# Java - Built-in Exceptions

## Built-in Exceptions in Java

Java defines several exception classes inside the standard package **java.lang**.

The most general of these exceptions are subclasses of the standard type RuntimeException. Since java.lang is implicitly imported into all Java programs, most exceptions derived from RuntimeException are automatically available.

## Types of Java Built-in Exceptions

Built-in Exceptions in Java are categorized into two categories Checked Exceptions and Unchecked Exceptions.

- **Checked Exceptions**: The checked exceptions are handled by the programmer during writing the code, they can be handled using the try-catch block. These exceptions are checked at compile-time.
- **Unchecked Exceptions**: The unchecked exceptions are not handled by the programmer. These exceptions are thrown on run-time. Some of the unchecked exceptions are NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException, etc.



## Common Built-in Exceptions in Java

Java defines several other types of exceptions that relate to its various class libraries. Following is the list of Java Unchecked and Checked RuntimeException.

| Sr.No. | Exception & Description |
|---|---|
| 1 | **ArithmeticException**<br>Arithmetic error, such as divide-by-zero. |
| 2 | **ArrayIndexOutOfBoundsException**<br>Array index is out-of-bounds. |
| 3 | **ArrayStoreException**<br>Assignment to an array element of an incompatible type. |

| | |
|---|---|
| 4 | **ClassCastException**<br>Invalid cast. |
| 5 | **IllegalArgumentException**<br>Illegal argument used to invoke a method. |
| 6 | **IllegalMonitorStateException**<br>Illegal monitor operation, such as waiting on an unlocked thread. |
| 7 | **IllegalStateException**<br>Environment or application is in incorrect state. |
| 8 | **IllegalThreadStateException**<br>Requested operation not compatible with the current thread state. |
| 9 | **IndexOutOfBoundsException**<br>Some type of index is out-of-bounds. |
| 10 | **NegativeArraySizeException**<br>Array created with a negative size. |
| 11 | **NullPointerException**<br>Invalid use of a null reference. |
| 12 | **NumberFormatException**<br>Invalid conversion of a string to a numeric format. |
| 13 | **SecurityException**<br>Attempt to violate security. |
| 14 | **StringIndexOutOfBounds**<br>Attempt to index outside the bounds of a string. |
| 15 | **UnsupportedOperationException**<br>An unsupported operation was encountered. |
| 16 | **ClassNotFoundException**<br>Class not found. |
| 17 | **CloneNotSupportedException**<br>Attempt to clone an object that does not implement the Cloneable interface. |
| 18 | **IllegalAccessException**<br>Access to a class is denied. |
| 19 | **InstantiationException**<br>Attempt to create an object of an abstract class or interface. |

| | |
|---|---|
| 20 | **InterruptedException**<br>One thread has been interrupted by another thread. |
| 21 | **NoSuchFieldException**<br>A requested field does not exist. |
| 22 | **NoSuchMethodException**<br>A requested method does not exist. |

## Examples of Java Built-in Exception

### Example 1: Demonstrating Arithmetic Exception Without try-catch

In this example, we're creating an error by dividing a value by 0. In this case, an unchecked exception will be raised. Being unchecked, compiler won't complain and program will compile successfully. Once program runs, the exception will be thrown and JVM will intercepts the same and terminate the program before printing the last statement.

</> Open Compiler

```
package com.tutorialspoint;

public class ExcepTest {

   public static void main(String args[]) {

      int b = 0;
      int c = 1/b;
      System.out.println("c :" + c);
   }
}
```

### Output

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at com.tutorialspoint.ExcepTest.main(ExcepTest.java:8)
```

### Example 2: Demonstrating Arithmetic Exception With try-catch

In this example, we're handling unchecked exception. As first step, we're generating an error by dividing a value by 0. In this case, an unchecked exception will be raised. We're handling via ArithmeticException. Once program runs, the exception will be thrown and catch block will intercepts the same and print the last statement.

</> Open Compiler

```
package com.tutorialspoint;

public class ExcepTest {

   public static void main(String args[]) {
      try {
         int b = 0;
         int c = 1/b;
         System.out.println("c :" + c);
      }
      catch (ArithmeticException e) {
         System.out.println("Exception thrown  :" + e);
      }
      System.out.println("Out of the block");
   }
}
```

## Output

```
Exception thrown  :java.lang.ArithmeticException: / by zero
Out of the block
```

## Example 3: Demonstrating No Such Method Exception

In this example, we're showcasing that a checked exception is to be handled by code otherwise compiler will complain. Whenever a method throws a checked exception, it has to either handle the exception or declare throws exception statement as we're doing for getName() method. When we try to run the method, JVM complains the compilation problem as shown in output listed below:

</> Open Compiler

```
package com.tutorialspoint;

public class ExcepTest {

   public static void main(String args[]) {
      ExcepTest excepTest = new ExcepTest();
      excepTest.getName();
   }

   private String getName() throws NoSuchMethodException {
       throw new NoSuchMethodException();
   }
}
```

## Output

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
   Unhandled exception type NoSuchMethodException

   at com.tutorialspoint.ExcepTest.main(ExcepTest.java:7)
```