Object Oriented System Laboratory Assignment-2

Name: Kaustav Mondal

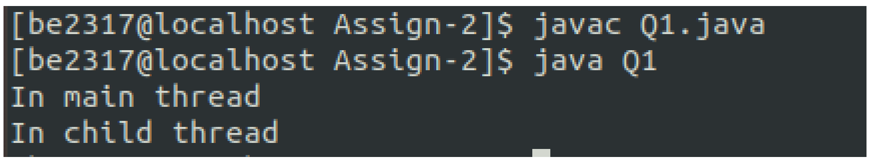
Section: A1

Roll No: 002311001017

1. **Write a program to create two threads. Print “In main thread” in main thread and “In child thread” in child thread.**

Ans :

```
class MyThread extends Thread

{

        MyThread(){

                System.out.println("In child thread");

        }

}

class Main

{

     public static void main(String args[]){

                System.out.println("In main thread");

                MyThread ob=new MyThread();

        }

}
```



```
[be2317@localhost Assign-2]$ javac Q1.java
[be2317@localhost Assign-2]$ java Q1
In main thread
In child thread
```

2. Create two threads and call them EvenThread and OddThread. EvenThread will print number as

2 4 6 8 10... and Odd Thread will print number as 1 3 5.... Now, synchronize these two threads

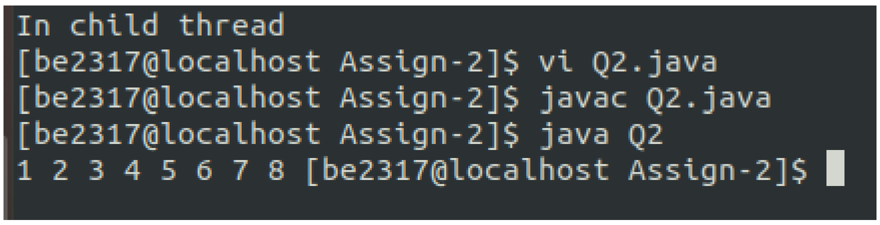
to get the output as:

1 2 3 4 5 6 7 8.

Ans:

```
class PrintNumbers {

        private static int counter = 1;

        private static final int MAX = 8;


        private static final Object lock = new Object();


        static class EvenThread extends Thread {

        public void run() {

        while (counter <= MAX) {

                synchronized (lock) {

                if (counter % 2 == 0) {

                System.out.print(counter + " ");

                counter++;

                }

                lock.notify();

                try {

                if (counter <= MAX) {

                        lock.wait();

                }

                } catch (InterruptedException e) {

                e.printStackTrace();

                }

                }
        }
        }
        }

        static class OddThread extends Thread {
        public void run() {
        while (counter <= MAX) {
                synchronized (lock) {
                if (counter % 2 != 0) {
                System.out.print(counter + " ");
                counter++;
                }
                lock.notify();
                try {
                if (counter <= MAX) {
                        lock.wait();
                }
                } catch (InterruptedException e) {
                e.printStackTrace();
                }
                }
        }
        }
        }
```

```
        public static void main(String[] args) {
        EvenThread evenThread = new EvenThread();
        OddThread oddThread = new OddThread();

        evenThread.start();
        oddThread.start();

        try {
        evenThread.join();
        oddThread.join();
        } catch (InterruptedException e) {
        e.printStackTrace();
        }
        }
}
```



```
In child thread
[be2317@localhost Assign-2]$ vi Q2.java
[be2317@localhost Assign-2]$ javac Q2.java
[be2317@localhost Assign-2]$ java Q2
1 2 3 4 5 6 7 8 [be2317@localhost Assign-2]$
```

3. Consider the following series

x = 1+1/1! +1/2! + 1/3! + ...1/10!

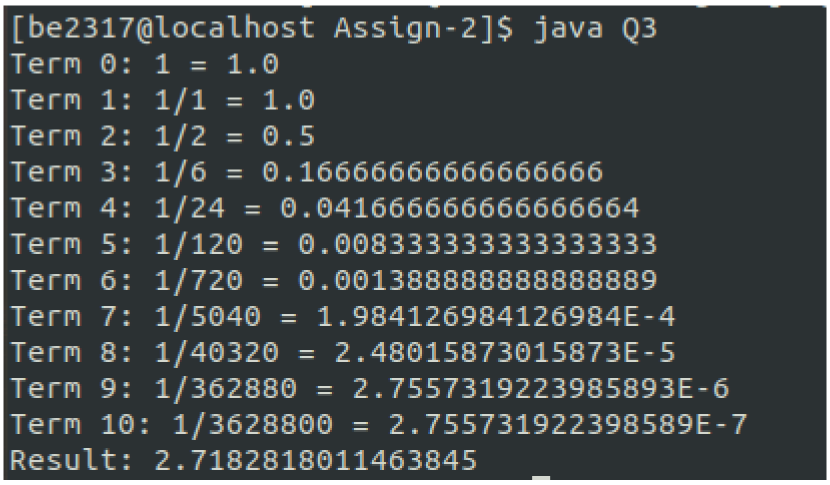
Create two threads t1 & t2. t1 will generate the denominators and t2 will form the term and add

them up. Finally print the result.

Ans:

```
public class SeriesSum {
        static int currentFactorial;
        static boolean ready = false;
        static final Object lock = new Object();

        public static void main(String[] args) {
        Thread t1 = new Thread(() -> {
        for (int i = 1; i <= 10; i++) {
                int fact = 1;
                for (int j = 2; j <= i; j++) fact *= j;

                synchronized(lock) {
                currentFactorial = fact;
                ready = true;
                lock.notify();
                while(ready) {
                try { lock.wait(); }
                catch (InterruptedException e) {}
                }
                }
        }
        });

        Thread t2 = new Thread(() -> {
```

```
        double sum = 1.0; // Initial value

        for (int i = 0; i < 10; i++) {

                synchronized(lock) {

                while(!ready) {

                try { lock.wait(); }

                catch (InterruptedException e) {}

                }

                sum += 1.0 / currentFactorial;

                System.out.println("Term " + (i+1) + ": 1/" + currentFactorial + " = " + 1.0
    / currentFactorial);

                ready = false;

                lock.notify();

                }

        }

        System.out.println("Result: " + sum);

        });


        System.out.println("Term 0: 1 = 1.0");

        t1.start();

        t2.start();

        }

}
```



```
[be2317@localhost Assign-2]$ java Q3
Term 0: 1 = 1.0
Term 1: 1/1 = 1.0
Term 2: 1/2 = 0.5
Term 3: 1/6 = 0.16666666666666666
Term 4: 1/24 = 0.041666666666666664
Term 5: 1/120 = 0.008333333333333333
Term 6: 1/720 = 0.001388888888888889
Term 7: 1/5040 = 1.984126984126984E-4
Term 8: 1/40320 = 2.48015873015873E-5
Term 9: 1/362880 = 2.7557319223985893E-6
Term 10: 1/3628800 = 2.755731922398589E-7
Result: 2.7182818011463845
```

4. Consider a file that contains a number of integers. Create two threads. Call them ‘producer’ and

‘consumer’ thread. Producer thread will be reading the integers from the file continuously while

consumer thread will add them up. Use proper synchronization mechanism if needed.

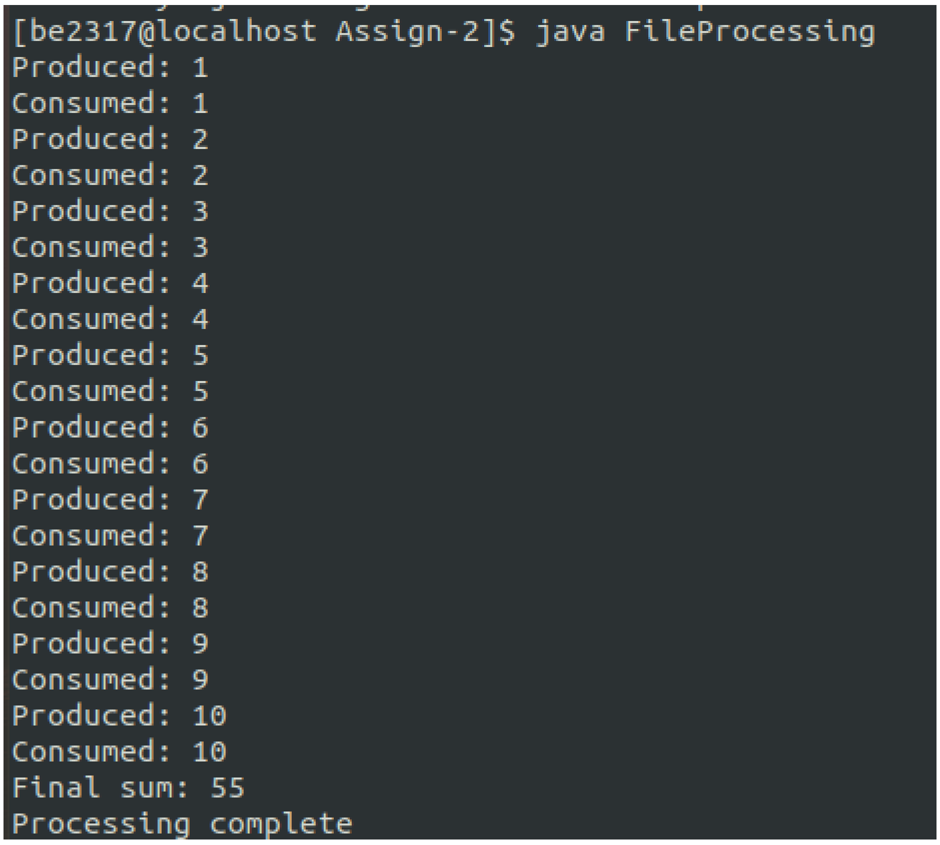
Ans:

```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.LinkedList;

import java.util.Queue;


class SharedBuffer {

        private Queue<Integer> buffer = new LinkedList<>();

        private final int CAPACITY = 5;

        private boolean isProducerDone = false;
```

```
synchronized void produce(int number) {

while (buffer.size() == CAPACITY) {

try {

        wait();

} catch (InterruptedException e) {

        Thread.currentThread().interrupt();

}

}

buffer.offer(number);

System.out.println("Produced: " + number);

notify();

}


synchronized Integer consume() {

while (buffer.isEmpty()) {

if (isProducerDone) {

        return null;

}

try {

        wait();

} catch (InterruptedException e) {

        Thread.currentThread().interrupt();

        return null;

}

}
```

```
        Integer number = buffer.poll();

        System.out.println("Consumed: " + number);

        notify();

        return number;

        }


        synchronized void setProducerDone() {

        isProducerDone = true;

        notifyAll();

        }

}


class Producer extends Thread {

        private SharedBuffer buffer;

        private String filename;


        Producer(SharedBuffer buffer, String filename) {

        this.buffer = buffer;

        this.filename = filename;

        }


        public void run() {

        try (BufferedReader reader = new BufferedReader(new
FileReader(filename))) {

        String line;
        while ((line = reader.readLine()) != null) {
                int number = Integer.parseInt(line.trim());
                buffer.produce(number);
        }
        buffer.setProducerDone();
        } catch (IOException | NumberFormatException e) {
        System.out.println("Error reading file: " + e.getMessage());
        buffer.setProducerDone();
        }
        }
}

class Consumer extends Thread {
        private SharedBuffer buffer;
        private int sum = 0;

        Consumer(SharedBuffer buffer) {
        this.buffer = buffer;
        }

        public void run() {
        while (true) {
        Integer number = buffer.consume();
        if (number == null) {
                break;
        }
        sum += number;
        }
        System.out.println("Final sum: " + sum);
        }

        public int getSum() {
        return sum;
        }
}

class FileProcessing {
        public static void main(String[] args) {
        SharedBuffer buffer = new SharedBuffer();
        Producer producer = new Producer(buffer, "numbers.txt");
        Consumer consumer = new Consumer(buffer);

        producer.start();
        consumer.start();

        try {
        producer.join();
        consumer.join();
        System.out.println("Processing complete");
        } catch (InterruptedException e) {
```

```
        System.out.println("Main thread interrupted");

        }

        }

}
```



```
[be2317@localhost Assign-2]$ java FileProcessing
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Produced: 7
Consumed: 7
Produced: 8
Consumed: 8
Produced: 9
Consumed: 9
Produced: 10
Consumed: 10
Final sum: 55
Processing complete
```

5. Consider the series 1+2+3+...+100.

This can be considered as (1+3+5+...+99)+(2+4+6+...+100). Create two threads to compute two

series in parallel (do not use simplified equation). Finally print the final sum.

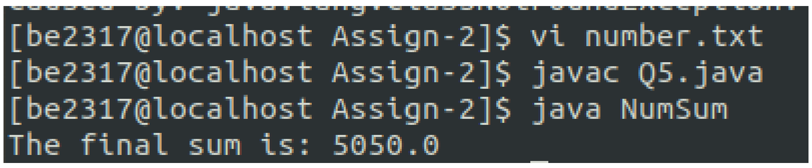
Ans:

```
class NumSum {

        public static double sum=0;

        public static boolean ready=false;
```

```
static final Object lock = new Object();

static class OddSum extends Thread{

public void run()
{   double os=0;
for (int i=1;i<=100;i+=2)
{
        os+=i;
}
synchronized(lock) {
        sum+=os;
        ready = true;
        lock.notify();
        while(ready) {
        try { lock.wait(); }
        catch (InterruptedException e) {}
        }
}
}
}

static class EvenSum extends Thread{

public void run()
```

```
{   double es=0;
for (int j=2;j<=100;j+=2)
{
        es+=j;
}
synchronized(lock) {
        while(!ready) {
        try { lock.wait(); }
        catch (InterruptedException e) {}
        }
        sum+=es;
        ready = false;
        lock.notify();
}
}
}

public static void main(String args[]){
OddSum o1=new OddSum();
EvenSum o2=new EvenSum();
o1.start();
o2.start();

try {
o1.join();
```

```
        o2.join();

        } catch (InterruptedException e) {

        e.printStackTrace();

        }


        System.out.println("The final sum is: " + sum);

        }

}
```



```
[be2317@localhost Assign-2]$ vi number.txt
[be2317@localhost Assign-2]$ javac Q5.java
[be2317@localhost Assign-2]$ java NumSum
The final sum is: 5050.0
```

6. Consider the following parallel binary search algorithm for series a1, a2...an sorted in increasing

order such that n mod 10 = 0. Element to be searched is e.

a) Create n/10 threads t1, t2,..,tn/10.

b) Distribute the numbers among threads such that ti will have numbers ai, ai+1, ...a2i-1.

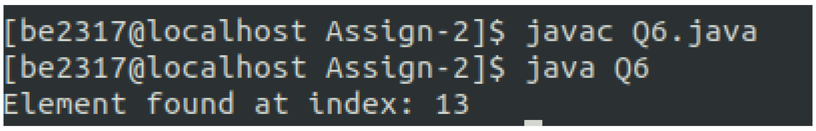
c) Distribute the element e to all threads.

d) Each thread searches the element e in its sub-array using binary search algorithm.

Ans :

```
import java.util.Arrays;


class ParallelBinarySearch extends Thread {

        private int[] subArray;
private int element;

private int startIndex;

private static volatile int foundIndex = -1;


public ParallelBinarySearch(int[] subArray, int element, int startIndex) {

this.subArray = subArray;

this.element = element;

this.startIndex = startIndex;

}


private int binarySearch(int[] arr, int target) {

int left = 0, right = arr.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) {

        return mid + startIndex;

} else if (arr[mid] < target) {

        left = mid + 1;

} else {

        right = mid - 1;

}

}

return -1;

}
```

```
@Override

public void run() {

if (foundIndex == -1) { // Only search if the element isn't found yet

int index = binarySearch(subArray, element);

if (index != -1) {

        foundIndex = index;

}

}

}


public static int parallelBinarySearch(int[] arr, int element) {

int n = arr.length;

int numThreads = n / 10;

ParallelBinarySearch[] threads = new ParallelBinarySearch[numThreads];


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

int start = i * 10;

int[] subArray = Arrays.copyOfRange(arr, start, start + 10);

threads[i] = new ParallelBinarySearch(subArray, element, start);

threads[i].start();

}


for (ParallelBinarySearch thread : threads) {

try {

        thread.join();
        } catch (InterruptedException e) {
                e.printStackTrace();
        }
        }
        return foundIndex;
        }

        public static void main(String[] args) {
        int[] arr = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19,
                21, 23, 25, 27, 29, 31, 33, 35, 37, 39,
                41, 43, 45, 47, 49, 51, 53, 55, 57, 59};
        int element = 27;

        int index = parallelBinarySearch(arr, element);

        if (index != -1) {
        System.out.println("Element found at index: " + index);
        } else {
        System.out.println("Element not found");
        }
        }
}
```



```
[be2317@localhost Assign-2]$ javac Q6.java
[be2317@localhost Assign-2]$ java Q6
Element found at index: 13
```