Java Exception Handling

Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, etc.

Advantage of Exception Handling

The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. Let's consider a scenario:

1. statement 1;
2. statement 2;
3. statement 3;
4. statement 4;
5. statement 5;//exception occurs
6. statement 6;
7. statement 7;
8. statement 8;
9. statement 9;
10. statement 10;

Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed. However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling in [Java](https://www.javatpoint.com/java-tutorial).



Major reasons why an exception occurs:

* Invalid user input
* Device failure
* Loss of network connection
* Physical limitations (out of disk memory)
* Code errors
* Opening an unavailable file

What is Exception?

Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program’s instruction.

Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

1. Checked exception
2. Unchecked exception
3. Error

Difference between checked and unchecked exceptions

1. Checked exception

Checked exceptions are exceptions that we developers should anticipate and handle properly.

For example ; if a file doesn’t exist maybe it got deleted , instead of letting runtime terminate our program we’d better display a friendly message to the user says ”hey file doesn’t exist ”.

The class that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc.

Checked exception are checked at compile-time.

1. Unchecked exception

The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullpointerException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

1. Error

Error is irrecoverable. Some example of errors are

OutOfMemeoryError, VirtualMachineError, infinite recursion etc.

## **Java Exception Keywords**

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| try | The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally. |
| catch | The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later. |
| finally | The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not. |
| throw | The "throw" keyword is used to throw an exception. |
| throws | The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature. |

Java provides five keywords that are used to handle the exception. The following table describes each.