PG DAC–March 2023
C-DAC THIRUVANANTHAPURAM
OOPs WITH JAVA-- LAB 14

Q1. Program to creating multiple thread where each thread displays numbers from 1 to 10. Also display all running threads

MultipleThread.java | *MultipleThreadMain.java

```
1  package com.javaassignment141.main;
2
3  public class MultipleThread extends Thread {
4      @Override
5      public void run() {
6          try {
7              for(int i = 1;  i<=10; i++) {
8                  System.out.println("Multiple Thread says: " +i);
9                  Thread.sleep(2000);
10             }
11             System.out.println("Multiple Thread ends here");
12         }
13         catch(InterruptedException e) {
14             e.printStackTrace();
15         }
16     }
17 }
18
```

MultipleThread.java | *MultipleThreadMain.java

```
1  package com.javaassignment141.main;
2
3  public class MultipleThreadMain {
4
5      public static void main(String[] args) {
6          MultipleThread t1 = new MultipleThread();
7          t1.start();
8          try {
9              for(int i = 1;  i<=10; i++) {
10             System.out.println("Multiple Thread Main says: " +i);
11             Thread.sleep(2000);
12             }
13             System.out.println("Multiple Thread Main ends here");
14         }
15         catch(InterruptedException e) {
16             e.printStackTrace();
17         }
18     }
19 }
20
```

Console

```
<terminated> MultipleThreadMain [Java Application] D:\Eclipse\
Multiple Thread Main says: 1
Multiple Thread says: 1
Multiple Thread Main says: 2
Multiple Thread says: 2
Multiple Thread Main says: 3
Multiple Thread says: 3
Multiple Thread Main says: 4
Multiple Thread says: 4
Multiple Thread Main says: 5
Multiple Thread says: 5
Multiple Thread Main says: 6
Multiple Thread says: 6
Multiple Thread Main says: 7
Multiple Thread says: 7
Multiple Thread Main says: 8
Multiple Thread says: 8
Multiple Thread Main says: 9
Multiple Thread says: 9
Multiple Thread Main says: 10
Multiple Thread says: 10
Multiple Thread Main ends here
Multiple Thread ends here
```

Q2. Repeat Q1 using Runnable Interface

MultipleThread.java | MultipleThreadMain.java | RunnableThread.java | RunnableThreadMain.java

```java
package com.javaassignment142.main;

public class RunnableThread implements Runnable {
    @Override
    public void run() {
        try {
            for(int i = 1;  i<=10; i++) {
                System.out.println("RunnableThread says: " +i);
                Thread.sleep(2500);
            }
            System.out.println("RunnableThread ends here");
        }
        catch(InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

MultipleThread.java | MultipleThreadMain.java | RunnableThread.java | RunnableThreadMain.java

```java
package com.javaassignment142.main;

public class RunnableThreadMain {

    public static void main(String[] args) {
        Thread t1 = new Thread(new RunnableThread());
        t1.start();
        try {
            for(int i = 1;  i<=10; i++) {
            System.out.println("RunnableThread Main says: " +i);
            Thread.sleep(1500);
            }
            System.out.println("RunnableThread Main ends here");
        }
        catch(InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Console

<terminated> RunnableThreadMain [Java Application] D:\Eclipse\ec

```
RunnableThread Main says: 1
RunnableThread says: 1
RunnableThread Main says: 2
RunnableThread says: 2
RunnableThread Main says: 3
RunnableThread Main says: 4
RunnableThread says: 3
RunnableThread Main says: 5
RunnableThread says: 4
RunnableThread Main says: 6
RunnableThread Main says: 7
RunnableThread says: 5
RunnableThread Main says: 8
RunnableThread Main says: 9
RunnableThread says: 6
RunnableThread Main says: 10
RunnableThread says: 7
RunnableThread Main ends here
RunnableThread says: 8
RunnableThread says: 9
RunnableThread says: 10
RunnableThread ends here
```

Q3. Create a child thread to display the factorial of a number using Lambda expression.

*LambdaThread.java × LambdaThreadMain.java

```
1  package com.javaassignment143.main;
2
3  public class LambdaThread implements Runnable {
4      @Override
5      public void run() {
6      try {
7          int n = 5;int factorial = 1;
8          for(int i = 1;  i<=n; i++) {
9          factorial *= i;
10         System.out.println("Factorial of " + n + " is " +factorial);
11         Thread.sleep(1500);
12         }
13         System.out.println("LambdaThread ends here");
14         }
15         catch(InterruptedException e) {
16         e.printStackTrace();
17         }
18     }
19 }
20
21
```

LambdaThread.java *LambdaThreadMain.java ×

```
1  package com.javaassignment143.main;
2
3  public class LambdaThreadMain {
4
5      public static void main(String[] args) {
6      Thread t1 = new Thread (() -> {
7      try {
8          int n = 5; int factorial = 1;
9          for(int i = 1;  i<=n; i++) {
10         factorial *= i;
11         System.out.println("Factorial of " + n + " is " + factorial);
12         Thread.sleep(1500);     }
13         System.out.println("LambdaThread Main ends here");
14         }
15         catch(InterruptedException e) {
16             e.printStackTrace();    }
17     });
18     t1.start();
19     try {
20         int n = 5;int factorial = 1;
21         for(int i = 1;  i<=n; i++) {
22         factorial *= i;
23         System.out.println("Factorial of " + n + " is " +factorial);
24         Thread.sleep(1500);     }
25         System.out.println("LambdaThread ends here");
26         }
27         catch(InterruptedException e) {
28         e.printStackTrace();    }
29 }
30     }
31
```

Console ×

<terminated> LambdaThreadMain [Java Application] D:\Ec

```
Factorial of 5 is 1
Factorial of 5 is 1
Factorial of 5 is 2
Factorial of 5 is 2
Factorial of 5 is 6
Factorial of 5 is 6
Factorial of 5 is 24
Factorial of 5 is 24
Factorial of 5 is 120
Factorial of 5 is 120
LambdaThread Main ends here
LambdaThread ends here
```

Q4. Write a program to create two child threads, one to compute the first 25 prime numbers and the other to compute the first 50 fibonacci numbers. After calculating 25 fibonacci numbers, make that thread to sleep and take up prime number computation.

*PrimeandFibo.java × PrimeFibonacciMain.java

```
1  package com.javaassignment144.main;
2
3  public class PrimeandFibo{
4      public synchronized void prime(int num) {
5        try {
6         int i, j;
7         boolean isPrime;
8         for (i= 2; i < 25; i++) {
9         isPrime = true;
10        for (j = 2; j <= i/2; j++) {
11        if (i % j == 0) {
12        isPrime = false;
13        break;
14         }
15           }
16        if (isPrime) {
17        System.out.println(i+ " ");
18         }
19        notify();
20         Thread.sleep(1000);
21            }    }
22        catch(InterruptedException exc) {
23        exc.printStackTrace();
24 }       System.out.println("Prime ends here");
25    }
26         public synchronized void Fibonacci(int num) {
27         try {
28          int n1=0,n2=1,n3,i,count=50;
29          System.out.println(n1+""+n2);
30          for(i=2;i<count;++i)
31          {
32         n3=n2+n1;
```

Console ×

<terminated> PrimeFibonacciMain [Java Application] D:\Eclipse

```
The Fibo series are:
01
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
75025
The Prime number series are:
2
3
5
7
```

```
32         n3=n2+n1;
33         System.out.println(""+n3);
34         n1=n2;
35         n2=n3;
36
37       if (i== 25) {
38       wait();
39       System.out.println("The remaining Fibonacci series is
40   }
41     }
42        Thread.sleep(1000);
43      } catch (InterruptedException e) {
44        e.printStackTrace();
45    }
46        System.out.println("Fibonacci Series ends here.");
47     }
48 }
```

```
7
11
13
17
19
23
Prime ends here
The remaining Fibonacci series is:
121393
196418
317811
514229
832040
1346269
2178309
3524578
5702887
```

```java
package com.javaassignment144.main;

public class PrimeFibonacciMain {

    public static void main(String[] args) {
        PrimeandFibo ap = new PrimeandFibo();
        Thread t1 = new Thread(() -> ap.prime(100));
        Thread t2 = new Thread(() -> ap.Fibonacci(50));
        System.out.println("The Fibo series are:");
        t2.start();
        try {

            Thread.sleep(1000);
        }catch (InterruptedException e) {
            e.printStackTrace();
        }
    System.out.println("The Prime number series are:");
        t1.start();

    }
}
```

```
9227465
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
1134903170
1836311903
-1323752223
512559680
-811192543
Fibonacci Series ends here.
```

Q5. Create a BankAccount class with data members accno, balance and methods deposit and withdraw. Create an object for it which is a joint account. Two threads are using the same account. One thread is trying to deposit an amount to that account and second thread trying to withdraw an amount from it after checking the minimum balance. Implement the program using synchronization.

*BankAccount.java × JointAccountDemo.java Reflect.java

```java
package com.javaassignment145.main;

public class BankAccount {
    private int balance = 8000;

    public synchronized void withdraw(int amount) {
        System.out.println("going to withdraw, checking for sufficient balance");
        try {
            Thread.sleep(2000);
            if(balance < amount) {
                System.out.println("insufficient balance available, waiting for deposit");
            //  wait();
            }
            System.out.println("Sufficient balance available, updating the balance");
            Thread.sleep(2500);
            balance =balance- amount;
            System.out.println("Withdrawl Completed!!!");
        } catch(InterruptedException e) {
            e.printStackTrace();    }
        }
    public synchronized void deposit(int amount) {
        System.out.println("going to deposit, updating the balance");
        try {
            Thread.sleep(2000);
            balance =balance+ amount;
            System.out.println("Deposit Completed!!!");
        } catch(InterruptedException e) {
            e.printStackTrace();
        }
        }
    }

```

*BankAccount.java | JointAccountDemo.java | Reflect.java

```
1  package com.javaassignment145.main;
2
3  public class JointAccountDemo {
4      public static void main(String[] args) {
5          BankAccount acct = new BankAccount();
6          Thread t1 = new Thread(() -> acct.withdraw(14000));
7          Thread t2 = new Thread(() -> acct.deposit(10000));
8
9          t1.start();
10         try {
11             Thread.sleep(2500);
12         } catch(InterruptedException e) {
13             e.printStackTrace();
14         }
15         t2.start();
16     }
17 }
18
```

Console

```
<terminated> JointAccountDemo [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk
going to withdraw, checking for sufficient balance
insufficient balance available, waiting for deposit
Sufficient balance available, updating the balance
Withdrawl Completed!!!
going to deposit, updating the balance
Deposit Completed!!!
```

Q6. Write a Java program to display the name of a class, name of superclass, constructors, fields and methods using reflection

*Employee.java | AppMain.java

```
1  package com.javaassignment146.main;
2  import java.io.Serializable;
3
4  public class Employee implements Serializable {
5      private int empno;
6      private String empname;
7      public Employee(int empno,String empname) {
8          this.empno = empno;
9          this.empname = empname;
10     }
11     public Employee() {
12         empno= 34;          }
13     public int getEmpno() {
14         return empno;           }
15     public void setEmpno(int empno) {
16         this.empno = empno;         }
17     public String getEmpname() {
18         return empname;
19     }
20     public void setEmpname(String empname) {
21         this.empname = empname;
22     }
23 }
24
```

*AppMain.java

```
1  package com.javaassignment146.main;
2  import java.lang.reflect.Constructor;
5
6  public class AppMain {
7      public static void main(String[] args) throws ClassNotFoundException {
8          Employee emp = new Employee();
9          Class obj = emp.getClass();
10
11         System.out.println("Emp refernece belong to class: " + obj.getName());
12         System.out.println("Class name: " + obj.getSimpleName());
13
14         Class superClass = obj.getSuperclass();
15         System.out.println("Superclass of Employee is: " + superClass.getName());
16
17         Field[] empFields = obj.getFields();
18
19         for(Field f : empFields ){
20         f.setAccessible(true);
21         System.out.println(f.getName() + " : " + f.getType());
22         }
23
```

*AppMain.java

```
24         Constructor[] arr = obj.getConstructors();
25         for(Constructor c : arr) {
26         System.out.println("Constructor has : "
27                     + c.getParameterCount());
28
29         if(c.getParameterCount() > 0) {
30         Class[] paramTypes = c.getParameterTypes();
31
32         for(Class param : paramTypes )
33         System.out.println(param.getName());
34     }
35         }
36         System.out.println("Methods:");
37         Method[] methods = obj.getDeclaredMethods();
38         for (Method method : methods) {
39             System.out.println(method.toString());
40         }
41     }
42 }
43
```

Console

<terminated> AppMain [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\

```
Emp refernece belong to class: com.javaassignment146.main.Employee
Class name: Employee
Superclass of Employee is: java.lang.Object
Constructor has : 2
int
java.lang.String
Constructor has : 0
Methods:
public int com.javaassignment146.main.Employee.getEmpno()
public void com.javaassignment146.main.Employee.setEmpno(int)
public java.lang.String com.javaassignment146.main.Employee.getEmpname()
public void com.javaassignment146.main.Employee.setEmpname(java.lang.String)
```