Multithreading

Java’s Multithreading Model

Agenda

1

Introduction to Java's Multithreading Model

2

3

4

Objectives

At the end of this module, you will be able to:

- Understand Java’s Multithreading model

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)*

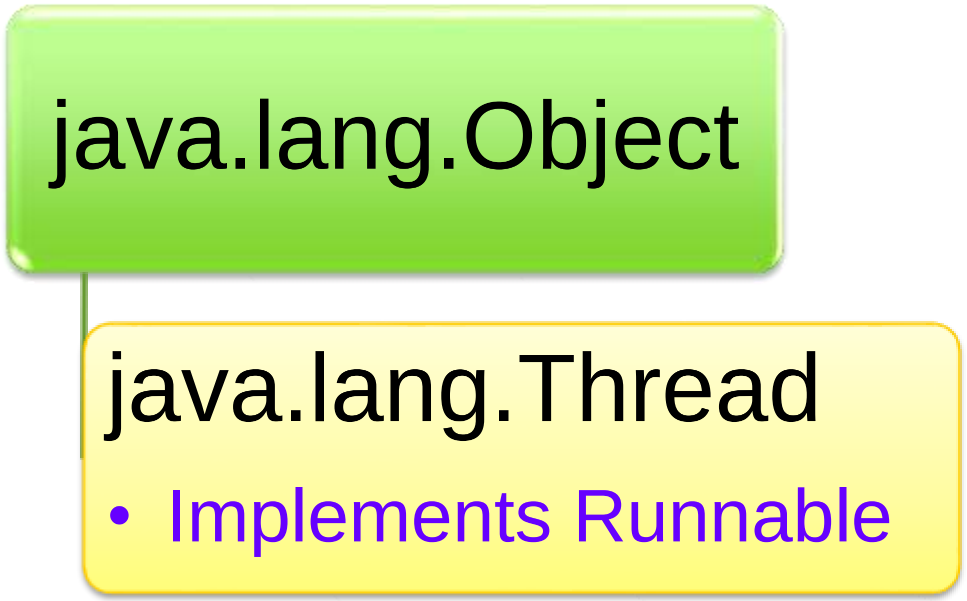
- *In Java*
  - *All the libraries and classes are designed with multithreading in mind*
  - *This enables the entire system to be asynchronous*

- 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

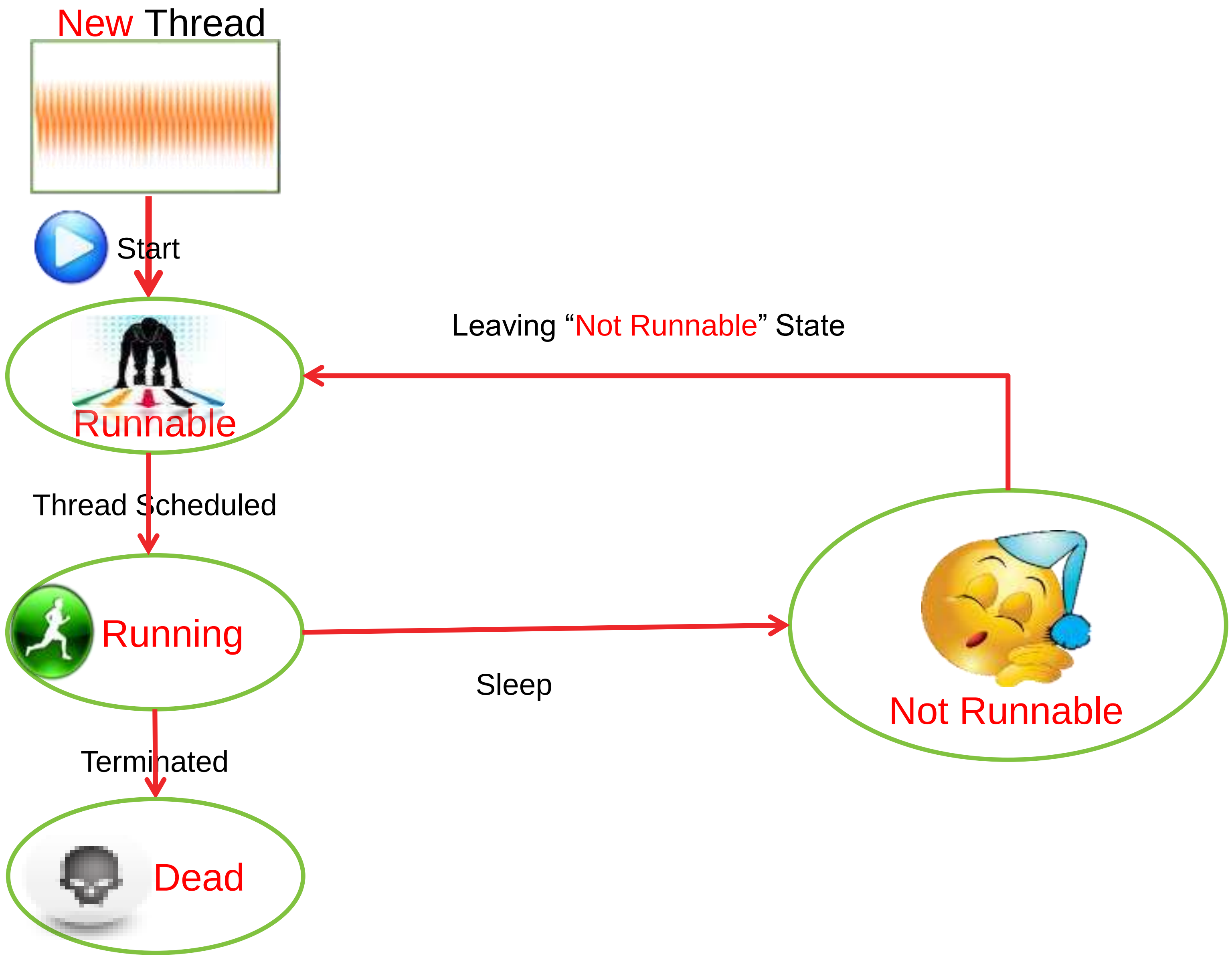
- The **Thread** class has two primary thread control methods:

Thread

- public void start() ➔ The **start()** method starts a thread execution
- public void run() ➔ The **run()** method actually performs the work of the thread and is the entry point for the thread

- 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



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 :

implementing the **Runnable** interface

extending the **Thread** class

- We will discuss both the ways in the next sections

Summary

- Java’s multithreading Model
- Thread Class
- Different states of threads
- Creating Threads

Thank You