COMPUTER CODING CLUB
MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY, PRAYAGRAJ

SOFTABLITZ

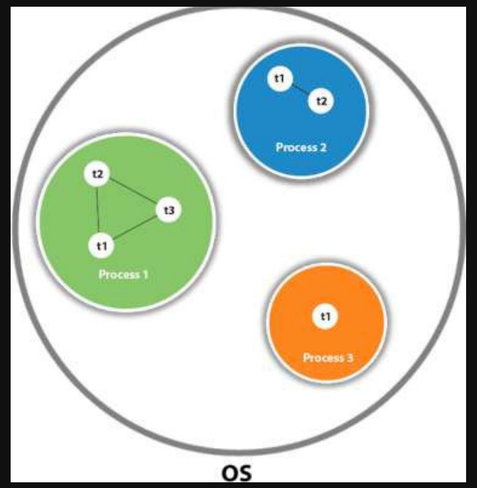
Multithreading

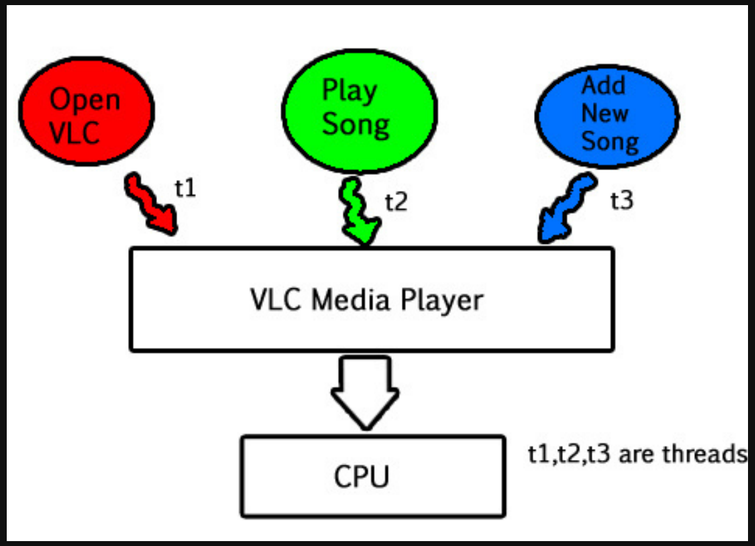
Agenda for today

- Multithreading
- Process vs Thread
- Lifecycle of a thread
- Multithreading in Java
- Implementation

What is Multithreading?

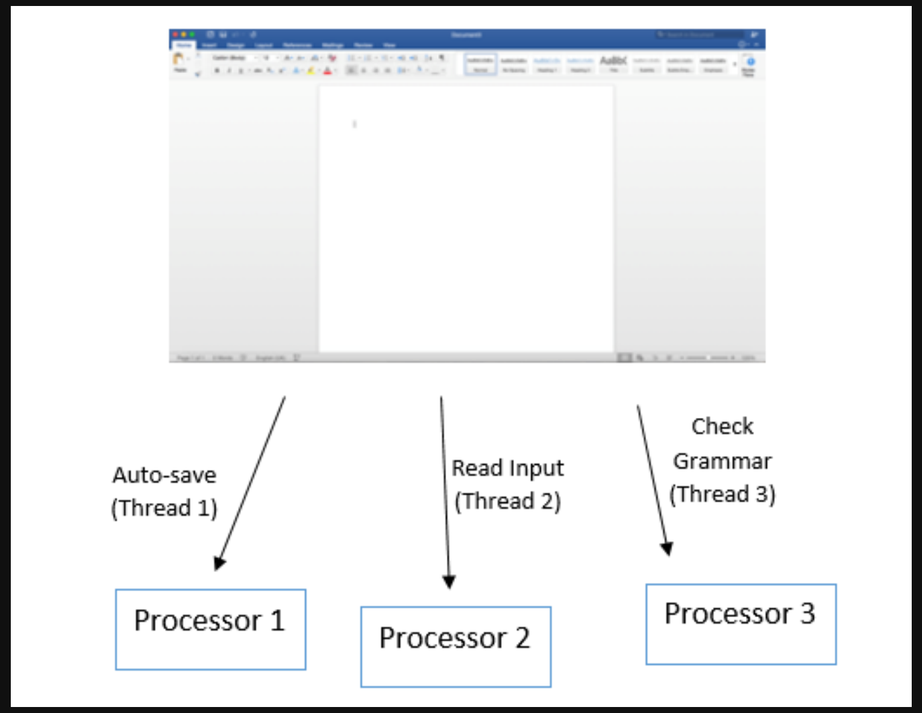
- Multi threading is an execution model that allows a single process to have multiple code segments (i.e., threads) running concurrently within the “context” of that process.
- Why multithreading:
  - To increase parallelism
  - To make most of the available CPU resources.
  - To improve application responsiveness and give better interaction with the user.
- e.g. VLC media player, where one thread is used for opening the VLC media player, one thread for playing a particular song and another thread for adding new songs to the playlist.





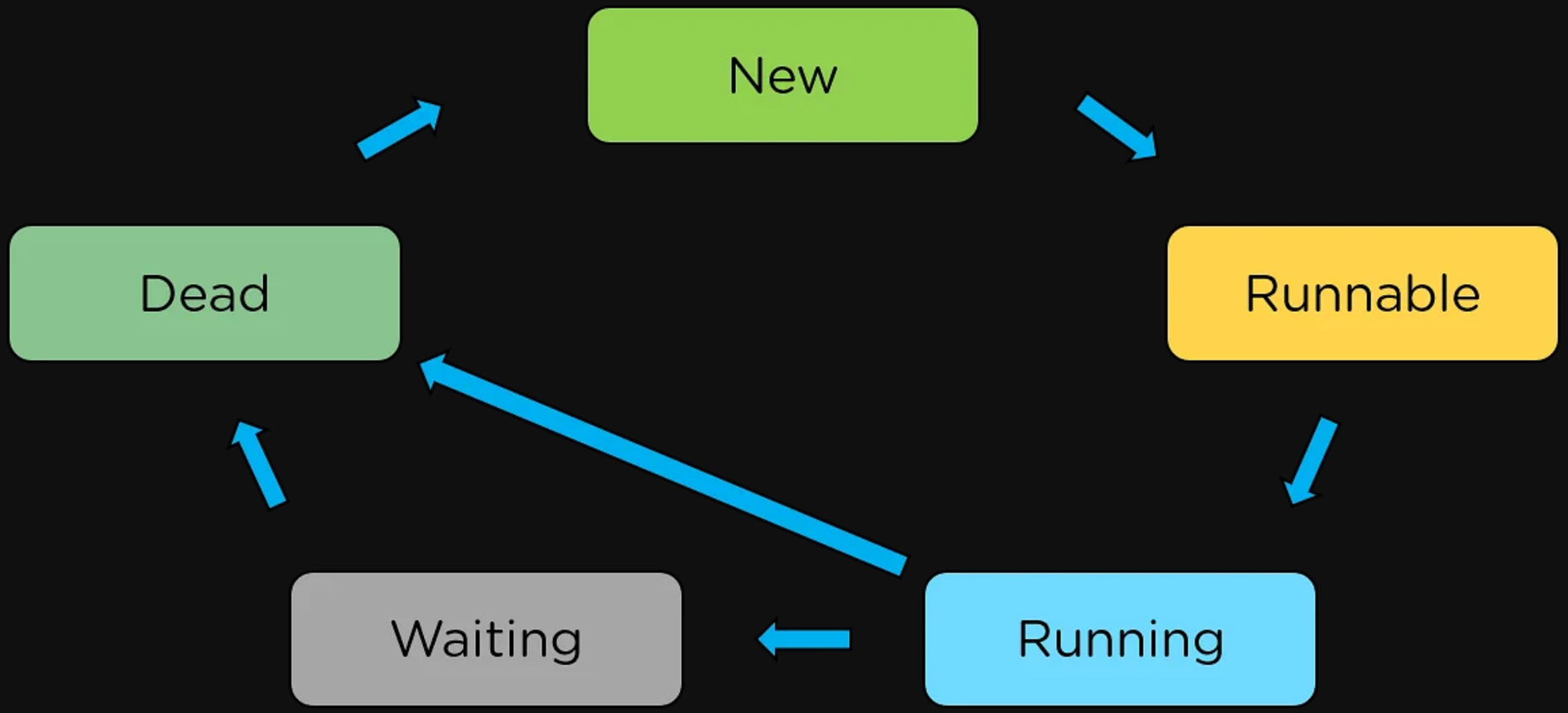
Process vs Thread

- When we execute a program or an application, a process is initiated. Each process comprises of one or more threads.
- A process is a program in execution whereas a thread is nothing but a segment of a process.



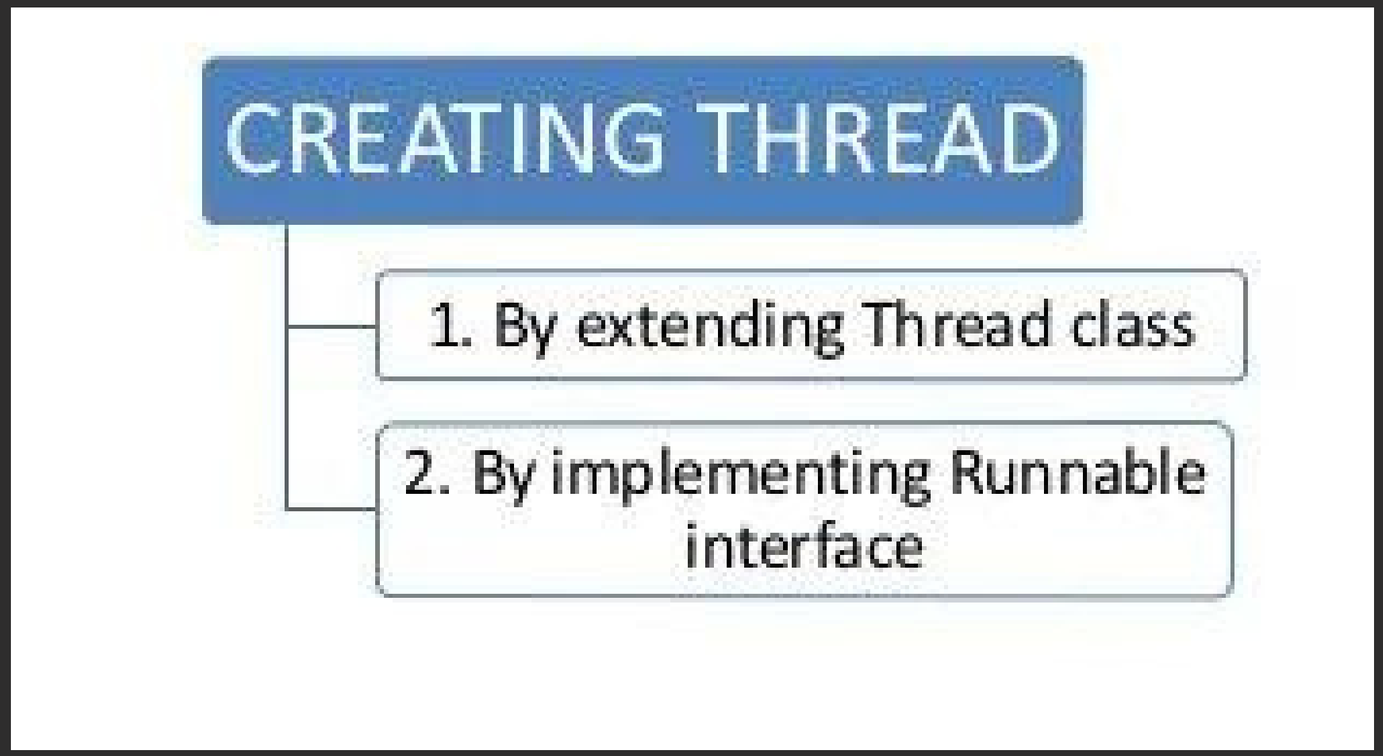
Consider a word application, It is a single process and when you type in word, a spell checker will keep working in the background to correct your words, this job can be considered as a thread.

Lifecycle of a thread

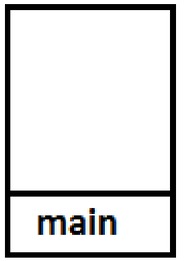


Multithreading in Java

- Java supports multithreading through Thread class.
- Each thread belongs to the Java.lang.Thread class.



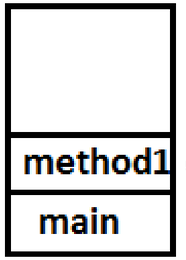
1) When we enter main() method



stack 1 (main thread's)

As soon as main is called by JVM it is pushed on Stack

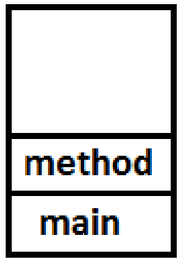
2) When main() calls method1() method



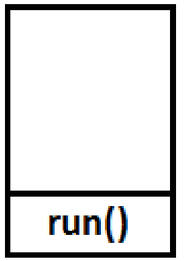
stack 1 (main thread's)

As soon as main calls method1 (), method1 pushed on Stack

3) When methd1() calls thread.start()



stack 1
(main thread's)



stack 2
(Thread-1's)

method1() creates new thread by calling thread.start (), as threads have their own stack - new stack is created.

```java
class MyThread extends Thread{
    public void run(){
        System.out.println("in run() method");
    }
}

public class MyClass {
    public static void main(String...args){
        System.out.println("In main() method");
        method1();
    }

    static void method1(){
        MyThread obj=new MyThread();
        obj.start();
    }
}

/*OUTPUT

currentThreadName= main
in run() method
currentThreadName= Thread-1

*/
```

Implementation

Thread class

- Extend the class Thread
- Override run() method
- Create object of above class
- Start thread using this object only

Runnable interface

- Implement interface Runnable
- Implement Abstract function run()
- Create an object of this class
- Create an object of thread class by passing above object in constructor and name of thread.
- Start thread