Web Technology
(KCS-602)
Unit 1

Prepared By

Abhishek Kesharwani

Assistant Professor,UCER Naini,Allahabad

Multithreading in Java

- **Multithreading in java** is a process of executing multiple threads simultaneously.
- Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.
- 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 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.

What is Thread in java

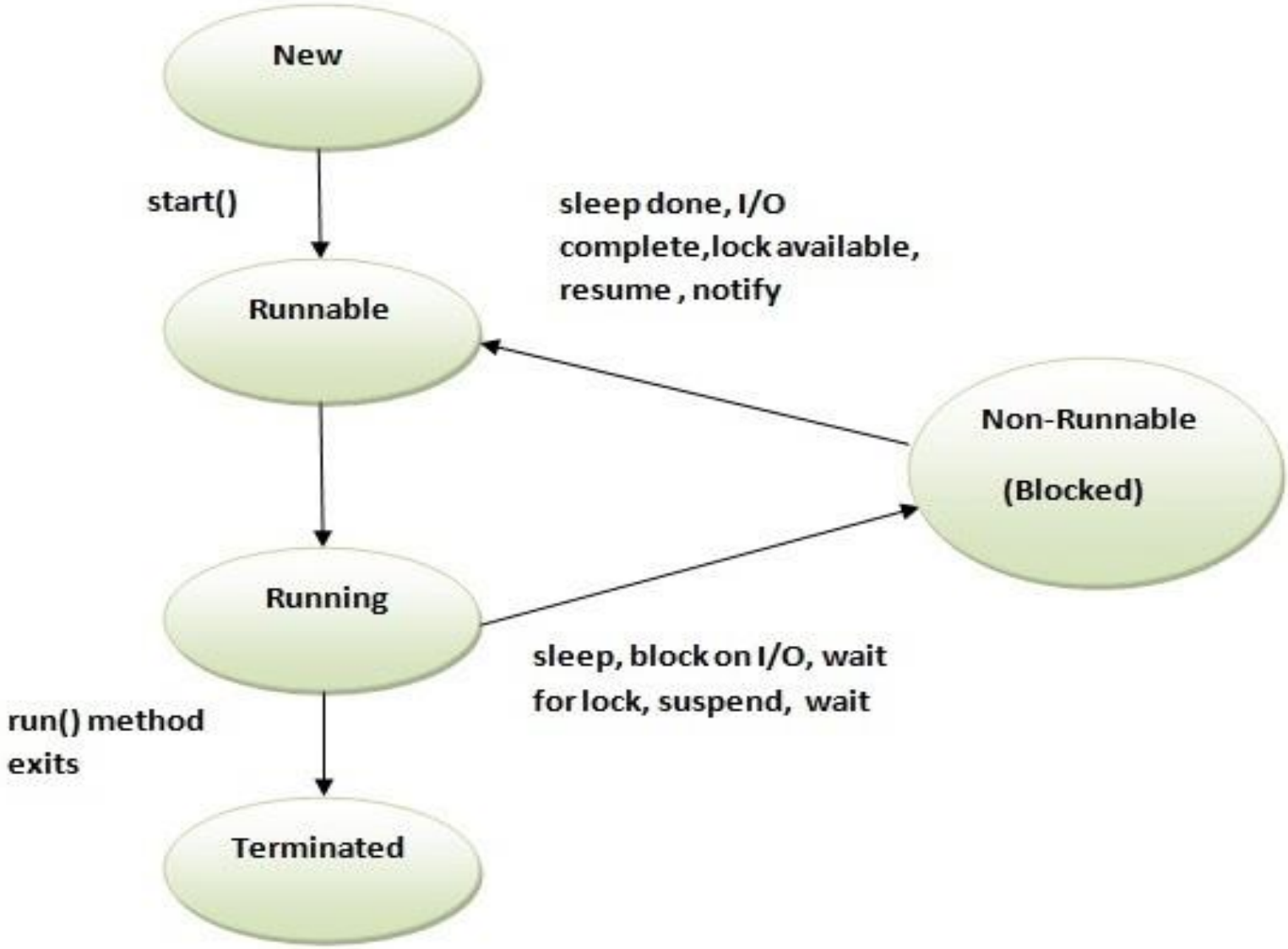
- A thread is a lightweight sub process, a smallest unit of processing. It is a separate path of execution.
- Threads are independent, if there occurs exception in one thread, it doesn't affect other threads. It shares a common memory area.

Note: At a time one thread is executed only.

Life cycle of a Thread (Thread States)

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

- New
- Runnable
- Running
- Non-Runnable (Blocked)
- Terminated


New
start()
Runnable
sleep done, I/O
complete,lock available,
resume , notify
Non-Runnable
(Blocked)
Running
sleep, block on I/O, wait
for lock, suspend, wait
run() method
exits
Terminated

1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

How to create thread

There are two ways to create a thread:

- By extending Thread class
- By implementing Runnable interface.

Thread class:

- Thread class provide constructors and methods to create and perform operations on a thread.
- Thread class extends Object class and implements Runnable interface.

Commonly used methods of Thread class:

- **public void run():** is used to perform action for a thread.
- **public void start():** starts the execution of the thread.JVM calls the run() method on the thread.
- **public void sleep(long miliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
- **public int getPriority():** returns the priority of the thread.
- **public int setPriority(int priority):** changes the priority of the thread.

- **public String getName():** returns the name of the thread.
- **public void setName(String name):** changes the name of the thread.
- **public Thread currentThread():** returns the reference of currently executing thread.
- **public int getId():** returns the id of the thread.
- **public Thread.State getState():** returns the state of the thread.
- **public boolean isAlive():** tests if the thread is alive.
- **public void yield():** causes the currently executing thread object to temporarily pause and allow other threads to execute.

- **public void suspend():** is used to suspend the thread(depricated).
- **public void resume():** is used to resume the suspended thread(depricated).
- **public void stop():** is used to stop the thread(depricated).

Starting a thread

- **start() method** of Thread class is used to start a newly created thread. It performs following tasks:A new thread starts(with new callstack).
- The thread moves from New state to the Runnable state.
- When the thread gets a chance to execute, its target run() method will run.

Java Thread Example by extending Thread class

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

Java Thread Example by implementing Runnable interface

```
class Multi3 implements Runnable{
public void run(){
System.out.println("thread is running...");
}
  public static void main(String args[]){
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);
t1.start();
 }
}
```

Sleep method in java

- The sleep() method of Thread class is used to sleep a thread for the specified amount of time.

Syntax of sleep() method in java

```
public static void sleep(long miliseconds)throws InterruptedException
```

```
class TestSleepMethod1 extends Thread{
 public void run(){
  for(int i=1;i<5;i++){
    try{Thread.sleep(500);}catch(InterruptedException e)
{System.out.println(e);}
    System.out.println(i);
  }
 }
 public static void main(String args[]){
  TestSleepMethod1 t1=new TestSleepMethod1();
  TestSleepMethod1 t2=new TestSleepMethod1();
     t1.start();
  t2.start();
 }
}
```

Naming Thread

The Thread class provides methods to change and get the name of a thread. By default, each thread has a name i.e. thread-0, thread-1 and so on.

public String getName(): is used to return the name of a thread.

public void setName(String name): is used to change the name of a thread.

```
class TestMultiNaming1 extends Thread{
  public void run(){
   System.out.println("running...");
  }
 public static void main(String args[]){
  TestMultiNaming1 t1=new TestMultiNaming1();
  TestMultiNaming1 t2=new TestMultiNaming1();
  System.out.println("Name of t1:"+t1.getName());
  System.out.println("Name of t2:"+t2.getName());

  t1.start();
  t2.start();

  t1.setName(“CSA Webtech");
  System.out.println("After changing name of t1:"+t1.getName());
 }
}
```

Priority of a Thread (Thread Priority)

- Each thread have a priority.
- Priorities are represented by a number between 1 and 10.
- In most cases, thread schedular schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses.

Three constants defined in Thread class

- public static int MIN_PRIORITY
- public static int NORM_PRIORITY
- public static int MAX_PRIORITY

Note: Default priority of a thread is 5 (NORM_PRIORITY). The value of MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10.

```
class TestMultiPriority1 extends Thread{
 public void run(){
   System.out.println("running thread name is:"+Thread.currentT
    hread().getName());
   System.out.println("running thread priority is:"+Thread.current
    Thread().getPriority());
    }
 public static void main(String args[]){
  TestMultiPriority1 m1=new TestMultiPriority1();
  TestMultiPriority1 m2=new TestMultiPriority1();
  m1.setPriority(Thread.MIN_PRIORITY);
  m2.setPriority(Thread.MAX_PRIORITY);
  m1.start();
  m2.start();
   }}
```