Distributed and Cloud Computing Laboratory

B.Tech. 6th Semester

Name	**: Deepak R**
Roll Number	**: 18ETCS002041**
Department	**: Computer Science and Engineering**

Faculty of Engineering & Technology

Ramaiah University of Applied Sciences

Ramaiah University of Applied Sciences

Private University Established in Karnataka State by Act No. 15 of 2013

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Year/Semester	2nd Year / 6th Semester
Name of the Laboratory	Distributed and Cloud Computing Laboratory
Laboratory Code	19CSL316A

Laboratory 1

Title of the Laboratory Exercise: Multithreaded Programs in Java

1. **Introduction and Purpose of Experiment**

   Multithreading is the ability of a single core or a multi-core processor to execute multiple threads concurrently, supported by Java run time system. By solving this students will be able to manipulate multiple threads in a Java program.

   Aim and Objectives

   Aim

   - To develop Java multithreaded programs

2. **Experimental Procedure**

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in Java language
   iv. Compile the Java program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

3. **Questions**

   Implement the following:

   a. Create two Java threads and display Hello World by them
   b. Create four Java threads and display the results of addition, subtraction, multiplication and division of two numbers by each thread.

4. Calculations/Computations/Algorithms

4.1 Algorithm for Program to Create two Java threads and display Hello World by them

1: Start

2: Main Class should be Created

3: Two objects of Thread class Should be declared

4: We Should Pass newly created objects of the subclass for each Thread class object with threadNo as parameter of the constructor to be declared in the subclass

5: We should Use the created thread objects to implement Runnable interface

6: Create a subclass that implements Runnable interface

7: Parameterized constructor to initialize threadNo Should be Declared.

8: Declare run() method for subclass to print message "Hello World"

9: End

4.2 Algorithm for Program Create four Java threads and display the results of addition, subtraction, multiplication and division of two numbers by each thread.

1: Start

2: Create a Main class

3: Declare and input two integers

4: Declare four objects of Thread class

5: Create four subclasses for Addition, Subtraction, Multiplication and Division where each implements Runnable interface

6: In the Main class, pass newly created objects of the subclasses for each of the Thread class objects with threadNo, x and y as parameters of the constructor to be declared in the subclasses

7: Declare a constructor to initialize num1, num2 and threadNo inside each of the subclasses with the values passed by the objects of Main class

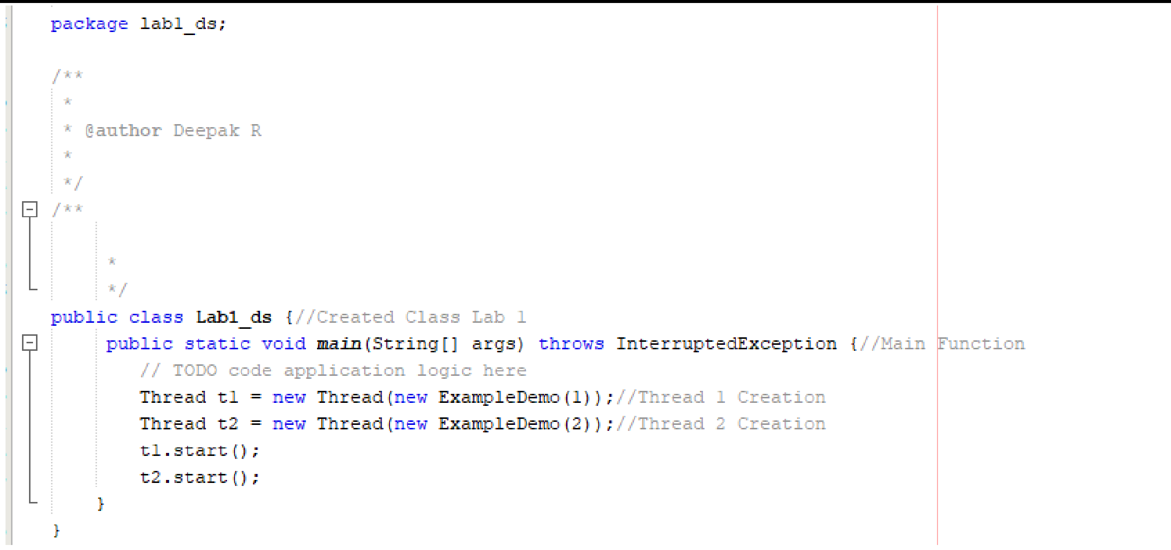
8: Declare run() method for each subclass to compute sum, difference, product and quotient and print the results when invoked by the respective thread object.

9: Use the thread objects to implement Runnable interface for each subclass

10: End

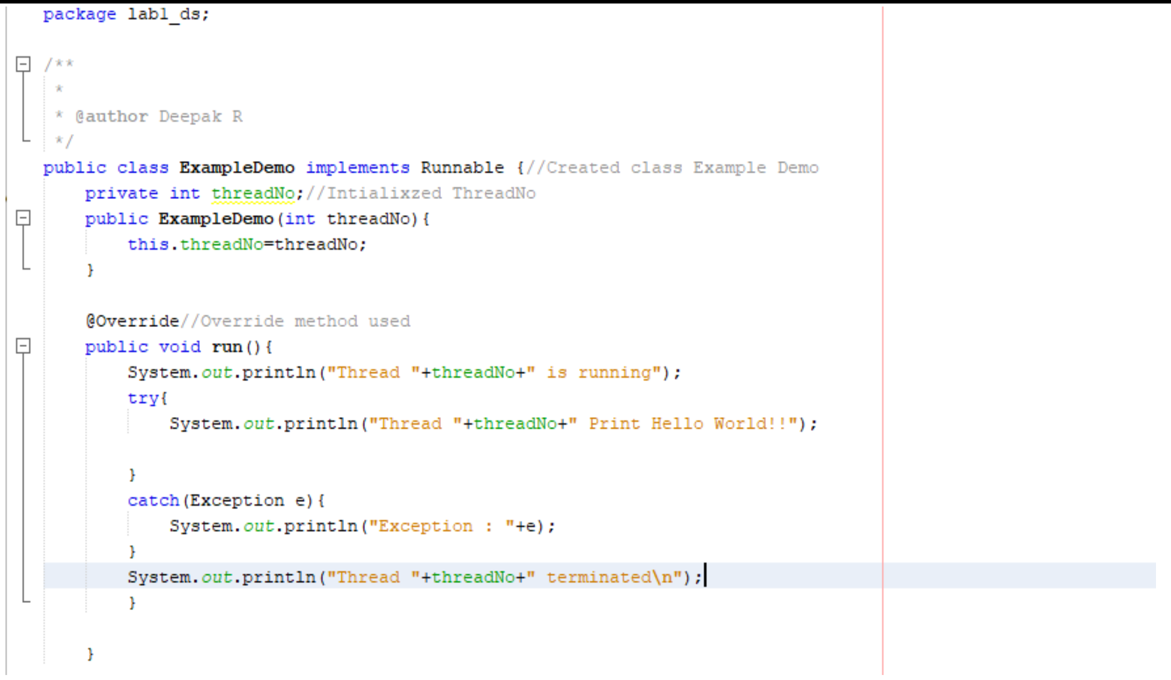
5. Presentation of Results

1.Program to Create two Java threads and display Hello World by them



```
package lab1_ds;

/**
 *
 * @author Deepak R
 *
 */
/**

      *
      */
public class Lab1_ds {//Created Class Lab 1
    public static void main(String[] args) throws InterruptedException {//Main Function
        // TODO code application logic here
        Thread t1 = new Thread(new ExampleDemo(1));//Thread 1 Creation
        Thread t2 = new Thread(new ExampleDemo(2));//Thread 2 Creation
        t1.start();
        t2.start();
    }
}
```

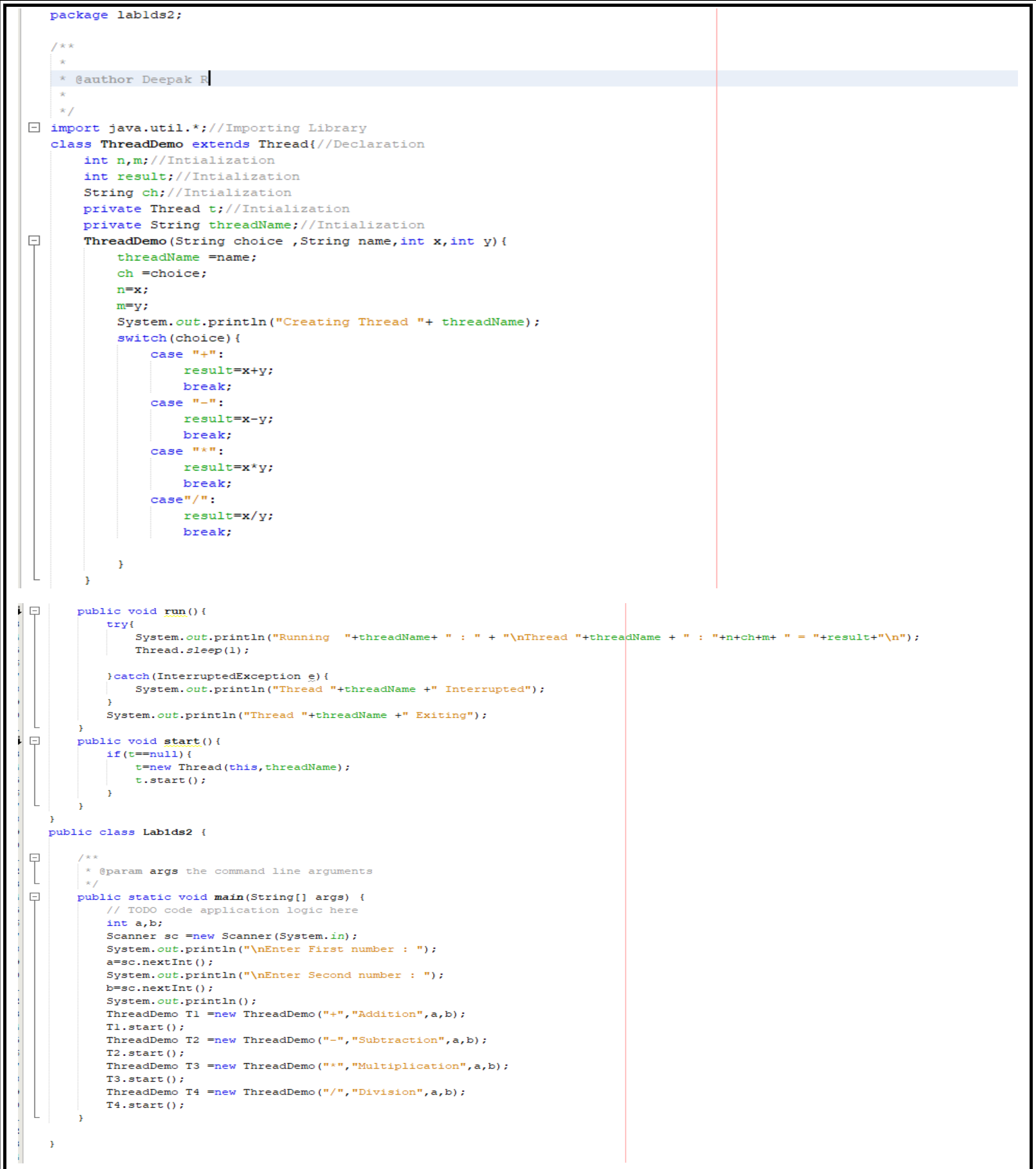
Fig 5.1 Main Class with 2 Threads



```
package lab1_ds;

/**
 *
 * @author Deepak R
 */
public class ExampleDemo implements Runnable {//Created class Example Demo
    private int threadNo;//Intialixzed ThreadNo
    public ExampleDemo(int threadNo){
        this.threadNo=threadNo;
    }

    @Override//Override method used
    public void run(){
        System.out.println("Thread "+threadNo+" is running");
        try{
            System.out.println("Thread "+threadNo+" Print Hello World!!");

        }
        catch(Exception e){
            System.out.println("Exception : "+e);
        }
        System.out.println("Thread "+threadNo+" terminated\n");
        }

    }
```

Fig 5.2 Implementation in Subclass

2.Program to Create four Java threads and display the results of addition, subtraction, multiplication and division of two numbers by each thread.



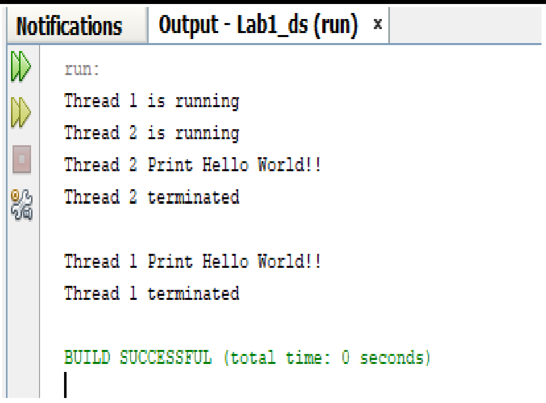
```java
package lab1ds2;

/**
 *
 * @author Deepak R
 *
 */
import java.util.*;//Importing Library
class ThreadDemo extends Thread{//Declaration
    int n,m;//Intialization
    int result;//Intialization
    String ch;//Intialization
    private Thread t;//Intialization
    private String threadName;//Intialization
    ThreadDemo(String choice ,String name,int x,int y){
        threadName =name;
        ch =choice;
        n=x;
        m=y;
        System.out.println("Creating Thread "+ threadName);
        switch(choice){
            case "+":
                result=x+y;
                break;
            case "-":
                result=x-y;
                break;
            case "*":
                result=x*y;
                break;
            case"/":
                result=x/y;
                break;

        }
    }

    public void run(){
        try{
            System.out.println("Running  "+threadName+ " : " + "\nThread "+threadName + " : "+n+ch+m+ " = "+result+"\n");
            Thread.sleep(1);

        }catch(InterruptedException e){
            System.out.println("Thread "+threadName +" Interrupted");
        }
        System.out.println("Thread "+threadName +" Exiting");
    }
    public void start(){
        if(t==null){
            t=new Thread(this,threadName);
            t.start();
        }
    }
}
public class Lab1ds2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int a,b;
        Scanner sc =new Scanner(System.in);
        System.out.println("\nEnter First number : ");
        a=sc.nextInt();
        System.out.println("\nEnter Second number : ");
        b=sc.nextInt();
        System.out.println();
        ThreadDemo T1 =new ThreadDemo("+","Addition",a,b);
        T1.start();
        ThreadDemo T2 =new ThreadDemo("-","Subtraction",a,b);
        T2.start();
        ThreadDemo T3 =new ThreadDemo("*","Multiplication",a,b);
        T3.start();
        ThreadDemo T4 =new ThreadDemo("/","Division",a,b);
        T4.start();
    }

}
```

Fig 5.3 Implementation of Program 2

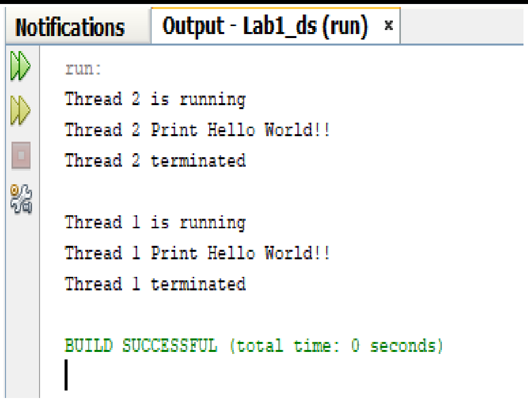
Output

Test Cases

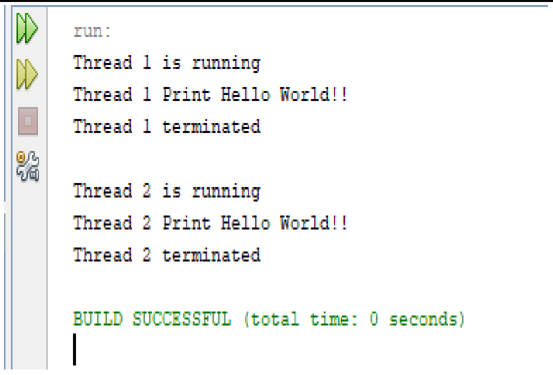
Output for Program to Create two Java threads and display Hello World by them



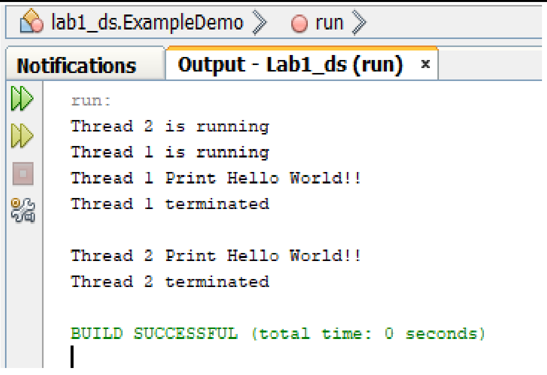
```
run:
Thread 1 is running
Thread 2 is running
Thread 2 Print Hello World!!
Thread 2 terminated

Thread 1 Print Hello World!!
Thread 1 terminated

BUILD SUCCESSFUL (total time: 0 seconds)
```



```
run:
Thread 2 is running
Thread 2 Print Hello World!!
Thread 2 terminated

Thread 1 is running
Thread 1 Print Hello World!!
Thread 1 terminated

BUILD SUCCESSFUL (total time: 0 seconds)
```



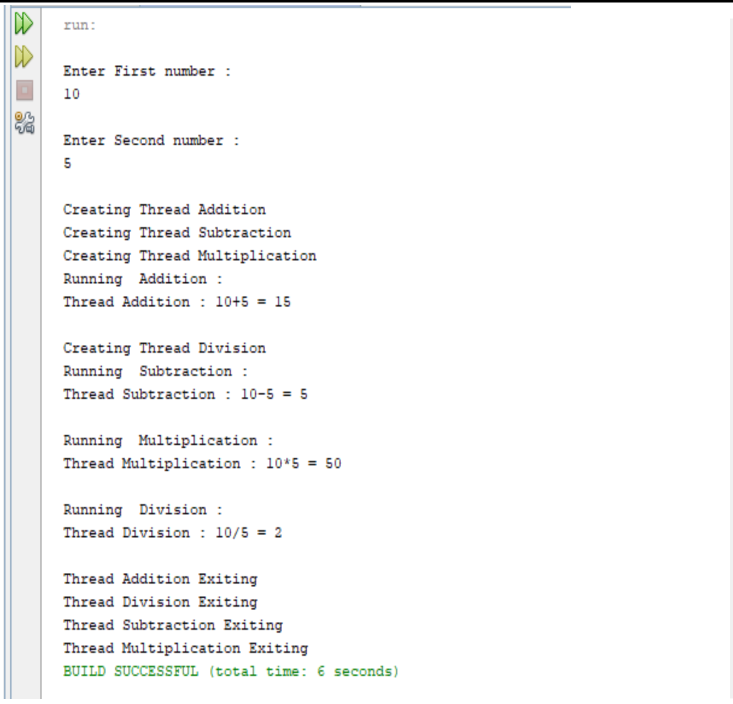
```
run:
Thread 1 is running
Thread 1 Print Hello World!!
Thread 1 terminated

Thread 2 is running
Thread 2 Print Hello World!!
Thread 2 terminated

BUILD SUCCESSFUL (total time: 0 seconds)
```



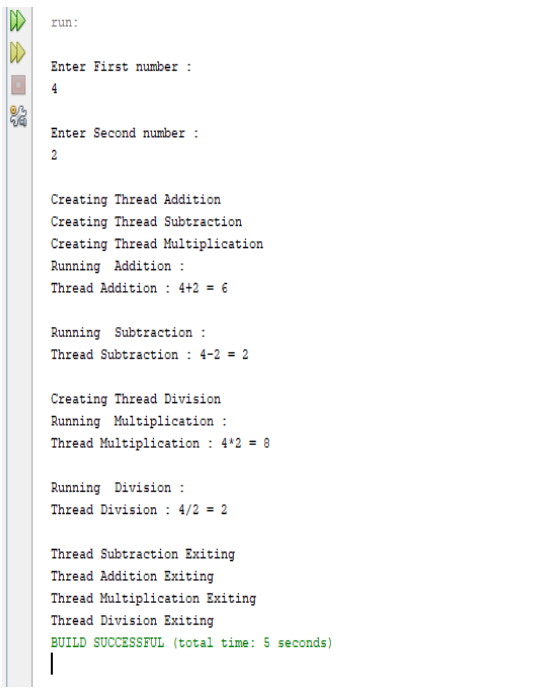
```
run:
Thread 2 is running
Thread 1 is running
Thread 1 Print Hello World!!
Thread 1 terminated

Thread 2 Print Hello World!!
Thread 2 terminated

BUILD SUCCESSFUL (total time: 0 seconds)
```

Test cases

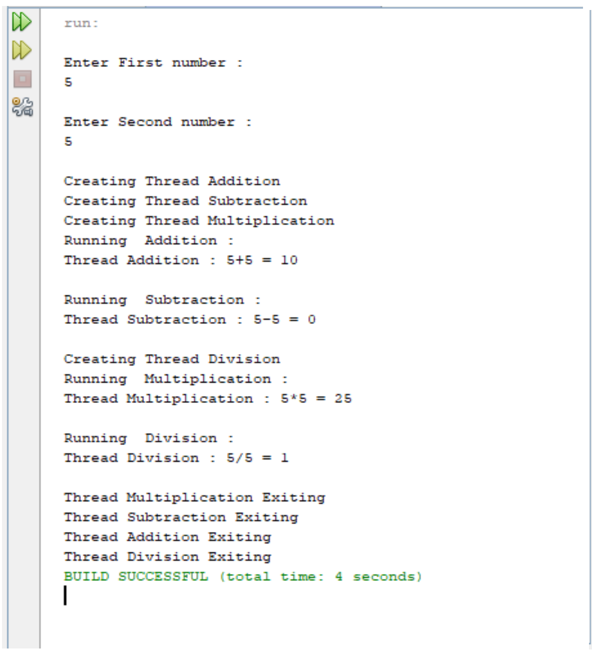
Output for Program to Create four Java threads and display the results of addition, subtraction, multiplication and division of two numbers by each thread.



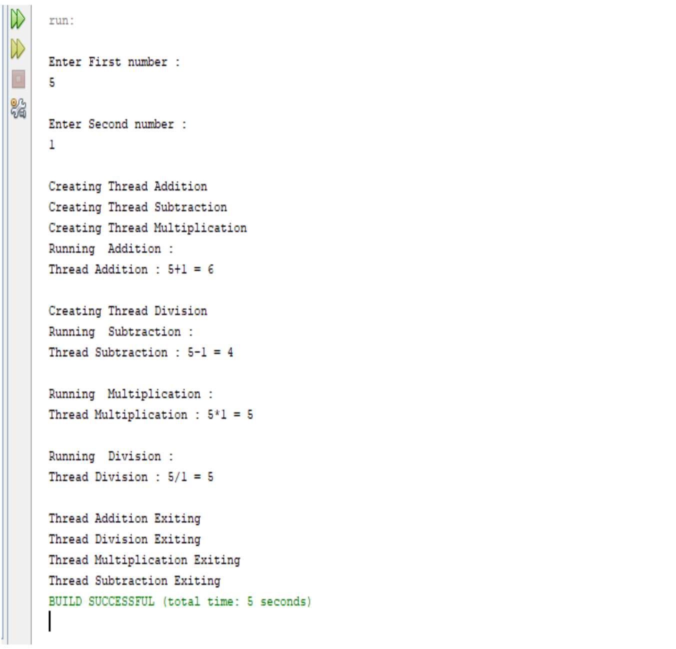
```
run:

Enter First number :
10

Enter Second number :
5

Creating Thread Addition
Creating Thread Subtraction
Creating Thread Multiplication
Running  Addition :
Thread Addition : 10+5 = 15

Creating Thread Division
Running  Subtraction :
Thread Subtraction : 10-5 = 5

Running  Multiplication :
Thread Multiplication : 10*5 = 50

Running  Division :
Thread Division : 10/5 = 2

Thread Addition Exiting
Thread Division Exiting
Thread Subtraction Exiting
Thread Multiplication Exiting
BUILD SUCCESSFUL (total time: 6 seconds)
```



```
run:

Enter First number :
4

Enter Second number :
2

Creating Thread Addition
Creating Thread Subtraction
Creating Thread Multiplication
Running  Addition :
Thread Addition : 4+2 = 6

Running  Subtraction :
Thread Subtraction : 4-2 = 2

Creating Thread Division
Running  Multiplication :
Thread Multiplication : 4*2 = 8

Running  Division :
Thread Division : 4/2 = 2

Thread Subtraction Exiting
Thread Addition Exiting
Thread Multiplication Exiting
Thread Division Exiting
BUILD SUCCESSFUL (total time: 5 seconds)
```



```
run:

Enter First number :
5

Enter Second number :
5

Creating Thread Addition
Creating Thread Subtraction
Creating Thread Multiplication
Running  Addition :
Thread Addition : 5+5 = 10

Running  Subtraction :
Thread Subtraction : 5-5 = 0

Creating Thread Division
Running  Multiplication :
Thread Multiplication : 5*5 = 25

Running  Division :
Thread Division : 5/5 = 1

Thread Multiplication Exiting
Thread Subtraction Exiting
Thread Addition Exiting
Thread Division Exiting
BUILD SUCCESSFUL (total time: 4 seconds)
```



```
run:

Enter First number :
5

Enter Second number :
1

Creating Thread Addition
Creating Thread Subtraction
Creating Thread Multiplication
Running  Addition :
Thread Addition : 5+1 = 6

Creating Thread Division
Running  Subtraction :
Thread Subtraction : 5-1 = 4

Running  Multiplication :
Thread Multiplication : 5*1 = 5

Running  Division :
Thread Division : 5/1 = 5

Thread Addition Exiting
Thread Division Exiting
Thread Multiplication Exiting
Thread Subtraction Exiting
BUILD SUCCESSFUL (total time: 5 seconds)
```

Analysis and Discussions

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. Thus, multithreading is a specialized form of multitasking. Here we have done arithmetic operation by using Multithreaded method.

1. Limitations of Experiments

- Difficulty of writing code

Multithreaded applications are not easy to write.

- Difficulty of debugging

It is much harder to replicate an error in a multithreaded orapplication than it is to do so in a single-threaded, single-contexted application. As a result, it is more difficult, in the former case, to identify and verify root causes when errors occur.

- Difficulty of managing concurrency

The task of managing concurrency among threads is difficult and has the potential to introduce new problems into an application.

2. Limitations of Results

Unpredictable Results Sometime

3. Learning happened

We learnt how to do Multithreaded operations and its application.

4. Recommendations

Above code can be further optimized.