Java Exceptions

When executing Java code, different errors can occur:

coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).

errors due to wrong input

Consider the following example, where we input a String which is an invalid one to convert to an integer:

```
Scanner scan = new Scanner(System.in);
System.out.println("Please enter an integer number: ");
// int iNumber = Integer.parseInt(scan.nextLine());
int iNumber = Integer.parseInt("a"); // This will generate an error.
```

The error message will be something like this:

java.lang.NumberFormatException: 'For input string: "a"'

If an error occurs, we can use try...catch to catch the error and execute some code to handle it.

Java try and catch

The try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code to be executed, if an error occurs in the try block. The try and catch keywords come in pairs:

Syntax

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

In the following example, we use the variable inside the catch block (e) together with the built-in Message property, which outputs a message that describes the exception:

Example

```
import java.util.Scanner;

try
{
   Scanner scan = new Scanner(System.in);

   System.out.println("Please enter an integer number: ");
   // int iNumber = Integer.parseInt(scan.nextLine());
   int iNumber = Integer.parseInt("a"); // This will generate an error.

   System.out.println(iNumber);
   scan.close();
}
catch (Exception e)
{
   System.out.println(e); // Print entirely exception message.
   // System.out.println(e.getMessage()); // Print the error message.
}
```

Explanation

The output will be: "Input String was not in a correct format."

By the way, you can also output your own error message:

Example

```
import java.util.Scanner;

try
{
   Scanner scan = new Scanner(System.in);

   System.out.println("Please enter an integer number: ");
   // int iNumber = Integer.parseInt(scan.nextLine());
```

```
   int iNumber = Integer.parseInt("a"); // This will generate an error.

   System.out.println(iNumber);
   scan.close();
}
catch (Exception e)
{
   System.out.println("Something went wrong.");
}
```

Explanation

The output will be: "Something went wrong."

Finally

The finally statement lets you execute code, after try...catch, regardless of the result:

Example

```
import java.util.Scanner;

try
{
   Scanner scan = new Scanner(System.in);

   System.out.println("Please enter an integer number: ");
   System.out.println();
   // int iNumber = Integer.parseInt(scan.nextLine());
   int iNumber = Integer.parseInt("a"); // This will generate an error.

   System.out.println(iNumber);
   scan.close();
}
catch (Exception e)
{
   System.out.println("Something went wrong.");
   System.out.println();
}
finally
{
   System.out.println("The 'try catch' is finished.");
```

```
}
```

Explanation

The output will be:

Something went wrong.

The 'try catch' is finished.

The throw keyword

The throw statement allows you to create a custom error.

The throw statement is used together with an exception class. There are many exception classes available in Java:

ArithmeticException, InputMismatchException, IndexOutOfBoundsException, NumberFormatException, ...

Example

```
public static void checkArrayIndex(int iNumber)
{
   int[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

   if (iNumber > array.length - 1)
   {
      throw new IndexOutOfBoundsException();
   }
   else
   {
      System.out.println("The value of array[" + iNumber + "] is " + array[iNumber] + ".");
   }
}

public static void main(String[] args)
{
   checkArrayIndex(10);
}
```

Explanation

The error message displayed in the program will be:

IndexOutOfBoundsException: 'Index was outside the bounds of the array.

As mentioned above, when an error occurs due to wrong input, we can use try...catch to catch the error and execute some code to handle it. In addition to that, we can use Integer.parseInt() or scan.nextInt() method to check if the input is an integer number.

Example

```
import java.util.Scanner;

public static void main(String[] args)
{
   //String str = "2";
   String str = "a";

   try
   {
      Scanner scan = new Scanner(System.in);

      System.out.println("Please enter an integer number: ");
      System.out.println();
      // int iNumber = scan.nextInt(); // Scanner class object
      // int iNumber = Integer.parseInt(scan.nextLine());
      int iNumber = Integer.parseInt(str); // This will generate an error.

      System.out.println(iNumber);
      scan.close();
   }
   catch (Exception e)
   {
      System.out.println("An invalid number or String!");
   }
}
```