Part 6

Github link : https://github.com/kalp104/java_21CE084_part6.git

Program 1	Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface

Code:

```java
// by extending Thread Class

import java.util.*;
public class prac1 extends Thread {
    public void run(){
        System.out.println("HELLO WORLD...!");
    }
    public static void main(String[] args) {
        prac1 obj = new prac1();
        obj.run();
    }
}
```

```java
// by using Runnable Interface
import java.util.*;
public class prac1part2 implements Runnable{

    public void run(){
        System.out.println("HELLO WORLD...2");
    }
    public static void main(String[] args) {
        prac1part2 obj = new prac1part2();
        Thread t1 = new Thread(obj);
        t1.run();
    }
}
```

Program 2	Generate 15 random numbers from 1 to 100 and store it in an int array. Write a program to display the numbers stored at odd indexes by thread1 and display numbers stored at even indexes by thread2.

Code:

```java
import java.util.Scanner;

// I don't know if I've actually done multithreading but
anyways
```

```java
class DistributedSummation extends Thread {
    public static int sum = 0;
    public static int assignedNumbers;
    public int startNumber;
    public int endNumber;

    public void setValue(int a, int b) {
        startNumber = a;
        endNumber = b;
    }

    synchronized public void sum() {
        for (int i = startNumber; i < endNumber; i++) {
            sum += i;
        }
    }

    public void run() {
        System.out.println(Thread.currentThread().getName()
+ " is running");
    }

}

public class pra_6_2{
    public static void main(String[] args) throws Exception
{
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the number upto you wanna
find sum:");
        int n = scan.nextInt();
        System.out.println("Enter the no. of threads you
want to sum" + n + " nos. :");
        int numberOfThreads = scan.nextInt();
        scan.close();
        int numberTracker = 1;

        DistributedSummation[] t = new
DistributedSummation[numberOfThreads];
        for (int i = 0; i < numberOfThreads; i++) {
            t[i] = new DistributedSummation();
        }

        DistributedSummation.assignedNumbers = n /
numberOfThreads;
        int remainingNumbers = n % numberOfThreads;
        for (int i = 0; i < numberOfThreads; i++) {
            t[i].start();
            t[i].setValue(numberTracker,
DistributedSummation.assignedNumbers * (i + 1));
            numberTracker =
DistributedSummation.assignedNumbers * (i + 1);
        }
        for (int i = 0; i < numberOfThreads; i++) {
            t[i].sum();
        }

        if (remainingNumbers != 0) {
            t[0].setValue(numberTracker + 1, n + 1);
            t[0].sum();
        }
        if (remainingNumbers != 0)
            System.out.println("The sum of the " + n + "
numbers using " + numberOfThreads + " is "
                    + (DistributedSummation.sum + n -
remainingNumbers));
        else
            System.out.println("The sum of the " + n + "
numbers using " + numberOfThreads + " is "
                    + (DistributedSummation.sum + n));
    }
}
```

Program 3	Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.

Code :

```
class Mythread extends Thread {
    public static int counter = 0;

    public void run() {
        System.out.println(
Thread.currentThread().getName() + " is running");
    }

    static void increment() {
        counter++;
    }
}
class pra_6_3{
    public static void main(String[] args) {
        Mythread t1 = new Mythread();
        t1.start();
        System.out.println("Before increment is called the
value of counter is : " + t1.counter);
```

```java
        System.out.println("\nThread t1 sleep method called");
        try {
            t1.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println(e);
        }
        t1.increment();
        System.out.println("After increment is called the value of counter is : " + t1.counter);
    }
}
```

Program 4

Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.

Code

```java
class Mythread extends Thread {
    public void run() {
        System.out.println("Thread " + Thread.currentThread().getName() + " is running");
    }
}

public class pra_6_4{
    public static void main(String[] args) {
        Mythread t1 = new Mythread();
        Mythread t2 = new Mythread();
        Mythread t3 = new Mythread();

        t1.setName("First");
        t2.setName("Second");
        t3.setName("Third");
        t1.setPriority(3);
        t2.setPriority(5);
        t3.setPriority(7);
        t1.start();
        t2.start();
        t3.start();
    }
}
```

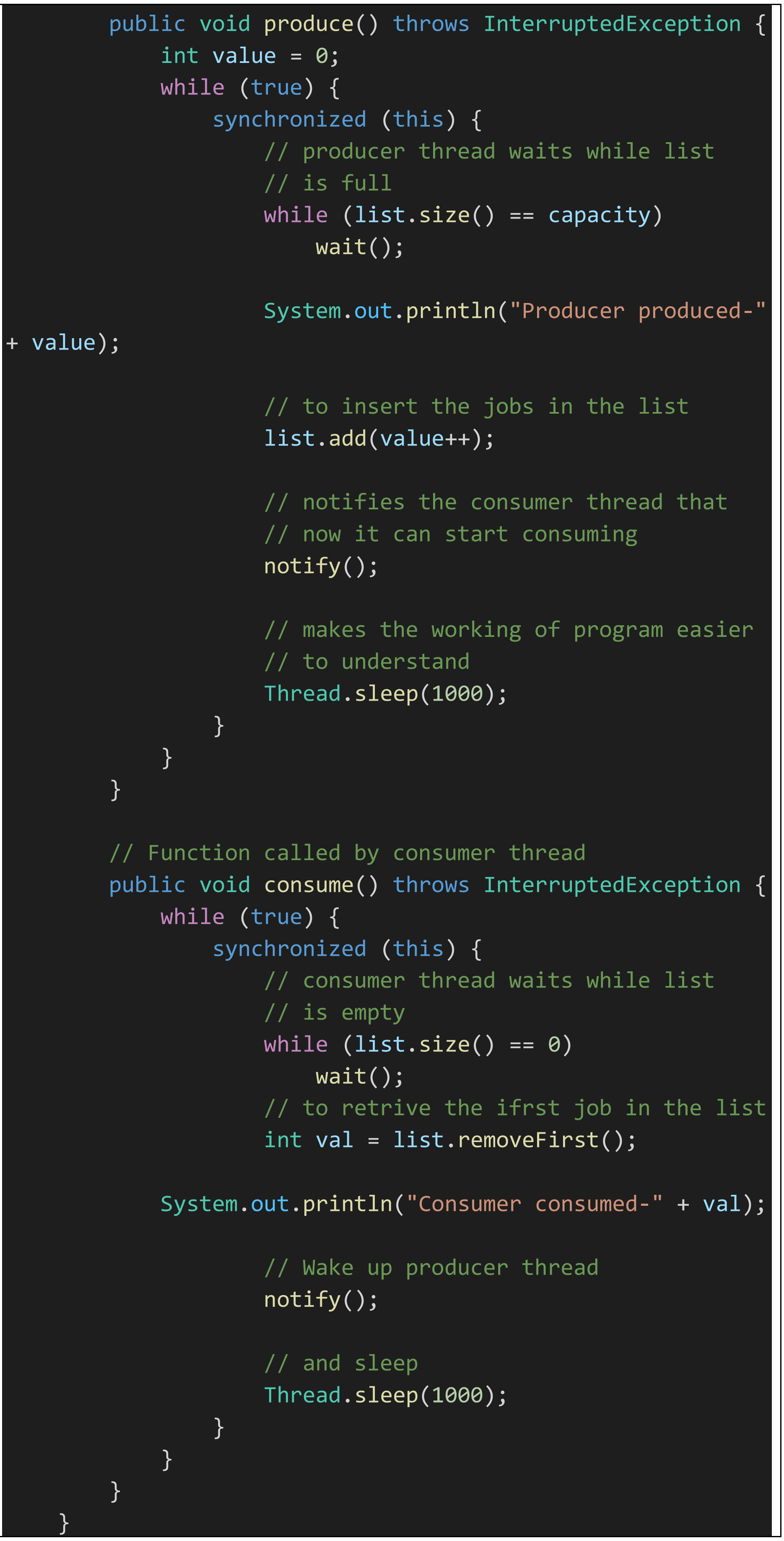
Program 5

Write a program to solve producer-consumer problem using thread Synchronization

Code

```java
import java.util.LinkedList;
```

```
public class pra_6_5{
    public static void main(String[] args) throws
InterruptedException {
        // Object of a class that has both produce()
        // and consume() methods
        final PC pc = new PC();
        // Create producer thread
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    pc.produce();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        // Create consumer thread
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    pc.consume();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        // Start both threads
        t1.start();
        t2.start();

        // t1 finishes before t2
        t1.join();
        t2.join();
    }

    // This class has a list, producer (adds items to list
    // and consumber (removes items).
    public static class PC {
        // Create a list shared by producer and consumer
        // Size of list is 2.
        LinkedList<Integer> list = new LinkedList<>();
        int capacity = 2;

        // Function called by producer thread
```



```
        public void produce() throws InterruptedException {
            int value = 0;
            while (true) {
                synchronized (this) {
                    // producer thread waits while list
                    // is full
                    while (list.size() == capacity)
                        wait();

                    System.out.println("Producer produced-"
+ value);

                    // to insert the jobs in the list
                    list.add(value++);

                    // notifies the consumer thread that
                    // now it can start consuming
                    notify();

                    // makes the working of program easier
                    // to understand
                    Thread.sleep(1000);
                }
            }
        }

        // Function called by consumer thread
        public void consume() throws InterruptedException {
            while (true) {
                synchronized (this) {
                    // consumer thread waits while list
                    // is empty
                    while (list.size() == 0)
                        wait();
                    // to retrive the ifrst job in the list
                    int val = list.removeFirst();

            System.out.println("Consumer consumed-" + val);

                    // Wake up producer thread
                    notify();

                    // and sleep
                    Thread.sleep(1000);
                }
            }
        }
    }
```

	 }