MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Multithreading using Thread Class and Runnable Interface in Java

Practical Report
Assignment 7

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 19, 2022

Contents

1 Aim and Objectives **1**

2 Problem Statements **1**
2.1 Problem 1 in Java 1
2.2 Problem 2 in Java 1

3 Theory **1**

4 Platform **1**

5 Input **1**

6 Output **2**

7 Code **2**
7.1 Java Implementation of Problem 1 2
7.1.1 Java Output 4
7.2 Java Implementation of Problem 2 5
7.2.1 Java Output 6

8 Conclusion **7**

9 FAQs **8**

1 Aim and Objectives

Aim

Implementing Solutions on Multithreading using Thread Class and Runnable Interface

Objectives

1. To understand Multithreading in Java
2. To learn two different ways to create threads in Java

2 Problem Statements

2.1 Problem 1 in Java

Write a program to create a multithreaded calculator that does addition, subtraction, multiplication, and division using separate threads. Additionally also handle '/ by zero' exception by the division method.

2.2 Problem 2 in Java

Print even and odd numbers in increasing order using two threads in Java

3 Theory

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For Problem 1

1. 2 numbers
2. Choice about what to do with those numbers

For Problem 2

1. The Maximum limit up to which the user wants to see the odd and even numbers printed

6 Output

For Problem 1

1. Menu about what to do with numbers
2. Output of the calculation done with those numbers

For Problem 2

1. Even numebers and Odd numbers in Ascending order upto the specified limit.

7 Code

7.1 Java Implementation of Problem 1

```java
// Krishnaraj Thadesar
// Batch A1, PA20
// OOPCJ Assignment 7.1
// Write a program to create a multithreaded calculator that does addition, subtraction,
// multiplication, and division using separate threads.
// Additionally also handle '/ by zero' exception by the division method.


import java.lang.Thread;
import java.util.Scanner;

class Calculator extends Thread implements Runnable {
    public int a, b, what_to_do = 0;

    Calculator(int a, int b, int choice, String name) {
        this.a = a;
        this.b = b;
        this.what_to_do = choice;
        this.setName(name);
    }

    @Override
    public synchronized void start() {
        System.out.println("Starting the Thread");
        System.out.println("The Name of this Thread is: " + getName());
        super.start();
    }

    @Override
    public void run() {
        switch (what_to_do) {
            case 1:
                System.out.println(a + b);
                break;
            case 2:
                System.out.println(a - b);
                break;
            case 3:
                System.out.println(a * b);
```

```
                break;
            case 4:
                try {
                    System.out.println(a / b);
                } catch (ArithmeticException e) {
                    System.out.println("You cant Divide by Zero!");
                }
                break;
            default:
                break;
        }
    }
}

public class assignment_7_problem_1 {
    public static Calculator add, sub, mul, div;
    public static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        int choice = 0;
        int a, b;
        System.out.println("Welcome To Thread Calculator of Assignment 7");
        while (choice != 5) {
            System.out.println("What would you like to do? ");
            System.out.println(
                    "1. Addition of 2 Numbers\n2. Subtraction of 2 Numbers\n3. Multiplication of 2 Numbers\n4. Division of 2 Numbers\n\n");
            choice = input.nextInt();
            if (choice == 5) {
                break;
            }
            System.out.println("Enter the 2 Numbers");
            a = input.nextInt();
            b = input.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("You have chosen Addition!");
                    add = new Calculator(a, b, choice, "Adder");
                    try {
                        add.start();
                        add.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the thread!");
                        System.out.println(e);
                    }
                    break;
                case 2:
                    System.out.println("You have chosen Subtraction!");
                    sub = new Calculator(a, b, choice, "Subtractor");
                    try {
                        sub.start();
                        sub.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the thread!");
                        System.out.println(e);
                    }
                    break;
```

```
                case 3:
                    System.out.println("You have chosen Multiplication!");
                    mul = new Calculator(a, b, choice, "Multiplier");
                    try {
                        mul.start();
                        mul.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the
    thread!");
                        System.out.println(e);
                    }
                    break;
                case 4:
                    System.out.println("You have chosen Division!");
                    div = new Calculator(a, b, choice, "Divider");
                    try {
                        div.start();
                        div.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the
    thread!");
                        System.out.println(e);
                    }
                    break;
                case 5:
                    System.out.println("You have chosed to Exit!");
                default:
                    break;
            }

        }
        System.exit(0);
    }
}
```

Listing 1: Probelm 1.java

7.1.1 Java Output

```
Welcome To Thread Calculator of Assignment 7
What would you like to do?
1. Addition of 2 Numbers
2. Subtraction of 2 Numbers
3. Multiplication of 2 Numbers
4. Division of 2 Numbers


1
Enter the 2 Numbers
2
2
You have chosen Addition!
Starting the Thread
The Name of this Thread is: Adder
4
What would you like to do?
1. Addition of 2 Numbers
2. Subtraction of 2 Numbers
3. Multiplication of 2 Numbers
```

```
4. Division of 2 Numbers


4
Enter the 2 Numbers
5
0
You have chosen Division!
Starting the Thread
The Name of this Thread is: Divider
You cant Divide by Zero!
What would you like to do?
1. Addition of 2 Numbers
2. Subtraction of 2 Numbers
3. Multiplication of 2 Numbers
4. Division of 2 Numbers


5
```

Listing 2: Output for Problem 1 - calculations

7.2 Java Implementation of Problem 2

```java
// Krishnaraj Thadesar
// Batch A1, PA20
// OOPCJ Assignment 7.2
// Print even and odd numbers in increasing order using two threads in Java

import java.security.ProtectionDomain;
import java.util.Scanner;

import javax.swing.InputMap;

class printEven extends Thread implements Runnable {
    int limit;

    printEven(int limit) {
        this.limit = limit;
    }

    @Override
    public synchronized void start() {
        super.start();
        System.out.println("Printing Even Numbers");
    }

    @Override
    public void run() {
        for (int i = 0; i < limit; i++) {
            if (i % 2 == 0) {
                System.out.println(i);
            }
        }
    }
}

class printOdd extends Thread implements Runnable {
    int limit;
```

```
36
37      printOdd(int limit) {
38          this.limit = limit;
39      }
40
41      @Override
42      public synchronized void start() {
43          super.start();
44          System.out.println("Printing Odd Numbers");
45      }
46
47      @Override
48      public void run() {
49          for (int i = 0; i < limit; i++) {
50              if (i % 2 != 0) {
51                  System.out.println(i);
52              }
53          }
54      }
55  }
56
57  public class assignment_7_problem_2 {
58      static printEven pe;
59      static printOdd po;
60      static Scanner input = new Scanner(System.in);
61
62      public static void main(String[] args) {
63          int limit = 0;
64          System.out.println("Enter To what limit Even or Odd numbers you want to
        See");
65          limit = input.nextInt();
66          pe = new printEven(limit);
67          po = new printOdd(limit);
68          try {
69              pe.start();
70              pe.join();
71              po.start();
72              po.join();
73          } catch (Exception e) {
74              System.out.println(e);
75          }
76      }
77  }
```

Listing 3: Multithreading Even Odd

7.2.1 Java Output

```
1  Enter To what limit Even or Odd numbers you want to See
2  10
3  Printing Even Numbers
4  0
5  2
6  4
7  6
8  8
9  Printing Odd Numbers
10 1
11 3
```

```
5
7
9
```

Listing 4: Output for ProblemHillStation 2

8 Conclusion

Thus, learnt the use of thread class in java and performed multithreading operations.

9 FAQs

1. *Why do we use collection framework?*
2. *Which is best collection framework in Java?*
3. *What is difference between array and collection?*
4. *What is HashMap in Java?*