

# Threads : Java



# What is a Thread?

- ▶ **facility to allow multiple activities to coexist within a single process.**
- ▶ **Represents a separate path of execution of a group of statements**
- ▶ Java is the first language to include threading within the language, rather than treating it as a facility of the OS
- ▶ **Video Game example**
  - 1.one thread for graphics
  - 2.one thread for user interaction
  - 3.one thread for networking
- ▶ **Server Example**
  - 1.Do various jobs
  - 2.Handle Several Clients

# Main Thread

- ❑ Default Thread in any Java Program
- ❑ JVM uses to execute program statements
- Program To Find the Main Thread

Class Current

```
{  
    public static void main(String args[])  
    {  
        Thread t=Thread.currentThread();  
        System.out.println("Current Thread: "+t);  
        System.out.println("Name is: "+t.getName());  
    }  
}
```

**Output:**

```
C:\> javac Current.java
```

```
C:\> java Current
```

```
Current Thread: Thread[main,5,main]
```

```
Name is: main
```

# Threads in Java

## Creating threads in Java:

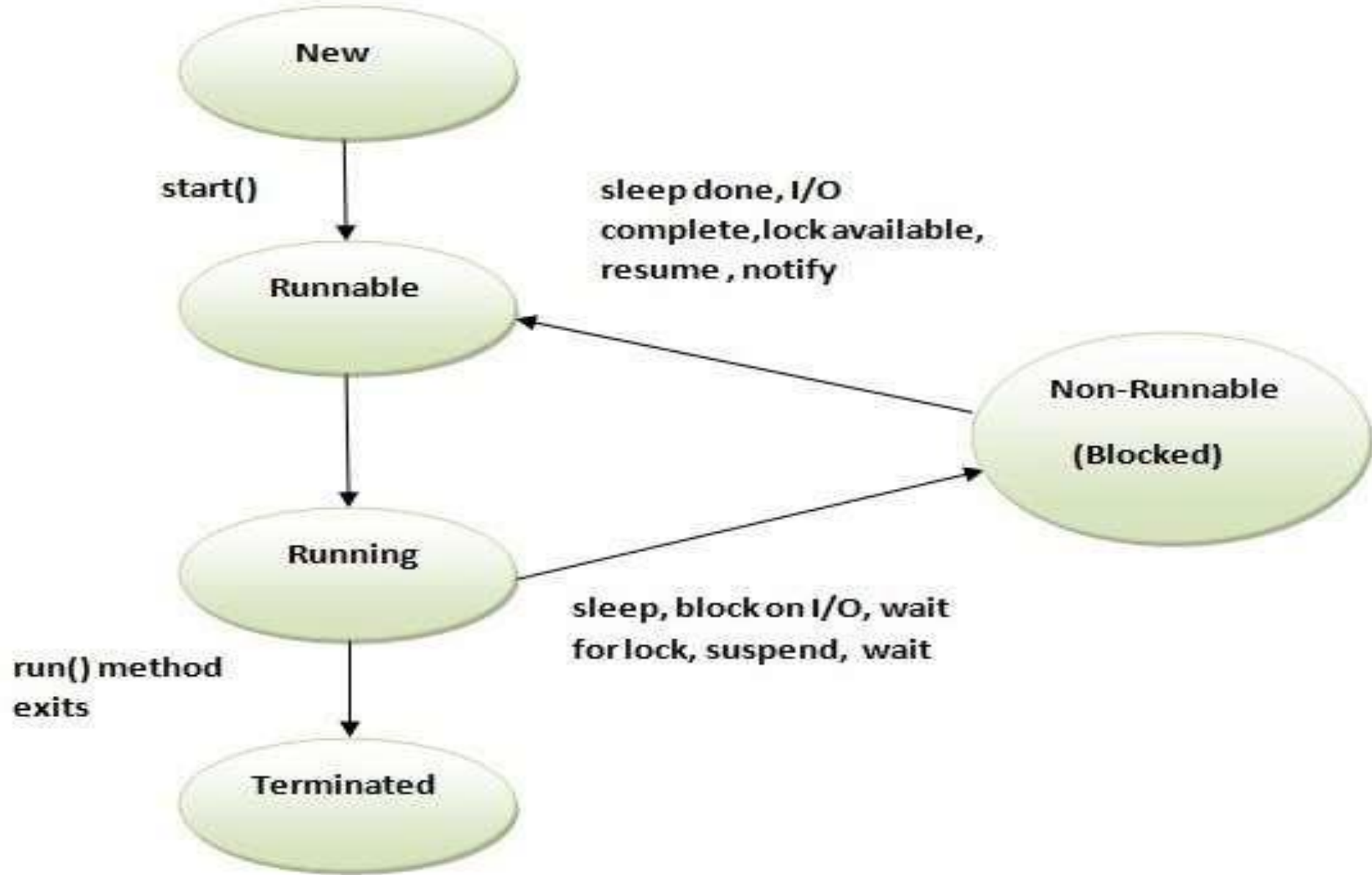
### ➔ Extend `java.lang.Thread` class

- ❑ `run()` method must be overridden (similar to main method of sequential program)
- ❑ `run()` is called when execution of the thread begins
- ❑ A thread terminates when `run()` returns
- ❑ `start()` method invokes `run()`

OR

### ➔ Implement `java.lang.Runnable` interface

# Life cycle of a Thread





## ❑ **New**

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

## ❑ **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.

## ❑ **Running**

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

## ❑ **Non-Runnable (Blocked)**

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

## ❑ **Terminated**

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

# Thread Priority

7

- Each thread is assigned a default priority of `Thread.NORM_PRIORITY` (constant of 5).
- You can reset the priority using `setPriority(int priority)`.
- Some constants for priorities include:
  - `Thread.MIN_PRIORITY`
  - `Thread.MAX_PRIORITY`
  - `Thread.NORM_PRIORITY`
- By default, a thread has the priority level of the thread that created it.

## ❖ Thread Synchronization

A shared resource may be corrupted if it is accessed simultaneously by multiple threads.

**Example:** two unsynchronized threads accessing the same bank account may cause conflict.

- Known as a *race condition* in multithreaded programs.
- ❑ A *thread-safe* class does not cause a race condition in the presence of multiple threads.



# Synchronized

9

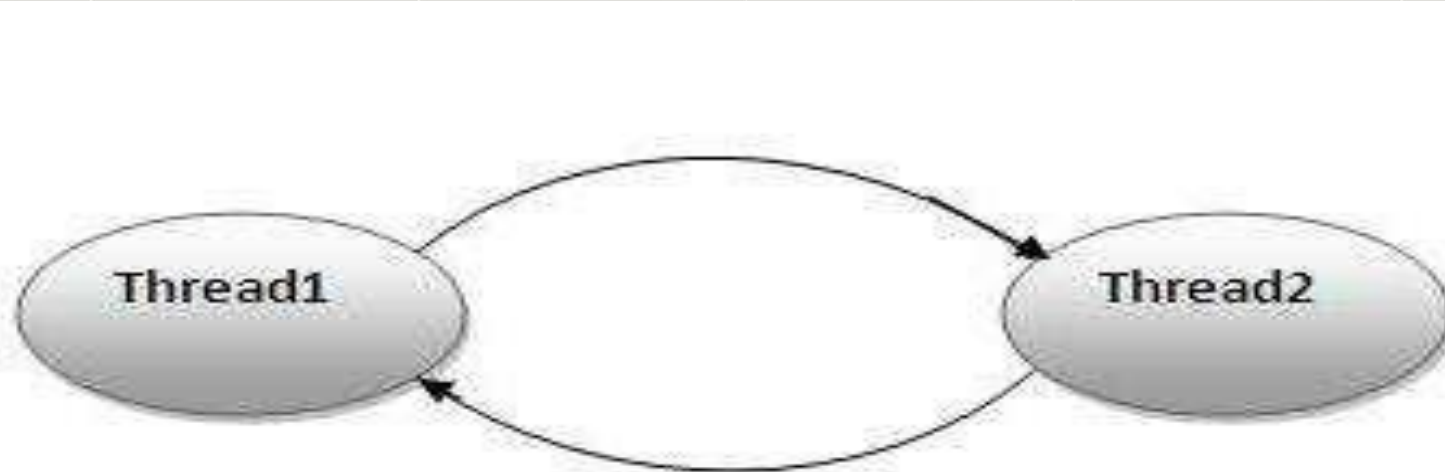
- **Problem** : race conditions
- **Solution** : give exclusive access to one thread at a time to code that manipulates a shared object.
- Synchronization keeps other threads waiting until the object is available.
- The synchronized keyword synchronizes the method so that only one thread can access the method at a time.

```
public synchronized void xMethod() {  
    // method body  
}
```



# Deadlock :

- ❑ a part of multithreading
- ❑ can occur when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread
- ❑ Since, both threads are waiting for each other to release the lock, the condition is called deadlock



# Preventing Deadlock

- ❑ Deadlock can be easily avoided by resource ordering.
- ❑ With this technique, assign an order on all the objects whose locks must be acquired and ensure that the locks are acquired in that order.
- ❑ Example:

Thread 1:

lock A

lock B

Thread 2:

wait for A

lock C (when A is locked)

Thread 3:


wait for A

wait for B

wait for C



# Advantages of Threads:

- easier to program
  - provide better performance
  - allow any program to perform multiple tasks at once.
  - multiple threads can share resources
  - an Internet-aware language such as Java, this is a very important tool
- 



# References

- ❑ <http://www.slideshare.net/parag/multithreading-in-java>
- ❑ <https://code.google.com/p/googleappengine/wiki/SdkForJavaReleaseNotes>
- ❑ <http://stackoverflow.com/questions/2213340/what-is-daemon-thread-in-java>
- ❑ <https://github.com/orientechnologies/orientdb/wiki/Java-Multi-Threading>
- ❑ <http://www.javatpoint.com/creating-thread>
- ❑ [http://www.tutorialspoint.com/java/java\\_multithreading.htm](http://www.tutorialspoint.com/java/java_multithreading.htm)
- ❑ <http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html>
- ❑ <http://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html>
- ❑ <http://www.javabeginner.com/learn-java/java-threads-tutorial>
- ❑ <http://www.geeksforgeeks.org/java/>
- ❑ <http://www.javacodegeeks.com/2014/08/java-concurrency-tutorial-visibility-between-threads.html>



**Thank You**