Batch: A3 **Roll No.: 16010121045**

Experiment / assignment / tutorial No.08

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 Programming 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.");
  }
}
```

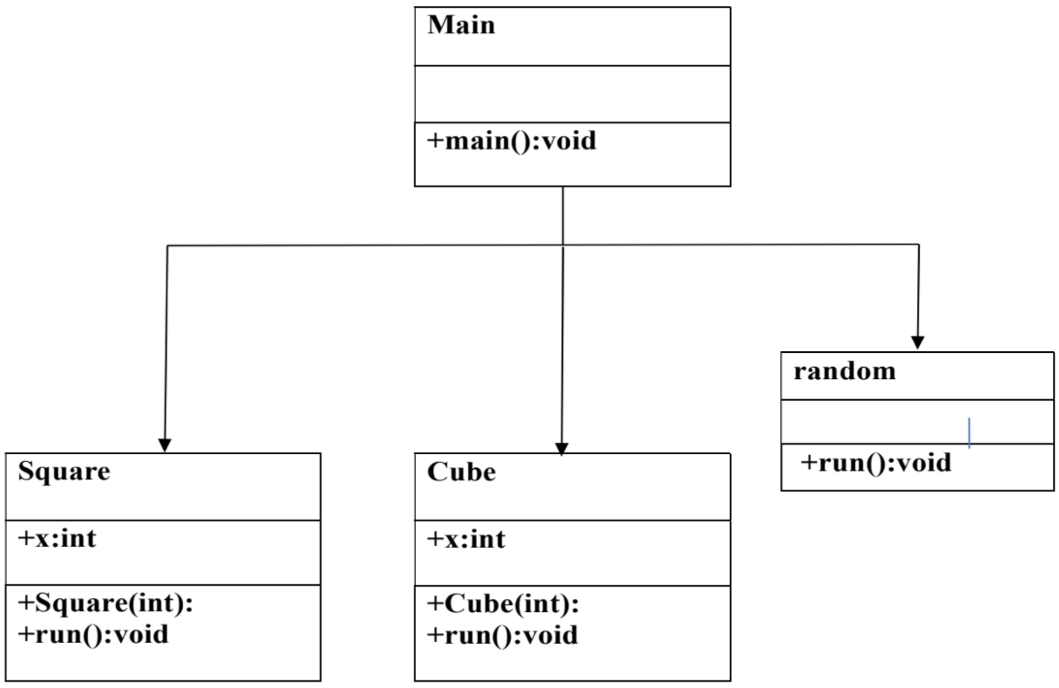
The second way to create a thread is to create a new class that extends Thread, and then to create an instance of that class.
The extending class must override the run( ) method, which is the entry point for the new thread. It must also call start( ) to begin execution of the new thread.

```
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.

Class Diagram:



Algorithm:

Step 1: Create class random and extend Thread to it

Step 2: Override the function run

i. In function run generate a random number by using random package

ii. Create objects of thread Square and thread cube

iii. Check if the number generated is even, if so , then start thread Square else start thread Cube

Step 3: In class Square, create a parametrized constructor to initialize the class variables

Step 4: Override the function run , which prints the square of the number.

Step 5: In class Cube, create a parametrized constructor to initialize the class variables

Step 6: Override the function run , which prints the cube of the number .

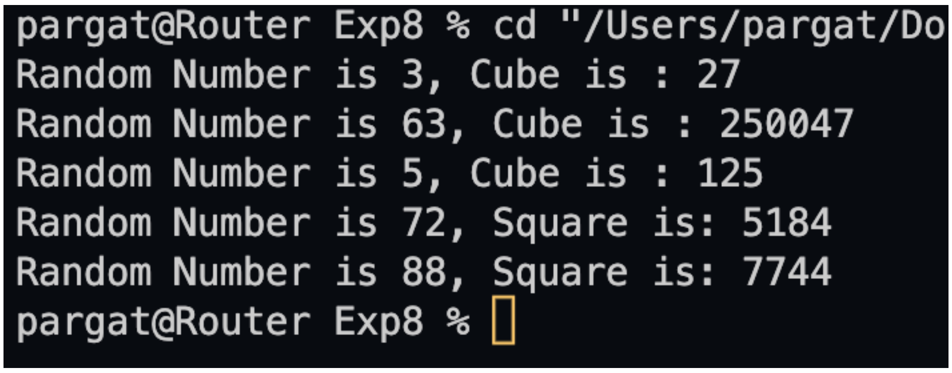
Step 7: Create a class Main , with a main function.

Step 8: The main function is used to create an object of class Random .This object is then used to start the Random thread.

Implementation details:

```
import java.util.Random;

class random extends Thread {
    public void run() {
        Random r = new Random();
        for (int i = 0; i < 5; i++) {
            int x = r.nextInt(100);
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
            } finally {
                Square obj2 = new Square(x);
                Cube obj3 = new Cube(x);
                if (x % 2 == 0)
                    obj2.start();
                else
                    obj3.start();
            }
        }
    }
}

class Square extends Thread {
    public int x;

    public Square(int x) {
        this.x = x;
    }

public void run()
{
System.out.println("Random Number is "+x+", Square is:
"+x*x);
}
}

class Cube extends Thread {
    int x;

    Cube(int x) {
        this.x = x;
    }

public void run()
{
System.out.println("Random Number is "+x+", Cube is :
"+x*x*x);
}
}

public class exp8 {
    public static void main(String args[]) {
        random obj1 = new random();
        obj1.start();
    }
}
```

Output:



```
pargat@Router Exp8 % cd "/Users/pargat/Do
Random Number is 3, Cube is : 27
Random Number is 63, Cube is : 250047
Random Number is 5, Cube is : 125
Random Number is 72, Square is: 5184
Random Number is 88, Square is: 7744
pargat@Router Exp8 % 
```

Conclusion:

Thus, multithreading was understood and implemented.

Date:________   **Signature of faculty in-charge**

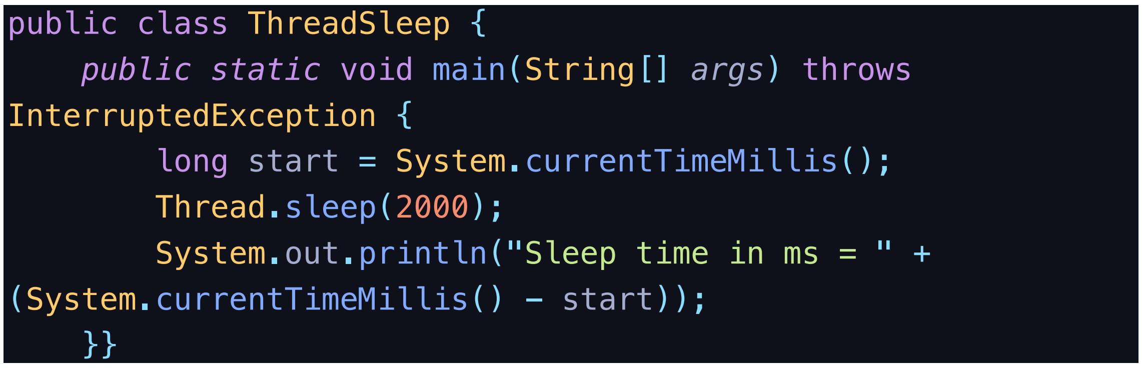
Post Lab Descriptive Questions

1. What do you mean by multithreading?

Multithreading in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU. Multithreaded applications execute two or more threads run concurrently. Hence, it is also known as Concurrency in Java. Each thread runs parallel to each other. Multiple threads don't allocate separate memory area, hence they save memory. Also, context switching between threads takes less time.

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

Thread.sleep() method can be used to pause the execution of current thread for specified time in milliseconds. The argument value for milliseconds can't be negative, else it throws IllegalArgumentException.
Example :



```
public class ThreadSleep {
    public static void main(String[] args) throws
InterruptedException {
        long start = System.currentTimeMillis();
        Thread.sleep(2000);
        System.out.println("Sleep time in ms = " +
(System.currentTimeMillis() - start));
    }}
```

The run() method of thread class is called if the thread was constructed using a separate Runnable object otherwise this method does nothing and returns. When the run() method calls, the code specified in the run() method is executed. You can call the run() method multiple times

```
public class RunExp1 implements Runnable {
    public void run() {
        System.out.println("Thread is running...");
    }

    public static void main(String args[]) {
        RunExp1 r1 = new RunExp1();
        Thread t1 = new Thread(r1);
        // this will call run() method
        t1.start();
    }
}
```

3. Explain any five methods of Thread class with Example ?

activeCount() : Returns an estimate of the number of active threads in the current thread’s thread group and its subgroups.

checkAccess() : Determines if the currently running thread has permission to modify this thread.

Clone(): Throws CloneNotSupportedException as a Thread can not be meaningfully cloned

currentThread() : Returns a reference to the currently executing thread object

dumpStack(): Prints a stack trace of the current thread to the standard error stream

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running created by
extending to parent Thread class");
    }
    public static void main(String[] args) {
        MyThread myThread = new MyThread();
        myThread.start();
    }
}
```