**29th & 30th November 2021 - Batch 01 and Batch 02**

1. **Write a Programme to show the use Simple Inheritances (1 Level of Inheritances)**

* Create a package simpleinheritances
* Create a Class Car and declare following attributes, datatypes and visibility:

*private double speed;*

*private String modalName;*

*private String engineModal;*

*private String engineStatus;*

* Declare Constructor without parameter and assign the values

public Car(){

this.modalName = "";

this.engineModal = "";

this.speed=0.0;

this.engineStatus="";

}

* Declare Constructor with parameters and assign the values

public Car(String modalName, String engineModal) {

this.modalName = modalName;

this.engineModal = engineModal;

this.speed=0.0;

this.engineStatus="";

}

* Define methods start, stop, startStop, accelerate, deAccelerate, applyBrake

public void start(){

this.engineStatus="start Mode";

this.speed=0.0;

}

public void stop(){

this.engineStatus="stop Mode";

this.speed=0.0;

}

public void startStop(String s){

if(s.equalsIgnoreCase("stop")){

this.stop();

}else if (s.equalsIgnoreCase("start")){

this.start();

}

}

public void accelerate(){

speed=speed+3;

}

public void deaccelerate(){

speed=speed-2;

}

public void applyBrake(){

this.speed=0.0;

}

* Define getter methods to get / return the values of speed, engine modal, engine status and modal name.
* Define setter methods to set the values of engine modal and modal name.
* Define a display method to display the information about engine modal and modal name of the car.
* Create CarClient Class and within main method create object of Car Class. Call suitable methods and see if Program works or not.

Car one = new Car();

one.setModalName("Beetle");

one.setEngineModal("WB880X82");

//one.display();

Car two = new Car("POLO 550", "V580C82");

two.display();

two.startStop("start");

System.out.println("Engine Status :" + two.getEngineStatus());

two.accelerate();

two.accelerate();

System.out.println("Speed Status :" + two.getSpeed());

* **NOW,**

Create another Class FlyingCar and declare this FlyingCar class to extend the Car class. Add attribute specific to FlyingCar class i.e. flyingHeight.

* After use of the **extends** keyword, FlyingCar class will start inheriting the properties and behaviours of Car Class.
* Define constructors with and without parameters, similar to as defined in Car Class.
* super() is used as first statement inside the constructors to call the respective constructors of the super class i.e. Car Class. Parameters inside super method are used to send information from this class to super class.

public class FlyingCar **extends** Car{

private double flyingHeight;

public FlyingCar(){

super();

this.flyingHeight=0.0;

}

public FlyingCar(String modalName, String engineModal){

super(modalName, engineModal);

this.flyingHeight=0.0;

}

}

* Add method fly to FlyingCar class as only Flying Car can Fly.
* Add method ascend and descend to increase the height while flying.

public void fly(){

System.out.println("Flying");

}

public void ascend(){

flyingHeight=flyingHeight+2;

System.out.println("Flying Height :" + this.flyingHeight + "Meters");

}

* Since, the acceleration of Flying Car will be different from Car, Hence, you can redefine accelerate method of Car class in FlyingCar class using the concept called method overriding

public void accelerate(){

speed=speed+10;

}

* Similarly, override display method also as shown below

public void display(){

super.display();

System.out.println("Flying Height :"+ this.flyingHeight);

}

* **NOW,** Create FlyingCarClient Class to see the working of Simple Inheritance

Inside the main method create the objects of FlyingCar Class and call suitable methods to observe the working of simple Inheritance.

public static void main(String[] args) {

FlyingCar fc=null;

fc=new FlyingCar("FCT8967FGR", "TRFE6756UM");

fc.display();

fc.start();

fc.accelerate();

System.out.println("Speed :" + fc.getSpeed());

}

Simple Inheritance

Car

Car

FlyingCarClient

FlyingCar

CarClient

1. **Write a Programme to show the use Hierarchical Inheritances**

Car

Hierarchical Inheritance

WaterCar

FlyingCar

**NOW,** Similar to that ofFlyingCar Class, create another class WaterCar to extends Car class.

public class WaterCar extends Car{

private double floatingSpeed;

public WaterCar(){

super();

this.floatingSpeed=0.0;

}

public WaterCar(String modalName, String engineModal){

super(modalName, engineModal);

this.floatingSpeed=0.0;

}

public void swim(){

System.out.println("Swimming");

}

public void accelerate(){

speed=speed+3;

}

public void display(){

super.display();

System.out.println("Floating Speed"+ this.floatingSpeed);

}

}

* **NOW,** Create HierarchicalClient Class to see the working of Hierarchical Inheritance

public class HierarchicalClient {

public static void main(String[] args) {

Car fc = new FlyingCar("FC102", "ERE1");

cc = fc;

cc.display();

WaterCar wc = new WaterCar("Wc 103", "Wdc03");

//wc.display();

}

}

1. **Write a Programme to show the use Multilevel Inheritances**

Car

Multilevel Inheritance

FlyingCar

SpaceCar

public class SpaceCar extends FlyingCar{

private double airPressure;

public SpaceCar(){

super();

this.airPressure=0.0;

}

public SpaceCar(String modalName, String engineModal){

super(modalName, engineModal);

this.airPressure=0.0;

}

public void fly(){

System.out.println("Flying");

}

public void checkPressure(){

System.out.println("Checking Pressure");

}

public void accelerate(){

speed=speed+100;

}

public void display(){

super.display();

System.out.println("Airpressure"+ this.airPressure);

}

}

public class MulitlevelClient {

public static void main(String[] args) {

Car cc; // reference

Car c = new Car("AV1010", "QWT01");

//c.display();

FlyingCar fc = new FlyingCar("FC102", "ERE1");

//fc.display();

SpaceCar sc = new SpaceCar("Sp187i", "67Tw");

sc.display();

}

}