**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG THÀNH PHỐ HỒ CHÍ MINH**

**KHOA HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**

****

**BÀI THI KẾT THÚC HỌC PHẦN**

**MÔN:LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

Sinh viên thực hiện: **Lê Trang Nguyên**

Lớp **:** ĐHCNTT1

Khoá : 09

Giảng viên hướng dẫn : **Trần Văn Định**

***TP. Hồ Chí Minh, 15 tháng 05 năm 2022***

**TÓM TẮT LÝ THUYẾT**

**I. THẾ NÀO LÀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG?**

- Lập trình hướng Đối tượng (OOP) là một phương pháp thiết kế và phát triển phần mềm. Những ngôn ngữ OOP không chỉ bao gồm cú pháp và một trình biên dịch (compiler) mà còn có một môi trường phát triển toàn diện. Môi trường này bao gồm một thư viện được thiết kế tốt, thuận lợi cho việc sử dụng các đối tượng.

- Hướng đối tượng là kĩ thuật mô hình hóa một hệ thống thế giới thực trong phần mềm dựa trên các đối tượng. Đối tượng (object) là khái niệm trung tâm của OOP, nó là một mô hình của một thực thể hay khái niệm trong thế giới thực. Việc mô hình hóa này bao gồm xác định các đối tượng tham gia bài toán – những cái làm nhiệmvụ gì đó hoặc bị làm gì đó. Lập trình theo kiểu hướng đối tượng là hoạt động định nghĩa các thể loại của các đối tượng đó ở hình thức các khuôn mẫu để tạo ra chúng.

- OOP dựa chủ yếu vào các đối tượng, nên khi có nhu cầu thay đổi thì chỉ cần thay đổi hoặc bổ sung các lớp mới trên cơ sở kế thừa.

- OOP hỗ trợ rất mạnh nguyên lý sử dụng lại nhiều nhất có thể để tạo ra mọi khả năng để kế thừa những lớp đã được thiết kế, lập trình tốt để nhanh chóng tạo ra những phần mềm có chất lượng tốt, giá thành rẻ hơn và đáp ứng các nhu cầu của người sử dụng.

**II. CÁC KHÁI NIỆM CƠ BẢN**

**1. Class(Lớp)**

Về cơ bản, lớp là các kiểu dữ liệu do người lập trình tự định nghĩa. Lớp là nơi chúng ta tạo ra một bản thiết kế (blueprint) mẫu cho cấu trúc của các đối tượng. Lớp là bản mẫu hay một kiểu chung cho tất cả các đối tượng có đặc trưng giống nhau, nghĩa là có các thuộc tính và phương thức giống nhau. Đối tượng là thể hiện (cá thể) của một lớp xác định.

**2.Đối tượng (Object):**

Đối tượng là khái niệm cơ sở, quan trọng nhất của phương pháp lập trình hướng đối tượng. Mỗi đối tượng có các đặc trung là thuộc tính mô tả tính chất của đối tượng và phương thức thể hiện các hành vi của đối tượng.

Đối tượng có thể biểu diễn người, vật, hay một hạng thức nào đó cần được xử lí trong chương trình. Nhiệm vụ của Lập trình hướng đối tượng là phân tích bài toán thành các đối tượng và xác định được bản chất trao đổi thông tin của chúng.

- Cách tạo đối tượng: khi tạo một đối tượng thường dùng từ khóa **new**

**-** Khai báo: **<** **objectType >** **Tên tham chiếu của đối tượng= new <Tên class>();**

Hoặc

**<** **objectType > Tên tham chiếu của đối tượng;**

**Tên tham chiếu của đối tượng= new <Tên class>();**

- Trong đó: +<class\_name>: Là kiểu dữ liệu của đối tượng bạn định tạo

+ ReferenceVariable: Là tên tham chiếu của đối tượng

+ new: Từ khóa để tạo đối tượng

+ <class\_name>(): Là class mà bạn tạo đối tượng từ nó

**- Ví dụ:**

Phuongtien Pt;

Pt=**new** Otocon();

**3. Attributes (Thuộc tính)**

**-** Có thể hiểu, thuộc tính chính là những thông tin cụ thể được lưu trữ. Thuộc tính sẽ được định nghĩa trong lớp. Khi một đối tượng được khởi tạo, thì sẽ có những dữ liệu khác nhau để lưu trữ thuộc tính của chúng. Lúc này, thuộc tính sẽ giúp chúng ta phân biệt giữa đối tượng này với đối tượng khác**.**

**-** Ví dụ: Con người có tên, tuổi ,năm sinh ….

- Khai báo thuộc tính: **<Phạm vi truy cập><Type><Tên thuộc tính>;**

Trong đó: + Phạm vi truy cập: public, private, protect

+ Type: Kieu dữ liệu

+ Tên thuộc tính: Do người lập trình đặt

**- Ví dụ:**

**private** String Hangsx;

**private** String Tenxe;

**private** **int** Congsuat;

**4. Phương thức (method)**

Trong lập trình hướng đối tượng, phương thức đại diện cho các hành vi. Cụ thể, phương thức **có** thể yêu cầu đối tượng thực hiện hành động, trả lại thông tin về một đối tượng hoặc cập nhật dữ liệu của một đối tượng.

Ví dụ: Con người có thể ăn nói đi đứng….

**-** Khai báo phương thức:

**<Phạm vi truy cập > <kiểu trả về/không trả về> <Tên phương thức> (<danh sách đối số>){**

**<khối lệnh>;**

**}**

**- Ví dụ:**

+ Kiểu không có kết quả trả về

**public** **void** Nhap() {

// **TODO** Auto-generated method stub

Scanner sc= **new** Scanner(System.***in***);

System.***out***.println("Nhap ma so oto con: ");

**this**.setMaso(sc.nextInt());

sc.nextLine();

System.***out***.println("Nhap ten hang san xuat: ");

**this**.setHangsx(sc.nextLine());

System.***out***.println("Nhap ten oto con: ");

}

+ Kiểu có kết quả trả về

String Phitruocba() {

**return** "2%";

}

**5. Hàm khởi tạo (constructor)**

Phương thức tạo dựng (constructor) trong java là một phương thức của lớp (nhưng khá đặc biệt) thường được sử dụng để khởi tạo các đối tượng. Thông thường người ta thường sử dụng hàm tạo để khởi gán giá trị cho các thuộc tính của đối tượng. Hàm khởi tạo phải có tên giống tên class

Hàm khởi tạo có 2 loai:

- Không có tham số: tạo ra đối tượng mà không cần tham số đầu vào

- Có tham số: là giá trị ban đầu cần gán cho đối tượng

**Ví dụ:**

**public** **abstract** **class** Thongtinsohuu {

**private** **int** Maso;

**private** String Tenchusohuu;

Thongtinsohuu() {

}

Thongtinsohuu(**int** Maso, String Tenchusohuu){

**this**.Maso=Maso;

**this**.Tenchusohuu=Tenchusohuu;

}

- Lưu ý: Nó được gọi là hàm tạo (constructor) vì nó xây dựng các giá trị tại thời điểm tạo đối tượng. Không cần thiết phải viết một constructor cho một class. Đó là bởi vì trình biên dịch java tạo ra một hàm tạo mặc định nếu class của bạn không có.

**6. Input trong java**

- Java là một ngôn ngữ lập trình thuần hướng đối tượng, nên cách nhập dữ liệu từ bàn phím cũng phải sử dụng theo hướng đổi tượng. Tức là phải khai báo một đối tượng, sau đó gọi hàm nhập, rồi gán giá trị cho biến.

- Scanner là một lớp trong thư viện Java.util giúp nhập và lưu dữ liệu từ bàn phím. Ngoài scanner, còn có nhiều lớp khác nữa, nhưng đây là lớp được sử dụng phổ biến.

- Để sử dụng Scanner cần khai báo thư viện của lớp này, sau đó tạo đối tượng cho lớp Scanner, **ví dụ:**

**import** java.util.Scanner;

**public** **abstract** **class** Thongtinsohuu {

**private** **int** Maso;

**public** **void** Nhap(){

Scanner sc= **new** Scanner(System.***in***);

System.***out***.println("Nhap ma so phuong tien: ");

**this**.setMaso(sc.nextInt());

}

- Các phương thức thường dùng trong lớp Scanner:

|  |  |
| --- | --- |
| **Tên phương thức** | **Mô tả** |
| nextByte | Đọc một số nguyên kiểu byte |
| nextShort | Đọc một số nguyên kiểu short |
| nextInt | Đọc một số nguyên kiểu int |
| nextLong | Đọc một số nguyên kiểu long |
| nextFloat | Đọc một số kiểu float |
| nextDouble | Đọc một số kiểu double |
| next | Đọc một string kết thức trước một ký tự trắng |
| nextLine | Đọc một line of text (kết thúc bằng phím Enter) |

**7. Output trong java**

Một cách đơn giản, bạn có thể sử dụng câu lệnh System.out.println(), câu lệnh System.out.print() hoặc câu lệnh System.out.printf() để đưa output ra màn hình.

Sự khác biệt của print(), println(), printf():

- print() – in chuỗi bên trong dấu ngoặc kép.

- println()- in chuỗi bên trong dấu ngoặc kép giống như hàm print(). Sau đó con trỏ di chuyển đến đầu dòng tiếp theo.

- printf()- cung cấp định dạng chuỗi (tương tự như printf trong lập trình C / C++ ).

Lưu ý:

- Để hiển thị số nguyên, biến, v.v., không sử dụng dấu ngoặc kép.

- Có thể sử dụng toán tử + để nối chuỗi và in chuỗi.

**Ví dụ:**

**import** java.util.Scanner;

**public** **abstract** **class** Thongtinsohuu {

**private** **int** Maso;

**public** **void** Xuat(){

System.***out***.println("Ma so phuong tien: "+getMaso());

}

**8. ArrayList**

Lớp ArrayList trong java là một lớp kế thừa lớp AbstractList và triển khai của List Interface trong Collections Framework nên nó sẽ có một vài đặc điểm và phương thức tương đồng với List. ArrayList được sử dụng như một mảng động để lưu trữ các phần tử.

Để khai báo một ArrayList, chúng ta cần phải import gói thư viện java.util.ArrayList của Java. Cú pháp import như sau:

**import** java.util.ArrayList;

**a. Khai báo một ArrayList**

ArrayList<String> listname = new ArrayList<>();

**Ví dụ:**

ArrayList<Thongtinsohuu>dsPt=**new** ArrayList<Thongtinsohuu>();

**b. Thêm phần tử vào ArrayList**

Sử dụng phương thức add() để thêm phần tử vào ArrayList

Ví dụ:

Pt=**new** Bantai();

Pt.Nhap();

dsPt.add(Pt);

**c. Truy cập phần tử**

Java cung cấp cho chúng ta phương thức get() để truy cập đến 1 phần tử bất kỳ trong ArrayList thông qua chỉ số của phần tử đó.

**Ví dụ:**

dsPt.get(i).Xuat();

**d. Cập nhật giá trị của phần tử**

Để cập nhật giá trị của phần tử trong ArrayList, Java cung cấp cho chúng ta phương thức set(index, element), trong đó index là chỉ số của phần tử cần cập nhật và element là phần tử mới để thay thế.

**Ví dụ:**

Pt.Nhap();

dsPt.set(1, Pt);

**e. Xóa phần tử**

Để xóa phần tử trong ArrayList, Java cung cấp phương thức có sẵn đó là phương thức remove(). Phương thức remove() sẽ xóa phần tử ra khỏi ArrayList theo 2 cách đó là xóa dựa vào chỉ số của phần tử và xóa trực tiếp phần tử đó (không cần biết đến chỉ số của nó).

**Ví dụ:**

dsPt.remove(i);

**III. 4 ĐẶC TÍNH CƠ BẢN CỦA OOP**

**1. Tính đóng gói (Encapsulation)**

**Khái niệm**

**-** Hiểu một cách đơn giản, “đóng gói” là việc đưa tất cả thông tin, dữ liệu quan trọng vào bên trong một đối tượng (object). Sau đó, khi một đối tượng được khởi tạo từ lớp (class), thì dữ liệu và phương thức (method) đã được đóng gói trong đối tượng đó. Khi sử dụng, ta chỉ cần gọi tên phương thức chứ không cần truy cập đến dữ liệu bên trong.

- Tính đóng gói cho phép che giấu thông tin và những tính chất xử lý bên trong của đối tượng. Các đối tượng khác không thể tác động trực tiếp đến dữ liệu bên trong và làm thay đổi trạng thái của đối tượng mà bắt buộc phải thông qua các phương thức công khai do đối tượng đó cung cấp.

- Tính chất này giúp tăng tính bảo mật cho đối tượng và tránh tình trạng dữ liệu bị hư hỏng ngoài ý muốn

**Ưu điểm**

**-** Tính linh hoạt: Mã được đóng gói sẽ linh hoạt, dễ sửa đổi hơn là những đoạn mã độc lập.

- Khả năng tái sử dụng: Mã đã đóng gói có thể được tái sử dụng trong một ứng dụng hoặc nhiều ứng dụng. Từ một đối tượng, người dùng có thể chuyển sang dùng một đối tượng khác mà không phải đổi mã. Bởi vì cả hai đối tượng đều có giao diện như nhau.

- Khả năng bảo trì: Mã được đóng gói trong những phần riêng biệt, như là lớp, phương thức, giao diện,… Do đó, việc thay đổi, cập nhật một phần của ứng dụng không ảnh hưởng đến những phần còn lại. Điều này giúp giảm công sức và tiết kiệm thời gian cho các nhà phát triển.

- Khả năng kiểm thử: Đối với một lớp được đóng gói, Tester sẽ diễn viết những bài kiểm thử hơn. Các biến thành viên sẽ tập trung ở một nơi chứ không nằm rải rác. Do đó, kiểm thử viên cũng tiết kiệm được thời gian và công sức hơn.

- Che giấu dữ liệu: Khi sử dụng phương thức, người dùng chỉ cần biết nó tạo ra kết quả gì. Họ không cần quan tâm đến những chi tiết bên trong của đối tượng để sử dụng nó.

**a. Từ khóa phạm vi truy cập(Access Modifier)**

Được dùng trước các khai báo của một lớp, biến hay phương thức để thể hiện khả năng truy cập của lớp, biến hay phương thức đó ở các lớp khác.

- Private: Truy cập trong nội bộ lớp

- Public: Thành phần công khai, truy cập tự do từ bên ngoài

- Protected: Thành phần được bảo vệ, bị hạn chế truy nhập từ bên ngoài: có thể truy cập bên trong package và bên ngoài package nhưng chỉ thông qua tính kế thừa.

**Ví dụ:**

**public** **abstract** **class** Thongtinsohuu {

**private** **int** Maso;

**public** **void** Nhap(){

Scanner sc= **new** Scanner(System.***in***);

System.***out***.println("Nhap ma so phuong tien: ");

Maso= sc.nextLine();

}

**b. Getter/setter**

Đây là hai tên gọi của hai thể loại phương thức. Chúng có liên quan đến đặc tính gói ghém dữ liệu trong lập trình hướng đối tượng, tiếng Anh gọi đặc tính này là Encapsulation. Trước đó, các định nghĩa về khả năng truy cập cũng là một dạng của gói ghém dữ liệu.

Các phương thức này giúp cho các thuộc tính private của một lớp hoặc chỉ được phép cho đọc (thông qua phương thức getter), hoặc chỉ cho cập nhật (thông qua phương thức setter) từ các lớp bên ngoài.

Để tạo ra sự gói ghém cho các thuộc tính trong một lớp thông qua getter và setter, thực hiện các bước sau:

– Định nghĩa khả năng truy cập cho các thuộc tính trong một lớp về private.

– Xây dựng phương thức getter hay setter cho từng thuộc tính private đó. Với setter nên đặt tên phương thức là setTênThuộcTính(). Còn với getter thì nên đặt tên phương thức là getTênThuộcTính(). Việc xuất hiện của getter và setter cho các thuộc tính private là không bắt buộc. Có thể không cần khai báo bất cứ getter hay setter nào cả nếu như không muốn thuộc tính đó được thấy bởi bất kỳ lớp nào. Hoặc bạn chỉ cần xây dựng getter cho một thuộc tính private nếu muốn thuộc tính đó chỉ được xem, không được sửa chữa. Hoặc chỉ xây dựng setter cho một thuộc tính private nếu muốn thuộc tính đó chỉ có khả năng chỉnh sửa từ bên ngoài mà không được xem.

- Cấu trúc:

+ Setter

**public void set <tên thuộc tính> ( < tham số giá trị mới> ){**

**this.<tên thuộc tính> = < tham số giá trị mới>;**

**}**

+ Getter

**< kiểu dữ liệu thuộc tính> get <tên thuộc tính> (){**

**Return this.<tên thuộc tính> ;**

**}**

**- Ví dụ:**

**public** **abstract** **class** Thongtinsohuu {

**private** **int** Maso;

**private** String Tenchusohuu;

**public** **int** getMaso() {

**return** Maso;

}

**public** **void** setMaso(**int** Maso) {

**this**.Maso=Maso;

}

**public** String getTenchusohuu() {

**return** Tenchusohuu;

}

**public** **void** setTenchusohuu(String Tenchusohuu) {

**this**.Tenchusohuu=Tenchusohuu;

}

}

**2. Tính kế thừa**

Kế thừa trong java là sự liên quan giữa hai class với nhau, trong đó có class cha (superclass) và class con (subclass). Khi kế thừa class con được hưởng tất cả các phương thức và thuộc tính của class cha. Tuy nhiên, nó chỉ được truy cập các thành viên public và protected của class cha. Nó không được phép truy cập đến thành viên private của class cha.

**Ưu điểm:**

- Tính kế thừa giúp tăng khả năng tái sử dụng. Khi một lớp kế thừa hoặc dẫn xuất một lớp khác, thì nó có thể truy cập tất cả các chức năng của lớp mà nó kế thừa.

- Khả năng tái sử dụng nâng cao độ tin cậy. Chúng ta chỉ cần kiểm tra và gỡ lỗi với mã lớp cha, chứ không cần kiểm tra từng lớp con.

- Khi mã được tái sử dụng, sẽ giúp giảm chi phí phát triển và bảo trì.

- Các lớp con sẽ tuân thủ theo một giao diện (interface) chuẩn.

- Tính kế thừa giúp hạn chế sự dư thừa mã và hỗ trợ khả năng mở rộng mã.

- Lập trình viên có điều kiện thuận lợi để tạo các thư viện lớp (class libraries).

**Từ khóa this,super:**

- this chỉ đến lớp hiện tại của đối tượng

- super chỉ đến lớp cha

**Ví dụ:**

- Lớp cha

**import** java.util.Scanner;

**public** **class** Thongtinsohuu {

**private** **int** Maso;

**public** **int** getMaso() {

**return** Maso;

}

**public** **void** setMaso(**int** Maso) {

**this**.Maso=Maso;

}

**public** **void** Nhap(){

Scanner sc= **new** Scanner(System.***in***);

System.***out***.println("Nhap ma so phuong tien: ");

**this**.setMaso(sc.nextInt());

}

**public** **void** Xuat(){

System.***out***.println("Ma so phuong tien: "+getMaso());

}

}

- Lớp con:\

**import** java.util.Scanner;

**public** **class** Otocon **extends** Thongtinsohuu {

**private** String Hangsx;

**private** String Tenxe;

**public** String getHangsx() {

**return** Hangsx;

}

**public** **void** setHangsx(String Hangsx) {

**this**.Hangsx=Hangsx;

}

**public** String getTenxe() {

**return** Tenxe;

}

**public** **void** setTenxe(String Tenxe) {

**this**.Tenxe=Tenxe;

}

**public** **void** Nhap() {

Scanner sc= **new** Scanner(System.***in***);

**super**.Nhap();

System.***out***.println("Nhap ten hang san xuat: ");

**this**.setHangsx(sc.nextLine());

System.***out***.println("Nhap ten oto con: ");

**this**.setTenxe(sc.nextLine());

}

**public** **void** Xuat() {

**super**.Xuat();

System.***out***.println("Hang san xuat: "+getHangsx());

System.***out***.println("Ten xe: "+getTenxe());

}

}

**3. Tính đa hình**

Tính đa hình có nghĩa là cùng một hành động nhưng ở những ngữ cảnh khác nhau thì cho ra những hành động khác nhau, hoặc cho ra các kết quả khác nhau.

**Ví dụ:**

Thongtinsohuu Pt;

Pt=**new** Otocon();

Pt=**new** Bantai();

Pt=**new** Xetai();

**Sử dụng đa hình khi:**

- Khi các lớp con cần dùng phương thức của lớp cha để bổ sung cho một phương thức khác.

- Khi một lớp cần có nhiều phương thức cùng tên nhưng khác tham số (parameter).

**Để có đa hình:** trong Java, chúng ta sử dụng nạp chồng phương thức (method overloading) và ghi đè phương thức (method overriding)

- Nạp chồng (Overloading): Đây là khả năng cho phép một lớp có nhiều thuộc tính, phương thức cùng tên nhưng với các tham số khác nhau về loại cũng như về số lượng. Khi được gọi, dựa vào tham số truyền vào, phương thức tương ứng sẽ được thực hiện.

- Ghi đè (Overriding): là hai phương thức cùng tên, cùng tham số, cùng kiểu trả về nhưng thằng con viết lại và dùng theo cách của nó, và xuất hiện ở lớp cha và tiếp tục xuất hiện ở lớp con. Khi dùng override, lúc thực thi, nếu lớp Con không có phương thức riêng, phương thức của lớp Cha sẽ được gọi, ngược lại nếu có, phương thức của lớp Con được gọi.

**Quan hệ cha con:**

- Upcasting và Downcasting:là quá trình ép kiểu qua lại giữa 2 đối tượng thuộc lớp cha và lớp con.

- Upcasting: khi biến tham chiếu của lớp cha tham chiếu tới đối tượng lớp con thid đó upcasting

**4. Tính trừu tượng**

Tính trừu tượng giúp bạn tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện.

Trong Java, chúng là sử dụng abstract class và abstract interface để có tính trừu tượng.

**a. Lớp trừu tượng (abstract class)**

**Khái niêm:**

- Là lớp chưa có phương thức xác định

- Từ khóa abstract được dùng để định nghĩa lớp và phương thức trừu tượng

- Phương thức trừu tượng không có phần thân hàm

- Lớp chứa phương thức trừu tượng là lớp trừu tượng

- Khi kế thừa lớp trừu tượng phải định nghĩa các phương thức của lớp trừu tượng đó

- Không thể sử dụng new để tạo đối tượng

**Ưu điểm:**

**-** Sử dụng abstract class để tạo tính kế thùa cho class

- Có thể linh động taoj các method giống như một class thông thường.

- Các class extend có thể override hoặc không override các method thường

**Cú pháp:**

- Khai báo lớp trừu tượng

**public abstract classname{**

**//khai báo thuộc tính;**

**//khai báo phương thức;**

**}**

**-** Khai báo phương thức trừu tượng

**public <kiểu trả về/không trả về> <Tên phương thức> (<danh sách đối số>);**

**-Ví dụ:**

+ Lớp abstract

**public** **abstract** **class** Thongtinsohuu {

**private** **int** Maso;

**private** String Tenchusohuu;

**public** **int** getMaso() {

**return** Maso;

}

**public** **void** setMaso(**int** Maso) {

**this**.Maso=Maso;

}

**public** String getTenchusohuu() {

**return** Tenchusohuu;

}

**public** **void** setTenchusohuu(String Tenchusohuu) {

**this**.Tenchusohuu=Tenchusohuu;

}

**abstract** String Phitruocba();

}

+ Lớp kế thừa lớp abstract:

**public** **class** Otocon **extends** Thongtinsohuu {

@Override

String Phitruocba() {

// **TODO** Auto-generated method stub

**return** "10%";

}

}

**b. Interface**

**Khái niệm**

- Là cơ chế hỗ trợ trừu trượng hóa trong java và thực hiện kế thừa

- Một Interface trong java là bản thiết kế của lớp

- Là một tập hợp các phương thức mà một class buộc phải định nghĩa khi triển khai

- Làm nhiệm vụ khai báo các lớp mà class cần cớ nhưng không có phần thân hàm

- Khi 1 class inplements từ interface nào đó thì bắt buộc phải ghi đè (override) tất cả phương thức trong interface tương

**Ưu điểm**

- Có thể kế thừa nhiều interface(đa kế thừa).

- Xây dựng được bộ khung mẫu mà các lớp phải follow theo.

- Giúp quản lý tốt, nắm bắt được các chức năng phải có cho một đối tượng nào đó.

**Cú pháp:**

**-** Khai báo interface

**public Interface TênInterface{**

**//khai báo phương thức trừ tượng;**

**}**

**- Ví dụ:**

+ Interface:

**public** **interface** Dongco {

**public** **int** getCongsuat();

**public** **void** setCongsuat(**int** Congsuat);

}

+ Class implements từ interface

**public** **class** Otocon **implements** Dongco {

**private** **int** Congsuat;

@Override

**public** **int** getCongsuat() {

// **TODO** Auto-generated method stub

**return** Congsuat;

}

@Override

**public** **void** setCongsuat(**int** Congsuat) {

// **TODO** Auto-generated method stub

**this**.Congsuat=Congsuat;

}

}

- Một interface có thể kế thừa một interface khác:

**- Ví dụ:**

**public** **interface** Phuongtien {

**public** String getHangsx ();

**public** **void** setHangsx(String Hangsx);

**public** String getTenxe();

**public** **void** setTenxe(String Tenxe);

}

**public** **interface** Dongco **extends** Phuongtien {

**public** **int** getCongsuat();

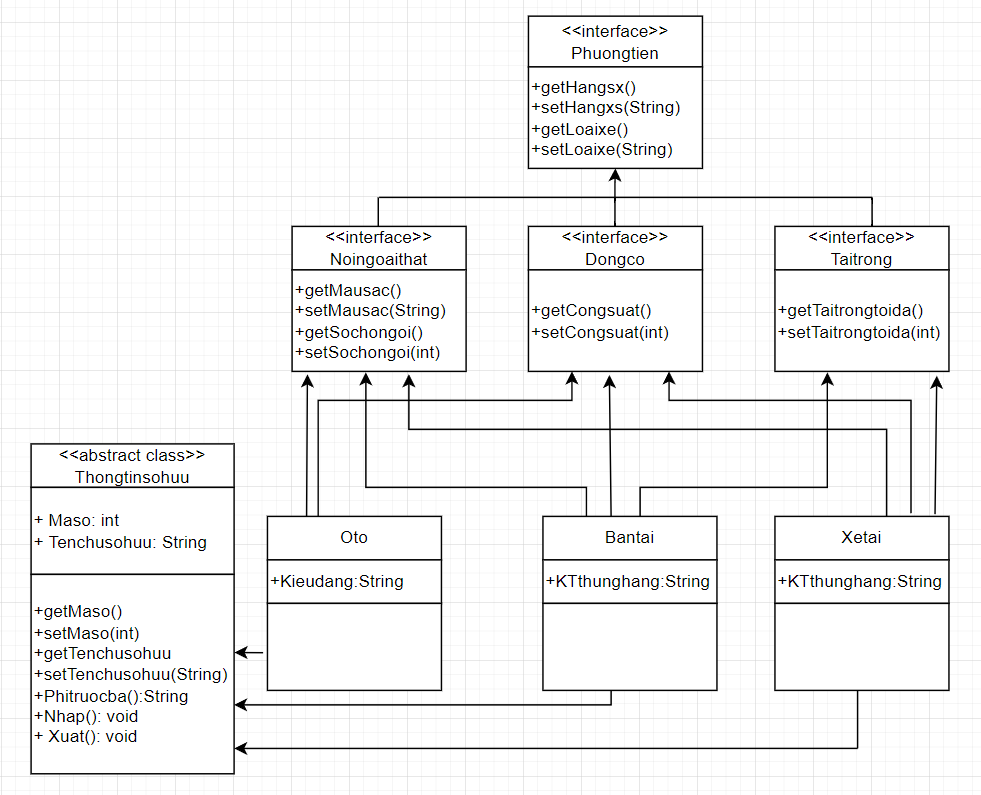
**public** **void** setCongsuat(**int** Congsuat);

}

**IV. Demo**

**1.** **Tên Demo:**

Demo quản lý phương tiện giao thông cho cơ quan chức năng tại địa phương

** 2. Sơ đồ lớp**

**3. Demo**

**Thongtinsohuu( abstract class)**

|  |
| --- |
| **package** demo;  **import** java.util.Scanner;  **public** **abstract** **class** Thongtinsohuu {  **private** **int** Maso;  **private** String Tenchusohuu;  Thongtinsohuu() {  }  Thongtinsohuu(**int** Maso, String Tenchusohuu){  **this**.Maso=Maso;  **this**.Tenchusohuu=Tenchusohuu;  }  **public** **int** getMaso() {  **return** Maso;  }  **public** **void** setMaso(**int** Maso) {  **this**.Maso=Maso;  }  **public** String getTenchusohuu() {  **return** Tenchusohuu;  }  **public** **void** setTenchusohuu(String Tenchusohuu) {  **this**.Tenchusohuu=Tenchusohuu;  }  **abstract** String Phitruocba();  **public** **void** Nhap(){  Scanner sc= **new** Scanner(System.***in***);  System.***out***.println("Nhap ma so phuong tien: ");  **this**.setMaso(sc.nextInt());  sc.nextLine();  System.***out***.println("Nhap ten chu phuong tien: ");  **this**.setTenchusohuu(sc.nextLine());  }  **public** **void** Xuat(){  System.***out***.println("Ma so phuong tien: "+getMaso());  System.***out***.println("Ten chu phuong tien: "+getTenchusohuu());    }  } |

**Phuongtien( Interface)**

|  |
| --- |
| **package** demo;  **public** **interface** Phuongtien {  **public** String getHangsx ();  **public** **void** setHangsx(String Hangsx);  **public** String getTenxe();  **public** **void** setTenxe(String Tenxe);    } |

**Dongco (Interface)**

|  |
| --- |
| **package** demo;  **public** **interface** Dongco **extends** Phuongtien {  **public** **int** getCongsuat();  **public** **void** setCongsuat(**int** Congsuat);  } |

**Noingoaithat (Interface)**

|  |
| --- |
| **package** demo;  **public** **interface** Noingoaithat **extends** Phuongtien {  **public** String getMausac();  **public** **void** setMausac(String Mausac);  **public** **int** getSochongoi();  **public** **void** setSochongoi(**int** Sochongoi);  } |

**Taitrong (Interface)**

|  |
| --- |
| **package** demo;  **public** **interface** Taitrong **extends** Phuongtien {  **public** **int** getTaitrongtoida();  **public** **void** setTaitrongtoida(**int** Taitrongtoida);  } |

**Otocon (class)**

|  |
| --- |
| **package** demo;  **import** java.util.Scanner;  **public** **class** Otocon **extends** Thongtinsohuu **implements** Noingoaithat, Dongco {  **private** String Hangsx;  **private** String Tenxe;  **private** **int** Congsuat;  **private** String Mausac;  **private** **int** Sochongoi;  **private** String Kieudang;    @Override  **public** String getHangsx() {  // **TODO** Auto-generated method stub  **return** Hangsx;  }  @Override  **public** **void** setHangsx(String Hangsx) {  // **TODO** Auto-generated method stub  **this**.Hangsx=Hangsx;  }  @Override  **public** String getTenxe() {  // **TODO** Auto-generated method stub  **return** Tenxe;  }  @Override  **public** **void** setTenxe(String Tenxe) {  // **TODO** Auto-generated method stub  **this**.Tenxe=Tenxe;  }    @Override  **public** **int** getCongsuat() {  // **TODO** Auto-generated method stub  **return** Congsuat;  }  @Override  **public** **void** setCongsuat(**int** Congsuat) {  // **TODO** Auto-generated method stub  **this**.Congsuat=Congsuat;  }  @Override  **public** String getMausac() {  // **TODO** Auto-generated method stub  **return** Mausac;  }  @Override  **public** **void** setMausac(String Mausac) {  // **TODO** Auto-generated method stub  **this**.Mausac=Mausac;  }  @Override  **public** **int** getSochongoi() {  // **TODO** Auto-generated method stub  **return** Sochongoi;  }  @Override  **public** **void** setSochongoi(**int** Sochongoi) {  // **TODO** Auto-generated method stub  **this**.Sochongoi=Sochongoi;  }  @Override  String Phitruocba() {  // **TODO** Auto-generated method stub  **return** "10%";  }  **public** String getKieudang() {  // **TODO** Auto-generated method stub  **return** Kieudang;  }    **public** **void** setKieudang(String Kieudang) {  // **TODO** Auto-generated method stub  **this**.Kieudang=Kieudang;  }  @Override  **public** **void** Nhap() {  // **TODO** Auto-generated method stub  System.***out***.println(" NHAP THONG TIN OTO CON ");  Scanner sc= **new** Scanner(System.***in***);  **super**.Nhap();  System.***out***.println("Nhap ten hang san xuat: ");  **this**.setHangsx(sc.nextLine());  System.***out***.println("Nhap ten oto con: ");  **this**.setTenxe(sc.nextLine());  System.***out***.println("Nhap mau sac: ");  **this**.setMausac(sc.nextLine());  System.***out***.println("Nhap kieu dang oto con: ");  **this**.setKieudang(sc.nextLine());  System.***out***.println("Nhap cong suat oto con(ma luc): ");  **this**.setCongsuat(sc.nextInt());  System.***out***.println("Nhap so cho ngoi: ");  **this**.setSochongoi(sc.nextInt());  System.***out***.println("-----------------------------------------");  }  @Override  **public** **void** Xuat() {  System.***out***.println(" XUAT THONG TIN OTO CON");  **super**.Xuat();  System.***out***.println("Hang san xuat: "+getHangsx());  System.***out***.println("Ten xe: "+getTenxe());  System.***out***.println("Kieu dang xe: "+getKieudang());  System.***out***.println("Cong suat xe: "+getCongsuat()+" ma luc");  System.***out***.println("Mau sac xe: "+getMausac());  System.***out***.println("So cho ngoi: "+getSochongoi());  System.***out***.println("Phi truoc ba: "+Phitruocba());  System.***out***.println("-----------------------------------------");    }  } |

**Bantai (class)**

|  |
| --- |
| **package** demo;  **import** java.util.Scanner;  **public** **class** Bantai **extends** Thongtinsohuu **implements** Noingoaithat, Dongco, Taitrong {  **private** String Hangsx;  **private** String Tenxe;  **private** **int** Congsuat;  **private** String Mausac;  **private** **int** Sochongoi;  **private** String Kichthuocthunghang;  **private** **int** Taitrongtoida;    @Override  **public** String getHangsx() {  // **TODO** Auto-generated method stub  **return** Hangsx;  }  @Override  **public** **void** setHangsx(String Hangsx) {  // **TODO** Auto-generated method stub  **this**.Hangsx=Hangsx;  }  @Override  **public** String getTenxe() {  // **TODO** Auto-generated method stub  **return** Tenxe;  }  @Override  **public** **void** setTenxe(String Tenxe) {  // **TODO** Auto-generated method stub  **this**.Tenxe=Tenxe;  }  @Override  **public** **int** getTaitrongtoida() {  // **TODO** Auto-generated method stub  **return** Taitrongtoida;  }  @Override  **public** **void** setTaitrongtoida(**int** Taitrongtoida) {  // **TODO** Auto-generated method stub  **this**.Taitrongtoida=Taitrongtoida;  }  @Override  **public** **int** getCongsuat() {  // **TODO** Auto-generated method stub  **return** Congsuat;  }  @Override  **public** **void** setCongsuat(**int** Congsuat) {  // **TODO** Auto-generated method stub  **this**.Congsuat=Congsuat;  }  @Override  **public** String getMausac() {  // **TODO** Auto-generated method stub  **return** Mausac;  }  @Override  **public** **void** setMausac(String Mausac) {  // **TODO** Auto-generated method stub  **this**.Mausac=Mausac;  }  @Override  **public** **int** getSochongoi() {  // **TODO** Auto-generated method stub  **return** Sochongoi;  }  @Override  **public** **void** setSochongoi(**int** Sochongoi) {  // **TODO** Auto-generated method stub  **this**.Sochongoi=Sochongoi;  }  @Override  String Phitruocba() {  // **TODO** Auto-generated method stub  **return** "10%";  }  **public** String getKichthuocthunghang() {  // **TODO** Auto-generated method stub  **return** Kichthuocthunghang;  }  **public** **void** setKichthuocthunghang(String Kichthuocthunghang) {  // **TODO** Auto-generated method stub  **this**.Kichthuocthunghang=Kichthuocthunghang;  }  @Override  **public** **void** Nhap() {  // **TODO** Auto-generated method stub  Scanner sc= **new** Scanner(System.***in***);  System.***out***.println(" NHAP THONG TIN XE BAN TAI");  **super**.Nhap();  System.***out***.println("Nhap ten hang san xuat: ");  **this**.setHangsx(sc.nextLine());  System.***out***.println("Nhap ten xe ban tai: ");  **this**.setTenxe(sc.nextLine());    System.***out***.println("Nhap cong suat xa ban tai(ma luc): ");  **this**.setCongsuat(sc.nextInt());  System.***out***.println("Nhap so cho ngoi: ");  **this**.setSochongoi(sc.nextInt());  sc.nextLine();  System.***out***.println("Nhap mau sac: ");  **this**.setMausac(sc.nextLine());  System.***out***.println("Nhap kich thuoc thung hang xe(Dai X Rong x Cao(m)): ");  **this**.setKichthuocthunghang(sc.nextLine());  System.***out***.println("Nhap tai trong toi da(Tan): ");  **this**.setTaitrongtoida(sc.nextInt());  System.***out***.println("-----------------------------------------");  }  @Override  **public** **void** Xuat() {  // **TODO** Auto-generated method stub  System.***out***.println(" XUAT THONG TIN XE BAN TAI");  **super**.Xuat();  System.***out***.println("Hang san xuat: "+getHangsx());  System.***out***.println("Ten xe: "+getTenxe());  System.***out***.println("Cong suat xe: "+getCongsuat()+" ma luc");  System.***out***.println("Mau sac xe: "+getMausac());  System.***out***.println("So cho ngoi: "+getSochongoi());  System.***out***.println("Kich thuoc thung hang(Dai x Rong x Cao): "+getKichthuocthunghang());  System.***out***.println("Tai trong toi da: "+getTaitrongtoida()+" tan");  System.***out***.println("Phi truoc ba: "+Phitruocba());  System.***out***.println("-----------------------------------------");  }  } |

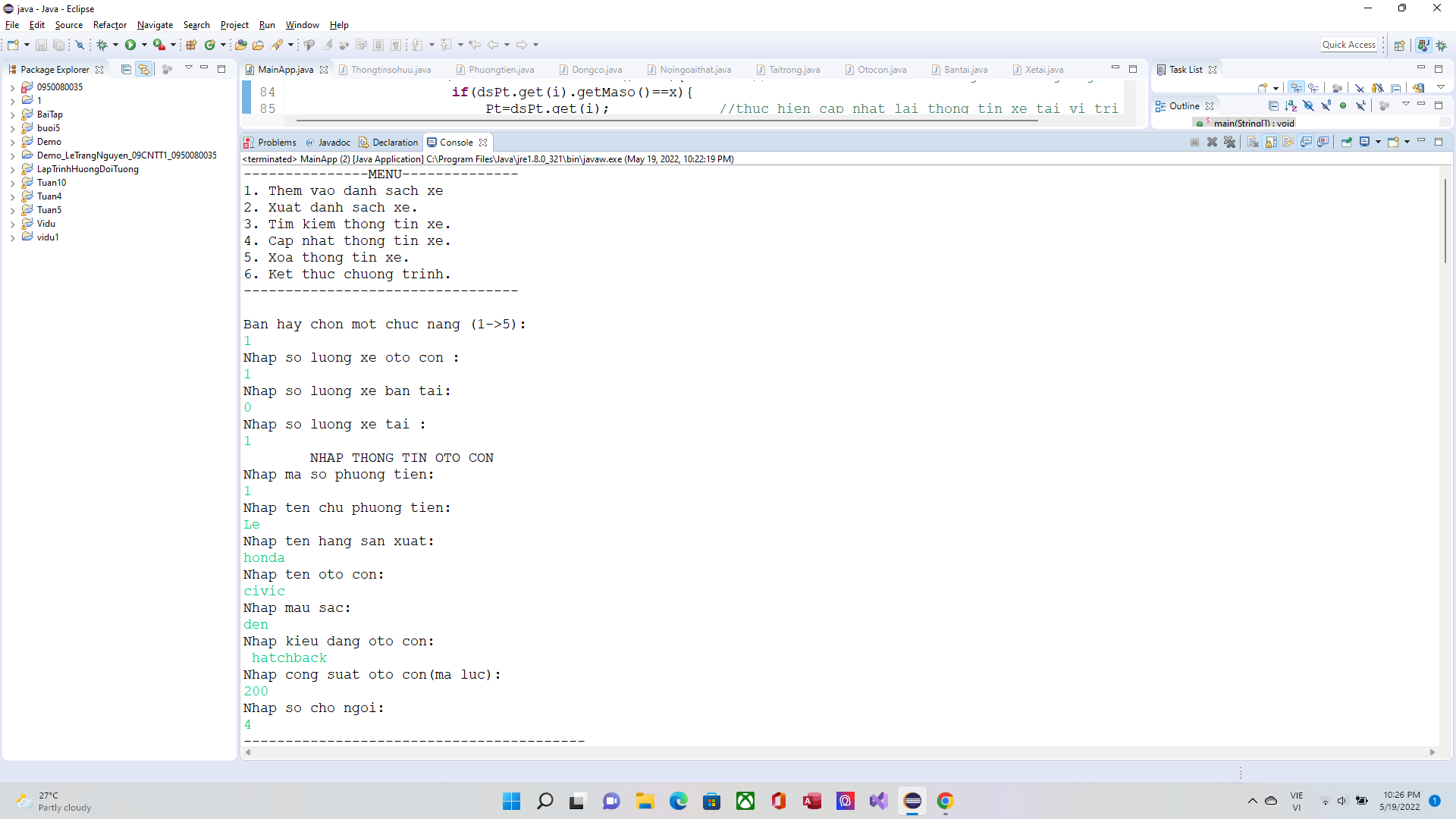
**Xetai (class)**

|  |
| --- |
| **package** demo;  **import** java.util.Scanner;  **public** **class** Xetai **extends** Thongtinsohuu **implements** Noingoaithat, Dongco, Taitrong {  **private** String Hangsx;  **private** String Tenxe;  **private** **int** Congsuat;  **private** String Mausac;  **private** **int** Sochongoi;  **private** String Kichthuocthunghang;  **private** **int** Taitrongtoida;  @Override  **public** String getHangsx() {  // **TODO** Auto-generated method stub  **return** Hangsx;  }  @Override  **public** **void** setHangsx(String Hangsx) {  // **TODO** Auto-generated method stub  **this**.Hangsx=Hangsx;  }  @Override  **public** String getTenxe() {  // **TODO** Auto-generated method stub  **return** Tenxe;  }  @Override  **public** **void** setTenxe(String Tenxe) {  // **TODO** Auto-generated method stub  **this**.Tenxe=Tenxe;  }  @Override  **public** **int** getTaitrongtoida() {  // **TODO** Auto-generated method stub  **return** Taitrongtoida;  }  @Override  **public** **void** setTaitrongtoida(**int** Taitrongtoida) {  // **TODO** Auto-generated method stub  **this**.Taitrongtoida=Taitrongtoida;  }  @Override  **public** **int** getCongsuat() {  // **TODO** Auto-generated method stub  **return** Congsuat;  }  @Override  **public** **void** setCongsuat(**int** Congsuat) {  // **TODO** Auto-generated method stub  **this**.Congsuat=Congsuat;  }  @Override  **public** String getMausac() {  // **TODO** Auto-generated method stub  **return** Mausac;  }  @Override  **public** **void** setMausac(String Mausac) {  // **TODO** Auto-generated method stub  **this**.Mausac=Mausac;  }  @Override  **public** **int** getSochongoi() {  // **TODO** Auto-generated method stub  **return** Sochongoi;  }  @Override  **public** **void** setSochongoi(**int** Sochongoi) {  // **TODO** Auto-generated method stub  **this**.Sochongoi=Sochongoi;  }  @Override  String Phitruocba() {  // **TODO** Auto-generated method stub  **return** "2%";  }  **public** String getKichthuocthunghang() {  // **TODO** Auto-generated method stub  **return** Kichthuocthunghang;  }  **public** **void** setKichthuocthunghang(String Kichthuocthunghang) {  // **TODO** Auto-generated method stub  **this**.Kichthuocthunghang=Kichthuocthunghang;  }  @Override  **public** **void** Nhap() {  // **TODO** Auto-generated method stub  System.***out***.println(" NHAP THONG TIN XE TAI ");  Scanner sc= **new** Scanner(System.***in***);  **super**.Nhap();  System.***out***.println("Nhap ten hang san xuat: ");  **this**.setHangsx(sc.nextLine());  System.***out***.println("Nhap ten xe tai: ");  **this**.setTenxe(sc.nextLine());    System.***out***.println("Nhap cong suat xa tai(ma luc): ");  **this**.setCongsuat(sc.nextInt());  System.***out***.println("Nhap so cho ngoi: ");  **this**.setSochongoi(sc.nextInt());  sc.nextLine();  System.***out***.println("Nhap mau sac: ");  **this**.setMausac(sc.nextLine());  System.***out***.println("Nhap kich thuoc thung hang xe(Dai X Rong x Cao(m)): ");  **this**.setKichthuocthunghang(sc.nextLine());  System.***out***.println("Nhap tai trong toi da(tan): ");  **this**.setTaitrongtoida(sc.nextInt());  System.***out***.println("-----------------------------------------");  }  @Override  **public** **void** Xuat() {  // **TODO** Auto-generated method stub  System.***out***.println(" XUAT THONG TIN XE TAI");  **super**.Xuat();  System.***out***.println("Hang san xuat: "+getHangsx());  System.***out***.println("Ten xe: "+getTenxe());  System.***out***.println("Cong suat xe: "+getCongsuat()+" ma luc");  System.***out***.println("Mau sac xe: "+getMausac());  System.***out***.println("So cho ngoi: "+getSochongoi());  System.***out***.println("Kich thuoc thung hang(Dai x Rong x Cao: "+getKichthuocthunghang());  System.***out***.println("Tai trong toi da: "+getTaitrongtoida()+" tan");  System.***out***.println("Phi truoc ba: "+Phitruocba());  System.***out***.println("-----------------------------------------");  }  } |

**MainApp (class)**

|  |
| --- |
| **package** demo;  **import** java.util.ArrayList;  **import** java.util.Scanner;  **public** **class** MainApp {    **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **int** Chon;  **int** a,b,c;  **int** j;  **int** x;  Scanner sc= **new** Scanner(System.in);  ArrayList<Thongtinsohuu>dsPt=**new** ArrayList<Thongtinsohuu>();  Thongtinsohuu Pt;  **do**  {  System.out.println("---------------MENU--------------");  System.out.println("1. Them vao danh sach xe");  System.out.println("2. Xuat danh sach xe.");  System.out.println("3. Tim kiem thong tin xe.");  System.out.println("4. Cap nhat thong tin xe.");  System.out.println("5. Xoa thong tin xe.");  System.out.println("6. Ket thuc chuong trinh.");  System.out.println("---------------------------------");    **do**  {  System.out.println("\nBan hay chon mot chuc nang (1->5): ");  Chon=sc.nextInt();  }**while**(Chon <= 0 || Chon >6);  **switch**(Chon)  {  **case** 1:  System.out.println("Nhap so luong xe oto con : ");  a=sc.nextInt();  System.out.println("Nhap so luong xe ban tai: ");  b=sc.nextInt();  System.out.println("Nhap so luong xe tai : ");  c=sc.nextInt();    **for**(**int** i=0;i<a;i++) {  Pt=**new** Otocon();  Pt.Nhap();  dsPt.add(Pt);  }    **for**(**int** i=0;i<b;i++) {  Pt=**new** Bantai();  Pt.Nhap();  dsPt.add(Pt);  }    **for**(**int** i=0;i<c;i++) {  Pt=**new** Xetai();  Pt.Nhap();  dsPt.add(Pt);  }  **break**;  **case** 2:  **for**(**int** i=0; i<dsPt.size();i++){  dsPt.get(i).Xuat();  }  **break**;  **case** 3:  sc.nextLine();  System.out.println("Nhap ma so xe can tim : ");  x=sc.nextInt();  j=0;  **for**(**int** i=0; i<dsPt.size();i++){ //Tim kiem vi tri doi tuong co Maso giong Maso vua nhap vao  **if**(dsPt.get(i).getMaso()==x){  dsPt.get(i).Xuat(); //Xuat thong tin xe tai vi tri tuong ung  j++;  }  }  **if**(j==0){  System.out.println("Khong tim thay ma so xe: ");  }  **break**;  **case** 4:  System.out.println("Nhap ma so xe can cap nhat : ");  x=sc.nextInt();  j=0;  **for**(**int** i=0; i<dsPt.size();i++){ //Tim kiem vi tri doi tuong co Maso giong Maso vua nhap vao  **if**(dsPt.get(i).getMaso()==x){  Pt=dsPt.get(i); //thuc hien cap nhat lai thong tin xe tai vi tri tuong ung  Pt.Nhap();  dsPt.set(i, Pt);  System.out.println("DA CAP NHAT THANH CONG");  j++;  }  }  **if**(j==0){  System.out.println("Khong tim thay ma so xe: ");  }  **break**;  **case** 5:  System.out.println("Nhap ma so xe can xoa: ");  x=sc.nextInt();  j=0;  **for**(**int** i=0; i<dsPt.size();i++){ //Tim kiem vi tri doi tuong co Maso giong Maso vua nhap vao  **if**(dsPt.get(i).getMaso()==x){  dsPt.remove(i); //Thuc hien xoa doi tuong tai vi tri tuong ung  System.out.println("DA XOA THANH CONG");  j++;  }  }  **if**(j==0){  System.out.println("Khong tim thay ma so xe: ");  }  **break**;  }  }**while**(Chon != 6);  }  } |

**KẾT QUẢ**

****

