```
        // starting consumer thread
        new Consumer(q);

        // starting producer thread
        new Producer(q);
    }
}
```

Output :

iii) One form of communication in client-server system environment is Remote Method Invocation (RMI). RMI is a java feature similar to RPCs. RMI allow a thread to invoke a method on remote object. Objects are considered remote if they reside in a different java virtual machine (JVM). Demonstration RMI program for adding/subtracting/multiplying/dividing two numbers.

A) Server program :

```
import java.util.*;
import java.net.*;

class RPCServer {
        DatagramSocket ds;
        DatagramPacket dp;
```

```
String str, methodName, result;
int val1, val2;

RPCServer() {
        try {
                ds = new DatagramSocket(1200);
                byte b[] = new byte[4096];
                while (true) {
                        dp = new DatagramPacket(b, b.length);
                        ds.receive(dp);
                        str = new String(dp.getData(), 0, dp.getLength());
                        if (str.equalsIgnoreCase("q")) {
                                System.exit(1);
                        } else {
                                StringTokenizer st = new StringTokenizer(str, " ");
                                int i = 0;
                                while (st.hasMoreTokens()) {
                                        String token = st.nextToken();
                                        methodName = token;
                                        val1 = Integer.parseInt(st.nextToken());
                                        val2 = Integer.parseInt(st.nextToken());
                                }
                        }
                        System.out.println(str);
                        InetAddress ia = InetAddress.getLocalHost();
                        if (methodName.equalsIgnoreCase("add")) {
                                result = "" + add(val1, val2);
                        } else if (methodName.equalsIgnoreCase("sub")) {
                                result = "" + sub(val1, val2);
```

```
                        } else if (methodName.equalsIgnoreCase("mul")) {
                                result = "" + mul(val1, val2);
                        } else if (methodName.equalsIgnoreCase("div")) {
                                result = "" + div(val1, val2);
                        }
                        byte b1[] = result.getBytes();
                        DatagramSocket ds1 = new DatagramSocket();
                        DatagramPacket dp1 = new DatagramPacket(b1, b1.length,
InetAddress.getLocalHost(), 1300);
                        System.out.println("result : " + result + "\n");
                        ds1.send(dp1);
                }
        } catch (Exception e) {
                e.printStackTrace();
        }
}

public int add(int val1, int val2) {
        return val1 + val2;
}

public int sub(int val3, int val4) {
        return val3 - val4;
}

public int mul(int val3, int val4) {
        return val3 * val4;
}
```

```
    public int div(int val3, int val4) {
        return val3 / val4;
    }

    public static void main(String[] args) {
        new RPCServer();
    }
}
```

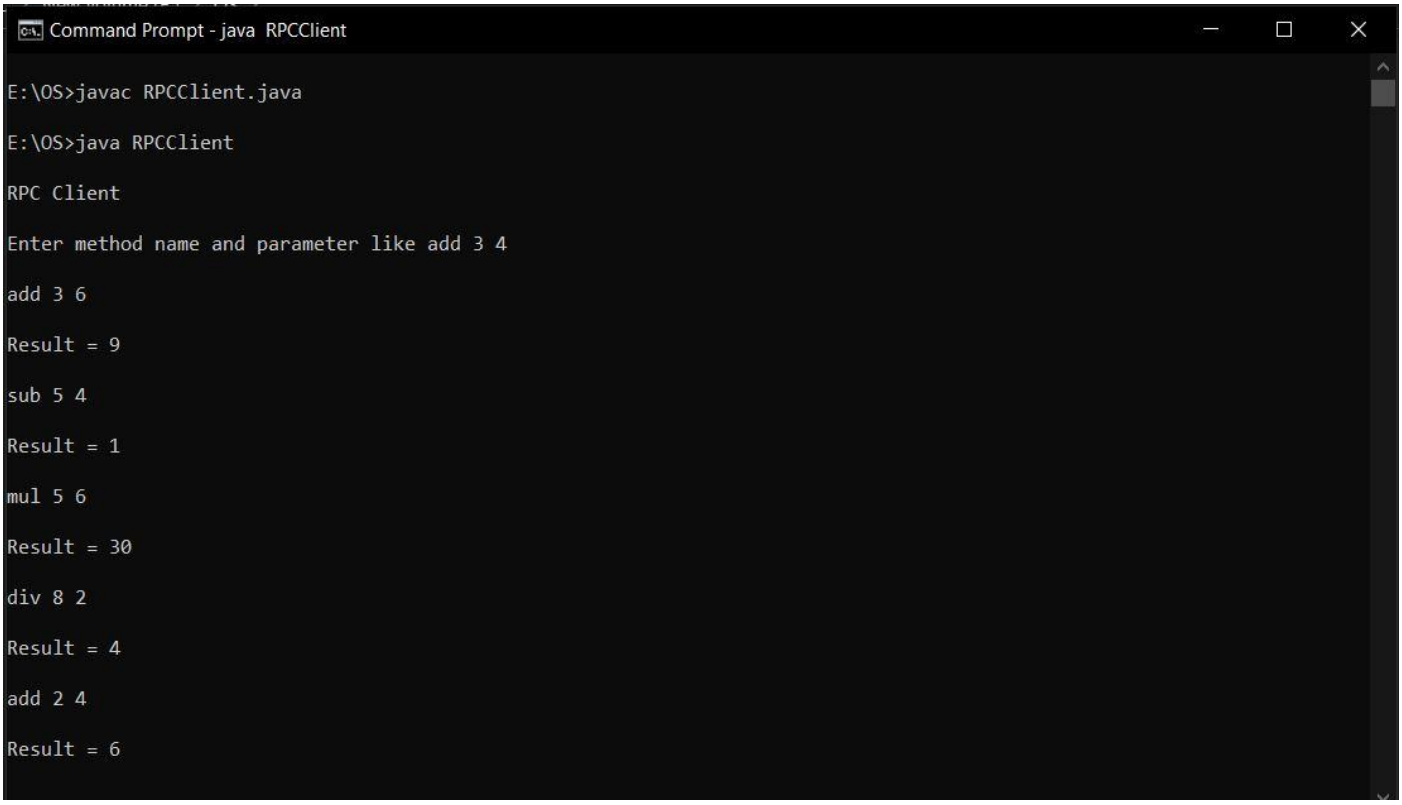
Output :

B) Client program:

```
import java.io.*;
import java.net.*;
class RPCClient
{
RPCClient()
    {
    try
    {
        InetAddress ia = InetAddress.getLocalHost();
        DatagramSocket ds = new DatagramSocket();
        DatagramSocket ds1 = new DatagramSocket(1300);
```

```
                System.out.println("\nRPC Client\n");
                System.out.println("Enter method name and parameter like add 3 4\n");
                while (true)
                {
                        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                        String str = br.readLine();
                        byte b[] = str.getBytes();
                        DatagramPacket dp = new DatagramPacket(b,b.length,ia,1200);
                        ds.send(dp);
                        dp = new DatagramPacket(b,b.length);
                        ds1.receive(dp);
                        String s = new String(dp.getData(),0,dp.getLength());
                        System.out.println("\nResult = " + s + "\n");
                }
        }
        catch (Exception e)
        {
        e.printStackTrace();
        }
        }

public static void main(String[] args)
{
        new RPCClient();
}
}
```

Output :



Practical 2

2. Threads

i) The java version of a multithreaded program that determines the summation of a non-negative integer. The summation class implements the Runnable interface. Thread creation is performed by creating an object instance of the Thread class and passing the constructor a Runnable object.

```
import java.util.*;
class summation implements Runnable{
   public void run(){
      int sum = 0;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the value of n : ");
      int n = sc.nextInt();
      for(int i = 0; i <= n; i++){
         sum = sum + i;
      }
```

```
        System.out.println("Summation of number is : " + sum);
    }
}

public class ThreadExample1{
    public static void main(String [] args){
        summation sum = new summation();
        Thread t1 = new Thread(sum);
        t1.start();
    }
}
```

Output :

ii) Write a multithreaded java program that outputs prime numbers. This program should work as follows : The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.

```
import java.util.*;
class Prime extends Thread{
    int a;
    public Prime(int a){
        this.a = a;
    }
```

```
    public void run(){
        for(int i = 0; i < a; i++){
            int counter=0;
            for(int num =i; num>=1; num--)
                {
               if(i%num==0)
                    {
                      counter = counter + 1;
                    }
                }
                if(counter == 2){
                   System.out.println(i);
                }
        }
    }
}

public class PrimeExample {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number : ");
        int num = sc.nextInt();
        Prime p = new Prime(num);
        p.start();
    }
}
```

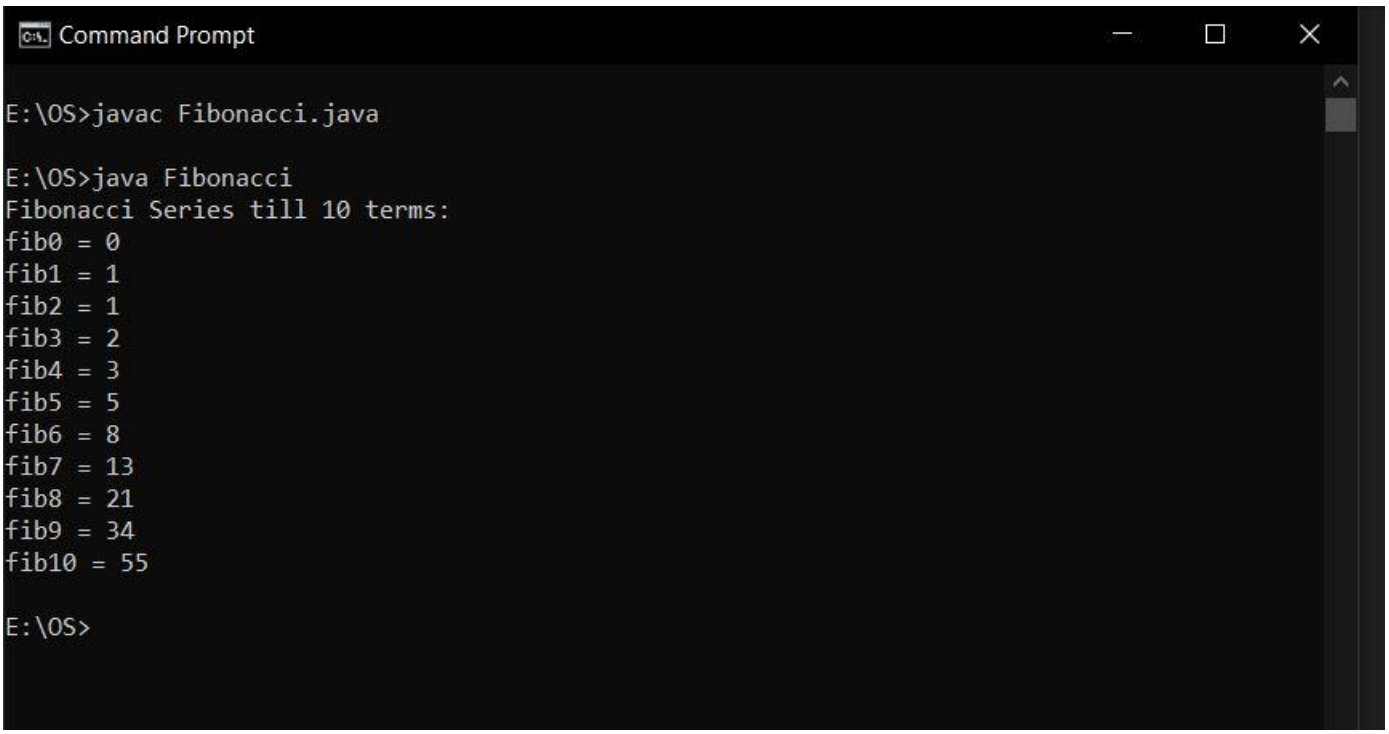
Output :

iii) The Fibonacci sequence is the series of numbers 0,1,1,2,3,5,8,… Formally, it can be expressed as : fib0 = 0, fib1 = 1, $fib_a = fib_{n-1} + fib_{n-2}$ Write a multithreaded program that generates the Fibonacci sequence using either the java.

```
class Fibo implements Runnable{
   @Override
   public void run(){
      int n = 10, firstTerm = 0, secondTerm = 1;
      System.out.println("Fibonacci Series till " + n + " terms:");

      for (int i = 0; i <= n; i++) {
        System.out.println("fib" + i + " = " + firstTerm);

        // compute the next term
        int nextTerm = firstTerm + secondTerm;
        firstTerm = secondTerm;
        secondTerm = nextTerm;
      }
   }
```

```
}

public class Fibonacci {
   public static void main(String[] args) {
      Fibo f1 = new Fibo();
      Thread t1 = new Thread(f1);
      t1.start();
   }
}
```

Output :



Practical 4

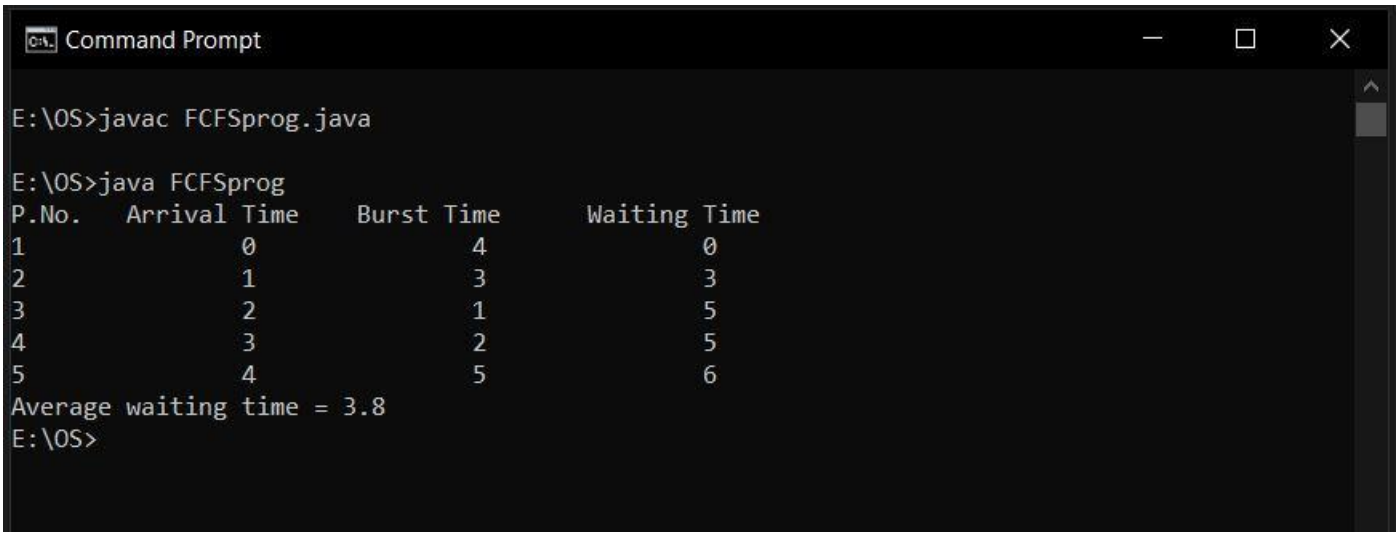
Implement FCFS scheduling algorithm in java.

```
class FCFSprog
{
static void CalculateWaitingTime(int at[], int bt[], int N)
```

```
{
        int []wt = new int[N];
        wt[0] = 0;
        System.out.print("P.No.\tArrival Time\t"
                + "Burst Time\tWaiting Time\n");

        System.out.print("1"
                + "\t\t" + at[0]+ "\t\t"
                + bt[0]+ "\t\t" + wt[0] +"\n");

        for (int i = 1; i < 5; i++) {
                wt[i] = (at[i - 1] + bt[i - 1] + wt[i - 1]) - at[i];

                System.out.print(i + 1+ "\t\t" + at[i]
                        + "\t\t" + bt[i]+ "\t\t"
                        + wt[i] +"\n");
        }
        float average;
        float sum = 0;

        for (int i = 0; i < 5; i++) {
                sum = sum + wt[i];
        }
        average = sum / 5;
        System.out.print("Average waiting time = "
                + average);
}
```

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

        int N = 5;

        int at[] = { 0, 1, 2, 3, 4 };

        int bt[] = { 4, 3, 1, 2, 5 };

        CalculateWaitingTime(at, bt, N);
}
}
```

Output :



Practical 5

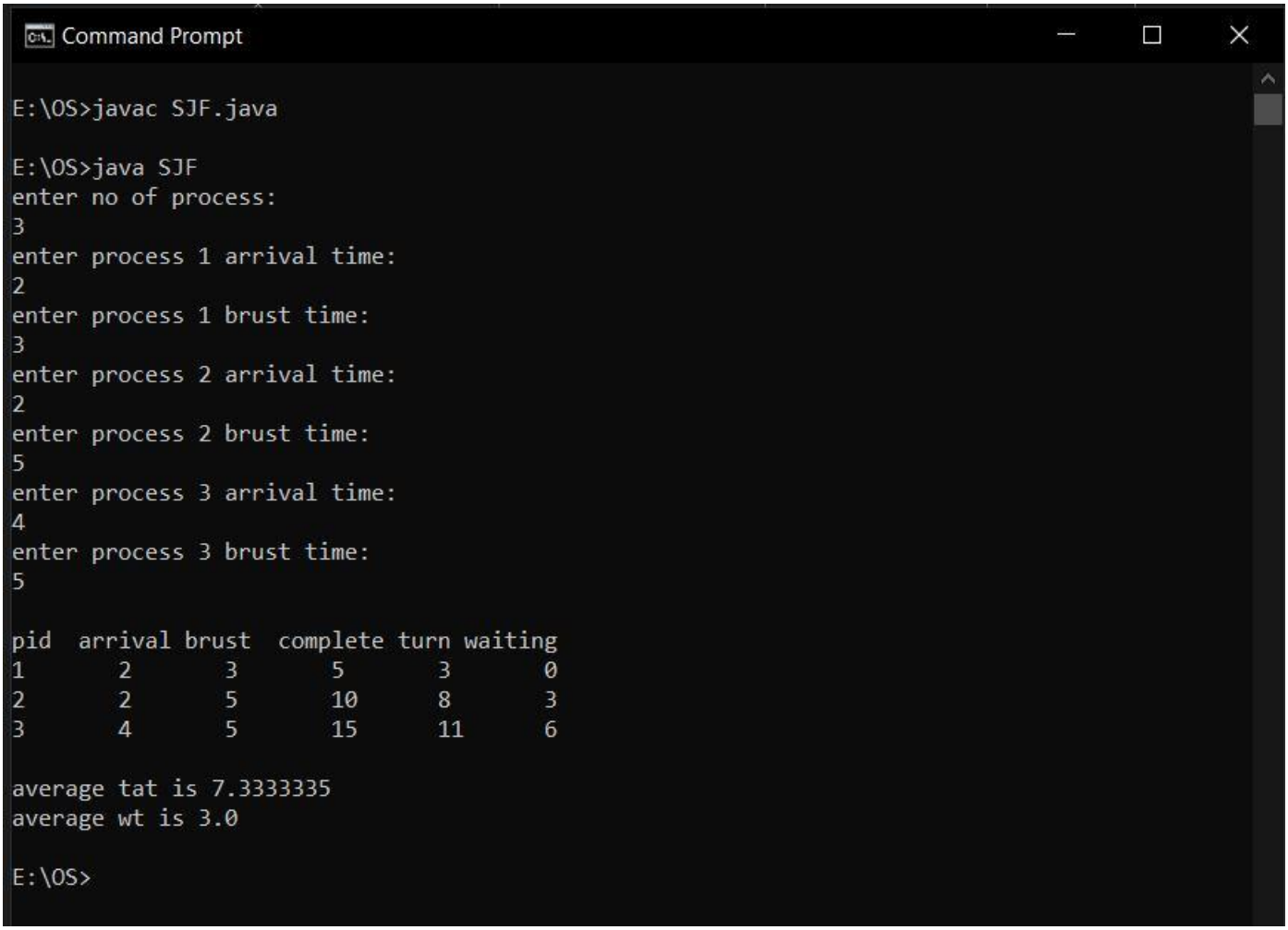
Implement SJF (with no pre-emption) scheduling algorithm in Java

```
import java.util.*;

public class SJF {
   public static void main(String args[]) {
```

```
Scanner sc = new Scanner(System.in);
System.out.println("enter no of process:");
int n = sc.nextInt();
int pid[] = new int[n];
int at[] = new int[n]; // at means arrival time
int bt[] = new int[n]; // bt means burst time
int ct[] = new int[n]; // ct means complete time
int ta[] = new int[n]; // ta means turn around time
int wt[] = new int[n]; // wt means waiting time
int f[] = new int[n]; // f means it is flag it checks process is completed or not
int st = 0, tot = 0;
float avgwt = 0, avgta = 0;

for (int i = 0; i < n; i++) {
   System.out.println("enter process " + (i + 1) + " arrival time:");
   at[i] = sc.nextInt();
   System.out.println("enter process " + (i + 1) + " brust time:");
   bt[i] = sc.nextInt();
   pid[i] = i + 1;
   f[i] = 0;
}
boolean a = true;
while (true) {
   int c = n, min = 999;
   if (tot == n) // total no of process = completed process loop will be terminated
      break;
   for (int i = 0; i < n; i++) {
      if ((at[i] <= st) && (f[i] == 0) && (bt[i] < min)) {
         min = bt[i];
```

```
                c = i;
            }
        }
        if (c == n)
            st++;
        else {
            ct[c] = st + bt[c];
            st += bt[c];
            ta[c] = ct[c] - at[c];
            wt[c] = ta[c] - bt[c];
            f[c] = 1;
            tot++;
        }
    }
    System.out.println("\npid  arrival brust  complete turn waiting");
    for (int i = 0; i < n; i++) {
        avgwt += wt[i];
        avgta += ta[i];
        System.out.println(pid[i] + "\t" + at[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" +
wt[i]);
    }
    System.out.println("\naverage tat is " + (float) (avgta / n));
    System.out.println("average wt is " + (float) (avgwt / n));
    sc.close();
  }
}
```

Output :



Practical 6

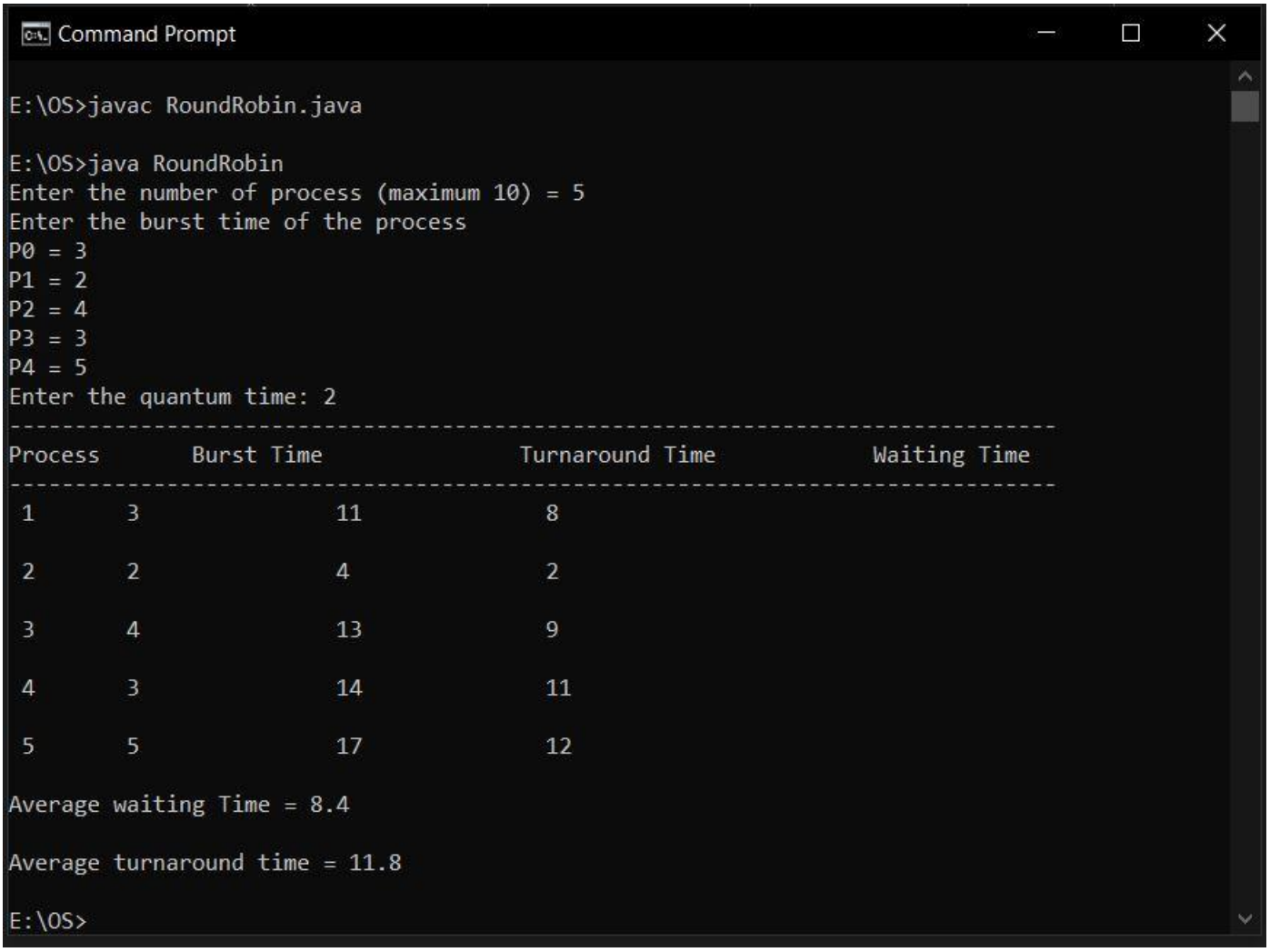
Implement RR scheduling algorithm in java

```
import java.util.Scanner;

public class RoundRobin {
   public static void main(String args[]) {
      int n, i, qt, count = 0, temp, sq = 0, bt[], wt[], tat[], rem_bt[];
      float awt = 0, atat = 0;
      bt = new int[10];
      wt = new int[10];
      tat = new int[10];
      rem_bt = new int[10];
```

```
Scanner s = new Scanner(System.in);
System.out.print("Enter the number of process (maximum 10) = ");
n = s.nextInt();
System.out.print("Enter the burst time of the process\n");
for (i = 0; i < n; i++) {
   System.out.print("P" + i + " = ");
   bt[i] = s.nextInt();
   rem_bt[i] = bt[i];
}
System.out.print("Enter the quantum time: ");
qt = s.nextInt();
while (true) {
   for (i = 0, count = 0; i < n; i++) {
      temp = qt;
      if (rem_bt[i] == 0) {
         count++;
         continue;
      }
      if (rem_bt[i] > qt)
         rem_bt[i] = rem_bt[i] - qt;
      else if (rem_bt[i] >= 0) {
         temp = rem_bt[i];
         rem_bt[i] = 0;
      }
      sq = sq + temp;
      tat[i] = sq;
   }
   if (n == count)
      break;
```

```
    }
    System.out.print("--------------------------------------------------------------------------------");
    System.out.print("\nProcess\t Burst Time\t Turnaround Time\t Waiting Time\n");
    System.out.print("--------------------------------------------------------------------------------");
    for (i = 0; i < n; i++) {
      wt[i] = tat[i] - bt[i];
      awt = awt + wt[i];
      atat = atat + tat[i];
      System.out.print("\n " + (i + 1) + "\t " + bt[i] + "\t\t " + tat[i] + "\t\t " + wt[i] + "\n");
    }
    awt = awt / n;
    atat = atat / n;
    System.out.println("\nAverage waiting Time = " + awt + "\n");
    System.out.println("Average turnaround time = " + atat);
  }
}
```

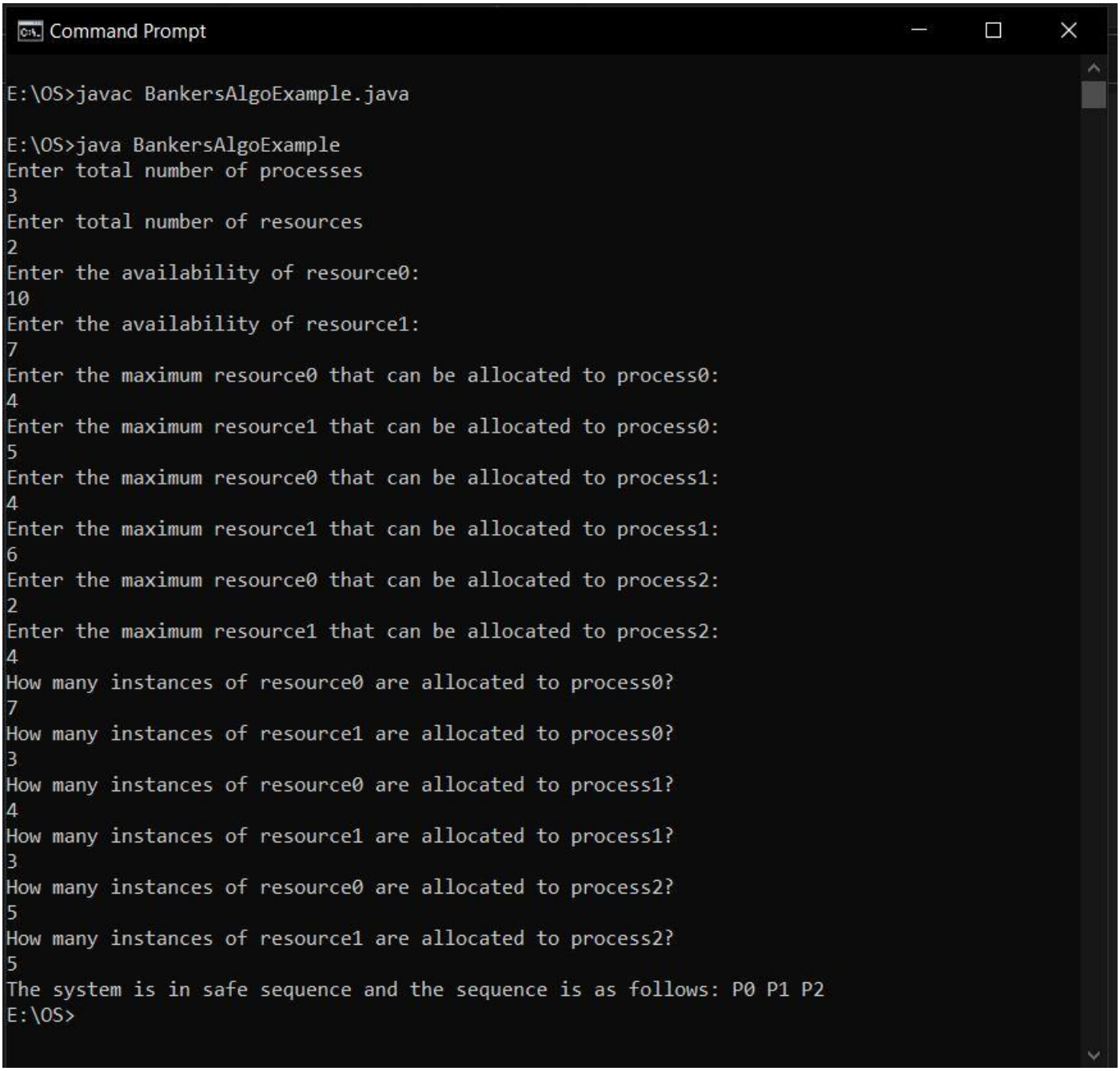
Output :



Practical 7

Write a java program that implements the banker's algorithm

```
// import required classes and packages
import java.util.*;
import java.io.*;
import java.util.Scanner;


class BankersAlgoExample
{


   static void findNeedValue(int needArray[][], int maxArray[][], int allocationArray[][], int totalProcess, int totalResources)
   {
```



Practical 8

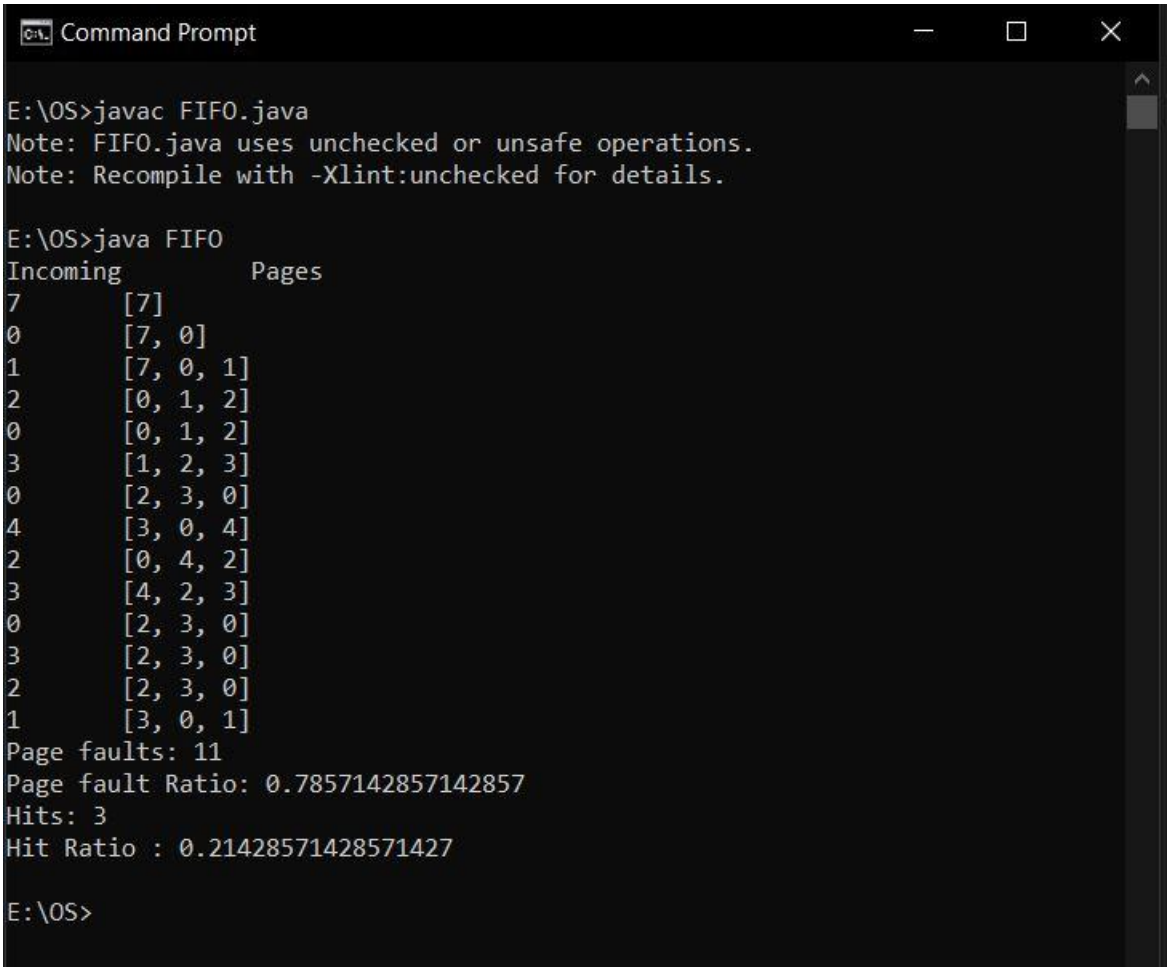
Write a java program that implements the FIFO page-replacement algorithm.

```
import java.util.HashSet;
import java.util.LinkedList;
import java.util.Queue;


public class FIFO
```

```
{
   static int pageFaults(int incomingStream[], int n, int frames)
   {
      System.out.println("Incoming \t Pages");

      HashSet s = new HashSet<>(frames);

      Queue queue = new LinkedList<>();

      int page_faults = 0;

      for (int i=0; i < n; i++)
      {
         if (s.size() < frames)
         {
            if (!s.contains(incomingStream[i]))
            {
               s.add(incomingStream[i]);
               page_faults++;
               queue.add(incomingStream[i]);
            }
         }

         else
         {
            if (!s.contains(incomingStream[i]))
            {
               int val = (int) queue.peek();
```

```
            queue.poll();

            s.remove(val);

            s.add(incomingStream[i]);

            queue.add(incomingStream[i]);
            page_faults++;
          }
        }
        System.out.print(incomingStream[i] + "\t");
        System.out.print(queue + " \n");
    }
    return page_faults;
}
public static void main(String args[])
{
    int incomingStream[] = {7, 0, 1, 2, 0 , 3, 0, 4, 2, 3, 0, 3, 2, 1};
    int frames = 3;

    int len = incomingStream.length;
    int pageFaults = pageFaults(incomingStream, len, frames);
    int hit = len - pageFaults;

    System.out.println("Page faults: " + pageFaults);
    System.out.println("Page fault Ratio: " + (double) pageFaults/len);
    System.out.println("Hits: " + hit);
    System.out.println("Hit Ratio : " + (double) hit/len);
}}
```

Output :


Command Prompt
E:\OS>javac FIFO.java
Note: FIFO.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
E:\OS>java FIFO
Incoming Pages
7 [7]
0 [7, 0]
1 [7, 0, 1]
2 [0, 1, 2]
0 [0, 1, 2]
3 [1, 2, 3]
0 [2, 3, 0]
4 [3, 0, 4]
2 [0, 4, 2]
3 [4, 2, 3]
0 [2, 3, 0]
3 [2, 3, 0]
2 [2, 3, 0]
1 [3, 0, 1]
Page faults: 11
Page fault Ratio: 0.7857142857142857
Hits: 3
Hit Ratio : 0.21428571428571427
E:\OS>