PSG COLLEGE OF TECHNOLOGY, COIMBATORE – 641 004
Department of Applied Mathematics and Computational Sciences
MSc Software Systems – Semester V
20XW57– Java Programming Lab
PROBLEM SHEET 7 – Multithreading

Completion Date: 14/09/2024

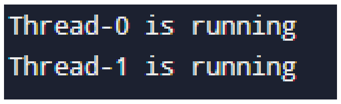
Note: ***Use Text Editors/IntelliJ IDEA/Apache NetBeans tools to develop, compile and execute the below programs***

1. **Write a program to illustrate creation of threads using Runnable interface (start method start each of the newly created thread. Inside the run method there is *sleep ( )* for suspend the thread for 500 milliseconds).**

Ans:

```
class MyRunnable implements Runnable {

    public void run() {

        try {

            System.out.println(Thread.currentThread().getName() + " is running");

            Thread.sleep(500);

        } catch (InterruptedException e) {

            System.out.println(e);

        }

    }
    public static void main(String[] args) {

        MyRunnable runnable = new MyRunnable();

        Thread t1 = new Thread(runnable);

        Thread t2 = new Thread(runnable);



        t1.start();

        t2.start();

    }

}
```

Output:



```
Thread-0 is running
Thread-1 is running
```

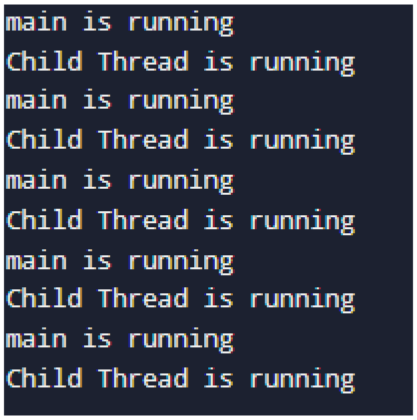
2. **Write a program to create a class MyThread in this class a constructor, call the base class constructor, using super and starts the thread. The run method of the class starts after this. It can be observed that both main thread and created**

child thread are executed concurrently.

Ans:

```
class MyThread extends Thread {

    public MyThread(String threadName) {

        super(threadName);

        start();

    }



    @Override

    public void run() {

        try {

            for (int i = 0; i < 5; i++) {

                System.out.println(Thread.currentThread().getName() + " is
running");

                Thread.sleep(500);

            }

        } catch (InterruptedException e) {

            System.out.println(e);

        }

    }
    public static void main(String[] args) {

        MyThread childThread = new MyThread("Child Thread");

        try {

            for (int i = 0; i < 5; i++) {

                System.out.println(Thread.currentThread().getName() + " is 
running");

                Thread.sleep(500);

            }

        } catch (InterruptedException e) {

            System.out.println(e);

        }

    }

}
```

Output:



```
main is running
Child Thread is running
main is running
Child Thread is running
main is running
Child Thread is running
main is running
Child Thread is running
main is running
Child Thread is running
```

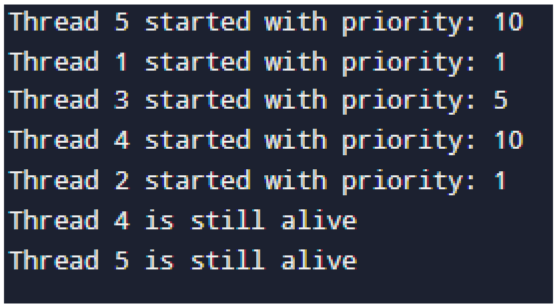
3. Write a java program to create five threads with different priorities. Send two threads of highest priority in sleep state. Check the aliveness of the threads and mark which thread is long listing.

Ans:

```java
class PriorityThread extends Thread {

    public PriorityThread(String name) {

        super(name);

    }



    public void run() {

        System.out.println(getName() + " started with priority: " + getPriority());

        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        PriorityThread t1 = new PriorityThread("Thread 1");
        PriorityThread t2 = new PriorityThread("Thread 2");
        PriorityThread t3 = new PriorityThread("Thread 3");
        PriorityThread t4 = new PriorityThread("Thread 4");
        PriorityThread t5 = new PriorityThread("Thread 5");

        t1.setPriority(Thread.MIN_PRIORITY);
        t2.setPriority(Thread.MIN_PRIORITY);
        t3.setPriority(Thread.NORM_PRIORITY);
        t4.setPriority(Thread.MAX_PRIORITY);
        t5.setPriority(Thread.MAX_PRIORITY);



        t1.start();

        t2.start();

        t3.start();

        t4.start();

        t5.start();



        try {

            t4.sleep(500);

            t5.sleep(500);

        } catch (InterruptedException e) {

            e.printStackTrace();

        }



        System.out.println(t4.isAlive() ? "Thread 4 is still alive" :
"Thread 4 is not alive");

        System.out.println(t5.isAlive() ? "Thread 5 is still alive" :
```

```
"Thread 5 is not alive");


    }


}
```

Output:



```
Thread 5 started with priority: 10
Thread 1 started with priority: 1
Thread 3 started with priority: 5
Thread 4 started with priority: 10
Thread 2 started with priority: 1
Thread 4 is still alive
Thread 5 is still alive
```

4. Write a multithreaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value. For example, suppose your program is passed the

integers 90 81 78 95 79 72 85 Output:

The average value is 82

The minimum value is 72

The maximum value is 95

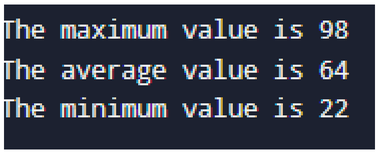
The variables representing the average, minimum, and maximum values will be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited.

Ans:

```
class StatisticsThread extends Thread {

    private int[] numbers;

    private String task;

    private static int average, minimum, maximum;



    public StatisticsThread(int[] numbers, String task) {

        this.numbers = numbers;

        this.task = task;

    }



    public void run() {

        if (task.equals("average")) {

            int sum = 0;

            for (int number : numbers) {

                sum += number;
```

```java
            }

            average = sum / numbers.length;

            System.out.println("The average value is " + average);

        } else if (task.equals("min")) {

            minimum = numbers[0];

            for (int number : numbers) {

                if (number < minimum) {

                    minimum = number;

                }

            }

            System.out.println("The minimum value is " + minimum);

        } else if (task.equals("max")) {

            maximum = numbers[0];

            for (int number : numbers) {

                if (number > maximum) {

                    maximum = number;

                }
            }

            System.out.println("The maximum value is " + maximum);

        }

    }



    public static void main(String[] args) {

        int[] numbers = {73, 41, 98, 22, 87};

        StatisticsThread avgThread = new StatisticsThread(numbers,
"average");

        StatisticsThread minThread = new StatisticsThread(numbers, "min");

        StatisticsThread maxThread = new StatisticsThread(numbers, "max");



        avgThread.start();

        minThread.start();

        maxThread.start();

    }

}
```

Output:



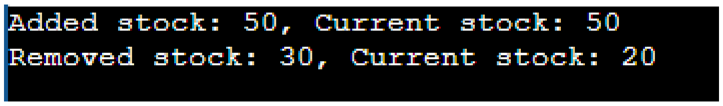
```
The maximum value is 98
The average value is 64
The minimum value is 22
```

5. Write a program for inventory problem, to illustrate the usage of synchronized keyword.

Ans:

```
class Inventory {

    private int stock = 0;


    public synchronized void addStock(int value) {

        stock += value;

        System.out.println("Added stock: " + value + ", Current stock: " + stock);

    }


    public synchronized void getStock(int value) {

        if (stock >= value) {
            stock -= value;

            System.out.println("Removed stock: " + value + ", Current
stock: " + stock);

        } else {

            System.out.println("Insufficient stock.");

        }

    }

}



public class StockOperation extends Thread {

    private Inventory inventory;

    private boolean add;

    private int value;



    public StockOperation(Inventory inventory, boolean add, int value) {

        this.inventory = inventory;

        this.add = add;

        this.value = value;
    }



    public void run() {

        if (add) {

            inventory.addStock(value);

        } else {

            inventory.getStock(value);

        }

    }



    public static void main(String[] args) {

        Inventory inventory = new Inventory();

        StockOperation addOp = new StockOperation(inventory, true, 50);

        StockOperation removeOp = new StockOperation(inventory, false,
30);



        addOp.start();

        removeOp.start();
    }

}
```

Output:



```
Added stock: 50, Current stock: 50
Removed stock: 30, Current stock: 20
```

6. Write a program for interthread communication process. In this they have three classes consumer, producer and stock.

addStock( )getStock( )

Producer Stock Consumer

notify( )wait( )

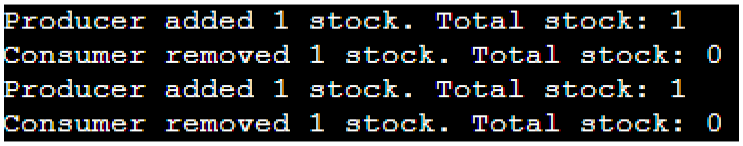
Ans:

```
class Stock {

    private int stock;



    public synchronized void addStock() throws InterruptedException {

        while (stock >= 1) {

            wait();
        }

        stock++;

        System.out.println("Producer added 1 stock. Total stock: " +
stock);

        notify();

    }



    public synchronized void getStock() throws InterruptedException {

        while (stock < 1) {

            wait();

        }

        stock--;

        System.out.println("Consumer removed 1 stock. Total stock: " +
stock);

        notify();

    }

}
```

```
class Producer extends Thread {

    Stock stock;



    Producer(Stock stock) {

        this.stock = stock;

    }



    public void run() {

        try {

            while (true) {

                stock.addStock();

                Thread.sleep(500);

            }

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }
```

```java
}



class Consumer extends Thread {

    Stock stock;



    Consumer(Stock stock) {

        this.stock = stock;

    }



    public void run() {

        try {

            while (true) {

                stock.getStock();

                Thread.sleep(500);

            }

        } catch (InterruptedException e) {

            e.printStackTrace();
        }
    }
}

public class MainDriver {
    public static void main(String[] args) {
        Stock stock = new Stock();
        Producer producer = new Producer(stock);
        Consumer consumer = new Consumer(stock);

        producer.start();
        consumer.start();
    }
}
```

Output:



```
Producer added 1 stock. Total stock: 1
Consumer removed 1 stock. Total stock: 0
Producer added 1 stock. Total stock: 1
Consumer removed 1 stock. Total stock: 0
```

7. Your English literature friend is very happy with the code you gave him. Now for his research, he used your application to find character frequency in many novels. For larger novels, the application takes a lot of time for computation. So he called you on a fine Sunday to discuss this with you. He wanted to know whether you can improve the speed of the application.

   You decided to modify the application by using multiple threads to reduce the computation time. For this, accept the number of counters or threads at the beginning of the problem and get the string for each counter or thread. Create a thread by extending the Thread class and take the user entered string as input. Each thread calculates the character frequency for the word assigned to that thread. All the counts are stored locally in the thread and once all the threads are completed print the character frequency for each of the threads.

   Create a class Main and test it.

   Input and Output format:

   Refer to sample Input and Output for formatting specifications.

   Sample input and output:

   Enter Number of Counters:2

   Enter text for counter 1: FrequencyCounter

   Enter text for counter 2: JavaTheCompleteReference

   Counter 1 Result :

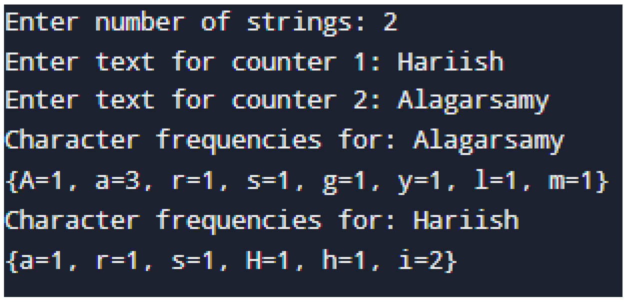
   C:1 F:1 c:1 e:3 n:2 o:1 q:1 r:2 t:1 u:2 y:1

```
Counter 2 Result :

C:1 J:1 R:1 T:1 a:2 c:1 e:7 f:1 h:1 l:1 m:1 n:1 o:1 p:1 r:1
t:1 v:1
```

Ans:

```java
import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;



class FrequencyCounter extends Thread {

    private String text;

    private Map<Character, Integer> frequencyMap = new HashMap<>();



    public FrequencyCounter(String text) {

        this.text = text;

    }



    public void run() {

        for (char c : text.toCharArray()) {

            frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);
        }
        System.out.println("Character frequencies for: " + text);
        System.out.println(frequencyMap);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of strings: ");
        int numOfStrings = scanner.nextInt();
        scanner.nextLine(); // Consume newline character

        String[] texts = new String[numOfStrings];
        for (int i = 0; i < numOfStrings; i++) {
            System.out.print("Enter text for counter " + (i + 1) + ": ");
            texts[i] = scanner.nextLine();
        }
        FrequencyCounter[] counters = new FrequencyCounter[texts.length];

        for (int i = 0; i < texts.length; i++) {

            counters[i] = new FrequencyCounter(texts[i]);

            counters[i].start();

        }



        for (FrequencyCounter counter : counters) {

            try {

                counter.join();

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }



        scanner.close();

    }

}
```

Output:



```
Enter number of strings: 2
Enter text for counter 1: Hariish
Enter text for counter 2: Alagarsamy
Character frequencies for: Alagarsamy
{A=1, a=3, r=1, s=1, g=1, y=1, l=1, m=1}
Character frequencies for: Hariish
{a=1, r=1, s=1, H=1, h=1, i=2}
```