

St Francis Institute of Technology, Mumbai-400 103

Class: SE-ITA/ITB Semester: III; A.Y. 2020-2021

Subject: Java Labs

Title-7: Java Program to implement Multithreading and Thread Synchronization.

1. Aim:

- i. Create two threads such that one thread will print even number and another will print odd number in an ordered fashion.(Use Thread Class)
- ii. Write java program to print Table of Five, Seven and Thirteen using Multithreading(Use Runnable Interface)

2. Prerequisite: Knowledge of Multithreading in Java.

3. Requirements: Personal Computer (PC), Windows Operating System, Net beans 8.0.

4. Pre-Experiment Exercise:

Theory:

a. Multithreading:

Java's multithreading support is centered around the `java.lang.Thread` class. The `Thread` class provides the capability to create objects of class `Thread`, each with its own separate flow of control. The `Thread` class encapsulates the data and methods associated with separate threads of execution and allows multithreading to be integrated within the object-oriented framework. Java provides two approaches to creating threads. In the first approach, you create a subclass of class `Thread` and override the `run()` method to provide an entry point into the thread's execution. When you create an instance of your `Thread` subclass, you invoke its `start()` method to cause the thread to execute as an independent sequence of instructions. The `start()` method is inherited from the `Thread` class. It initializes the `Thread` object using your operating system's multithreading capabilities and invokes the `run()` method.

Java's other approach to creating threads does not limit the location of your `Thread` objects within the class hierarchy. In this approach, your class implements the `java.lang.Runnable` interface. The `Runnable` interface consists of a single method, the `run()` method, which must be overridden by your class. The `run()` method provides an entry point into your thread's execution. In order to run an object of your class as an independent thread, you pass it as an argument to a constructor of class `Thread`.

b. Thread Synchronization

There are times when you might want to coordinate access to shared resources. For example, in a database system, you might not want one thread to be updating a database record while another thread is trying to read it. Java enables you to coordinate the actions of multiple threads using *synchronized methods* and *synchronized statements*.

An object for which access is to be coordinated is accessed through the use of synchronized methods. These methods are declared with the synchronized keyword. Only one synchronized method can be invoked for an object at a given point in time. This keeps synchronized methods in multiple threads from conflicting with each other.

All classes and objects are associated with a unique *monitor*. The monitor is used to control the way in which synchronized methods are allowed to access the class or object. When a synchronized method is invoked for a given object, it is said to *acquire* the monitor for that object. No other synchronized method may be invoked for that object until the monitor is released. A monitor is automatically released when the method completes its execution and returns. A monitor may also be released when a synchronized method executes certain methods, such as `wait()`. The thread associated with the currently executing synchronized method becomes not runnable until the wait condition is satisfied and no other method has acquired the object's monitor.

5. Laboratory Exercise

A. Procedure

- i. Open Net beans for Java.
- ii. Open File and Create New Java Project.
- iii. Inside the Java Project rename give name to your Java Class.
- iv. Click on Finish.
- v. Type the Java Code in the opened class.
- vi. Save the code by pressing Ctrl+S.
- vii. Run the code by pressing Shift+F6.

B. Program code with comments:

Write and execute your program code to achieve the given aim and attach it **with your own comments with neat indentation**.

6. Post-Experiments Exercise

A. Extended Theory:

1. Explain deadlock situation and how can it be eliminated.
2. Explain with diagram Java Thread Model and Thread Lifecycle Model.

B. Results/Observations/Program output:

Present the program input/output results and comment on the same.

C. Questions/Programs:

1. Write java program to implement the concept of Thread Synchronization.

D. Conclusion:

1. Write what was performed in the experiment/program.
2. What is the significance of experiment/program?
3. Mention few applications of what was studied.

E. References

1. Balguruswamy, “Programming with java A primer”, Fifth edition, Tata McGraw Hill Publication.
2. Let Us Java-Yashwant Kanetkar.
3. Learn to Master JAVA, from Star EDU solutions , by ScriptDemics.
4. Java 8 Programming-Black Book,by-Dreamtech Publications.
5. www.programmingsimplified.com
6. www.javatpoint.com
