Name: Rohan Kamble

Class :SYBSC CS

PRN No :2020420004

Subject :Operating System Practical

(USCS303)

Practical No.1

Aim :

Write a multithreaded program that generates the Fibonacci sequence using either the Java.

Program code :

```
import java.io.*;
import java.util.*;
class Multi extends Thread
{ public void
run()
{
int n1=0,n2=1,n3,i,count=20;
Scanner input = new Scanner(System.in);
System.out.print("Enter Number for limit :");
count=input.nextInt();
System.out.print(n1+" "+n2);//printing 0 and 1

for(i=2;i<count;++i)
{
n3=n1+n2;
System.out.print("
"+n3); n1=n2; n2=n3;
}
} public static void main(String[ ]
args)
{
Multi t1=new Multi();
t1.start();
}
}
```

Output :

```
C:\Users\hp\Desktop\java files>javac Multi.java
C:\Users\hp\Desktop\java files>java Multi
Enter Number for limit :8
0 1 1 2 3 5 8 13
```

Practical No.2

Aim :

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.

Program Code :

```
import java.util.*;
class MyRunnable1 implements Runnable
{ public void run(
)
{
int num,sum=0;
Scanner input = new Scanner(System.in);
System.out.print("Enter Numbers(Negative Number to Quit) :");
while(true)
{
num=input.nextInt();
if(num<0) break;
sum +=num;
}
System.out.println( "Sum is : " +sum);
}
}
class Summation
{ public static void main( String[ ] args
)
{
MyRunnable1 r = new MyRunnable1( ) ;
Thread t = new Thread( r ) ;
System.out.println( t ) ;
t.start( ) ;
}
}
```

Output :

```
C:\Users\hp\Desktop\java files>javac Summation.java
C:\Users\hp\Desktop\java files>java Summation
Thread[Thread-0,5,main]
Enter Numbers(Negative Number to Quit) :12 12 11 5 -3
Sum is : 40
```

Practical No.3

Aim :

Write a multithreaded Java program that outputs prime number .Create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.

Program Code :

```
import java.util.*;
public class Prime
{ public static void main(String
args[])
{ int s1, s2, s3, flag = 0, i,
j;
Scanner s = new Scanner(System.in);
System.out.println ("Enter the lower limit :");
s1 = s.nextInt();
System.out.println ("Enter the upper limit :");
s2 = s.nextInt();
System.out.println ("The prime numbers in between the entered limits
are
:"); for(i = s1; i <= s2;
i++)
{ for( j = 2; j < i;
j++)
{ if(i % j ==
0)
{ flag =
0;
break;
}
else
{ flag =
1;
}
} if(flag ==
1) {
System.out.println(i);
}
}
}
}
```

Output :

```
C:\Users\hp\Desktop\java files>javac Prime.java
C:\Users\hp\Desktop\java files>java Prime
Enter the lower limit :
20
Enter the upper limit :
50
The prime numbers in between the entered limits are
: 23
29
31
37
41
43
47
```

Practical No.4

Aim :

Give Java solution to Bounded buffer problem.

Program Code :

```
import java.io.BufferedWriter;
import java.io.File; import
java.io.FileOutputStream; import
java.io.IOException; import
java.io.OutputStreamWriter;
import java.io.Writer; import
```

```
java.util.*; import java.lang.*;
import java.io.*;

public class writer
{ public void
writing()
{
try
{
File statText = new File("test.txt");
FileOutputStream is = new FileOutputStream(statText);
OutputStreamWriter osw = new OutputStreamWriter(is);
Writer w = new BufferedWriter(osw);
w.write("POTATO!!! IT IS A VEGETABLE");
w.close();
}
catch (IOException e)
{
System.err.println("Problem writing to the file test.txt");
}
} public static void
main(String[]args)
{ writer write = new
writer(); write.writing();
File f=new File("test.txt");
System.out.println(f.getName());
System.out.println(f.getPath());
System.out.println(f.canRead()?"Readable":"Not Readable");
System.out.println(f.exists()?"Exist":"Non Exist");
System.out.println(f.canWrite()?" " :" ");
System.out.println(f.length()+"bytes"); try (BufferedReader
br = new BufferedReader(new FileReader ("test.txt")))
{
String sCurrentLine; while ((sCurrentLine =
br.readLine()) != null)
{
System.out.println(sCurrentLine);
}

}catch (IOException e)
{
e.printStackTrace();
}
}
}
```

Output :

```
C:\Users\hp\Desktop\java files>javac
writer.java C:\Users\hp\Desktop\java files>java
writer test.txt test.txt
Readable
Exist

28bytes
POTATO!!! IT IS A VEGETABLE
```

Practical No.5

Aim :

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

Program Code :

```
import java.io.*;
class FIFO
{
public static void main(String args[]) throws IOException
{ int fifo[]=new
int[3]; int n;
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter the number of inputs
:"); n=Integer.parseInt(br.readLine()); int
inp[]=new int[n];
System.out.println("Enter the inputs:");
for(int i=0;i<n;i++)
inp[i]=Integer.parseInt(br.readLine()); System.out.println("—————
");
for(int
i=0;i<3;i++)
fifo[i]=-1; int
Hit=0; int
Fault=0; int j=0;
boolean check;
for(int i=0;i<n;i++)
{
check=false;
for(int k=0;k<3;k++)
if(fifo[k]==inp[i])
{
check=true;
Hit=Hit+1;
}
if(check==false)
{
fifo[j]=inp[i];
j++;
if(j>=3)
j=0;
Fault=Fault+1;
}
}
System.out.println("HIT:"+Hit+" FAULT;"+Fault);
}
}
```

Output :

```
C:\Users\hp\Desktop\java files>javac FIFO.java

C:\Users\hp\Desktop\java files>java FIFO
Enter the number of inputs :
5
Enter the inputs:
5
6
8
```

```
8 4
â??â??â??â??â?
? HIT:1 FAULT;4
```

Practical No.6

Aim :

Implement FCFS scheduling algorithm in Java.

Program Code :

```
import java.io.*;
class FCFS
{ public static void main(String args[]) throws
Exception
{ int
n,AT[],BT[],WT[],TAT[];
float AWT=0;
InputStreamReader isr=new
InputStreamReader(System.in); BufferedReader br=new
BufferedReader(isr);
System.out.println("Enter no of
process");
n=Integer.parseInt(br.readLine());
BT=new int[n];
WT=new int[n];
TAT=new int[n];
AT=new int[n];
System.out.println("Enter Burst time for each process\n************");
for(int i=0;i<n;i++)
{
System.out.println("Enter BT for process "+(i+1));
BT[i]=Integer.parseInt(br.readLine());
}
System.out.println("****************");
for(int i=0;i<n;i++)
{
System.out.println("Enter AT for process"+i);
AT[i]=Integer.parseInt(br.readLine());
}
System.out.println
("****************");
WT[0]=0;
for(int i=1;i<n;i++)
{
WT[i]=WT[i-1]+BT[i-1];
WT[i]=WT[i]-AT[i];
} for(int
i=0;i<n;i++)
{
TAT[i]=WT[i]+BT[i];
AWT=AWT+WT[i];
}
System.out.println(" PROCESS BT WT TAT ");
for(int i=0;i<n;i++)
{
System.out.println(" "+ i + " "+BT[i]+" "+WT[i]+" "+TAT[i]); }
AWT=AWT/n;
System.out.println
("****************");
System.out.println("Avg waiting time="+AWT+"\n****************");
}
}
```

Output :

```
C:\Users\hp\Desktop\java files>javac FCFS.java
C:\Users\hp\Desktop\java files>java FCFS
Enter no of process
3
Enter Burst time for each process
************
Enter BT for process 1
5
Enter BT for process 2
4
Enter BT for process 3
3
*****************
Enter AT for process0
3
Enter AT for process1
4
Enter AT for process2
5
*****************
PROCESS BT WT TAT
0 5 0 5
1 4 1 5
2 3 0 3
*****************
Avg waiting time=0.33333334
*****************
```

Practical No.7

Aim :

Implement SJF (with no preemption) scheduling algorithm in Java.

Program Code :

```
import java.util.*;
public class SHORTESTJOBFIRST
{ public static void
main(String[]args)
{
Scanner in = new Scanner(System.in);
System.out.print("Enter Number of Process: ");
int process = in.nextInt();
String name[] = new String[process];
int BurstT[] = new int[process]; int
ArrivalT[] = new int[process];
for(int x = 0; x<process; x++)
{
System.out.print("Name of process: ");
name[x] = in.next();
System.out.print("Enter BurstTime: ");
BurstT[x] = in.nextInt();
System.out.print("Enter ArrivalT: ");
ArrivalT[x] = in.nextInt();
}
System.out.print("\f");
System.out.println("******[USER INPUTTED VALUES]******");
System.out.println("============================================
= =====");
System.out.println("||\tPROCESS\t||\tBT\t||\tAT\t| |");
System.out.println("============================================
= =====");
for(int x=0;x!=process;x++) {
```

```
System.out.println("||\t" + name[x] + "\t||\t" + BurstT[x] + "\t||\t"
+ ArrivalT[x] + "\t||");
}
System.out.println("==========================================
= ======"); int
tempAT, tempBT;
String tempname;
boolean check = false;

do{ check=false; for (int i=0;i <
(process-1); i++){
if (BurstT[i] > BurstT[i+1]){

tempAT=ArrivalT[i];
ArrivalT[i]=ArrivalT[i+1];
ArrivalT[i+1]=tempAT;

tempBT=BurstT[i];
BurstT[i]=BurstT[i+1];
BurstT[i+1]=tempBT;

tempname= name[i];
name[i]= name[i+1];
name[i+1]=tempname;

check=true;
}
}
}while(check);
System.out.println("***[AFTER SORTING INPUTTED VALUES]****");
System.out.println("==========================================
= =====");
```

```
System.out.println("||\tPROCESS\t||\tBT\t||\tAT\t| |");
System.out.println("===========================================
= =====");
for(int x=0;x!=process;x++) {
System.out.println("||\t" + name[x] + "\t||\t" + BurstT[x] + "\t||\t"
+ ArrivalT[x] + "\t||");
}
System.out.println("===========================================
= ====="); System.out.println(); double sumBT = 0;
//nevermind this computing for the waiting time coz i'm still not finished
in this
System.out.println("Computing for Waiting Time: ");
for(int x=0;x!=process;x++)
{
System.out.println(name[x]+" = "+BurstT[x]+" "+ArrivalT[x]);
}
System.out.println("Turn Around Time(TAT) for each processes: ");
for(int x=0;x!=process;x++)
{
System.out.println(name[x]+" = "+BurstT[x]);
}


for(int x=0;x!=process;x++)
{
sumBT +=BurstT[x];
}
System.out.printf("Average Turn Around Time(ATAT): %5.2f
ms",(sumBT/process));
}
}
```

Output :

```
C:\Users\hp\Desktop\java files>java SHORTESTJOBFIRST
Enter Number of Process: 3
Name of process: input
Enter BurstTime: 4
Enter ArrivalT: 2
Name of process: storage
Enter BurstTime: 6
Enter ArrivalT: 4
Name of process: output
Enter BurstTime: 2
Enter ArrivalT: 6

******[USER INPUTTED VALUES]******
==================================================
= || PROCESS || BT || AT | |
==================================================
= || input || 4 || 2 ||
|| storage || 6 || 4 ||
|| output || 2 || 6 ||
==================================================
= ***[AFTER SORTING INPUTTED VALUES]****
==================================================
= || PROCESS || BT || AT | |
==================================================
= || output || 2 || 6 ||
|| input || 4 || 2 ||
|| storage || 6 || 4 ||
===================================================

Computing for Waiting Time:
```

```
output = 2 - 6
input = 4 - 2
storage = 6 - 4
Turn Around Time(TAT) for each processes:
output = 2
input = 4
storage = 6
Average Turn Around Time(ATAT): 4.00 ms
```