exceptions

```
public static void printInfo(String sentence) {
   try {
      // get first and last char before the dot
      char first = sentence.charAt(0);
      char last = sentence.charAt(sentence.indexOf(".") - 1);
      String out = String.format("First: %c Last: %c",first, last);
      System.out.println(out);

   } catch (StringIndexOutOfBoundsException e1) {
      System.out.println("Wrong sentence, no dot?");

   } catch (NullPointerException e2) {
      System.out.println("Non valid string");

   } finally {
      System.out.println("done!");
   }
}
```

It is possible to add multiple catch blocks and optionally a single finally clause

Handling multiple exceptions

```
public static void printInfo(String sentence) throws {
   try {
     // get first and last char before the dot
     char first = sentence.charAt(0);
     char last = sentence.charAt(sentence.indexOf(".") - 1);
     String out = String.format("First: %c Last: %c",first, last);
     System.out.println(out);

   } catch (StringIndexOutOfBoundsException| NullPointerException e) {
     System.out.println("Non valid string");

   } finally {
       System.out.println("done!");
   }
}
```

Or join catch blocks if the action is the same

Throwing Exceptions

```
import java.io.*;

class WriteFile {
  public static void main(String[] args) {
    FileWriter f;
    BufferedWriter bf;
    try {
      f = new FileWriter("file1.text");
      bf = new BufferedWriter(f);
      String s = "Hello World!";
      bf.write(s,0,s.length());
      bf.newLine();
      bf.write("Java is nice!!!",8,5);
      bf.newLine();
      bf.close();
    } catch (IOException e) {
      System.out.println("Error with files:"+e.toString());
  }
}
```

Exception can be handled in the method...

Throwing Exceptions

```
public class WriteFile {
  public static void main(String[] args) {
    WriteFile wf = new WriteFile();
    try {
      wf.write();
    } catch (IOException e) {
      System.out.println("Error with files:"+e.toString());
    }
  }

  public void write() throws IOException {
    FileWriter f;
    BufferedWriter bf;
    f = new FileWriter("file1.text");
    bf = new BufferedWriter(f);
    String s = "Hello World!";
    bf.write(s,0,s.length());   bf.newLine();
    bf.write("Java is nice!!!",8,5);  bf.newLine();
    bf.close();
  }
}
```

Or thrown to the upper level

Creating new Exceptions

```
public class ExceptionTest {
   public static void main(String[] args) {
      ExceptionTest t = new ExceptionTest();
      try {
         t.test(3);
      } catch (MyException e) {
         e.printStackTrace();
      }
   }

   public int test(int x) throws MyException {
      if (x < 0)
         throw new MyException();
      return x - 1;
   }
}
```

New exceptions can be created and integrated into the hierarchy of exceptions

Thank you for your attention!

EXPLORE DESIGN PERFECTION