VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

Introduction to Multithreading

Objectives

This session will give the knowledge about

- What is a multithreading
- What is Multitasking?
- Uses of Multithreading
- Java’s multithreading Model
- Thread Class
- Different states of threads

Need for Multithreading

- Have you faced the following situations:
- Your browser cannot skip to the next web page because it is downloading a file?
- You cannot enter text into your current document until your word processor completes the task of saving the document to disk

What is Multitasking?

- Multitasking is synonymous with process-based multitasking, whereas multithreading is synonymous with thread-based multitasking
- All modern operating systems support multitasking
- A process is an executing instance of a program
- Process-based multitasking is the feature by which the operating system runs two or more programs concurrently

Example: Multitasking

- You might have come across people doing multiple things at the same time: A person talking on the phone while having lunch or watching TV
- In a computer, you can run multiple programs at the same time. Like you can play a song using Media Player while typing a Word document

What is Multithreading?

- In multithreading, the thread is the smallest unit of code that can be dispatched by the thread scheduler
- A single program can perform two tasks using two threads
- Only one thread will be executing at any given point of time given a single-processor architecture

Single-Threaded Systems

Single-threaded systems use an approach called an event loop with polling. In this model:

- A single thread of control runs in an infinite loop
- Polling a single event queue to decide which instruction to execute next
- Until this instruction returns, nothing else can happen in the system
- This results in wastage of precious CPU cycles

Examples: Multithread

- Computer games are best examples of multithreading
- You might have seen that in most of the ‘race’ games, other cars or bikes will be competing with your car/bike. These are nothing but threads.

Examples: Multithread

- Imagine that you need to paint your house. You can employ one painter who will take 10 days to complete the work or you can employ 10 painters who will finish the work in one day.
- In this case the 10 painters will be painting at the same time. That is 10 threads will be executing at the same time.

One thread

Multiple threads

Examples: Multithread

- When you start typing in a MS Word document, you can see that the incorrectly spelled words are underlined with a red wavy line. Like:

- This is done by spell check feature of MS Word. The spell check is running in parallel while we type in a Word document. This is handled by a separate thread.

Multitasking Vs. Multithreading

Compared to multithreading, multitasking is characterized by the following:

- Each process requires its own separate address space
- Context switching from one process to another is a CPU- intensive task needing more time
- Inter-process communication between processes is again expensive as the communication mechanism has to span separate address spaces

These are the reasons why processes are referred to as heavyweight tasks

Multitasking Vs. Multithreading

Threads cost less in terms of processor overhead because of the following reasons:

- Multiple threads in a program share the same address space and they are part of the same process
- Switching from one thread to another is less CPU- intensive
- Inter-thread communication, on the other hand, is less expensive as threads in a program communicate within the same address space

Threads are therefore called lightweight processes

Uses of Multithreading

- A multithreaded application performs two or more activities concurrently
- It is accomplished by having each activity performed by a separate thread
- Threads are the lightest tasks within a program, and they share memory space and resources with each other

VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

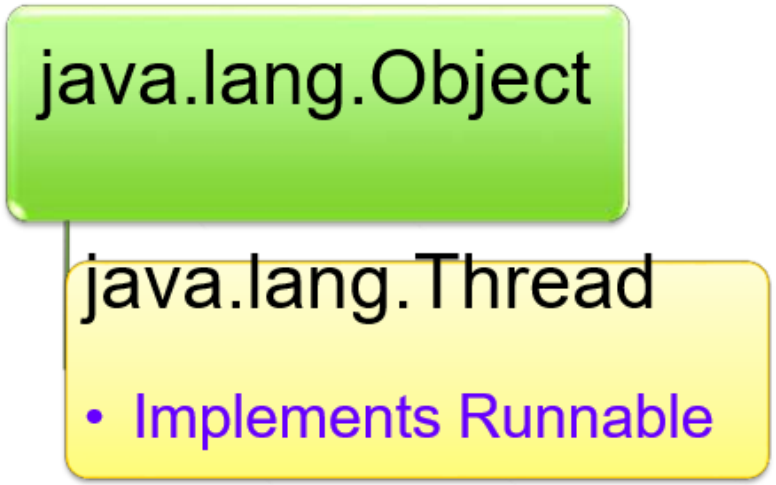
Java's Multithreading Model

Java's Multithreading Model

- Java has completely done away with the event loop/polling mechanism (Event loop/polling means - Executing one process after another which results in CPU time wastage)
- In Java
  - All the libraries and classes are designed with multithreading in mind.
  - This enables the entire system to be asynchronous
- In Java the java.lang.Thread class is used to create thread-based code, imported into all Java applications by default

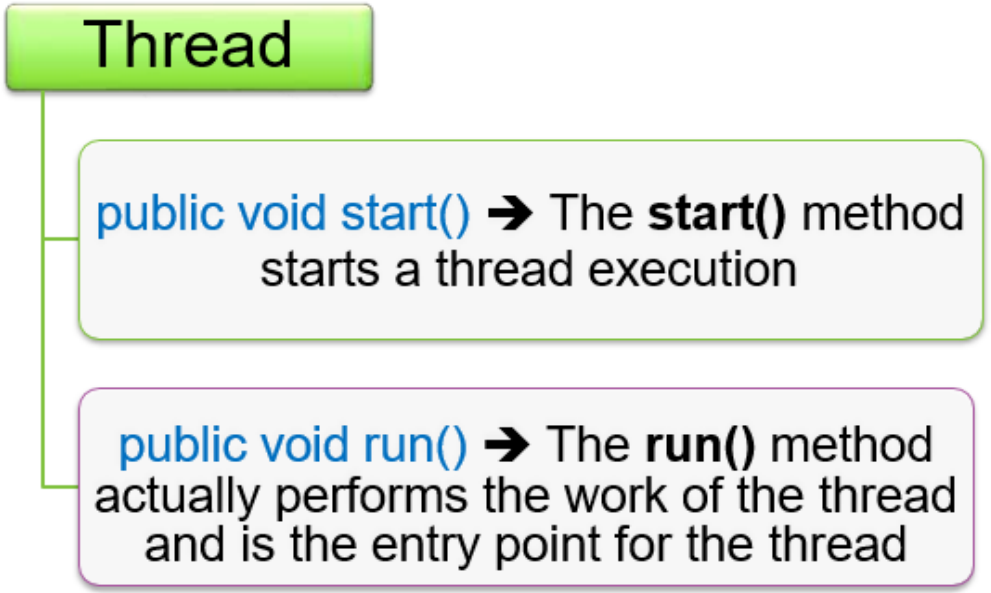
The Thread class

- Java's multithreading feature is built into the Thread class
- Thread Class Hierarchy:

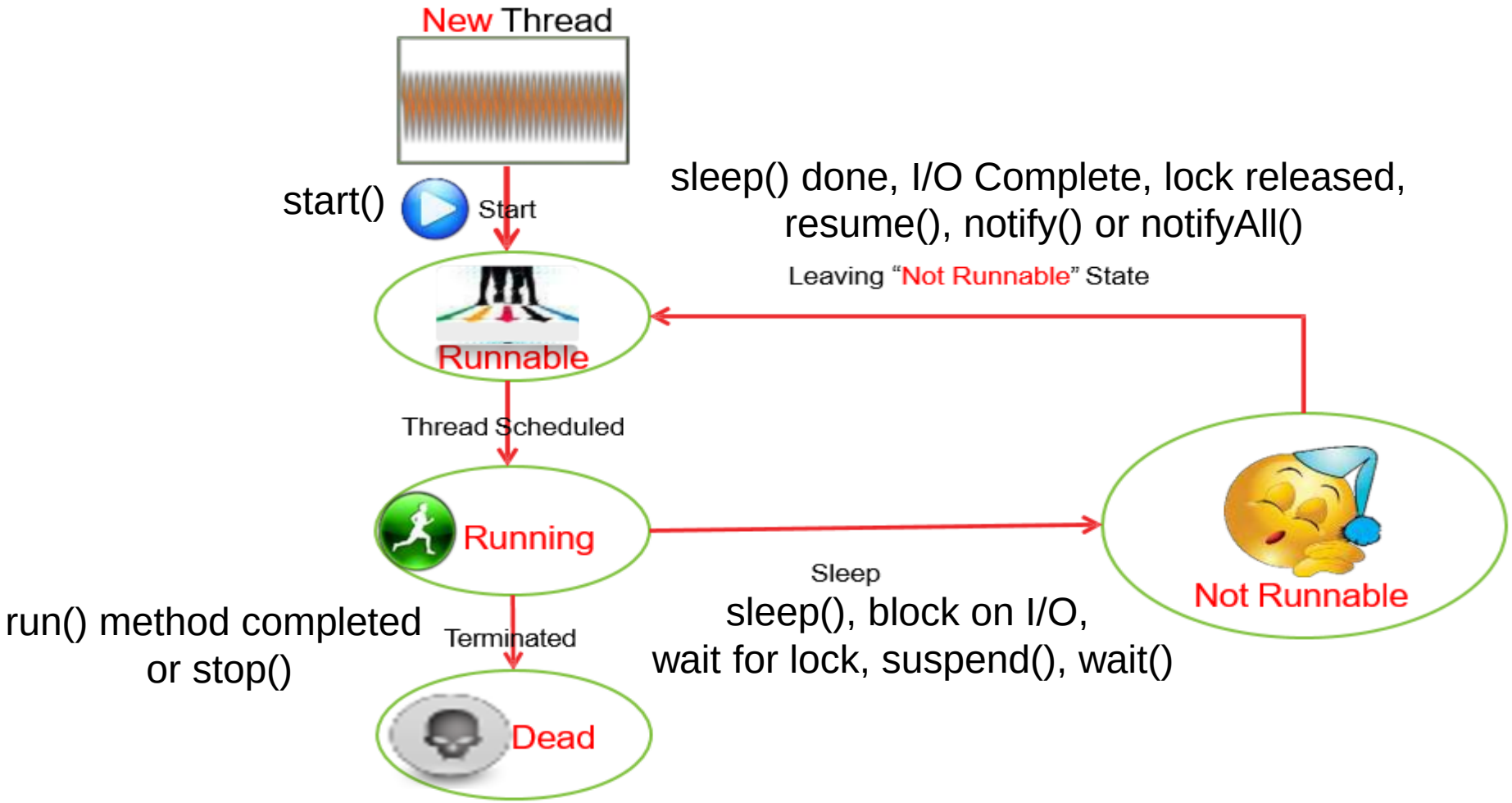


The Thread class

- The **Thread** class has two primary thread control methods:
- The thread *dies* when the **run( )** method terminates
- You never call **run( )** explicitly
- The **start( )** method called on a thread automatically initiates a call to the thread's **run( )** method



Different States of a Thread



Life Cycle of a Thread

Following are the stages of the life cycle −

- New − A new thread begins its life cycle in the new state. It remains in this state until the program starts the 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.

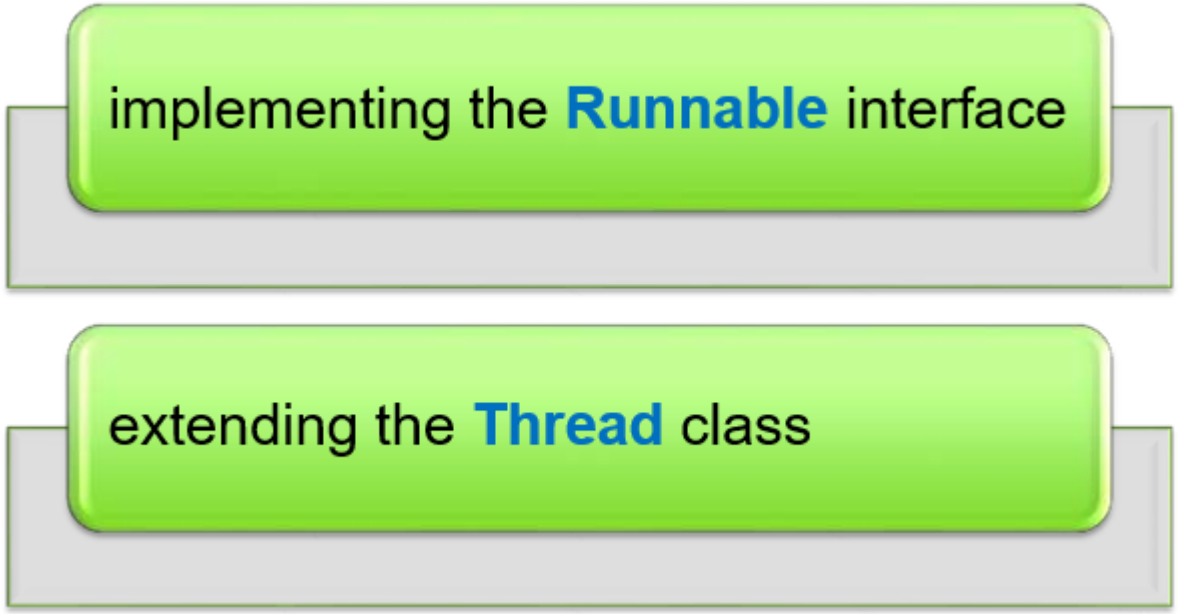
Life Cycle of a Thread

- 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.

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

Creating Threads

- A thread can be created by instantiating an object of type Thread.
- This can be achieved in any of the following two ways :



Summary

We have discussed about

- What is a multithreading
- What is Multitasking?
- Uses of Multithreading
- Java’s multithreading Model
- Thread Class
- Different states of threads