Java – Learn From Home

Assignment – Chapter 5

Concept: Multi-threading

Objective: At the end of the assignment, participants will be able to:

- Implement multi-tasking using threads
- Create and manage threads
- Synchronize threads

Problems:

Exercise 1:

Write a program to implement Runnable interface and override the run method.

Guided Solution:

Step 1: Create a class name RunnableThread and implement the Runnable Interface.
Step 2: Override the run() method inside the RunnableThread class
Step 3: Inside the run() method use System.out.println("Run Method Overriden")
Step 4: Write another class name RunnableDemo
Step 5: Write main() method inside the RunnableDemo class.
Step 6: Create an instance of RunnableThread class.
Step 7: Create a Thread Object and pass the RunnableThread instance into the Thread Object.
Step 8: Start the Thread using start() method.

Exercise 2: Write a program to print "Hello Thread!..." using Thread implementation.

Guided Solution:

Step 1: Create a class name HelloThread and extend the Thread class
Step 2: Override the run() method inside the HelloThread class
Step 3: Inside the run() method use System.out.println("Hello Thread")
Step 4: Write another class name ThreadDemo
Step 5: Write main() method inside the ThreadDemo class.
Step 6: Create an instance of HelloThread class.
Step 7: Create a Thread Object and pass the HelloThread instance into the Thread Object.
Step 8: Start the Thread using start() method.
Step 9: Compile the class using "javac HelloThread.java" and Execute as "java HelloThread".

Exercise 3: Write a Java application to simply lists the numbers 1 to 10 using Thread Code.

Problem Statement:

The application will have two thread instances; the thread will append the thread instance name to the output number so that we can distinguish between the instances. Finally, we will use the sleep method within the thread's run method to cease execution every half second so that we can interleave the two thread instances' output

Guided Solution:

Step 1: Create a class named PrintNumbersThread and extend the Thread class.

E.g: public class PrintNumbersThread extends Thread {}

Step 2: declare an instance variable named name of type String

String name;

Step 3: Create an argument contructor to get the Thread Names.

E.g:

```
public PrintNumbersThread(String threadName){
name = threadName;
}
```

Step 4: Write down the run method implementing from the Thread class.

E.g:

```
public void run() {
}
```

Step 5: Inside the **run()** method declare a local variable **i** of type int.

Step 6: write a for loop that runs for **10 times**.

E.g. : for (i=1; i<11; i++)

Step 7: Inside the for loop print the name of the thread.

System.out.println(name + ": " + i);

Step 8: Invoke a sleep() method after printing the thread names inside the for loop and surround the sleep() method using try and catch by catching the InterruptedException

E.g:

```
try {
 Thread.sleep(500);
} catch (InterruptedException e) {}
```

Step 9: Write another class named RunThreads.

Step 10: Declare a main method inside the class.

Step 11: Create two instance of PrintThreadNumber class by passing the String value to its constructor.

```
PrintNumbersThread  thread1 = new PrintNumbersThread("Thread1");
PrintNumbersThread  thread2 = new PrintNumbersThread("Thread2");
```

Step 12: Start both thread.

```
thread1.start();
thread2.start();
```

Step 13: Compile the class using "javac RunThreads.java" and Execute as "java RunThreads"

Exercise 4: Write down Two Game thread and define the Priority for both the thread as [Thread1: low and Thread2: High] and observe their Output.

Note: Using runnable interface.

Guided Solution:

Step 1: Create a class named GameThread and extend the Thread class.

```
E.g.: public class GameThread extends Thread {}
```

Step 2: declare an instance variable named threadName of type String

```
String threadName;
```

Step 3: Create an argument contructor to get the ThreadNames.

```
E.g.:
public GameThread(String threadName){
this.name = threadName;
}
```

Step 4: Write down the run method implementing from the Thread class.

```
E.g:
public void run() {
}
```

Step 5: Inside the run() method print the name of the thread.

Step 6: Write another class named PriorityDemo.

Step 7: Declare a main method inside the class.

Step 8: Create two instance of GameThread class by passing the String value to its constructor.

```
GameThread g1 = new GameThread("Thread1");
GameThread g2 = new GameThread("Thread2");
```

Step 9: Create two Thread Objects and pass the GameThread object into its constructor argument.

```
Thread t1 = new Thread(g1);
Thread t2 = new Thread(g2);
```

Step 10: Call setPriority() method on t1 and t2 tread Object by passing the MIN and MAX priority correspondingly.

```
t1.setPriority(Thread.MIN_PRIORITY);
t2.setPriority(Thread.MAX_PRIORITY);
```

Step 11: Start both the thread.

```
t1.start();
t2.start();
```

Step 12: Compile the class using "javac PriorityDemo.java" and Execute as "java PriorityDemo".

Exercise 5: Write a thread program where three different threads are started simultaneously and ensuring only one thread

is allowed to run and other threads have to wait to complete for the first thread.

Note: Use the concept of Synchronization.

Guided Solution:

Step 1: Create a class named ThreadSync

Step 2: Let the ThreadSynch class implements the Runnable interface.

E.g.: class ThreadSync implements Runnable.

Step 3: Override the run() method inside the ThreadSync class

Step 4: Ensure the run() method is applied with synchronized keyword.

E.g: public synchronized void run(){}

Step 5: Inside the run() method print the name of the thread using Thread.currentThread().getName() method.

```
System.out.print(Thread.currentThread().getName());
```

Step 6: After printing the name of the thread call the sleep() method with 1 millisecond duration.

```
try {
Thread.sleep(1000);
}catch(Exception e){
}
```

Step 7: Write the main() method inside this ThreadSync class

Step 8: Create the Object ThreadSynch inside the main method.

```
ThreadSync r = new ThreadSync();
```

Step 9: Create three thread object and pass the ThreadSync class instance to all the Thread instance constructor argument.

```
Thread t1=new Thread(r);
Thread t2=new Thread(r);
Thread t3=new Thread(r);
```

Step 10: Start all the Three thread simultaneously.

```
t1.start();t2.start();t3.start();
```

Step 11: Compile the class using "javac ThreadSync.java" and Execute as "java ThreadSync"

Step 12: Observer the O/P.