CO_2 Assignment

Atharva Jain
24200300092

4) Make 2 interfaces swimmable and flyable. Implement the behaviours in a bird class, demonstrating how a single class can support independent capabilities through interface.

Ans)

This program demonstrates how a single class "Bird" can have multiple independent capabilities by implementing the interfaces swimmable and flyable.

```
// Flyable.java
interface Flyable {
    void fly();
}

// Swimmable.java
interface Swimmable {
    void swim();
}

// Bird.java
class Bird implements Flyable, Swimmable {
    private String name;
    Bird(String name) {
        this.name = name;
    }
```

```
public void fly (){
  System.out.println (name + " is flying");
  }
public void swim (){
  System.out.println (name + " is swimming");
    }
  }

public class TestBird {
  public static void main (String [] args){
    Bird duck = new Bird ("Duck");
        duck.fly();
        duck.swim();
    }
}
```

(2) Represent workforce in a structured manner, create multilevel inheritance hierarchy, where person = base class, employee extends person, manager extends employee. Each class should add unique attribute

```
class Person {
    protected String name;
    public Person (String name) {
        this.name = name;
    }
    public void display () {
public class WorkForceHierarchy {
    public static void main (String[] args) {
        Person P1 = new Person ("John");
        em
        System.out.println(" Person name : " + name);
    }
}
class Employee extends Person
    protected String department;
    public Employee (String name, String
                     department) {
        super (name);
        this.department = department;
    }
@ Override
    public void display () {
    super.display ();
    System.out.println (" Department : " +
                        department);
    }
}
```

```
class Manager extends Employee {
  private int teamsize;
public Manager (String name, String department,
                int size) {
  super ( name, department);
  this . teamsize = teamsize;
}
@ Override
  public void display () {
    super . display ();
    System . out . println (" Manages a team of"
                            + teamsize);
        }
    }
public class Workforce {
    public static void main (String [] args){
      Manager m = new Manager ("Alice",
                               " IT ", 10);
          m . display ();
    }
}
```

Output →

```
Person name : Alice
Department : IT
Manges a team of 10
```

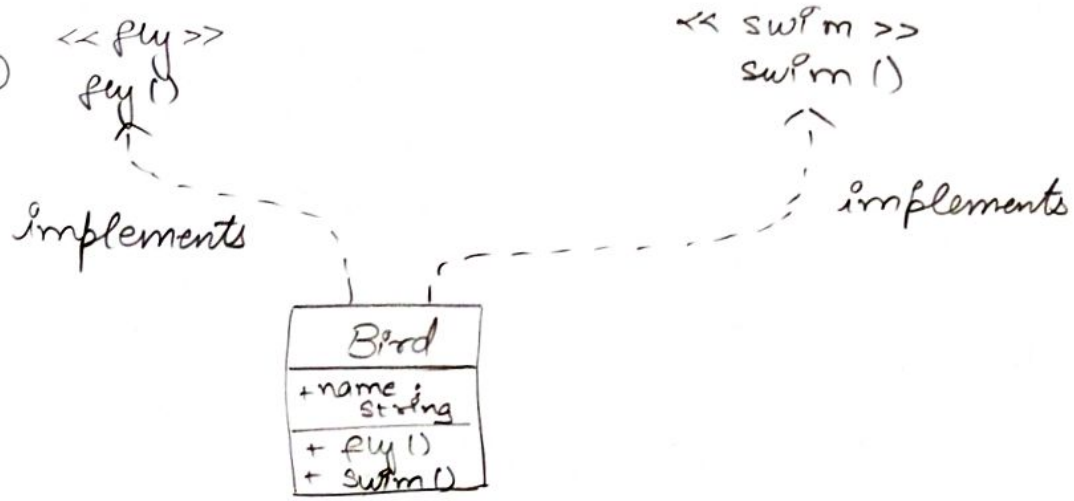
Q) Develop a Java program to illustrate single inheritance. Define a base class Vehicle and extend it with a subclass Car. Implement and Override a method to display relevant details, showing how inheritance allows method specialization.

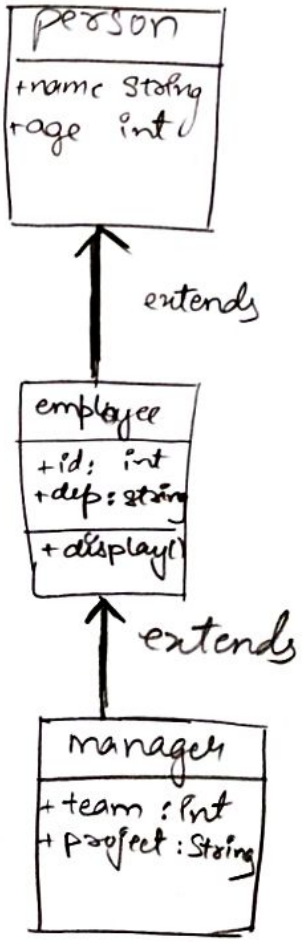
A)

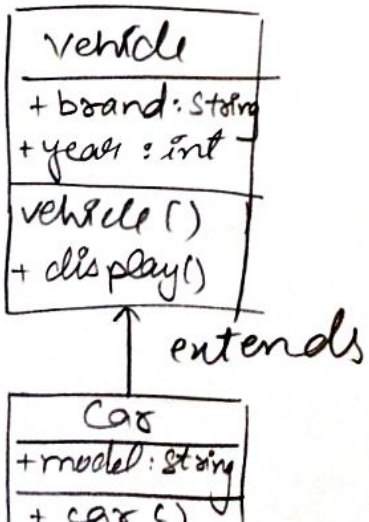
```
class Vehicle {
    String name;

    Vehicle (String name) {
        this.name = name
    }
    void display () {
    System.out.println (" The name is "+ name);
    }
}
class Car extends Vehicle {
    int speed;
Car ( String name, int speed) {
    super ( name);
    this.speed = speed;
}
@ Override
    void display () {
        super.display ();
        System.out.println( " Speed = "+ speed);
            }
    }
public class VehicleDemo {
    public static void main (String [] args) {
        Car c = new Car (" Toyota", 180);
        c.display ();
    }
}
```

Model a smart home appliance that supports 2 features : being switchable and being timer-operated. Define 2 interface : Switchable with a turnOn() method and TimerOperated with a setTimer() method. Implement both in a SmartFan class and demonstrate their usage by simulating turning the fan on and setting a timer

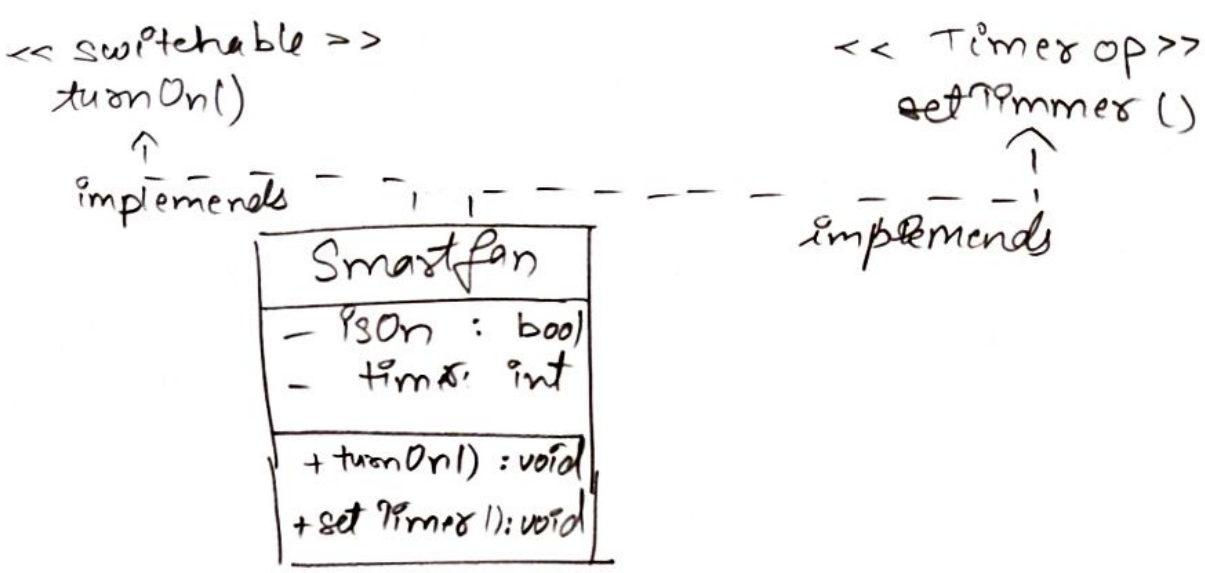
```
Inte  interface Switch {
        void turnOn();
    }
    interface Timer {
      void setTimer ( int min);
    }
class SFan implements Switch, Timer {
   @ Override
     public void turnOn(){
   System.out.println (" Sfan is ON");
    }
  @ Override
  public void setTimer() {
    System.out.println (" smart fan timer
                          set of " + min);
     }
  }
public class SFanDemo {
    public static void main( String [] args ){
      Sma SFan fan = new SFan ();
        fan. turnON();
        fan. setTimer (30);
      }
}
```

① 
<< fly >>
fly()
<< swim >>
swim()
implements
implements
Bird
+name String
+ fly()
+ swim()

② 
person
+name String
+age int
extends
employee
+id: int
+dep: String
+display()
extends
manager
+team : Int
+project : String

③ 
Vehicle
+ brand: String
+ year : int
vehicle()
+ display()
extends
Car
+model: String
+ car()

④


<< switchable >>
turnOn()
<< Timer op >>
setTimmer()
implements
implements
Smartfan
- isOn : bool
- time: int
+ turnOn() : void
+ setTimer(): void