Experiment No. 14

Title: Write a Java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

Objectives:

1) To learn how to create Thread.
2) To learn multithreading

Theory:

Java is a *multithreaded programming language* which means we can develop multithreaded program using Java. A multithreaded 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 especially when your computer has multiple CPUs.
By definition multitasking is when multiple processes share common processing resources such as a CPU. Multithreading 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.
Multithreading enables you to write in a way where multiple activities can proceed concurrently in the same program.

Life Cycle of a Thread:
A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. Following diagram shows complete life cycle of a thread.

Above-mentioned stages are explained here

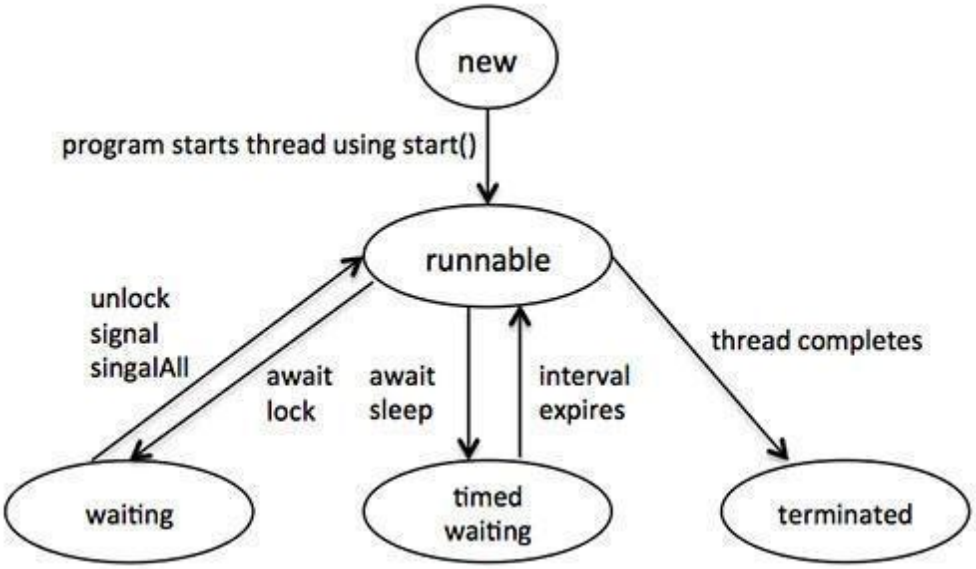
- **New:** A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.

- **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 t **exp**ires or when the event it is waiting for occurs.

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



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

Thread Priorities:

Every Java thread has a priority that helps the operating system determine the order in which threads are scheduled.
Java thread priorities are in the range between MIN_PRIORITY (a constant of 1) and MAX_PRIORITY (a constant of 10). By default, every thread is given priority NORM_PRIORITY (a constant of 5).
Threads with higher priority are more important to a program and should be allocated processor time before lower-priority threads. However, thread priorities cannot guarantee the order in which threads execute and very much platform dependent.

Step 2:
At second step you will instantiate a **Thread** object using the following constructor:
Thread(Runnable threadObj, String threadName);

Where, threadObj is an instance of a class that implements the Runnable interface and threadName is the name given to the new thread.

Step 3
Once Thread object is created, you can start it by calling start( ) method, which executes a call to run( ) method. Following is simple syntax of start() method:

void start( );

Create Thread by Extending Thread Class:
The second way to create a thread is to create a new class that extends Thread class using the following two simple steps. This approach provides more flexibility in handling multiple threads created using available methods in Thread class.

Step 1
You will need to override run( ) method available in Thread class. This method provides entry point for the thread and you will put you complete business logic inside this method. Following is simple syntax of run() method:

public void run( )

Step 2
Once Thread object is created, you can start it by calling start( ) method, which executes a call to run( ) method. Following is simple syntax of start() method:

void start( );

Keywords: Thread, Runnable,

Algorithm:

1) Create a class RandomNumberThread that extends Thread class.
2) In RandomNumberThread class write run() method which generates Random Integer number.
3) Create class SquareThread that extends Thread class
4) In SquareThread class write run() method that computes the square of the number.
5) Create class CubeThread that extends Thread class
6) In CubeThread class write run() method that computes the cube of the number
7) Run the thread using start() method of Thread class to show square and cube operations.
8) Using the sleep() method assign the time to generate random Integer number.
9) Display the output.