Lập trình Java

Tuần 3: Lớp, đối tượng

Nội dung

- Khái niệm về đối tượng và lớp
- Định nghĩa được lớp và tạo đối tượng
- Định nghĩa các trường và các phương thức

Khái niệm về đối tượng

Biểu diễn đối tượng trong thế giới thực, ứng với mỗi đối tượng sẽ có các thuộc tính và hành vi riêng của nó

Đặc điểm và hành vi

Ví dụ:

Đối tượng: Xe hơi

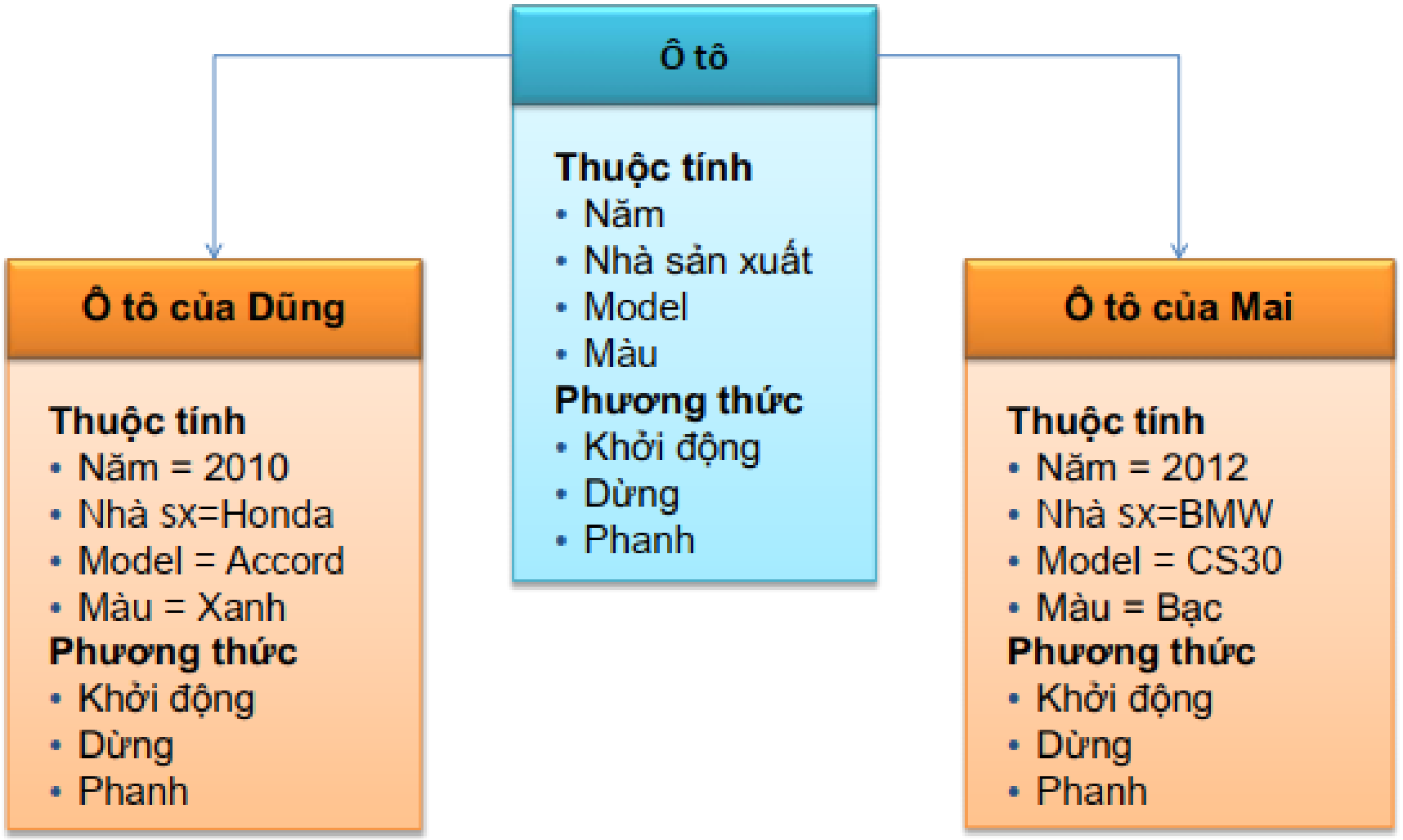
Đặc điểm: Hãng sản xuất, năm, màu,.. (Danh từ)

Hành vi: khởi động, dừng, bật đèn,.. (Động từ)

Định nghĩa lớp

- Lớp (Class) là một khuôn mẫu để mô tả các đối tượng cùng loại
- Ví dụ: Lớp Sinh Viên mô tả đối tượng là sinh viên bao gồm các thuộc tính như tên, mã số sinh viên, giới tính,..
- Lớp bao gồm các thuộc tính (các đặc điểm tiêu biểu của đối tượng) và các phương thức (các hành vi của đối tượng)

Mô hình lớp và đối tượng



Tính trừu tượng (Abstraction)

Thể hiện ở việc mô tả 1 thực thể trong cuộc sống thông qua lựa chọn các thuộc tính và hành vi đặc trưng mà không cần liệt kê tất cả các thuộc tính và hành vi.

Định nghĩa class

```
class <<ClassName>>
{
    <<type>> <<field1>>;
    ...
    <<type>> <<fieldN>>;

    <<type>> <<method1>>([parameters]) {
        // body of method
    }
    ...
    <<type>> <<methodN>>([parameters]) {
        // body of method
    }
}
```

Khai báo các trường

Khai báo các phương thức

Định nghĩa class

```
public class Employee{
    public String fullname;
    public double salary;

    public void input(){
        Scanner scanner = new Scanner(System.in);
        System.out.print(" >> Full Name: ");
        this.fullname = scanner.nextLine();

        System.out.print(" >> Salary: ");
        this.salary = scanner.nextDouble();
    }

    public void output(){
        System.out.println(this.fullname);
        System.out.println(this.salary);
    }
}
```

Trường

Phương thức

Lớp Employee có 2 thuộc tính là fullname và salary và 2 phương thức là input() và output()

Tạo đối tượng

Sau khi đã tạo class Employee, để tạo ra một nhân viên cụ thể ta sử dụng đoạn mã:

```
public static void main(String[] args) {

    Employee emp = new Employee();
    emp.input();
    emp.output();
}
```

Trong đó:

- emp là biến kiểu đối tượng Employee
- Toán tử new dùng để tạo đối tượng mới
- emp.input và emp.output bằng cách sử dụng dấu (.) để truy xuất các trường và phương thức có trong class.

Định nghĩa phương thức

Cú pháp

```
<<kiểu trả về>> <<tên phương thức>> ([danh sách tham số])
{
        // thân phương thức
}
```

Phương thức có thể có 1 hoặc nhiều tham số

Kiểu trả về có thể có không trả về giá trị nào (void)

Định nghĩa phương thức

```
public class Employee{
    public String fullname;
    public double salary;

    public void input(){…}
    public void output(){…}

    public void setInfo(String fullname, double salary) {
        this.fullname = fullname;
        this.salary = salary;
    }

    public double incomeTax(){
        if(this.salary < 5000000){
            return 0;
        }
        double tax = (this.salary - 5000000) * 10/100;
        return tax;
    }
}
```

Kiểu trả về là **void** nên thân phương thức **không chứa lệnh return giá trị**

Kiểu trả về là **double** nên thân phương thức phải chứa lệnh **return số thực**

Nạp chồng phương thức (Overloading)

Trong một class có thể có nhiều phương thức cùng tên nhưng khác tham số (kiểu, số lượng, thứ tự)

```
public class MyClass{
    void method(){...}
    void method(int x){...}
    void method(float x){...}
    void method(int x, double y){...}
}
```

Nạp chồng phương thức (Overloading)

```
class MayTinh{
    int tong(int a, int b){return a + b;}
    int tong(int a, int b, int c){return a + b + c;}
}
```

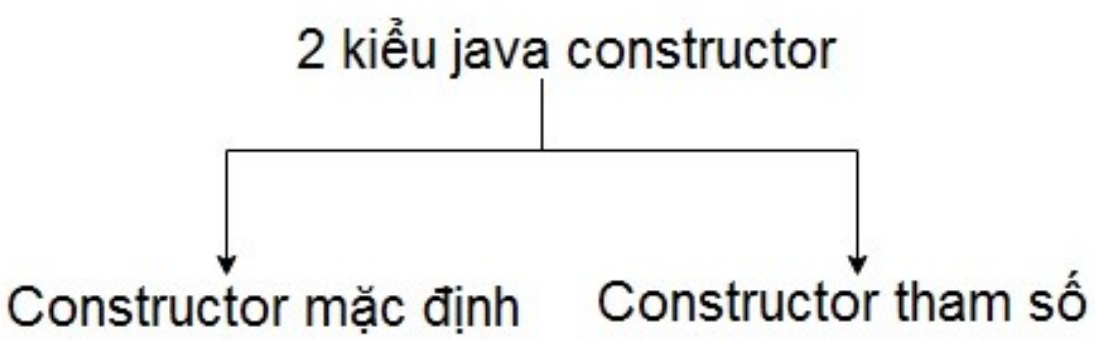
```
MayTinh mt = new MayTinh();
int t1 = mt.tong(5, 7);
int t2 = mt.tong(5, 7, 9);
```

Hàm tạo (Constructor)

- Constructor là một phương thức đặc biệt được sử dụng để tạo đối tượng.
- Đặc điểm:

➢ Tên constructor trùng với tên lớp

➢ Không trả lại giá trị



Hàm tạo (Constructor)

Ví dụ: Constructor tham số

Trong một lớp có thể định nghĩa nhiều hàm constructor khác tham số, mỗi hàm cung cấp 1 cách tạo đối tượng.

Ví dụ

Lớp

```
public class ChuNhat{
        double dai, rong;
        public ChuNhat(double dai, double rong){
                this.dai = dai;
                this.rong = rong;
        }
}
```

Đối tượng

```
ChuNhat cn1 = new ChuNhat(20, 15);
ChuNhat cn2 = new ChuNhat(50, 25);
```

Hàm tạo (Constructor)

```
public class ChuNhat{
        double dai, rong;
        public ChuNhat(double dai, double rong){
                this.dai = dai;
                this.rong = rong;
        }
        public ChuNhat(double canh){
                this.dai = canh;
                this.rong = canh;
        }
}
```

```
ChuNhat cn = new ChuNhat(20, 15);
ChuNhat vu= new ChuNhat(30);
```

Hàm tạo (Constructor)

- Nếu không khai báo hàm constructor thì Java sẽ tự động cung cấp hàm tạo mặc định (không tham số)
- Constructor mặc định cung cấp các giá trị mặc định như 0, null, (tùy thuộc vào kiểu dữ liệu) … tới đối tượng được khởi tạo.

Hàm tạo (Constructor)

```
class Student3{
    int id;
    String name;

    void display() {
        System.out.println(id+" "+name);
    }

    public static void main(String args[]) {
        Student3 s1=new Student3();
        Student3 s2=new Student3();
        s1.display();
        s2.display();
    }
}
```

```
0 null
0 null
```

Từ khóa this

- This được sử dụng để đại diện cho một đối tượng hiện tại, được sử dụng trong lớp để tham chiếu tới các thuộc tính và phương thức trong lớp (field và method)
- This.field để phân biết thuộc tính của các biến cục bộ hoặc tham số của phương thức.

```
public class MyClass{
    int field;
    void method(int field){
        this.field = field;
    }
}
```

Trường

Tham số

Package

- Package dùng để chia các class và các interface thành từng gói khác nhau, tương tự như quản lý file(ứng với class) và folder (ứng với package)
- Có nhiều package chia theo chức năng:

java.util: chứa các lớp tiện ích
java.io: chứa các lớp vào/ra dữ liệu
java.lang: chứa các lớp thường dùng...

Modifier

- Modifier được chia làm 2 nhóm:

  +Access Modifiers: kiểm soát mức độ truy cập
- +Non-Access Modifiers: không kiểm soát mức độ truy cập nhưng cung cấp các chức năng khác

Phạm vi truy cập (*access_modifier)*

- *Cú pháp*

<access_modifier> <ClassName> ([list of parameters])

- Access_modifier được sử dụng để định. Có 4 đặc tả khác nhau:

private: chỉ được phép sử dụng nội bộ trong class

public: công khai hoàn toàn

{default}:

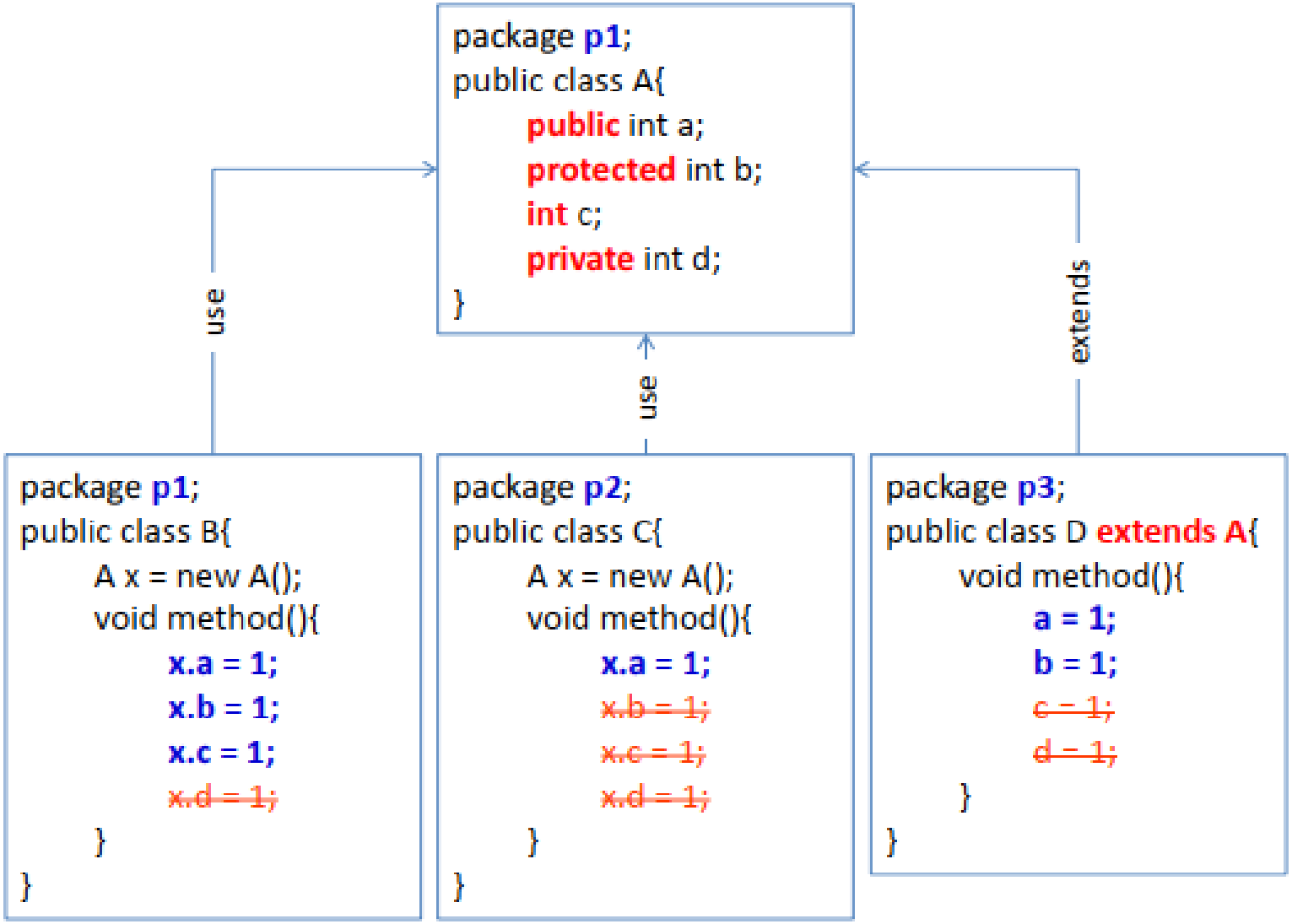
- Là public đối với các lớp truy xuất cùng gói
- Là private với các lớp truy xuất khác gói.

protected: tương tự {default} nhưng cho phép kế thừa dù lớp con và cha khác gói.

Phạm vi truy cập (*access_modifier)*

Mức độ bảo mật che giấu thông tin tang theo chiều mũi tên:

Phạm vi truy cập (*access_modifier*)



Non-Access Modifiers

- Có những Non-Access Modifier: final, static, abstract, synchronized, transient, volatile,...

  + Final: class(không được thừa kế), phương thức (không được overight), thuộc tính (hằng số)

  + Static: phương thức, thuộc tính sử dụng chung cho các đối tượng

  + Abstract (trừu tượng): class (phương thức bình thường và phương thức abstract), phương thức (chỉ có khởi tạo, không có thân hàm và kết thúc bằng dấu ;)

Từ khóa static

- Phương thức static có thể được truy cập trực tiếp không cần tạo đối tượng và có thể thay đổi giá trị của biến static.
- Biến static hay còn được gọi là biến tĩnh được sử dụng chung cho tất cả đối tượng trong class (ví dụ như tên trường học). Biến static lấy bộ nhớ một lần nên giúp tiết kiệm bộ nhớ.

```
public class Student{
    int id;
    String name;
    static String college = "US";
```

Từ khóa static

```
public class MyClass {
  // Static method
  static void myStaticMethod() {
    System.out.println("Phương thức static");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println("Phương thức public");
  }

  // Main method
  public static void main(String[ ] args) {
    myStaticMethod(); // Gọi phương thức static

    MyClass myObj = new MyClass(); // Tạo đối tượng từ lớp MyClass
    myObj.myPublicMethod(); // Gọi phương thức public
  }
}
```

Từ khóa static

```
public class Student{
    int id;
    String name;
    static String college = "US";

    static void change(){
    college = "QuocGia";
    }

    Student(int id, String name){
    this.id = id;
    this.name = name;
    }

    void display (){System.out.println(id+" "+name+" "+college);}

   public static void main(String args[]){
   Student.change();

   Student s1 = new Student (111,"AAAAA");
   Student s2 = new Student (222,"BBBBB");

   s1.display();
   s2.display();
   }  }
```

Tính đóng gói (encapsulation)

- Tính đóng gới thể hiện sự che dấu thông tin trong đối tượng, bao gồm che dấu trường dữ liệu và buộc phải dùng phương thức để truy xuất các trường dữ liệu.
- Tính đóng gói nhằm bảo vệ dữ liệu và tang cường khả năng mở rộng.
- Sử dụng private cho các trường dữ liệu, bổ sung các phương thức getter và setter để đọc và ghi các trường đã che dấu

Tính đóng gói (encapsulation)

```
private double diem;
```

```
public void setDiem(double diem){
    this.diem = diem;
}
public String getDiem(){
    return this.diem;
}
```

Tính đóng gói (encapsulation)

```
public class SinhVien{
        private String hoTen;
        private double diem;
        public void setHoTen(String hoTen){
                this.hoTen = hoTen;
        }
        public String getHoTen(){
                return this.hoTen;
        }
        public void setDiem(double diem){
                if(diem < 0 || > 10){
                        System.out.println("Điểm không hợp lệ");
                }
                else{
                        this.diem = diem;
                }
        }
        public String getDiem(){
                return this.diem;
        }
}
```

```
public class MyClass{
        public static void main(String[] args){
                SinhVien sv = new SinhVien();
                sv.setHoTen("Nguyễn Văn Tèo");
                sv.setDiem(20);
        }
}
```

Bài tập điểm danh

- Tạo một lớp Nhân viên gồm các thuộc tính:

  int maso;

  string ten;

  double luong;

  + Tạo contructor gồm 2 tham số (maso, ten)

  + Tạo Get, Set cho các thuộc tính

- Trong hàm main sử dụng contrutor để tạo 2 đối tượng nhân viên A, nhân viên B. Sau đó in ra màn hình: ten + maso + luong.

Qui tắc đặt tên trong java

Tên (class, field, method, package, interface, variable) được đặt theo qui ước (mềm) như sau:

- Tên package: toàn bộ ký tự thường và dấu chấm
  - java.util, com.poly
- Tên class, interface: Các từ phải viết hoa ký tự đầu
  - class **E**mployee{}, class **S**inh**V**ien{}, class **H**inh**C**hu**N**hat()
- Tên field, method, variable: Các từ phải viết hoa ký tự đầu ngoại trừ từ đầu tiên phải viết thường
  - **h**o**T**en, diem, **f**ull**N**ame, **m**ark
  - **s**et**H**o**T**en(), **i**nput(), **s**et**D**iem()

Tên class, field và variable sử dụng danh từ

Tên phương thức sử dụng động từ

Bài tập tuần 3

a. Create a super class called Car. The Car class has the following fields and methods.

- int speed;
- double regularPrice;
- String color;
- double getSalePrice();

b. Create a sub class of Car class and name it as Truck. The Truck class has the following fields and methods.

- int weight;
- double getSalePrice();//If weight>2000, 10% discount. Otherwise,20% discount.

c. Create a subclass of Car class and name it as Ford. The Ford class has the following fields and methods

- int year;
- int manufacturerDiscount;
- double getSalePrice();//FromthesalepricecomputedfromCarclass,subtractthemanufacturerDiscoun.

d. Create a subclass of Car class and name it as Sedan. The Sedan class has the following fields and methods.

- int length;
- double getSalePrice();//If length>20 feet, 5% discount, Otherwise, 10% discount.

e. Create MyOwnAutoShop class which contains the main() method. Perform the following within the main() method.

- Create an instance of Sedan class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the superclass.
- Create two instances of the Ford class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the super class.
- Create an instance of Car class and initialize all the fields with appropriate values.