PG-DAC THIRUVANANTHAPURAM OOPs WITH JAVA

Q1. Write a program to accept your name print Hello your name.
i) From Console
ii) Using buffered reader.
iii) Using Scanner

```java
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;


public class PrintName {
Run | Debug
public static void main(String[] args) throws IOException {
    // Using buffered reader to input data from user
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter your name: ");
        String myname = reader.readLine(); // Store the input in the 'myname' variable
        System.out.println("Hello, " + myname);

        // Using Console to input data from user
        String name = System.console().readLine();
        System.out.println("Your name is " + name);

        // Using Scanner to input data from user
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String name1 = scanner.nextLine();
        System.out.println("My name is " + name1);
        scanner.close();
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignment/day9/NameS
canner$  /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Co
de/User/workspaceStorage/7828303b2bf78dff9f4b51634b610e20/redhat.java/jdt_ws/NameScanner_4e078c
1b/bin PrintName
Enter your name: Nisha
Hello, Nisha
Nisha
Your name is Nisha
Enter your name: Nisha
My name is Nisha
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignment/day9/NameS
canner$
```

Q2. Write a program to serialize the Employee object with id, name and dept. Create 2 objects for it and store it in a file and then deserialize it and print the details.

```java
import java.io.*;
class Employee implements Serializable {
    private int id;
    private String name;
    private String dept;

    public Employee(int id, String name, String dept) {
        this.id = id;
        this.name = name;
        this.dept = dept;
    }
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getDept() {
        return dept;
    }
    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ", dept=" + dept + "]";
    }
}
```

```java
public class SerializationQuestion {
    Run | Debug
    public static void main(String[] args) {
        Employee employee1 = new Employee(id:1, name:"Nisha", dept:"HR");
        Employee employee2 = new Employee(id:2, name:"Jerry", dept:"IT");
        // Serialization
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("employee.ser"))) {
            oos.writeObject(employee1);
            oos.writeObject(employee2);
            System.out.println("Employee objects serialized successfully.");
            // Print the serialized employee objects
            System.out.println("Serialized Employee 1: " + employee1);
            System.out.println("Serialized Employee 2: " + employee2);
        } catch (IOException e) {
            e.printStackTrace();
        } // Deserialization
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("employee.ser"))){
            Employee deserializedEmployee1 = (Employee) ois.readObject();
            Employee deserializedEmployee2 = (Employee) ois.readObject();

            System.out.println("Deserialized Employee 1: " + deserializedEmployee1);
            System.out.println("Deserialized Employee 2: " + deserializedEmployee2);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignment/day9/Seria
l$  cd /media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignment/day9/Serial ; /usr/bi
n/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspac
eStorage/d019aeac223d6610c31393dd2836fac4/redhat.java/jdt_ws/Serial_e73a49cc/bin SerializationQ
uestion
Employee objects serialized successfully.
Serialized Employee 1: Employee [id=1, name=Nisha, dept=HR]
Serialized Employee 2: Employee [id=2, name=Jerry, dept=IT]
Deserialized Employee 1: Employee [id=1, name=Nisha, dept=HR]
Deserialized Employee 2: Employee [id=2, name=Jerry, dept=IT]
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignment/day9/Seria
l$
```

Q3. Create a class named Department with data members deptid and deptname. Create another class . Employee with data members empid,empname and Department. Create an object of Employee and make a shallow copy of it. Change the values in the copy and observe the behavior.

```java
class Department {
    int deptid;
    String deptname;

    public Department(int deptid, String deptname) {
        this.deptid = deptid;
        this.deptname = deptname;
    }
}

class Employee implements Cloneable {
    int empid;
    String empname;
    Department department;

    public Employee(int empid, String empname, Department department) {
        this.empid = empid;
        this.empname = empname;
        this.department = department;
    }

    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}
```

```java
public class ShallowCopy_latest {
    Run | Debug
    public static void main(String[] args) {
        Department hr = new Department(deptid:1, deptname:"HR");
        Employee employee1 = new Employee(empid:101, empname:"Nisha", hr);
        Employee employee2 = null;

        try {
            employee2 = (Employee) employee1.clone();
        } catch (CloneNotSupportedException cns) {
            cns.printStackTrace();
        }

        System.out.println("Employee1 Department Name: " + employee1.department.deptname + ", Name: " + employee1.empname);

employee2.department.deptname = "IT";
System.out.println("After the changes by employee2: Employee1 Department Name: " + employee1.department.deptname + ", Name: " + employee1.empname);
    }
}
```

```
y$ javac ShallowCopy_latest.java
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignmen
t/day9/ShallowCopy$ java ShallowCopy_latest
Employee1 Department Name: HR, Name: Nisha
After the changes by employee2: Employee1 Department Name: IT, Name: Nisha
```

Q4. Create a class named Department with data members deptid and deptname.
Create another class Employee with data members empid, empname and Department. Create an object of Employee and make a deep copy of it. Change the values in the copy and observe the behavior.

```java
class Department implements Cloneable {
    private int deptid;
    private String deptname;

    public Department(int deptid, String deptname) {
        this.deptid = deptid;
        this.deptname = deptname;
    }

    // Getter and setter methods for deptid and deptname

    public int getDeptid() {
        return deptid;
    }

    public void setDeptid(int deptid) {
        this.deptid = deptid;
    }

    public String getDeptname() {
        return deptname;
    }

    public void setDeptname(String deptname) {
        this.deptname = deptname;
    }

    @Override
    public Department clone() {
        try {
            return (Department) super.clone();
        } catch (CloneNotSupportedException e) {
            throw new AssertionError("Clone not supported for Department");
        }
    }
}
```

```java
class Employee implements Cloneable {
    private int empid;
    private String empname;
    private Department department;

    public Employee(int empid, String empname, Department department) {
        this.empid = empid;
        this.empname = empname;
        this.department = department;
    }

    // Getter and setter methods for empid and empname
    public int getEmpid() {
        return empid;
    }

    public void setEmpid(int empid) {
        this.empid = empid;
    }

    public String getEmpname() {
        return empname;
    }

    public void setEmpname(String empname) {
        this.empname = empname;
    }

    public Department getDepartment() {
        return department;
    }

    public void setDepartment(Department department) {
        this.department = department;
    }
```

```java
    @Override
    public Employee clone() {
        try {
            Employee clonedEmployee = (Employee) super.clone();
            clonedEmployee.department = department.clone(); // Deep copy the Department object
            return clonedEmployee;
        } catch (CloneNotSupportedException e) {
            throw new AssertionError("Clone not supported for Employee");
        }
    }
}
public class DeepCopyQuestion {
    Run | Debug
    public static void main(String[] args) {
        // Create a Department
        Department originalDepartment = new Department(deptid:1, deptname:"HR");

        // Create an Employee with the original department
        Employee originalEmployee = new Employee(empid:101, empname:"John", originalDepartment);

        // Create a deep copy of the Employee
        Employee clonedEmployee = originalEmployee.clone();

        // Modify the cloned Employee's details
        clonedEmployee.setEmpid(empid:102);
        clonedEmployee.setEmpname(empname:"Jerry");
        clonedEmployee.getDepartment().setDeptid(deptid:2);
        clonedEmployee.getDepartment().setDeptname(deptname:"IT");

        // Print original and cloned Employee
        System.out.println("Original Employee: " + originalEmployee.getEmpid() + ", " + originalEmployee.getEmpname() +
                originalEmployee.getDepartment().getDeptid() + ", " + originalEmployee.getDepartment().getDeptname());

        System.out.println("Cloned Employee: " + clonedEmployee.getEmpid() + ", " + clonedEmployee.getEmpname() + ", " +
                clonedEmployee.getDepartment().getDeptid() + ", " + clonedEmployee.getDepartment().getDeptname());
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignmen
t/day9/DeepCopy$  cd /media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assign
ment/day9/DeepCopy ; /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -c
p /home/nisha/.config/Code/User/workspaceStorage/ea6cdf677ae5823cec7809d8ce983c13/r
edhat.java/jdt_ws/DeepCopy_d5c38059/bin DeepCopyQuestion
Original Employee: 101, John, 1, HR
Cloned Employee: 102, Jerry, 2, IT
```

Q5. Program to creating multiple thread where each thread displays numbers from 1 to 10.

```java
public class NumberPrinter implements Runnable {
    private final int threadNumber;

    public NumberPrinter(int threadNumber) {
        this.threadNumber = threadNumber;
    }

    @Override
    public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println("Thread " + threadNumber + ": " + i);
        }
    }

    Run | Debug
    public static void main(String[] args) {
        int numThreads = 5; // You can change this to the number of threads you want
        Thread[] threads = new Thread[numThreads];

        for (int i = 0; i < numThreads; i++) {
            threads[i] = new Thread(new NumberPrinter(i + 1));
            threads[i].start();
        }

        for (int i = 0; i < numThreads; i++) {
            try {
                threads[i].join(); // Wait for all threads to finish
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignment/day9/1-10$  /usr/bin/env /usr/lib/jvm/java-8-o
penjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspaceStorage/16b2f1889e840a1a208cd06c4c0172bd/redhat.java/jdt_ws/1-
10_7a2ee2f3/bin NumberPrinter
Thread 2: 1
Thread 2: 2
Thread 1: 1
Thread 2: 3
Thread 2: 4
Thread 2: 5
Thread 2: 6
Thread 2: 7
Thread 2: 8
Thread 2: 9
Thread 2: 10
Thread 3: 1
Thread 3: 2
Thread 3: 3
Thread 3: 4
Thread 3: 5
Thread 3: 6
Thread 3: 7
Thread 3: 8
Thread 3: 9
Thread 3: 10
Thread 5: 1
Thread 5: 2
Thread 5: 3
Thread 5: 4
Thread 5: 5
Thread 5: 6
Thread 5: 7
Thread 5: 8
Thread 5: 9
Thread 5: 10
Thread 4: 1
Thread 1: 2
Thread 1: 3
Thread 1: 4
Thread 1: 5
Thread 1: 6
Thread 1: 7
Thread 1: 8
Thread 1: 9
Thread 1: 10
Thread 4: 2
Thread 4: 3
Thread 4: 4
Thread 4: 5
Thread 4: 6
Thread 4: 7
Thread 4: 8
```

Q6. Repeat Q4 using Runnable Interface

```java
class Department {
    private int deptid;
    private String deptname;

    public Department(int deptid, String deptname) {
        this.deptid = deptid;
        this.deptname = deptname;
    }

    // Getters for deptid and deptname
    public int getDeptid() {
        return deptid;
    }

    public String getDeptname() {
        return deptname;
    }
}

class Employee implements Runnable {
    private int empid;
    private String empname;
    private Department department;

    public Employee(int empid, String empname, Department department) {
        this.empid = empid;
        this.empname = empname;
        this.department = department;
    }
```

```java
    // Getters for empid, empname, and department
    public int getEmpid() {
        return empid;
    }

    public String getEmpname() {
        return empname;
    }

    public Department getDepartment() {
        return department;
    }

    @Override
    public void run() {
        System.out.println("Employee ID: " + empid);
        System.out.println("Employee Name: " + empname);
        System.out.println("Department ID: " + department.getDeptid());
        System.out.println("Department Name: " + department.getDeptname());
    }
}

public class RunnableEmployee {
    Run | Debug
    public static void main(String[] args) {
        // Create a Department object
        Department hrDepartment = new Department(deptid:1, deptname:"HR");

        // Create Employee objects
        Employee employee1 = new Employee(empid:101, empname:"Arya", hrDepartment);
        Employee employee2 = new Employee(empid:102, empname:"Diya", hrDepartment);

        // Create threads and execute the Employee objects concurrently
        Thread thread1 = new Thread(employee1);
        Thread thread2 = new Thread(employee2);

        thread1.start();
        thread2.start();
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assign
ment/day9/Runnable$  cd /media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_
Assignment/day9/Runnable ; /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bi
n/java -cp /home/nisha/.config/Code/User/workspaceStorage/a4822e06f2e4460b044df7
ee1fde856d/redhat.java/jdt_ws/Runnable_e1c488f5/bin RunnableEmployee
Employee ID: 101
Employee Name: Arya
Department ID: 1
Department Name: HR
Employee ID: 102
Employee Name: Diya
Department ID: 1
Department Name: HR
```

Q7. Create a child thread to display the factorial of a number using Lambda expression.

FactorialThread.java > FactorialThread > main(String[])

```java
public class FactorialThread{
    Run | Debug
    public static void main(String[] args) {
        int number = 6;

        // Create a thread using a lambda expression
        Thread factorialThread = new Thread(() -> {
            long factorial = 1;
            for (int i = 1; i <= number; i++) {
                factorial *= i;
            }
            System.out.println("Factorial of " + number + " is " + factorial);
        });

        // Start the thread
        factorialThread.start();
    }
}
```

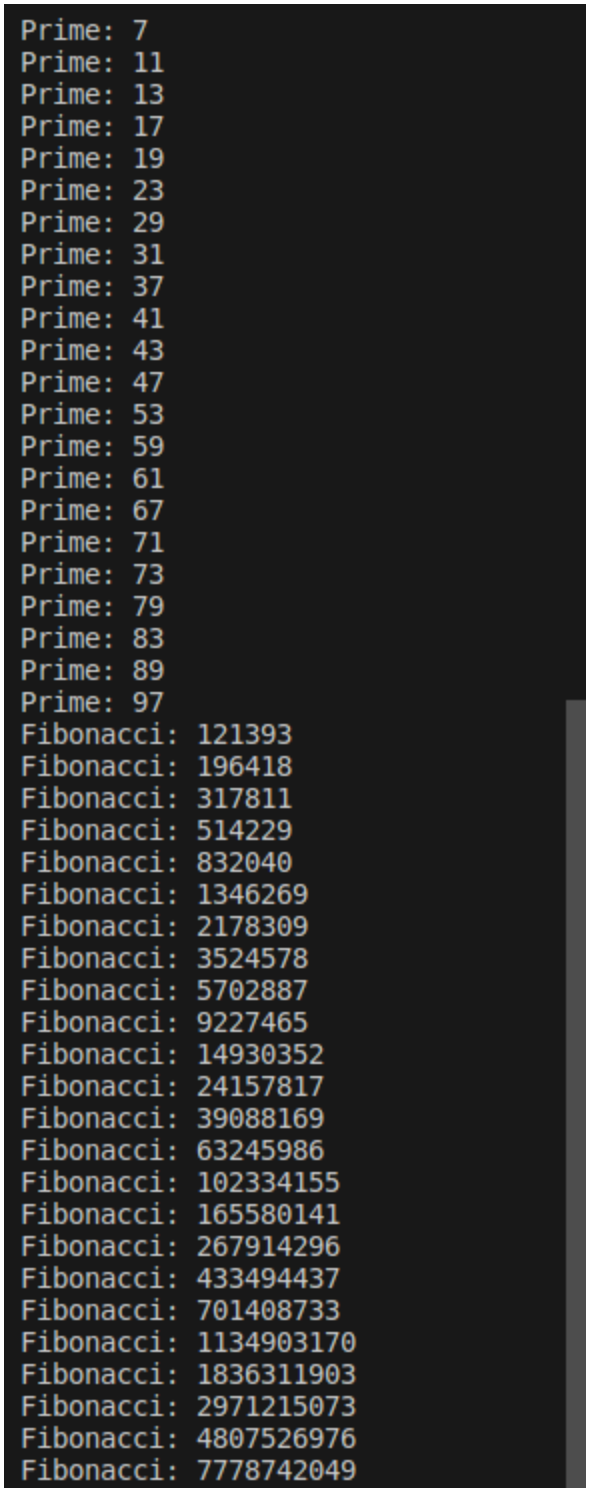
```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/
Java/Java_Assignment/day9/Fact$  /usr/bin/env /usr/lib/jvm/java-
8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/w
orkspaceStorage/a56ec80a3dea82dbeeda82b72d7f77dd/redhat.java/jdt
_ws/Fact_7a393844/bin FactorialThread
Factorial of 6 is 720
```

Q8. Write a program to create two child threads, one to compute the first 25 prime numbers and another to compute the first 50 fibonacci numbers. After calculating 25 fibonacci numbers, make that thread to sleep and take up prime number computation.

```java
class PrimeNumberThread extends Thread {
    @Override
    public void run() {
        int count = 0;
        int num = 2;

        while (count < 25) {
            boolean isPrime = true;
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }

            if (isPrime) {
                System.out.println("Prime: " + num);
                count++;
            }

            num++;
        }
    }
}
```

```java
class FibonacciNumberThread extends Thread {
    @Override
    public void run() {
        int n = 50;
        long[] fibonacci = new long[n];
        fibonacci[0] = 0;
        fibonacci[1] = 1;

        for (int i = 2; i < n; i++) {
            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];
            System.out.println("Fibonacci: " + fibonacci[i]);

            if (i == 25) {
                try {
                    System.out.println("Fibonacci thread sleeping...");
                    Thread.sleep(2000); // Sleep for 2 seconds
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

public class TwoThreads {
    Run | Debug
    public static void main(String[] args) {
        PrimeNumberThread primeThread = new PrimeNumberThread();
        FibonacciNumberThread fibonacciThread = new FibonacciNumberThread();

        primeThread.start();
        fibonacciThread.start();
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vert
ual_Box_Share/Nisha_Ubuntu/Cdac/
Java/Java_Assignment/day9/q8$  /
usr/bin/env /usr/lib/jvm/java-8-
openjdk-amd64/jre/bin/java -cp /
home/nisha/.config/Code/User/wor
kspaceStorage/a14ea6c92fc7e7baca
922f7e39e6772e/redhat.java/jdt_w
s/q8_e428099f/bin TwoThreads
Fibonacci: 1
Fibonacci: 2
Fibonacci: 3
Fibonacci: 5
Fibonacci: 8
Fibonacci: 13
Fibonacci: 21
Fibonacci: 34
Fibonacci: 55
Fibonacci: 89
Fibonacci: 144
Fibonacci: 233
Fibonacci: 377
Fibonacci: 610
Fibonacci: 987
Fibonacci: 1597
Fibonacci: 2584
Fibonacci: 4181
Fibonacci: 6765
Fibonacci: 10946
Fibonacci: 17711
Fibonacci: 28657
Fibonacci: 46368
Fibonacci: 75025
Fibonacci thread sleeping...
Prime: 2
Prime: 3
Prime: 5
Prime: 7
Prime: 11
Prime: 13
Prime: 17
Prime: 19
Prime: 23
Prime: 29
Prime: 31
Prime: 37
Prime: 41
Prime: 43
Prime: 47
Prime: 53
Prime: 59
```



Q9. 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.

```java
class BankAccount {
    private int accno;
    private double balance;

    public BankAccount(int accno, double balance) {
        this.accno = accno;
        this.balance = balance;
    }
    public synchronized void deposit(double amount) {
        System.out.println("Depositing " + amount);
        balance += amount;
        System.out.println("New balance after deposit: " + balance);
    }

    public synchronized void withdraw(double amount) {
        if (balance - amount >= 1000) { // Checking minimum balance
            System.out.println("Withdrawing " + amount);
            balance -= amount;
            System.out.println("New balance after withdrawal: " + balance);
        } else {
            System.out.println("Insufficient balance for withdrawal.");
        }
    }
}
public class JointAccount{
    Run | Debug
    public static void main(String[] args) {
        BankAccount jointAccount = new BankAccount(accno:12345, balance:5000.0);

        Thread depositThread = new Thread(() -> {
            jointAccount.deposit(amount:2000.0);
        });

        Thread withdrawThread = new Thread(() -> {
            jointAccount.withdraw(amount:3000.0);
        });

        depositThread.start();
        withdrawThread.start();
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/Java/Java_Assignment/day9/q9$  /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspaceStorage/694c0b329bf94ca615df4458c197de12/redhat.java/jdt_ws/q9_e42809a0/bin JointAccount
Depositing 2000.0
New balance after deposit: 7000.0
Withdrawing 3000.0
New balance after withdrawal: 4000.0
```