\*\*\*\*\*\*\*\*\*\*Multithreading in Java:-

1. [Multithreading](https://www.javatpoint.com/multithreading-in-java)
2. [Multitasking](https://www.javatpoint.com/multithreading-in-java#multitasing)
3. [Process-based multitasking](https://www.javatpoint.com/multithreading-in-java#multiprocessing)
4. [Thread-based multitasking](https://www.javatpoint.com/multithreading-in-java#multithreading)
5. [What is Thread](https://www.javatpoint.com/multithreading-in-java#thread)

**Multithreading in**[**Java**](https://www.javatpoint.com/java-tutorial) is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

### Advantages of Java Multithreading

1) It **doesn't block the user** because threads are independent and you can perform multiple operations at the same time.

2) You **can perform many operations together, so it saves time**.

3) Threads are **independent**, so it doesn't affect other threads if an exception occurs in a single thread.

# Difference between throw and throws in Java

There are many differences between throw and throws keywords. A list of differences between throw and throws are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **throw** | **throws** |
| 1) | Java throw keyword is used to explicitly throw an exception. | Java throws keyword is used to declare an exception. |
| 2) | Checked exception cannot be propagated using throw only. | Checked exception can be propagated with throws. |
| 3) | Throw is followed by an instance. | Throws is followed by class. |
| 4) | Throw is used within the method. | Throws is used with the method signature. |
| 5) | You cannot throw multiple exceptions. | You can declare multiple exceptions e.g. public void method()throws IOException,SQLException. |

# Difference between final, finally and finalize

There are many differences between final, finally and finalize. A list of differences between final, finally and finalize are given below:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **final** | **finally** | **finalize** |
| 1) | **Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.** | **Finally is used to place important code, it will be executed whether exception is handled or not.** | **Finalize is used to perform clean up processing just before object is garbage collected.** |
| 2) | **Final is a keyword.** | **Finally is a block.** | **Finalize is a method.** |

**\*\*\*\*\*\*\*\*\*\*\*Java provides three types of control flow statements.**

1. **Decision Making statements**
   * if statements
   * switch statement
2. **Loop statements**
   * do while loop
   * while loop
   * for loop
   * for-each loop
3. **Jump statements**
   * break statement
   * continue statement

## \*\*\*\*\*What is Exception in Java:-

**\*\*\*** Exception is an abnormal condition.

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

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

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

The java.lang.Throwable class is the root class of Java Exception hierarchy which is inherited by two subclasses: Exception and Error. A hierarchy of Java Exception classes are given below:



## Difference between Checked and Unchecked Exceptions:-

### 1) Checked Exception

The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

### 2) Unchecked Exception

The classes which inherit RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

### 3) Error

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

## \*\*\*Java Exception Keywords:-

There are 5 keywords which are used in handling exceptions in Java.

## 1. try:- The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.

## 2. 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.

## 3. finally:- The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.

## 4. throw:- The "throw" keyword is used to throw an exception.

## 5. throws:- The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

## \*\*\*Common Scenarios of Java Exceptions:-

1. **1) A scenario where Arithmetic Exception occurs:- nt** a=50/0;//ArithmeticException

### 2) A scenario where NullPointerException occurs

If we have a null value in any [variable](https://www.javatpoint.com/java-variables), performing any operation on the variable throws a NullPointerException.

1. String s=**null**;

2. System.out.println(s.length());//NullPointerException

### 3) A scenario where NumberFormatException occurs

The wrong formatting of any value may occur NumberFormatException. Suppose I have a [string](https://www.javatpoint.com/java-string) variable that has characters, converting this variable into digit will occur NumberFormatException.

1. String s="abc";
2. **int** i=Integer.parseInt(s);//NumberFormatException

### 4) A scenario where ArrayIndexOutOfBoundsException occurs

If you are inserting any value in the wrong index, it would result in ArrayIndexOutOfBoundsException as shown below:

1. **int** a[]=**new** **int**[5];
2. a[10]=50; //ArrayIndexOutOfBoundsException