# Multithreading🡪

Multithreading in Java is a process of executing multiple threads simultaneously.

**Multitasking-**

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved in two ways:

* Process-based Multitasking (Multiprocessing)
* Thread-based Multitasking (Multithreading)

**Thread-**

A thread is a lightweight subprocess, the smallest unit of processing.

Threads are independent. If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area.

**Life cycle of a Thread-**

1. New
2. Active
3. Blocked / Waiting
4. Timed Waiting
5. Terminated

**Create Thread🡪**

1. By extending Thread class
2. By implementing Runnable interface.

# Java8

## Functional Interface🡪

The Interface which has only one Abstract method. Also, can have any default and static method.

This can also contain the Object class method. Like hasCode, toString etc.

Functional interface🡪

1. **Predicate:**

Purpose: The Predicate interface represents a boolean-valued function of one argument. It is often used for filtering and conditional checks.

Method: It declares a single abstract method test(T t), which takes an argument of type T and returns a boolean result.

Example: You can use a Predicate to filter elements in a collection based on certain criteria. For instance, you can check if a number is even or odd, if a string contains a certain substring, etc.

1. **Function:**

Purpose: The Function interface represents a function that accepts one argument and produces a result. It is commonly used for data transformation and mapping.

Method: It declares a single abstract method apply(T t), which takes an argument of type T and returns a result of type R.

Example: You can use a Function to convert or transform data from one form to another. For instance, you can convert a string to uppercase, parse a string to an integer, apply mathematical operations, etc.

1. **Consumer:**

Purpose: The Consumer interface represents an operation that accepts a single input argument and returns no result. It is typically used for performing side effects, such as printing output, modifying state, or interacting with external systems.

Method: It declares a single abstract method accept(T t), which takes an argument of type T and performs an action.

Example: You can use a Consumer to iterate over elements in a collection and perform some action on each element, such as printing, logging, updating a database, etc.

**4-Supplier:**

Purpose: The Supplier interface represents a supplier of results. It has no input arguments and produces a result upon request. It is commonly used for lazy initialization and providing values on demand.

Method: It declares a single abstract method get(), which returns a result of type T.

Example: You can use a Supplier to provide values dynamically, such as generating random numbers, fetching data from an external source, creating new objects, etc.

## 2.2