Batch No.__110___	**Roll No. __B2__**
Experiment / Assignment/ Tutorial No. 8	
Grade : AA/AB/BB/BC/CC/CD/DD	
Signature of the Staff In-charge with date	

TITLE : Multithreading Programming

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

Expected OUTCOME of Experiment:

CO1: Understand the features of object oriented programming compared with procedural approach with C++ and Java
CO4: Explore the interface, exceptions, multithreading, packages.

Books/ Journals/ Websites referred:

1. Ralph Bravaco , Shai Simoson , “Java Programing From the Group Up” Tata McGraw-Hill.

2.Grady Booch, Object Oriented Analysis and Design .

Pre Lab/ Prior Concepts:
Java provides built-in support for multithreaded programming. A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each

thread defines a separate path of execution. A multithreading is a specialized form of multitasking. Multithreading requires less overhead than multitasking processing.
Multithreading enables you to write very efficient programs that make maximum use of the CPU, because idle time can be kept to a minimum.

Creating a Thread:
Java defines two ways in which this can be accomplished:

1. You can implement the Runnable interface.
2. You can extend the Thread class itself.

Create Thread by Implementing Runnable:
The easiest way to create a thread is to create a class that implements the Runnable interface.
To implement Runnable, a class needs to only implement a single method called run( ), which is declared like this:

public void run( )

You will define the code that constitutes the new thread inside run() method. It is important to understand that run() can call other methods, use other classes, and declare variables, just like the main thread can.

After you create a class that implements Runnable, you will instantiate an object of type Thread from within that class. Thread defines several constructors. The one that we will use is shown here:

Thread(Runnable threadOb, String threadName);

Here, threadOb is an instance of a class that implements the Runnable interface and the name of the new thread is specified by threadName.
After the new thread is created, it will not start running until you call its start( ) method, which is declared within Thread. The start( ) method is shown here:

void start( );

Here is an example that creates a new thread and starts it running:

```
class NewThread implements Runnable {
  Thread t;
  NewThread() {
    t = new Thread(this, "Demo Thread");
    System.out.println("Child thread: " + t);
    t.start(); // Start the thread
  }
    public void run() {
    try {
      for(int i = 5; i > 0; i--) {
        System.out.println("Child Thread: " + i);
        // Let the thread sleep for a while.
        Thread.sleep(50);
      }
   } catch (InterruptedException e) {
      System.out.println("Child interrupted.");
```

```
    }
    System.out.println("Exiting child thread.");
  }
}

public class ThreadDemo {
  public static void main(String args[]) {
    new NewThread();
    try {
      for(int i = 5; i > 0; i--) {
        System.out.println("Main Thread: " + i);
        Thread.sleep(100);
      }
    } catch (InterruptedException e) {
      System.out.println("Main thread interrupted.");
    }
    System.out.println("Main thread exiting.");
  }
}
```

The second way to create a thread is to create a new class that extends Thread, and then to create an instance of that class.
The extending class must override the run( ) method, which is the entry point for the new thread. It must also call start( ) to begin execution of the new thread.

```
class NewThread extends Thread {
  NewThread() {
    super("Demo Thread");
    System.out.println("Child thread: " + this);
    start(); // Start the thread
  }
   public void run() {
    try {
      for(int i = 5; i > 0; i--) {
        System.out.println("Child Thread: " + i);
                // Let the thread sleep for a while.
        Thread.sleep(50);
      }
    } catch (InterruptedException e) {
      System.out.println("Child interrupted.");
    }
    System.out.println("Exiting child thread.");
  }
}

public class ExtendThread {
  public static void main(String args[]) {
```

```
    new NewThread(); // create a new thread
    try {
      for(int i = 5; i > 0; i--) {
        System.out.println("Main Thread: " + i);
        Thread.sleep(100);
      }
    } catch (InterruptedException e) {
      System.out.println("Main thread interrupted.");
    }
    System.out.println("Main thread exiting.");
  }
}
```

Some of the Thread methods

Methods	Description
void setName(String name)	Changes the name of the Thread object. There is also a getName() method for retrieving the name
Void setPriority(int priority)	Sets the priority of this Thread object. The possible values are between 1 and 10. 5
boolean isAlive()	Returns true if the thread is alive, which is any time after the thread has been started but before it runs to completion.
void yield()	Causes the currently running thread to yield to any other threads of the same priority that are waiting to be scheduled.
void sleep(long millisec)	Causes the currently running thread to block for at least the specified number of milliseconds.
Thread currentThread()	Returns a reference to the currently running thread, which is the thread that invokes this method.

Explaination

Here we are using synchronized keyword for the function . if the synchronized keyword is not used, then all the three threads may generate randome numbers. But since the synchronized keyword is used, all threads must follow the order od execution and therefore we see that number is first squared and then cubed.

Implementation details:

```
import java.util.Random;

public class program implements Runnable{
    Thread t1;
    Thread t2;
    Thread t3;
     int number;
    public  program(){
        t1 = new Thread(this);

        t1.start();

        t2 = new Thread(this);
        t2.start();

        t3 = new Thread(this);
        t3.start();
    }

    @Override
    synchronized public void run() {
        if(number==0){
            //first thread runs
        Random myRandom = new Random();
         number = myRandom.nextInt(50);
        System.out.println(number);
        }
        else if(number%2==1){
            number = number*number;
            System.out.println(number);
        }
        else if(number%2==0){
            number = number*number*number;
            System.out.println(number);
        }

    }
}
```

```java
public class Main{

    public static void main(String args[]){
        System.out.println("Hello world");
        program myProgram = new program();
        //myProgram.go();
    }
}
```

Output:

```
[Running] cd "c:\Users\student\Documents\Multithreading\" && javac Main.java && java Main

Hello world
7
49
2401
```

Conclusion:

Thus we understood the working of java multithreading and implemented the multithreading in java. Multithreading is a way by with we can save the time and resources required. But this approach may generate a few problems related to synchronization which can be solved by using the synchronized keyword.

Date 2 Ddec 22 **Signature of faculty in-charge**

Post Lab Descriptive Questions

1. What do you mean by multithreading?

Multithreading is **the ability of a program or an operating system to enable more than one user at a time without requiring multiple copies of the program running on the computer**. Multithreading can also handle multiple requests from the same user.

https://www.techtarget.com/whatis/definition/multithreading#:~:text=Multithreading%20is%20the%20ability%20of,requests%20from%20the%20same%20user.

2. Explain the use of sleep and run function with an example?

The Java Thread. sleep() method can be used to pause the execution of the current thread for a specified time in milliseconds. The argument value for milliseconds cannot be negative. Otherwise, it throws IllegalArgumentException .

Example of these function can be found in the prelab code and implementations.