Java Exceptions - Try...Catch

❮ Previous Next ❯

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).

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
}
```

Consider the following example:

This example will generate an error, because **myNumbers[10]** does not exist.

```
public class MyClass {
  public static void main(String[ ] args) {
    int[] myNumbers = new int[5];
    System.out.println(myNumbers[10]);
  }
}
```

Run example »

The output (error) will be something like this:

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

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

Example

```
public class MyClass {
  public static void main(String[ ] args) {
    try {
      int[] myNumbers = new int[5];
      System.out.println(myNumbers[10]);
    } catch (Exception e) {
      System.out.println("Something went wrong.");
    }
  }
}
// Outputs "Something went wrong."
```

Run example »

Finally

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

Example

```java
public class MyClass {
  public static void main(String[] args) {
    try {
      int[] myNumbers = new int[5];
      System.out.println(myNumbers[10]);
    } catch (Exception e) {
      System.out.println("Something went wrong.");
    } finally {
      System.out.println("The 'try catch' is finished.");
    }
  }
}

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

Run example »

The throw keyword

The `throw` statement allows you to create a custom error.

The `throw` statement is used together with an **exception type**. There are many exception types available in Java: `ArithmeticException`, `ClassNotFoundException`, `ArrayIndexOutOfBoundsException`, `SecurityException`, etc.

The exception type is often used together with a custom method. Don't worry if you don't understand the example below, you will learn more about methods in the next chapter:

Example

Throw an exception if **age** is below 18 (print "Access denied"). If the age is 18 or older, print "Access granted":

```
public class MyClass {
  static void checkAge(int age){
    if(age < 18) {
      throw new ArithmeticException("Access denied - You must be at least 18 years old.");
    }
    else {
      System.out.println("Access granted - You are old enough!");
    }
  }

  public static void main(String[] args) {
    checkAge(15); // Set age to 15 (which is below 18...)
  }
}
```

Run example »

If age was 20, you would not get an exception:

Example

```
checkAge(20);
```

Run example »

❮ Previous Next ❯