Multithreading

Presented by,

Prashant Srivastava

Assistant Professor

CSE Dept (SGI-Allahabad)

Table Of Contents

What is Threads

- **Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and ‘multithreading, both are used to achieve multitasking.**
- **multitasking is when multiple processes share common processing resources such as a CPU. Multi threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.**

Demo

What is Multithreaded Applications

Multi threaded programming language which means we can develop multi threaded program using Java. A multi threaded program contains two or more parts that can run concurrently and each part can handle different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

Multithreading is a conceptual programming paradigm where a program is divided into two or more subprogram (processes), which can implemented at the same time in parallel.

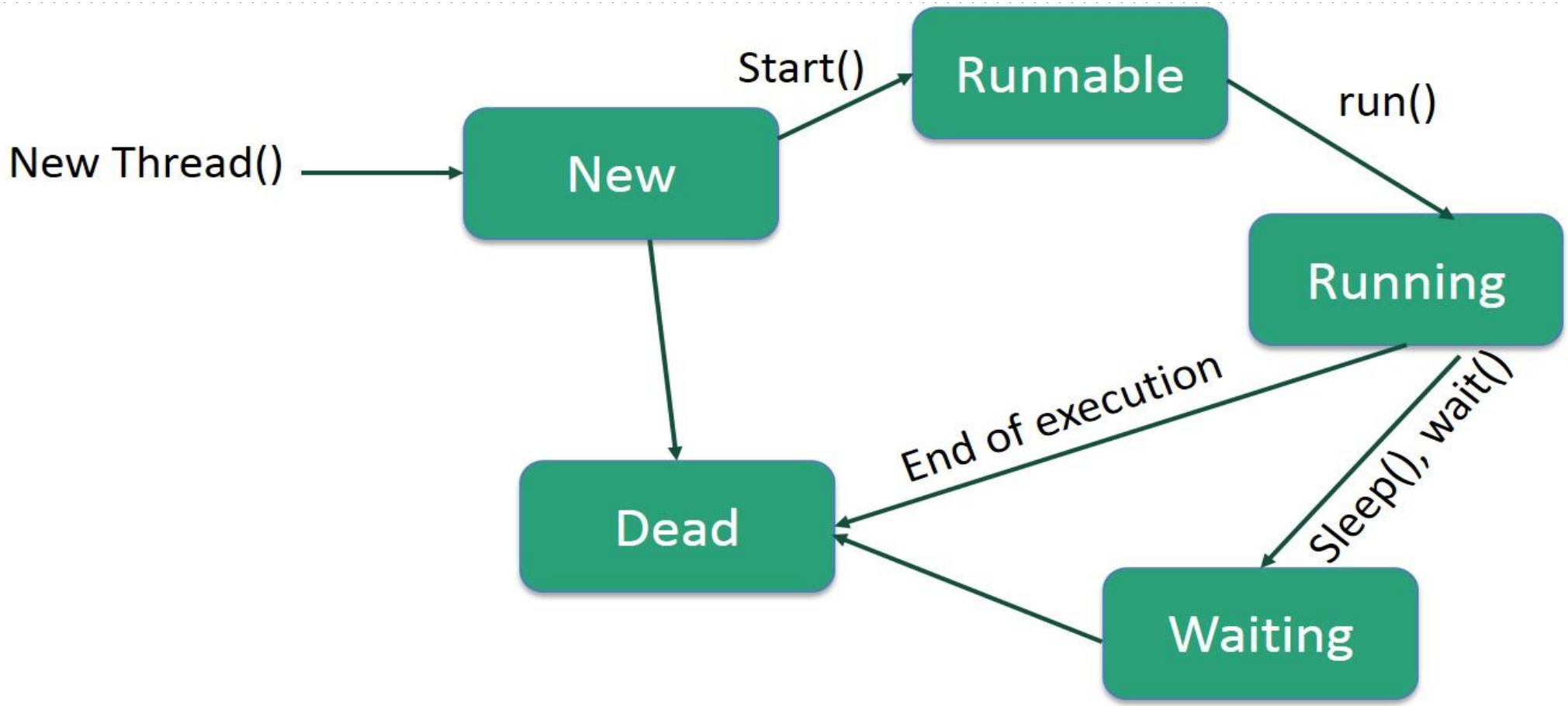
Advantage of Java Multithreading

1) It doesn't block the user because threads are independent and you can perform multiple operations at 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 exception occur in a single thread.

Life Cycles Of Multithreading



Contd..

New: A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.

Runnable: After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

Waiting: Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task.A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.

Timed waiting: A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

Contd..

Terminated (Dead): A runnable thread enters the terminated state when it completes its task or otherwise terminates.

Multithreading Implementation Using java

A new thread can be created in two ways

- **By creating a thread class:** Define a class that extends Thread class and override its run() method with the code required by the thread.
- **By converting s class to a thread:** Define a class that implements Runnable interface. The Runnable interface has only one method, **run(),** that is to be defined in the method with the code to be executed by the thread

Creating threads using the thread class

```
Class A extends Thread
{
Public void run()
{
for(int i=1;i<=5;i++)
{
System.out.println("\t From Thread A: i="+i);
}
System.out.println("Exit From A");
}
}
Class B extends Thread
{
Public void run()
{
for(int j=1;j<=5;j++)
{
System.out.println("\t From Thread B: j="+j);
}
```