Task Five – Vehicle Management System

General Instructions

Create a **Java project** named **TaskFive**.

- Create a **Main** class that tests all required functionality.

Rules:

- Do **NOT** use any AI tools
- Project name must be **TaskFive**
- Upload you solution into GitHub Repository and submit the link.

Part A – Interface: `PriceAdmin`

Create an interface named **PriceAdmin** that contains:

```
public double calculatePrice();
```

Part B – Interface: `VehicleAdmin`

Create an interface named **VehicleAdmin** that contains:

```
public void describeCar();
```

Part C – Class: `Wheel`

Requirements

- Declare **private int** data field `size`
- Declare **private static String** data field `color`
- All wheels share the same color

Methods

- Public setters and getters for `size`
- Public static setters and getters for `color`

Part D – Class: `Engine`

Requirements

- Declare **private int** data field `powerEngine`
- Create public setter and getter methods

Part E – Package `q1_b` → Class: `Seat`

- Create a package named **q1_b**
- Inside it, create class **Seat**
- Implement an **empty default constructor** only

Part F – Class: `Vehicle`

Class **Vehicle** must:

- Implement **PriceAdmin** and **VehicleAdmin** interfaces

Data Fields

- `protected Wheel[] warr = new Wheel[4];`
- `protected Seat[] sarr = new Seat[5];`
- `protected Engine e;`

Constructor

- Create a **default constructor**
- Inside it:
- Initialize all elements of `warr` using a loop
- Initialize all elements of `sarr` using a loop
- Instantiate the `Engine` object

Methods

- Override `toString()` to return the **class name**

Part G – Class: `Car`

Class **Car** extends **Vehicle**.

Data Fields

- `protected static int carCount = 0;`
- `protected static final String[] manufacturerName = {"Toyota", "Nissan", "Honda", "Hondai"};`

Constructor

- Default constructor
- Increment `carCount` each time a car object is created

Methods

- Override `toString()`
- Return `super.toString()` concatenated with `"2019"`

Part H – Class: `VanCar`

Class **VanCar** extends **Car**.

Methods

1. `calculatePrice()`

- Base price: **35000 JD**
- Pricing rules:
- If `powerEngine <= 1500`
  - Tax = 14%
  - Add 1000 JD
- If `powerEngine == 2000`
  - Tax = 16%
  - Add 1500 JD
- Otherwise
  - Tax = 18%
  - Add 2000 JD

Return the **final price**.

2. `describeCar()`

- Print the calculated price

3. `equals(VanCar r)`

- Return `true` if:
- `r` is instance of `VanCar`
- Both cars have the **same price**
- Otherwise return `false`

Part I – Class: `SportCar`

Class **SportCar** extends **Car**.

Data Fields

- `private int modelYear;`
- `private int modelNumber;`

Constructors

- Default constructor
- One-argument constructor (`modelYear`)
- Two-argument constructor (`modelYear` , `modelNumber`)
- Use constructor chaining

Methods

1. `calculatePrice()`

- Base price: **2000 JD**
- Tax: **20%**

2. `describeCar()`

- Print `modelNumber` and `modelYear`

3. `equals(Object obj)`

- Return `true` if both sport cars have the **same engine power**
- Otherwise return `false`

Part J – Main Class

In the `main` method:

1. Print the **initial number of cars**
2. Declare `Car[] carArr = new Car[3];`

3. Assign:
4. `carArr[0] = new VanCar();`
5. `carArr[1] = new SportCar();`
6. `carArr[2] = new VanCar();`
7. Invoke `toString()` on the first element
8. Change the color of **all wheels** to `"gray"`

Additional Method in Main Class

Create a method:

```
public static int searchModel(String m)
```

- Search in `manufacturerName` array inside `Car`
- Use `equalsIgnoreCase()`
- Return index if found
- Return `-1` if not found

---

1. Invoke `searchModel("Honda")`
2. Print the **final number of cars**