**Batch: B3 Roll No.: 121**

**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 .

3. <https://www.javatpoint.com/multithreading-in-java>

4. <https://www.javatpoint.com/java-thread-run-method>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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.

(daemon is a thread which runs in the background)

**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.");

}

}

//when threads are running paralelly, and there is a racing-like condition. Then, there is competition. It is called deadlock.

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:**

Thread

NumberGenerator

Square

max: int

randomnumber: Random

n: int

random\_int: int

Cube

num: int

num: int

Square(num: int)

+ run() : void

Cube(num : int)

+ run(): void

+ run(): void

1..\*

1..\*

1..\*

1

EXP\_08

+ static main(args[]: String) : void

**Algorithm:**

STEP 1: Start

STEP 2: Define a class Square which inherits from Thread class.

STEP 2.1: Declare num variable of type int.

STEP 2.2: Define a parameterized constructor Square passing num of type integer as parameter. Initialize the instance variable num of the class using this operator.

STEP 2.3: Define a public method run of void return type and display the square the number.

STEP 3: Define a class Cube which inherits from Thread class.

STEP 3.1: Declare num variable of type int.

STEP 3.2: Define a parameterized constructor Cube passing num of type integer as parameter. Initialize the instance variable num of the class using this operator.

STEP 3.3: Define a public method run of void return type and display the cube the number.

STEP 4: Define a class NumberGenerator which inherits from Thread class.

STEP 4.1: Declare max variable of type int.

STEP 4.2: Define a public method run of void return type and display the square the number.

STEP 4.2.1: Declare an object randomnumber of class Random.

STEP 4.2.2: Declare variable n of type integer and read n.

STEP 4.2.3: Read max

STEP 4.2.4: Declare a for loop with iterator i such that initially, i 🡨 0, the loop goes on as long as i < n, and the value of i increments by 1 after each iteration.

Declare an integer variable random\_int

random\_int 🡨 randomnumber.nextInt(max)

Display random\_int

If(random\_int % 2 == 0)

Declare an object sq\_ob of Square class passing

random\_int as parameter.

sq\_ob.start()

else

Declare an object cb\_ob of Cube class passing

random\_int as parameter.

cb\_ob.start()

STEP 4.3: In try method, cause the thread to sleep for 1000 ms.

In catch block, pass a parameter e of type Exception. Display “Exception”.

STEP 5: Define a class EXP\_08

STEP 5.1: Define main function which is public, static and has void return type. Pass args[] variable of type String as parameter.

STEP 5.2: Declare object ob of class NumberGenerator.

STEP 5.3: Start thread (ob.start())

STEP 6: Stop

**Implementation details:**

import java.util.\*;

class Square extends Thread

{

    int num;

    Square(int num)

    {

        this.num = num;

    }

    public void run()

    {

        System.out.println("The square of " + this.num + " is " + (this.num \* this.num) + ".");

    }

}

class Cube extends Thread

{

    int num;

    Cube(int num)

    {

        this.num = num;

    }

    public void run()

    {

        System.out.println("The cube of " + this.num + " is " + (this.num \* this.num \* this.num) + ".");

    }

}

class NumberGenerator extends Thread

{

    int max;

    public void run()

    {

        Random randomnumber = new Random();

        Scanner in = new Scanner(System.in);

        System.out.println("Enter the number of random numbers to generate: ");

        int n = in.nextInt();

        System.out.println("Enter the maximum value which any generated random number must not exceed (which must be less than 1290): ");

        do

        {

            max = in.nextInt();

            if (max > 1290)

            {

                System.out.println("The maximum value of generated integer cannot be more than 1290.");

                System.out.println("The reason being that the cube of such a number cannot be stored in an integer type variable.");

            }

        } while (max > 1290);

        for (int i = 0; i < n; i++)

        {

            int random\_int = randomnumber.nextInt(max);

            System.out.println("The random integer generated is " + random\_int + ".");

            if (random\_int % 2 == 0)

            {

                Square sq\_ob = new Square(random\_int);

                sq\_ob.start();

            } else

            {

                Cube cb\_ob = new Cube(random\_int);

                cb\_ob.start();

            }

            try

            {

                Thread.sleep(1000);

            } catch (Exception e)

            {

                System.out.println("Exception");

            }

        }

    }

}

class EXP\_08

{

    public static void main(String args[])

    {

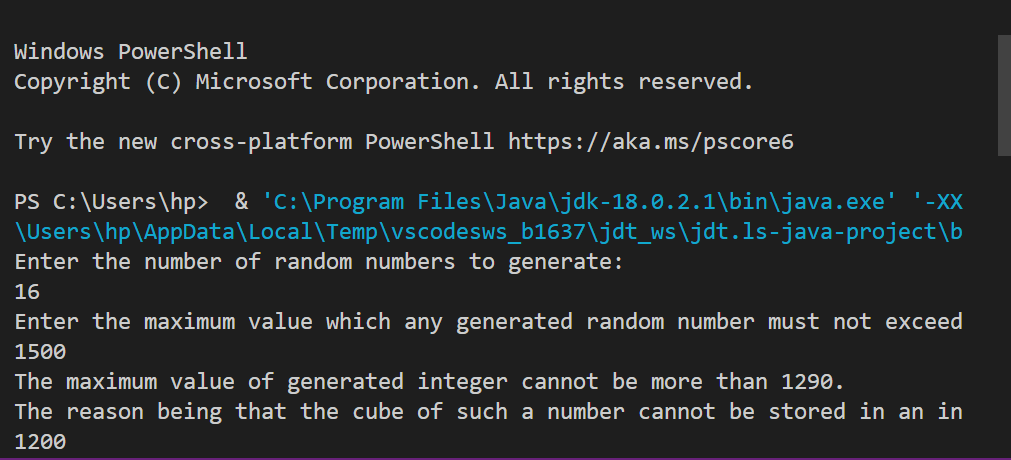
        NumberGenerator ob = new NumberGenerator();

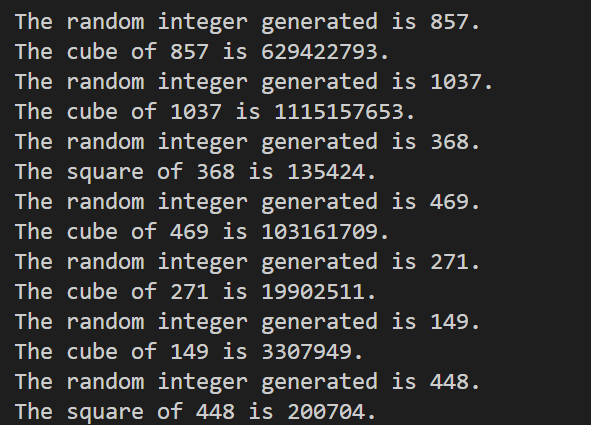
        ob.start();

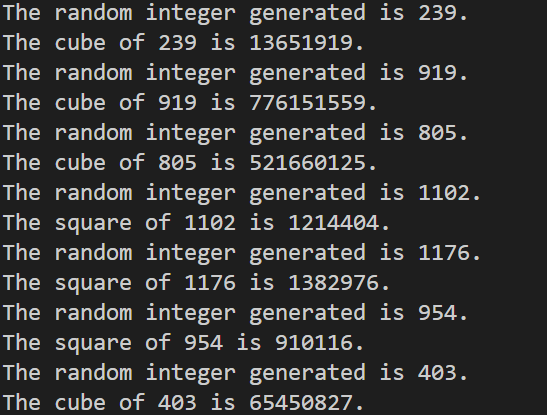
    }

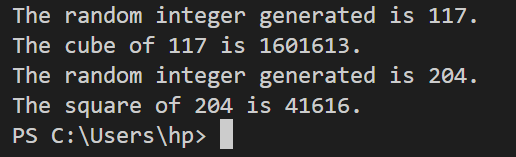
}

**Output:**









**Conclusion:**

Thus, in this experiment, the concept of Multithreading was learnt and implemented. Multithreading enables the programmer to save time as multiple processes can be performed together. While one process is waiting due to I/O operation or some other reason, another process can run. Thus, the idle time is reduced and energy efficiency is increased.

**Date: \_\_07-12-22\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

1. What do you mean by multithreading?

Ans. Multithreading in Java is a process of executing multiple threads simultaneously. A thread is a light-weight sub-process, the smallest unit of processing. Multiprocessing and multithreading both are used to achieve multitasking. However, multithreading is preferred because threads use a shared memory area. They don’t allocate separate memory area so memory is saved. Also, context-switching between threads takes less time.

Java Multithreading is used mostly in games, animation, etc.

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

Ans. The sleep function causes the current thread to suspend execution for a specified period. This is an efficient means of making processor time available to the other threads of a program.

Example: Thread.sleep(1000) causes the particular thread to sleep for 1,000 milliseconds.

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.

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

Ans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Modifier and Type** | **Method** | **Description** |
| **1.** | void | start() | It is used to start the execution of a thread. |
| **2.** | String | getName() | It returns the name of the thread. |
| **3.** | long | getId() | It returns the id of the thread. |
| **4.** | void | suspend() | It is used to suspend the thread. |
| **5.** | void | stop() | It is used to stop the thread. |