

**BÀI TẬP**  
**MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**  
**(OBJECT ORIENTED PROGRAMMING EXERCISES)**  
**HỆ: ĐẠI HỌC**

## MỤC LỤC

Module 0. CÁC KHÁI NIỆM CƠ BẢN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG .....	4
Bài 1.....	4
Bài 2.....	5
Bài 3.....	6
Bài 4.....	7
Bài 5. * .....	7
Bài 6.....	8
Bài 7. Cài đặt các lớp có quan hệ Aggregation.....	10
Bài 8. Cài đặt lớp chứa danh sách.....	10
Bài 9. Cài đặt lớp chứa danh sách.....	11
Bài 10. Cài đặt các lớp có quan hệ Composition .....	12
Bài 11.....	12
Bài 12.....	13
Phụ lục .....	<b>Error! Bookmark not defined.</b>
Module 1. KẾ THỪA – ĐA HÌNH .....	14
Bài 1.....	14
Bài 2.....	14
Bài 3.....	15
Bài 4.....	15
Bài 5.....	15
Bài 6. * .....	16
Bài 7. * .....	17
Bài 8.....	18
Bài 9.....	18
Bài 10.....	19
Bài 11.....	19
Bài 12.....	20
Module 2. TẬP HỢP .....	21
Bài 1.....	22
Bài 2.....	22
Bài 3.....	22
Bài 4.....	22

Bài 5.....	22
Bài 6. * .....	23
Bài 7. * .....	23
Bài 8.....	25
Bài 9.....	26
Bài 10.....	26
Bài 11.....	27

## Module 0. CÁC KHÁI NIỆM CƠ BẢN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

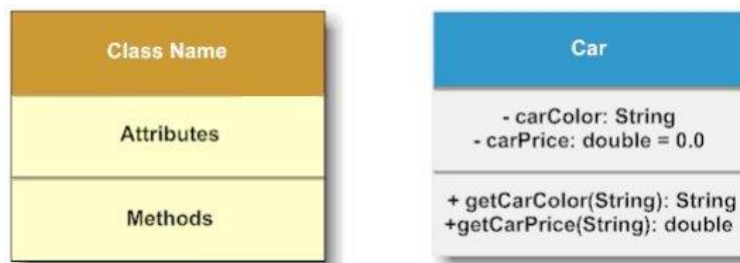
Mục tiêu:

- Cài đặt lớp đối tượng.
- Tạo, sử dụng đối tượng.
- Định nghĩa getter/setter cho thuộc tính (kiểm tra ràng buộc trên thuộc tính nếu có).
- Tạo constructor có kiểm soát tham số truyền cho thuộc tính.
- Tạo cơ chế liên lạc giữa các lớp.
- Ghi chú cho lớp, phương thức (theo dạng document).

Yêu cầu:

- Bài tập bắt buộc: 1, 3, 4, 5, 7, 8, 9.
- Viết documentation comment cho class.

Định nghĩa một lớp trong UML:



Cách truy xuất (visibility):

public	+
private	-
protected	#
package	~

```
public class Car {
    private String carColor;
    private double carPrice = 0.0;
    public String getCarColor(String model) {
        return carColor;
    }
    public double getCarPrice(String model) {
        return carPrice;
    }
}
```

### Bài 1.

Biết lớp tọa độ có:

- Các thuộc tính: tọa độ x, tọa độ y và tên tọa độ.
- Các phương thức thiết lập (set), lấy (get) thông tin x, y và tên tọa độ.
- Các phương thức khởi tạo: **một constructor mặc định** và **một constructor đầy đủ tham số**.
- Phương thức toString trả về thông tin theo mẫu: tên tọa độ (x, y).

Yêu cầu:

- a. Vẽ lược đồ lớp UML cho lớp tọa độ.
- b. Cài đặt lớp tọa độ theo thiết kế.
- c. Xây dựng lớp khác chứa hàm main cho phần kiểm nghiệm.

## **Bài 2.**

- a. Viết lớp Sinh viên như sau:

### Attributes:

- Mã sinh viên (số nguyên > 0),
- Họ tên (chuỗi, không được rỗng),
- Điểm LT, điểm TH (số thực, thuộc [0.0, 10.0]).

### Methods:

- Constructor mặc định (gán giá trị mặc định),
- Constructor nhận đầy đủ thông tin để khởi tạo giá trị cho tất cả các biến instance,
- Các getter và setter cho mỗi thuộc tính,
- Tính điểm trung bình ( $\text{trung bình} = (\text{điểm LT} + \text{điểm TH})/2$ ),
- Phương thức toString diễn tả đối tượng ở dạng chuỗi gồm 4 cột là mã sinh viên, họ tên, điểm LT, điểm TH, điểm TB.

Yêu cầu kiểm tra ràng buộc dữ liệu, gán giá trị mặc định khi dữ liệu không hợp lệ.

- b. Xây dựng class chứa hàm main: tạo 3 đối tượng sinh viên sv1, sv2, sv3, trong đó:
  - sv1 chứa thông tin của chính mình (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
  - sv2 là thông tin người bạn thân nhất của bạn (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
  - sv3 tạo bằng constructor mặc định: Nhập các thông tin cho sv3 từ bàn phím rồi sau đó dùng các setter để gán vào cho các thuộc tính tương ứng. Sau đó thay đổi cách gán dữ liệu dùng constructor đủ tham số.
  - In bảng danh sách sinh viên gồm 4 cột là MSSV, họ tên, điểm LT, điểm TH, điểm TB (bảng có 3 dòng cho 3 sinh viên) như hình bên dưới.

HD: Thông tin sinh viên in trên một dòng có định dạng. Sử dụng `String.format("chuỗi định dạng", đối số 1, đối số 2, ....)`; Trong đó chuỗi định dạng giống c++, ví dụ:

“%-30s”: chuỗi, chiếm 30 ký tự, dấu trừ canh lề trái.

“%5.2f”: số thực, chiếm 5 ký tự, bao gồm 2 ký số lẻ.

Ký tự định dạng:

- s : chuỗi
- d: số nguyên (byte, short, int, long)
- f: số thực (float, double)
- b: boolean

```

<terminated> SinhVienTest [Java Application] C:\Java\jre1.8.0_141\bin\javaw.exe (Aug 5, 2017, 2:00:05 PM)
Nhập mã số của sinh viên sv3:
33333
Nhập họ và tên của sinh viên sv3:
Nguyễn Hoàng Anh
Nhập điểm lý thuyết của sinh viên sv3:
5.0
Nhập điểm thực hành của sinh viên sv3:
9.0
masv    hoten                                diemlt    diemth    diemtb
11111   Nguyễn Thanh An                        6.50      8.50      7.50
22222   Lê Thị Bông                             7.50      8.00      7.75
33333   Nguyễn Hoàng Anh                       5.00      9.00      7.00
  
```

Kết quả thực thi chương trình.

HD: Tham khảo code ở phụ lục.

### Bài 3.

- a. Cài đặt lớp hình tam giác, biết tam giác có 3 cạnh ma, mb, mc, viết các phương thức sau:
  - Constructor mặc định (không tham số).
  - Constructor đủ tham số: *nếu giá trị truyền có số âm hoặc nếu 3 giá trị không lập thành hình tam giác thì gán 3 thuộc tính bằng 0.*
  - Các phương thức getter/setter: *nếu giá trị không hợp lệ thì không gán (giữ lại giá trị cũ).*
  - Các phương thức tính chu vi, phương thức tính diện tích.
  - Phương thức trả về thông tin kiểu tam giác (thường, cân, đều, không phải tam giác).
  - Phương thức `toString` để diễn tả đối tượng ở dạng chuỗi gồm: thông tin 3 cạnh, kiểu tam giác, chu vi, diện tích.
- b. Viết hàm main tạo 5 hình tam giác: 2 hình vi phạm ràng buộc, 3 hình là tam giác thường, cân, đều. Xuất thông tin các hình này theo dạng bảng.

HD:

- Ba giá trị lập thành một hình tam giác khi và chỉ khi tổng hai cạnh bất kỳ luôn lớn hơn cạnh còn lại.
- Công thức tính diện tích tam giác:  $S = \sqrt{p(p-a)(p-b)(p-c)}$  với  $p = \frac{a+b+c}{2}$ .

**Bài 4.**

Cho mô tả bài toán:

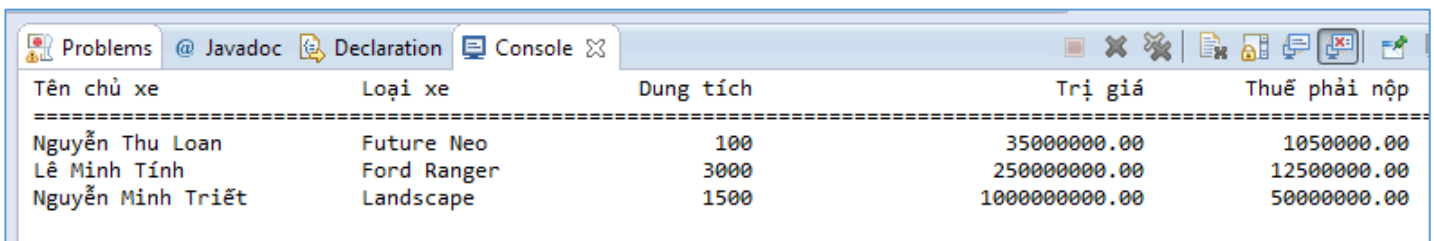
Sở giao thông cần theo dõi việc đăng ký xe (Vehicle) của người dân, biết mỗi xe cần lưu các thông tin là: *chủ xe*, *loại xe*, *trị giá xe* ( $\geq 0$ ), *dung tích xylanh* ( $\geq 0$ ). Dựa vào thông tin *trị giá xe* và *dung tích xylanh*, sở giao thông cũng tính mức thuế phải đóng trước bạ khi mua xe như sau:

- Dưới 100cc, 1% trị giá xe.
- Từ 100 đến 200cc, 3% trị giá xe.
- Trên 200cc, 5% trị giá xe.

a. Hãy thiết kế và cài đặt lớp Vehicle với các attributes và methods phù hợp (có kiểm tra ràng buộc dữ liệu). Lớp phải có các constructor và phải bảo đảm tính encapsulation.

b. Xây dựng class chứa hàm main, có các công việc:

- Tạo 3 đối tượng Vehicle xe1, xe2, xe3. Dữ liệu được gán sẵn hoặc cho người dùng nhập.
- Xuất bảng kê khai tiền thuế trước bạ của các xe như mẫu:



Tên chủ xe	Loại xe	Dung tích	Trị giá	Thuế phải nộp
Nguyễn Thu Loan	Future Neo	100	35000000.00	1050000.00
Lê Minh Tính	Ford Ranger	3000	25000000.00	1250000.00
Nguyễn Minh Triết	Landscape	1500	100000000.00	5000000.00

Mẫu kết xuất của chương trình.

**Bài 5. \***

Lớp **HangThucPham** mô tả một hàng hóa là hàng thực phẩm trong kho của một cửa hàng bán thực phẩm, có các thuộc tính: **mã hàng** (không cho phép sửa, không được để rỗng), **tên hàng** (không được để rỗng, mặc định là "xxx"), **đơn giá** ( $\geq 0$ ), **ngày sản xuất** (phải trước ngày hiện tại, mặc định là ngày hiện tại) và **ngày hết hạn** (phải sau ngày sản xuất, mặc định là ngày sản xuất). Yêu cầu:

- Viết các phương thức setters/getters cho lớp HangThucPham. Trong các setter, nếu dữ liệu muốn gán không hợp lệ thì gán giá trị mặc định cho phép tương ứng của trường đó.
- Tạo một constructor có đầy đủ tham số và một constructor có tham số là mã hàng. Nếu mã hàng rỗng thì phát sinh lỗi và không cho phép tạo đối tượng đó, các giá trị khác nếu không hợp lệ thì để mặc định.
- Viết phương thức kiểm tra một hàng thực phẩm đã hết hạn chưa.
- Phương thức toString trả về chuỗi chứa thông tin của hàng thực phẩm, bao gồm thông tin đã hết hạn chưa. Trong đó: định dạng đơn giá có phân cách hàng nghìn; định dạng kiểu ngày là dd/MM/yyyy.

- e. Viết lớp khác chứa hàm main: Tạo 3 đối tượng, trong đó có một đối tượng có ngày hết hạn trước ngày sản xuất. Xuất thông tin 3 mặt hàng này như mẫu:

Mã_Hàng	Tên_Hàng	Đơn_Giá	Ngày_Sản_Xuất	Ngày_Hết_Hạn	Ghi_Chú
001	Gạo	100,000.00VND	10/07/2018	10/07/2018	
002	Mì	5,000.00VND	01/03/2018	01/09/2018	
003	Nước	10,000.00VND	01/03/2017	01/03/2018	Hàng hết hạn

- f. Kiểm thử lớp HangThucPham với các ràng buộc đã cho.

HD:

- **Kiểm tra mã hàng:**

```
private void setMaHang(String maHang) throws Exception {
    if (!maHang.trim().equals(""))
        this.maHang = maHang;
    else
        throw new Exception("Lỗi: Mã hàng rỗng!");
}
```

- **Kiểu ngày:** dùng lớp định nghĩa sẵn *LocalDate*.
- **Kiểm tra ngày này trước hoặc sau ngày kia:** dùng phương thức *isBefore*, *isAfter*.
- **Kiểm tra hàng đã hết hạn chưa:**

```
public boolean hetHan() {
    return ngayHetHan.isBefore(LocalDate.now()) ? true : false;
}
```

- **Định dạng số phân cách hàng nghìn:** `DecimalFormat df = new DecimalFormat("#,##0.00");`
- **Định dạng kiểu ngày là dd/MM/yyyy:** `DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");`

## Bài 6.

Cài đặt cho lớp tài khoản (Account) được thiết kế như sau:

Account
<ul style="list-style-type: none"> <li>- accountNumber : long</li> <li>- name : String</li> <li>- balance : double</li> <li>- RATE = 0.035 : final double</li> </ul>
<<constructors>> + Account() + Account(accNumber : long, name : String, balance : double) + Account(accNumber : long, name : String)
<<property get>> + getAccountNumber() : long + getBalance() : double
<<other methods>>



+ deposit(amount : double) : boolean + withdraw(amount : double, fee : double) : boolean + addInterest() : void + transfer(acc2 : Account, amount : double): boolean + toString() : String
--

Mô tả lớp Account:

- Ba constructor: **constructor mặc định** (số tài khoản = 999999, tên tài khoản = “chưa xác định”, số dư = 50000); **constructor đầy đủ tham số**; **constructor có hai tham số**. Trong trường hợp giá trị tham số không hợp lệ thì gán giá trị mặc định cho thuộc tính tương ứng (dữ liệu gán là hợp lệ khi số tài khoản >0, tên tài khoản khác rỗng, số dư >=50000).
- **deposit**(amount : double): cho phép gửi thêm số tiền amount vào tài khoản, hàm trả về true nếu gửi thành công (amount>0).
- **withdraw**(amount : double, fee : double): cho phép rút số tiền amount từ tài khoản, hàm trả về true nếu rút tiền thành công (amount>0 và amount+fee <= balance).
- **addInterest**() : tính tiền lãi,  $balance = balance + balance * RATE$ .
- **transfer**(acc2 : Account, amount : double): chuyển một khoản tiền amount từ account này sang account kia, trả về true nếu chuyển thành công.
- **toString** trả về chuỗi chứa toàn bộ thông tin tài khoản, yêu cầu định dạng kiểu tiền tệ.

*HD: Định dạng tiền tệ:*

1. Tạo đối tượng *Locale*: Xác định ngôn ngữ và quốc gia áp dụng: *Locale local= new Locale(“mã NN”, “mã QG”)*. Ví dụ:

```
Locale local = new Locale(“vi”, “vn”);
```

2. Tạo đối tượng *NumberFormatter* với tham số *Locale* ở trên: Ví dụ:

```
NumberFormat formatter = NumberFormat.getCurrencyInstance(local);
```

3. Dùng *formatter* để định dạng: ví dụ: *formatter(456953.12)*;

[formatter sau khi định dạng sẽ trả về chuỗi số đã định dạng]

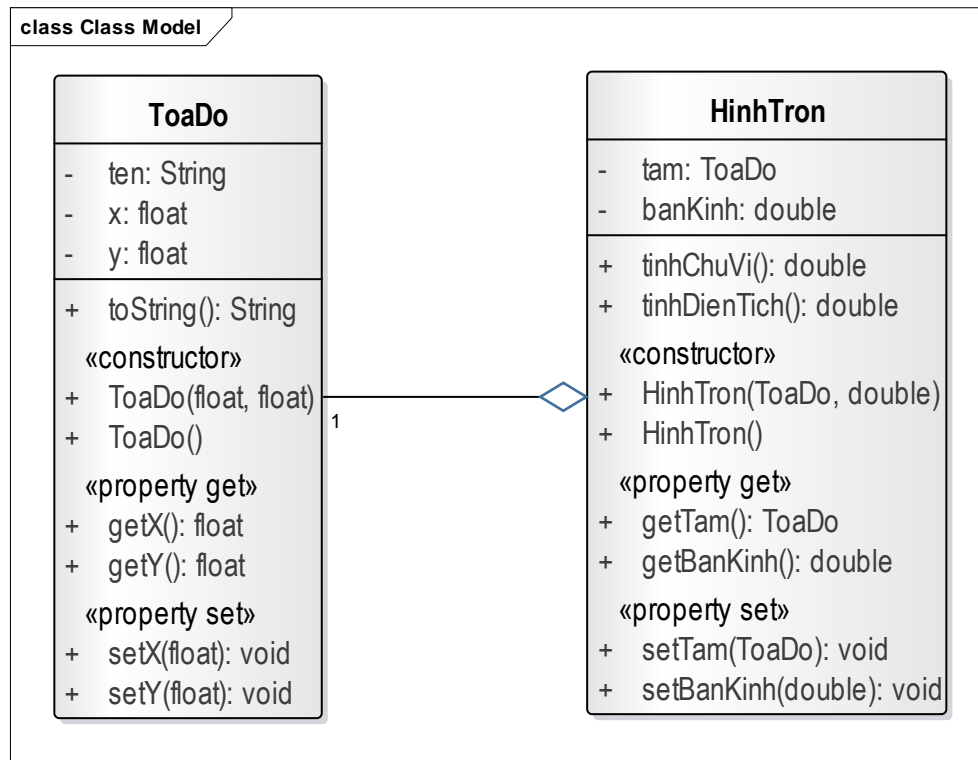
(c) . Test lớp Account như sau :

- Tạo 3 đối tượng acc1, acc2, acc3 với các giá trị name, accountNumber, balance lần lượt như sau: {"Ted Murphy", 72354, 102.56}; {"Jane Smith", 69713, 40.00}; {"Edward Demsey", 93757, 759.32}.
- Gửi thêm số tiền cho acc1 là 25.85, cho acc2 là 500.00.
- Rút khỏi acc2 số tiền là 430.75, mức phí 1.50.
- Tính tiền lãi cho acc3.
- Xuất thông tin của acc1, acc2, acc3 (kiểm tra lại kết quả).
- Chuyển tiền từ acc2 sang cho acc1 số tiền là 100.00.

- Xuất thông tin của acc1, acc2 (kiểm tra lại kết quả).

### Bài 7. Cài đặt các lớp có quan hệ Aggregation

Cài đặt cho mô hình lớp sau:



Yêu cầu kết xuất: Giả sử nhập vào hình tròn có tâm O với 2 tọa độ x,y là 5,5, có bán kính là 10.5 thì xuất kết quả tương ứng là: “*Hình tròn có tâm O(5,5) với bán kính 10.5 có diện tích và chu vi lần lượt là 346.185 và 65.940.*”

### Bài 8. Cài đặt lớp chứa danh sách

Lớp đối tượng CD gồm có các thuộc tính sau:

- Mã CD (là số nguyên >0, mặc định là 999999),
- Tựa CD (chuỗi không được rỗng, tên mặc định là "chưa xác định"),
- Số bài hát (số nguyên >0),
- Giá thành (số thực >0).

Yêu cầu (có vẽ mô hình lớp):

a. Viết lớp CD:

- Các thuộc tính khai báo private; định nghĩa các phương thức get/set cho từng thuộc tính và kiểm tra tính hợp lệ của dữ liệu.
- Viết các constructor để khởi tạo đối tượng CD.
- Viết phương thức toString trả về thông tin của một CD.

- b. Xây dựng lớp **CDList** để lưu danh sách các CD (dùng mảng):
- Phương thức khởi tạo n phần tử cho lớp **CDList**.
  - Phương thức thêm một CD vào danh sách, thêm thành công nếu không trùng mã CD và kích thước mảng còn cho phép.
  - Tính số lượng CD có trong danh sách.
  - Tính tổng giá thành của các CD.
  - Phương thức trả thông tin của toàn bộ CD có trong danh sách.
  - Phương thức sắp xếp danh sách giảm dần theo giá thành.
  - Phương thức sắp xếp danh sách tăng dần theo tựa CD.
- c. Viết lớp cho phần kiểm nghiệm. Có thể dùng menu case thực hiện các chức năng theo yêu cầu.

**Bài 9.** Cài đặt lớp chứa danh sách

- a. Cài đặt lớp **CongNhan**, biết:

- *maCN*, *mHo*, *mTen*, *mSoSP* lần lượt là các thuộc tính họ, tên và số sản phẩm (>0) của công nhân.
- Viết các phương thức getter/setter cho các thuộc tính của lớp.
- Viết các phương thức khởi tạo (mặc định và đầy đủ tham số).
- Viết phương thức  *tinhLuong()* để tính lương cho công nhân,  $\text{lương} = \text{số sản phẩm} * \text{đơn giá}$ , với đơn giá tính theo từng cấp như sau:

Số sản phẩm	Đơn giá
Từ 1 – 199 sản phẩm	0.5
Từ sản phẩm thứ 200 – 399	0.55
Từ sản phẩm thứ 400 – 599	0.6
Từ sản phẩm thứ 600 trở lên	0.65

- Viết phương thức *toString* trả về thông tin của một công nhân.

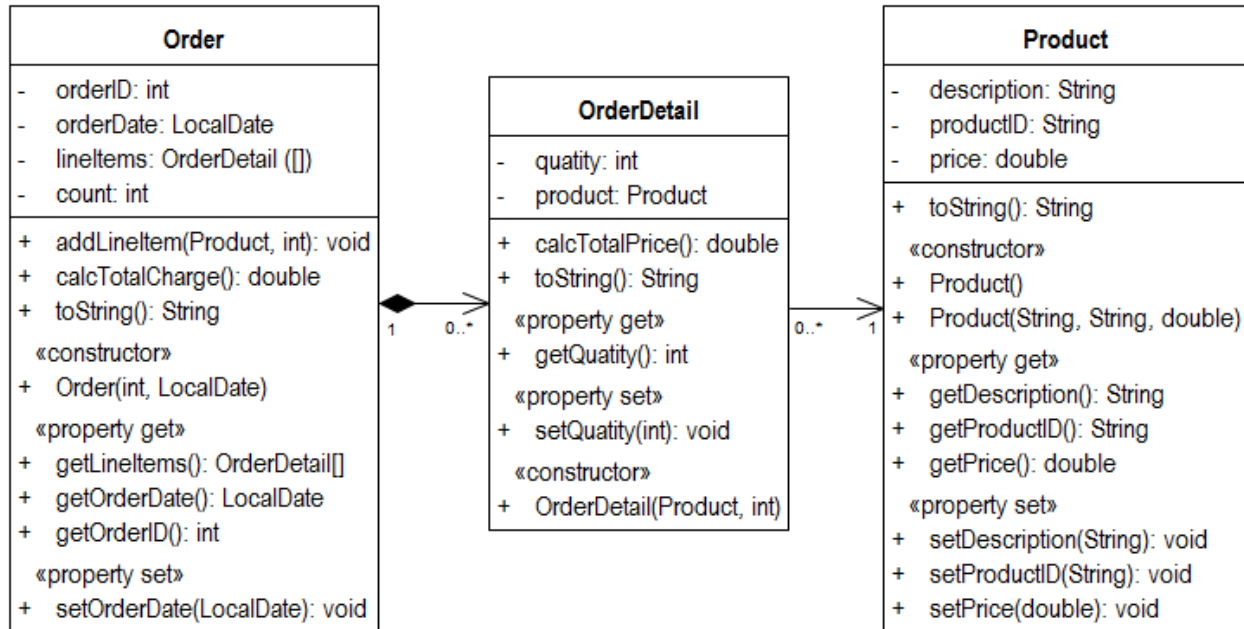
- b. Cài đặt lớp **DanhSachCongNhan**, có:

- Phương thức khởi tạo n phần tử cho lớp **DanhSachCongNhan**.
- Phương thức thêm một công nhân vào danh sách, thêm thành công nếu kích thước còn cho phép.
- Phương thức xuất toàn bộ thông tin công nhân viên.
- Phương thức tính số lượng công nhân viên có trong danh sách.
- Phương thức xuất thông tin các công nhân làm trên 200 sản phẩm.
- Phương thức sắp xếp công nhân theo số sản phẩm giảm dần.

- c. Viết hàm main sử dụng lớp **DanhSachCongNhan** theo yêu cầu sau: Cho nhập thông tin một số công nhân vào chương trình, thực thi các yêu cầu trong câu b với các đối tượng này.

**Bài 10.** Cài đặt các lớp có quan hệ Composition

Hiện thực mô hình lớp sau bằng ngôn ngữ lập trình Java.



Trong đó:

- Tính thành tiền ( $calcTotalPrice()$ ) = quantity \* price.

- Tính tổng tiền hóa đơn ( $calcTotalCharge()$ ) =  $\sum_{count=0}^n$  thành tiền.

- Phương thức  $addLineItem(Product\ p, int\ q): void$ , để thêm một sản phẩm p với số lượng q vào hóa đơn.

```

public void addLineItem(Product p, int x) {
    lineItems.add(new OrderDetail(p, x));
}
  
```

- Phần viết lớp cho phần kiểm nghiệm, in ra màn hình theo mẫu:

```

<terminated> Driver (15) [Java Application] C:\Java\jre1.8.0_25\bin\javaw.exe (Sep 10, 2015, 3:28:55 PM)
Mã HD: 1
Ngày lập hóa đơn: 10/09/2015
STT | Mã SP | Mô tả | Đơn giá | S Lượng | Thành tiền
1 | sp4 | Nước tương | 8,000 | 10 | 80,000 VND
2 | sp1 | Gạo | 18,000 | 5 | 90,000 VND
3 | sp3 | Đường | 10,000 | 1 | 10,000 VND
4 | sp1 | Gạo | 18,000 | 1 | 18,000 VND
Tổng tiền thanh toán: 198,000 VND
  
```

**Bài 11.**

Xây dựng lớp HocVien để quản lý họ tên, năm sinh, điểm 5 môn học của các học viên trong lớp học. Cho biết bao nhiêu người trong lớp được làm luận văn tốt nghiệp, bao nhiêu người thi tốt nghiệp, bao nhiêu người phải thi lại là tên các môn thi lại.

Tiêu chuẩn để xét như sau: *làm luận văn* phải có điểm trung bình lớn hơn 7 trong đó không có môn nào dưới 5; *thi tốt nghiệp* khi điểm trung bình không lớn hơn 7 và điểm các môn không dưới 5; *thi lại* khi có môn dưới 5 điểm.

### **Bài 12.**

Tại một trung tâm đào tạo, người quản lý muốn lưu trữ thông tin học viên của các khóa học, biết một khóa học có thể có tối đa 20 học viên. Thông tin học viên cần lưu trữ là: họ tên, địa chỉ, số điện thoại; thông tin khóa học là: tên khóa học, ngày mở khóa học, thời gian học, danh sách học viên. Ngoài ra người quản lý còn có nhu cầu muốn: tìm kiếm khóa học; biết một học viên đã từng học khóa học nào; các khóa học nào chưa kết thúc; các khóa học nào chưa bắt đầu...

Yêu cầu:

- a. Vẽ mô hình lớp và cài đặt các lớp đó để quản lý thông tin học viên và khóa học như mô tả trên.
- b. Viết hàm main để thực hiện theo nhu cầu của người quản lý.

## Module 1. KẾ THỪA – ĐA HÌNH

Mục tiêu:

- Hiểu và áp dụng được cách viết code kế thừa và đa hình trên Java.
- Hiểu và áp dụng được cách sử dụng mô hình lớp với mô tả kế thừa.

Yêu cầu:

- Bài làm lưu trong workspace có tên MaSV\_HoTen.
- Tạo project tên Module03.
- Mỗi bài làm trong một package có tên package là bai01, bai02, ...
- Viết documentation comment cho class.
- Bài tập bắt buộc: 1 → 8

### Bài 1.

Công ty du lịch X có quản lý thông tin các chuyến xe. Có 2 loại chuyến xe:

- Chuyến xe nội thành: Mã số chuyến, Họ tên tài xế, số xe, số tuyến, số km đi được, doanh thu.
- Chuyến xe ngoại thành: Mã số chuyến, Họ tên tài xế, số xe, nơi đến, số ngày đi được, doanh thu.

Thực hiện các yêu cầu sau:

- Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.
- Trong hàm main, tạo sẵn mỗi loại 2 chuyến xe vào danh sách (không cần cho nhập từ bàn phím). Xuất tổng doanh thu cho tất cả các chuyến xe và tổng doanh thu của từng loại chuyến xe.

### Bài 2.

Thư viện X quản lý danh sách các loại sách. Thông tin về các loại sách gồm:

- Sách giáo khoa: Mã sách, ngày nhập, đơn giá, số lượng, nhà xuất bản, tình trạng (“mới”, “cũ”).

Nếu tình trạng sách là “mới” thì: **thành tiền = số lượng \* đơn giá.**

Nếu tình trạng sách là “cũ” thì: **thành tiền = số lượng \* đơn giá \* 50%.**

- Sách tham khảo: Mã sách, ngày nhập, đơn giá, số lượng, nhà xuất bản, thuế. **Thành tiền = số lượng \* đơn giá + thuế.**

Thực hiện các yêu cầu sau:

- Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.
- Trong hàm main, tạo sẵn mỗi loại 3 cuốn sách vào danh sách (không cần cho nhập từ bàn phím), sau đó:
  - Tính tổng thành tiền cho từng loại.
  - Tính trung bình cộng đơn giá của các sách tham khảo.
  - Xuất ra các sách giáo khoa của nhà xuất bản K (yêu cầu nhập K).

**Bài 3.**

Viết chương trình quản lý danh sách các giao dịch. Hệ thống quản lý 2 loại giao dịch:

- Giao dịch vàng: Mã giao dịch, ngày giao dịch, đơn giá, số lượng, loại vàng. Thành tiền tính như sau:  

$$\text{Thành tiền} = \text{số lượng} * \text{đơn giá}.$$
- Giao dịch tiền tệ: Mã giao dịch, ngày giao dịch, đơn giá, số lượng, tỉ giá, loại tiền tệ có 3 loại: tiền Việt Nam, tiền USD, tiền Euro. Thành tiền tính như sau:
  - Nếu là tiền USD hoặc Euro thì:  $\text{thành tiền} = \text{số lượng} * \text{đơn giá} * \text{tỉ giá}.$
  - Nếu là tiền VN thì:  $\text{thành tiền} = \text{số lượng} * \text{đơn giá}.$

Thực hiện các yêu cầu sau:

- a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.
- b. Trong hàm main, tạo sẵn mỗi loại 3 giao dịch vào danh sách (không cần cho nhập từ bàn phím), sau đó:
  - Tính tổng số lượng cho từng loại.
  - Tính trung bình thành tiền của giao dịch tiền tệ.
  - Xuất ra các giao dịch có đơn giá > 1 tỷ.

**Bài 4.**

Viết chương trình quản lý danh sách các giao dịch nhà đất. Thông tin bao gồm:

- Giao dịch đất: Mã giao dịch, ngày giao dịch, đơn giá, loại đất (“A”, “B”, “C”), diện tích.
  - Nếu là loại B, C thì:  $\text{thành tiền} = \text{diện tích} * \text{đơn giá}.$
  - Nếu là loại A thì:  $\text{thành tiền} = \text{diện tích} * \text{đơn giá} * 1.5.$
- Giao dịch nhà: Mã giao dịch, ngày giao dịch, đơn giá, loại nhà (“cao cấp”, “thường”), địa chỉ, diện tích.
  - Nếu là loại nhà cao cấp thì:  $\text{thành tiền} = \text{diện tích} * \text{đơn giá}.$
  - Nếu là loại thường thì:  $\text{thành tiền} = \text{diện tích} * \text{đơn giá} * 90\%.$

Thực hiện các yêu cầu sau:

- a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.
- b. Trong hàm main, tạo sẵn mỗi loại 3 giao dịch vào danh sách (không cần cho nhập từ bàn phím), sau đó:
  - Tính tổng số lượng cho từng loại.
  - Tính trung bình thành tiền của giao dịch đất.
  - Xuất ra các giao dịch của tháng 9 năm 2013.

**Bài 5.**

Viết chương trình quản lý danh sách hoá đơn tiền điện của khách hàng. Thông tin bao gồm các loại khách hàng:

- Khách hàng Việt Nam: mã khách hàng, họ tên, ngày lập hoá đơn, đối tượng khách hàng (“sinh hoạt”, “kinh doanh”, “sản xuất”), số KW tiêu thụ, đơn giá, định mức. Thành tiền được tính như sau:
  - Nếu số KW tiêu thụ  $\leq$  định mức thì thành tiền = số KW tiêu thụ \* đơn giá.
  - Ngược lại, thành tiền = số KW tiêu thụ \* đơn giá \* định mức + số KW vượt định mức \* đơn giá \* 2.5.
- Khách hàng nước ngoài: mã khách hàng, họ tên, ngày lập hoá đơn, quốc tịch, số KW tiêu thụ, đơn giá. Thành tiền được tính = số KW tiêu thụ \* đơn giá.

Thực hiện các yêu cầu sau:

- Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.
- Trong hàm main, tạo sẵn mỗi loại 3 khách hàng vào danh sách (không cần nhập từ bàn phím), sau đó:
  - Tính tổng số lượng cho từng loại khách hàng.
  - Tính trung bình thành tiền của khách hàng người nước ngoài.
  - Xuất ra các hoá đơn trong tháng 09 năm 2018 (của cả 2 loại khách hàng).

### Bài 6. \*

Một khách sạn X cần quản lý các hóa đơn của khách hàng thuê phòng. Hóa đơn có 2 loại: hóa đơn theo giờ, hóa đơn theo ngày. Thông tin chung của chi tiết hóa đơn là: *mã hóa đơn, ngày hóa đơn, tên khách hàng, mã phòng, đơn giá*. Thông tin riêng của từng loại hóa đơn gồm:

- Hóa đơn theo giờ còn có *số giờ thuê*. Thành tiền = số giờ thuê \* đơn giá. Nếu trường hợp số giờ > 24 tiếng và < 30 tiếng thì cũng chỉ tính 24 giờ. Nếu trường hợp số giờ > 30 tiếng thì không dùng loại hóa đơn theo giờ (phát sinh ngoại lệ).
- Hóa đơn theo ngày sẽ có *số ngày thuê*. Thành tiền = số ngày thuê \* đơn giá. Nếu số ngày > 7 thì giảm 20% đơn giá cho những ngày còn lại.

Thực hiện các yêu cầu sau:

- Vẽ mô hình và cài đặt các lớp với chức năng thừa kế theo mô tả trên.
- Cài đặt thêm lớp chứa danh sách các hóa đơn theo mô hình sau:

DanhSachHoaDon
- danhSach : HoaDon[]
- count : int
+ DanhSachHoaDon(n: int)
+ them(hd: HoaDon) : boolean
+ xuat() : void
+ thongKeSoLuongHDTTheoGio() : int
+ thongKeSoLuongHDTTheoNgay() : int

Lưu số lượng hóa đơn hiện tại đang có trong danh sách.

Trả về *true* nếu thêm thành công (không trùng mã\*).

Xuất toàn bộ các hóa đơn theo dạng bảng, mỗi hóa đơn trên một dòng.



```
+ tinhTongThanhTien(thang : int, nam : int) :
double
```

(\*) Yêu cầu override phương thức equals của lớp Object để kiểm tra trùng mã.

c. Tạo menu case trong hàm main cho phép thực hiện các chức năng trong câu b.

### Bài 7. \*

**Hàng hóa** trong kho của một siêu thị gồm có **hàng thực phẩm**, **hàng sành sứ** và **hàng điện máy**.

Mỗi loại hàng đều có: **mã hàng** (không được sửa, không được để trống), **tên hàng** (không được rỗng, mặc định là “xxx”), **đơn giá** ( $>=0$ ), **số lượng tồn** ( $>=0$ ).

Hàng thực phẩm thì cần quan tâm đến thông tin: **nhà cung cấp**, **ngày sản xuất** (phải trước ngày hiện tại, mặc định là ngày hiện tại) và **ngày hết hạn** (phải sau ngày sản xuất, mặc định là ngày sản xuất).

Hàng điện máy cần biết: **thời gian bảo hành** bao nhiêu tháng ( $>=0$ ), **công suất** bao nhiêu KW ( $>=0$ ).

Hàng sành sứ thì cần biết thông tin về: **nhà sản xuất** và **ngày nhập kho** (phải trước ngày hiện tại, mặc định là ngày hiện tại).

Ngoài ra, người quản lý cần quan tâm đến số lượng tồn kho và các yếu tố khác của từng loại hàng hóa để **đánh giá mức độ bán buôn**, tiền **VAT** từng loại hàng hóa. Biết rằng VAT của hàng điện máy và sành sứ là 10%, VAT của hàng thực phẩm là 5%.

a) Dựa vào các thông tin trên, hãy xác định:

- Các lớp có thể có, lớp nào là lớp trừu tượng (abstract class), lớp nào là lớp cụ thể.
- Các thuộc tính cho từng lớp.
- Các phương thức cho từng lớp (phương thức nào là phương thức trừu tượng (abstract method), danh sách các tham số có thể có cho từng phương thức và kiểu trả về của phương thức).
- Thiết kế mô hình lớp (xây dựng cây thừa kế, các interface nếu có).

b) Thực hiện cài đặt cho mỗi loại hàng cụ thể trên. Trong đó, để **đánh giá mức độ bán buôn** thì:

- Hàng thực phẩm, nếu vẫn còn tồn kho và bị hết hạn thì đánh giá là *khó bán*.
- Hàng điện máy, nếu số lượng tồn kho  $<3$  thì được đánh giá là *bán được*.
- Hàng sành sứ, nếu số lượng tồn kho  $>50$  và thời gian lưu kho  $>10$  ngày thì đánh giá là *bán chậm*.
- Các trường hợp còn lại xem như *không đánh giá*.

c) Hãy viết lớp quản lý **danh sách hàng hóa**. Dùng mảng để lưu trữ danh sách hàng hóa.

- Tạo constructor **khởi tạo** danh sách n phần tử.
- Viết phương thức **thêm** một hàng hóa vào danh sách (*thêm thành công nếu không bị trùng mã hàng*).
- Viết phương thức **in toàn bộ** danh sách các hàng hóa.
- Viết các phương thức **in từng loại** hàng hóa.
- Viết phương thức **tìm kiếm** hàng hóa khi biết mã hàng (trả về hàng hóa tìm thấy).
- Viết phương thức **sắp xếp** hàng hóa theo tên hàng tăng dần.
- Viết phương thức **sắp xếp** hàng hóa theo số lượng tồn giảm dần.
- Viết phương thức **xuất** các hàng thực phẩm khó bán.

- Viết các phương thức **xóa** hàng hóa khi biết mã hàng.
- Viết các phương thức **sửa** thông tin đơn giá của hàng hóa khi biết mã hàng.

d) Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng trong câu c.

HD:

*Dùng `Arrays.sort` và interface `Comparator` để sắp xếp.*

### Bài 8.

Giả sử cần xây dựng chương trình quản lý dùng cho một học viện nghiên cứu giảng dạy và ứng dụng. Đối tượng quản lý bao gồm các sinh viên đang theo học, các nhân viên đang làm việc tại học viện, các khách hàng đến giao dịch mua bán sản phẩm ứng dụng. Dựa vào một số đặc tính của từng đối tượng, người quản lý cần đưa ra cách thức đánh giá khác nhau.

Hãy xây dựng các lớp sau:

- Lớp **Person**: bao gồm các thành phần *họ tên*, *địa chỉ*, phương thức *toString*.
- Các lớp **Student**, **Employee**, **Customer** (theo mô tả bên dưới) thừa kế lớp **Person**.
  - Lớp **Student**: bao gồm các thuộc tính *điểm môn học 1*, *điểm môn học 2*; các phương thức: *tính điểm trung bình*, *đánh giá*, *toString* trả về bảng điểm sinh viên (gồm thông tin thuộc tính và điểm trung bình).
  - Lớp **Employee**: bao gồm thuộc tính *hệ số lương*; các phương thức: *tính lương*, *đánh giá*, *toString* trả về bảng lương cho nhân viên (gồm thông tin thuộc tính và tiền lương).
  - Lớp **Customer**: bao gồm thuộc tính *tên công ty*, *trị giá hóa đơn*, *đánh giá*; phương thức *toString* trả về thông tin hóa đơn cho khách hàng (gồm các thuộc tính của đối tượng).
- Lớp **Management** lưu thông tin toàn bộ các sinh viên, nhân viên, khách hàng và tổng số người hiện tại có trong danh sách. Ngoài ra còn có các phương thức:
  - Constructor khởi tạo mảng với n phần tử.
  - Thêm một người vào danh sách.
  - Xóa một người khỏi danh sách (nhận thông số là họ tên của người cần xóa).
  - Sắp xếp danh sách theo thứ tự họ tên.
  - Xuất danh sách theo dạng bảng.
- Viết lớp Test có hàm main cho phần kiểm nghiệm. Giao tiếp với người dùng bằng menu (thể hiện tính đa hình – polymorphism bằng cách cho phép lựa chọn nhập thông tin là sinh viên, nhân viên hay khách hàng).

### Bài 9.

Tạo lớp trừu tượng **Shape** với 3 phương thức trừu tượng **draw()**, **erase()** và **move(int x, int y)**. Tạo các lớp con như liệt kê ở bảng dưới đây đồng thời override các phương thức trừu tượng (các phương thức này chỉ in câu thông báo tương ứng ra console).

Class	Superclass	Subclass
Shape	-	Circle, Quad, Triangle, Polygon
Circle	Shape	-

Quad	Shape	Rectangle
Rectangle	Quad	-
Triangle	Shape	-
Polygon	Shape	-

Viết lớp **Drawing** có phương thức **drawShape**(Shape *theShape*), phương thức có tham số là đối tượng Shape. Trong phương thức gọi tới **draw()** của từng đối tượng Shape. Thực thi phương thức này.

### Bài 10.

Với một tập mini các loại xe trong thế giới thực cho bên dưới:



Yêu cầu quản lý:

- Thông tin từng loại xe.
- Tính tiền thuế cho từng loại xe dựa trên giá trị xe như sau:
  - o **Xe đạp:** Không đóng thuế.
  - o **Xe máy:** gồm VAT=10% và thuế trước bạ 5%.
  - o **Xe ô tô khách:** gồm thuế tiêu thụ đặc biệt 30% (nếu số chỗ >=5), 50% (nếu số chỗ <5); VAT=10% và thuế trước bạ 20%.
  - o **Xe ô tô tải:** gồm VAT=10% và thuế trước bạ 2%.

Yêu cầu sinh viên:

- a. Thiết kế lược đồ lớp theo mô tả trên.
- b. Cài đặt cho lược đồ đã thiết kế.

### Bài 11.

Trường đại học X có nhiều cấp đào tạo, thông tin về sinh viên được tổ chức như sau:

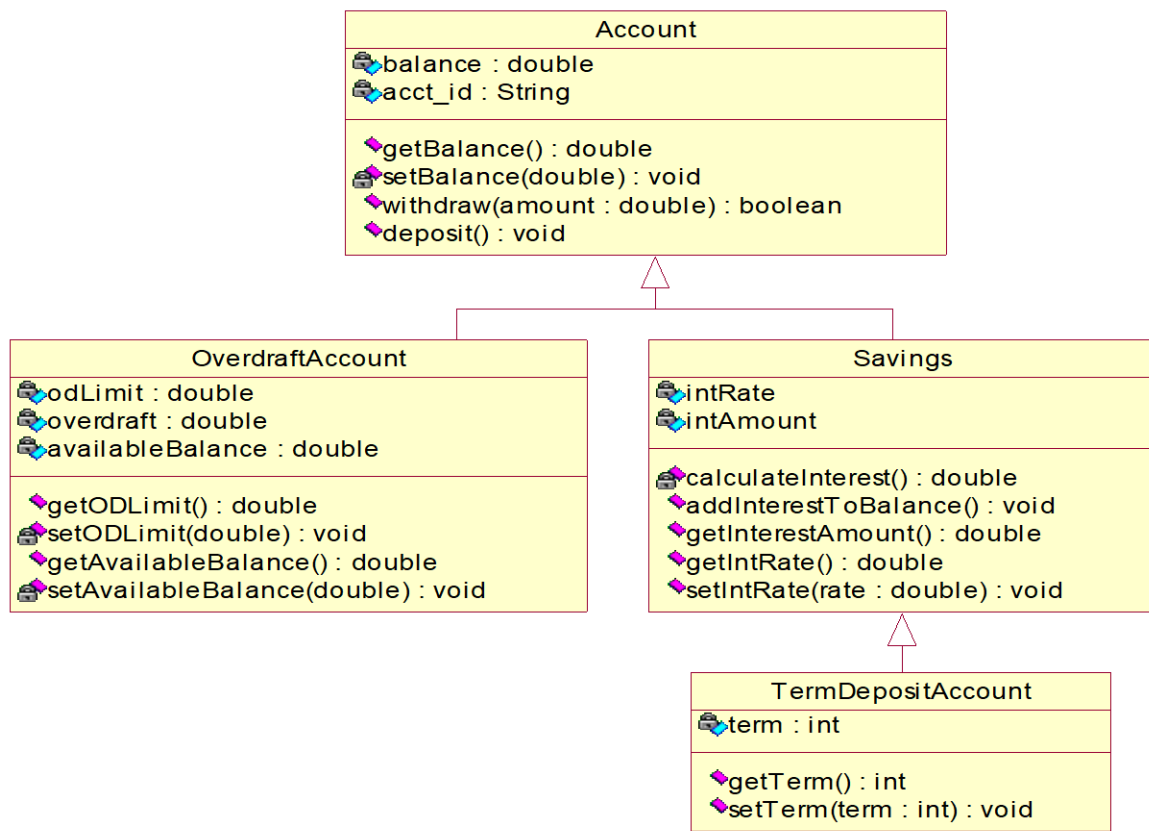
- Sinh viên trung cấp (trung cấp 2 năm)
- Sinh viên cao đẳng
- Sinh viên đại học
- Sinh viên liên kết (Australia, US)

## Yêu cầu

- Xác định các thông tin, dữ liệu chung, riêng.
- Định nghĩa các lớp (các thuộc tính, và phương thức) và mô hình phân cấp các lớp.
- Viết lớp cho phép nhập và hiển thị thông tin về sinh viên.

**Bài 12.**

Cho mô hình sau:



- Sử dụng mô hình lớp ở trên cho cây phân cấp của các đối tượng Account, hãy tạo các phương thức để thi hành phần thiết kế này. Cung cấp các đoạn mã lệnh để thực thi các phương thức.
- Viết chương trình kiểm tra trong lớp BankingServices, tạo các đối tượng thuộc các lớp cụ thể và kiểm tra các phương thức.

## Module 2. TẬP HỢP

Mục tiêu:

- Khai báo và khởi tạo tập hợp
- Các thao tác thêm, xóa, sửa, duyệt trên tập hợp
- Các thuật toán sắp xếp, tìm kiếm trên tập hợp

Yêu cầu:

- Bài tập bắt buộc: 1,2,3,4,5,6,7
- Viết documentation comment cho class.

Tóm tắt bài học

Interface	Description	Concrete Classes
Collection	A basic interface that defines the normal operations that allow a collection of objects to be maintained or handled as a single unit.	
Set	The Set interface extends the Collection interface to represent its mathematical namesake: a set of <b>unique</b> elements.	HashSet LinkedHashSet
SortedSet	The SortedSet interface extends the Set interface to provide the required functionality for maintaining a set in which the elements are stored in some <b>sorted</b> order.	TreeSet
List	The List interface extends the Collection interface to maintain a <b>sequence</b> of elements that need not be unique.	ArrayList Vector LinkedList
Map	A basic interface that defines operations for maintaining mappings of <b>keys to values</b> .	HashMap Hashtable LinkedHashMap
SortedMap	Extends the Map interface for maps that maintain their mappings <b>sorted</b> in key order.	TreeMap

**Bài 1.**

Thực hiện các yêu cầu sau:

- Công ty TrueLove cần lưu tên của các nhân viên của mình. Mỗi tháng một nhân viên sẽ được chọn ngẫu nhiên để nhận một quà tặng. Hãy dùng tập hợp viết chương trình quản lý danh sách nhân viên.
- Công ty TrueLove cần đặt tên cho sản phẩm mới, tên sản phẩm được chọn từ tên của nhân viên, vì vậy tên sản phẩm không được trùng và chỉ được dùng một lần. Hãy dùng tập hợp viết chương trình cung cấp tên cho sản phẩm.
- Công ty TrueLove muốn dùng tên phổ biến nhất cho sản phẩm của họ, tên phổ biến là tên giống nhau nhiều nhất. Hãy dùng tập hợp viết chương trình cung cấp tên cho sản phẩm.
- Công ty TrueLove muốn cho nhân viên đi du lịch, chính sách được tạo ra là ưu tiên cho những người đăng ký trước. Hãy dùng tập hợp viết chương trình đăng ký du lịch.
- Công ty TrueLove muốn tạo danh sách các khách hàng theo thứ tự tăng dần theo doanh số. Hãy dùng tập hợp viết chương trình quản lý danh sách khách hàng.

**Bài 2.**

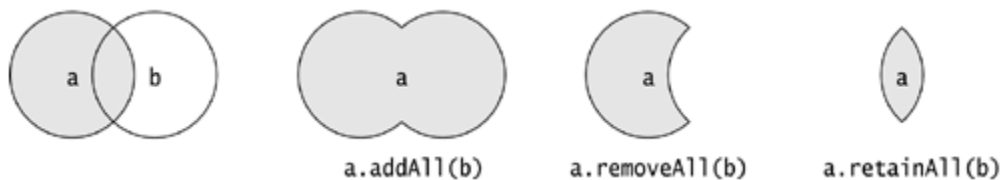
Viết phương thức nhận vào một chuỗi và trả về số ký tự duy nhất của chuỗi. Viết chương trình cho nhập vào nhiều chuỗi và xuất kết quả tương ứng.

Yêu cầu: Các chuỗi nhập vào có thể trùng nhau nên để tiết kiệm thời gian đếm ký tự, người ta muốn lưu lại các kết quả số ký tự của từng chuỗi để khi gặp phải chuỗi giống trước đó, chương trình chỉ truy xuất đến kết quả đã tính được. Lựa chọn sử dụng collection và map thích hợp.

**Bài 3.**

Viết chương trình tạo hai tập hợp số nguyên (Set). Tính giao, hội, hiệu hai tập trên, xuất kết quả tăng dần.

HD: Dùng TreeSet



- `a.addAll(b)` → tập a hội tập b
- `a.removeAll(b)` → tập a trừ tập b
- `a.retainAll(b)` → tập a giao tập b

**Bài 4.**

Tạo một ArrayList chứa tập các sinh viên, mỗi sinh viên gồm các thông tin mã, họ tên, năm sinh. Thêm vào danh sách 5 sinh viên tùy ý (không được trùng mã). Viết các phương thức: in danh sách sinh viên dạng bảng; thêm một sinh viên mới; xóa sinh viên khi biết mã; sửa thông tin sinh viên (không sửa mã); tìm kiếm sinh viên theo mã, theo tên, sắp xếp danh sách theo mã tăng dần.

**Bài 5.**

Tùy chọn sửa lại một trong các bài tập ở Module 3: lựa chọn sử dụng collection thích hợp để lưu trữ (không dùng mảng thông thường).

**Bài 6. \***

Phòng học được quản lý trong một trường đại học gồm: phòng học lý thuyết, phòng máy tính và phòng thí nghiệm. Mỗi phòng học đều có *mã phòng, dãy nhà, diện tích, số bóng đèn*. Ngoài ra còn:

- Phòng học lý thuyết thì cần quan tâm xem *có máy chiếu không*.
- Phòng máy tính thì cần biết là trang bị *bao nhiêu máy tính*.
- Phòng thí nghiệm thì thêm thông tin *chuyên ngành, sức chứa, có bồn rửa không* (để rửa dụng cụ thí nghiệm/rửa tay).

Thêm nữa, người quản lý cần phải xem xét phòng học *có đạt chuẩn không*. Phòng học đạt chuẩn nếu: tất cả các phòng đều phải đủ ánh sáng (*trung bình  $10m^2$  - 1 bóng đèn*), và:

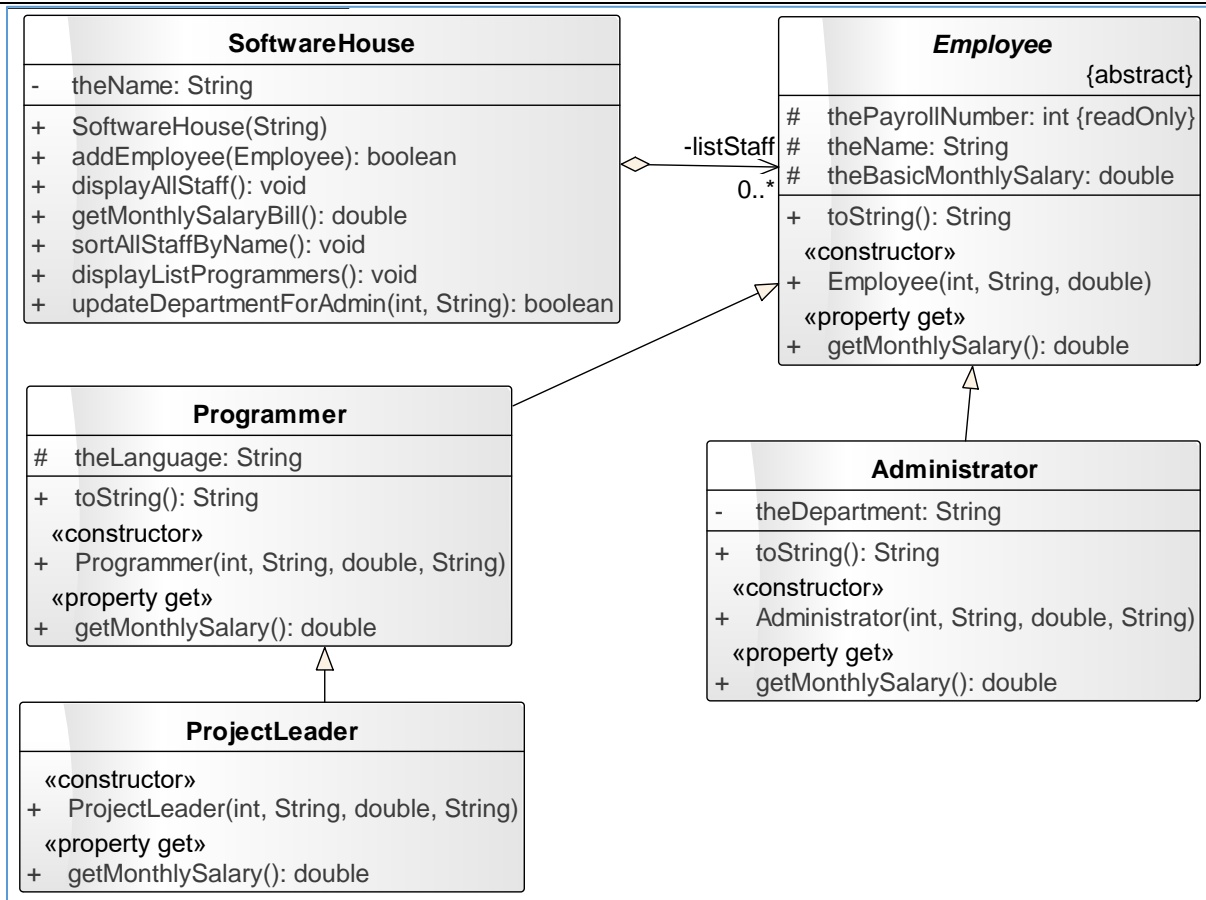
- Phòng lý thuyết: phải có máy chiếu.
- Phòng máy tính: trung bình  $1.5m^2$  đặt một máy.
- Phòng thí nghiệm: phải có bồn rửa đi kèm.

Yêu cầu:

- a. Thiết kế và thực hiện cài đặt tường minh cho mỗi loại phòng được mô tả trên.
- b. Viết lớp **quản lý danh sách phòng học**. Yêu cầu dùng một List (ArrayList, LinkedList, Vector) để lưu trữ danh sách phòng học. Trong đó:
  - Tạo constructor **khởi tạo** danh sách.
  - Viết phương thức **thêm** một phòng học vào danh sách (*thêm được nếu không trùng mã phòng*).
  - Viết phương thức **tìm kiếm** một phòng học nào đó khi biết mã phòng.
  - Viết phương thức **lấy thông tin toàn bộ** danh sách các phòng học.
  - Viết các phương thức để **lấy thông tin các phòng học đạt chuẩn**.
  - Viết phương thức để **sắp xếp danh sách tăng dần theo dãy nhà**.
  - Viết phương thức để **sắp xếp danh sách giảm dần theo diện tích**.
  - Viết phương thức để **sắp xếp danh sách tăng dần theo số bóng đèn**.
  - Viết phương thức để **cập nhật** số máy tính cho một phòng máy tính nào đó khi biết mã phòng.
  - Viết phương thức để **xóa** một phòng học nào đó khi biết mã phòng. *Lưu ý khi test chương trình, khi xóa cần phải xác minh rằng có chắc chắn xóa không*.
  - Viết phương thức để **tính tổng số phòng học**.
  - Viết các phương thức để **lấy danh sách các phòng máy có 60 máy**.
- c. Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng theo yêu cầu.

**Bài 7. \***

Hiện thực theo mô hình lớp bên dưới bằng ngôn ngữ lập trình Java.



Đây là một phần ứng dụng quản lý nhân viên trong một công ty phần mềm (*SoftwareHouse*). Công ty có nhiều nhân viên. Mỗi nhân viên (*Employee*) cần lưu trữ các thông tin: Mã số (*thePayrollNumber*), tên nhân viên (*theName*), lương cơ bản hằng tháng (*theBasicMonthlySalary*). Mã số của mỗi nhân viên chỉ được tạo một lần duy nhất và không được phép sửa.

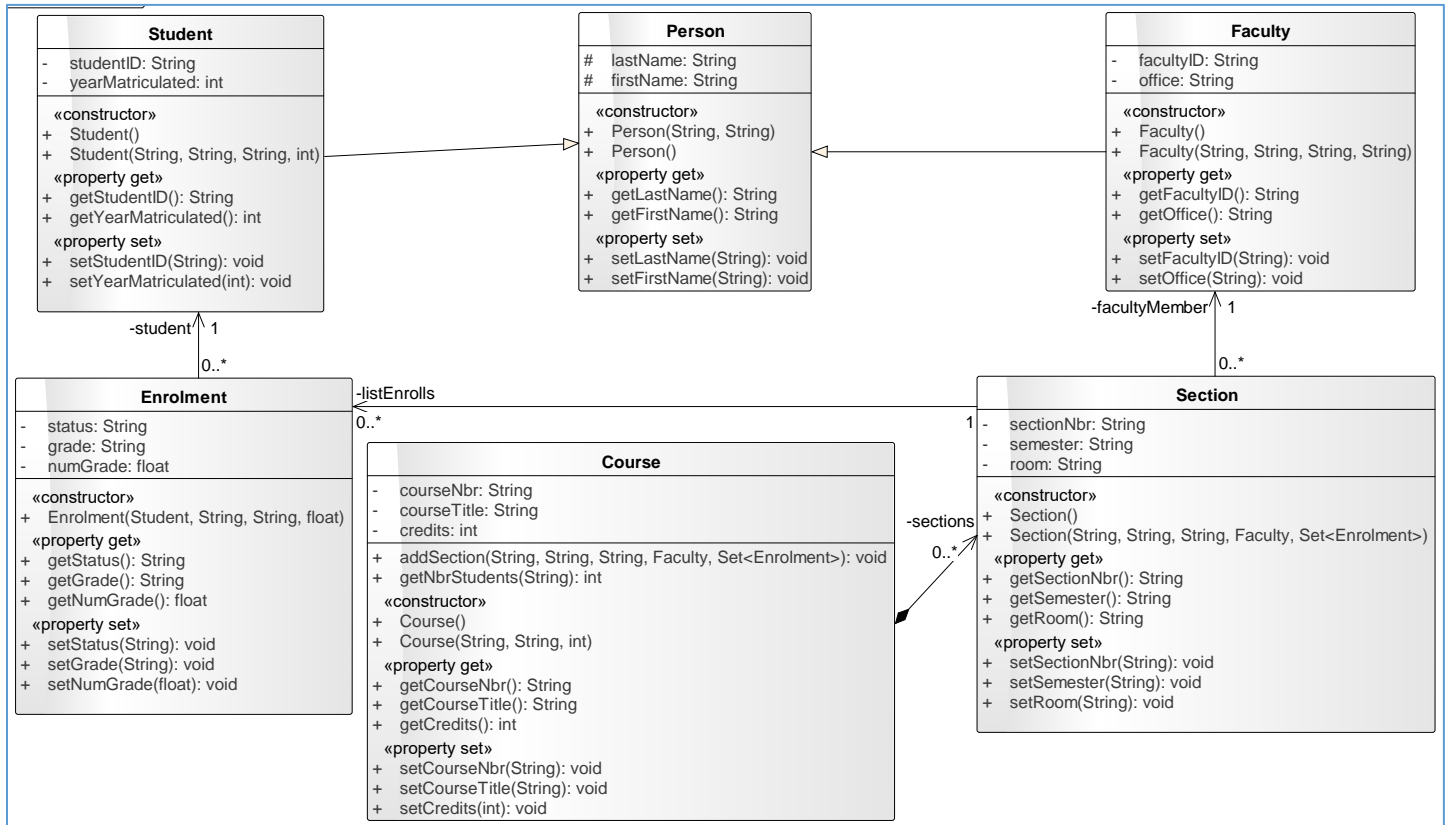
- Nhân viên trong công ty gồm: Lập trình viên (*Programmer*), Người quản lý (*Administrator*) và Người trưởng dự án (*ProjectLeader*).
  - Tạo các phương thức setters/getters cho các thuộc tính của các lớp.
  - Tiền lương hằng tháng (*getMonthlySalary*) là tiền lương cơ bản + phụ cấp. Phụ cấp được tính như sau:
    - Lập trình viên: Nếu ngôn ngữ lập trình là “Java” thì phụ cấp 20% của lương cơ bản.
    - Người quản lý: Phụ cấp 40% của lương cơ bản
    - Người trưởng dự án: Phụ cấp 20% lương cơ bản.
- Công ty phần mềm (*SoftwareHouse*) có tên gọi (*theName*), và danh sách các nhân viên. Tạo:
  - Constructor *SoftwareHouse(aName : String)*, tạo một công ty có tên là *aName* và khởi tạo danh sách nhân viên (mỗi phần tử trong mảng là một nhân viên).
  - Phương thức *addEmployee(emp : Employee): boolean*, dùng để thêm một nhân viên *emp* vào công ty. Thêm thành công nếu không trùng mã số.
  - Phương thức *displayAllStaff(): void*, hiển thị toàn bộ nhân viên trong công ty lên màn hình theo dạng cột, định dạng đơn vị tiền tệ là \$, phân cách hàng nghìn.
  - Phương thức *getMonthlySalaryBill(): double*, tính tổng tiền phải trả cho các nhân viên.
  - Phương thức *sortAllStaffByName(): void*, sắp xếp danh sách nhân viên theo tên.



- f. Phương thức *displayListProgrammers()*: *void*, hiển thị danh sách các lập trình viên.
- g. Phương thức *updateDepartmentForAdmin(aPayrollNo: int, deptNew: String)*: *boolean*, cập nhật phòng ban là *deptNew* cho người quản lý có mã số là *aPayrollNo*, trả về *true* nếu cập nhật thành công.
- h. Phương thức *deleteEmployee(id : int)*: *boolean*, xóa nhân viên theo mã số, trả về *true* nếu xóa được.
- c. Viết chương trình chính tạo menu case để thực hiện các chức năng trong câu b.

### Bài 8.

Quản lý thông tin các khóa học và thông tin sinh viên tham gia đăng ký học các khóa học.



Lớp **Person** (người) là lớp cơ sở. Lớp **Student** (sinh viên) và **Faculty** (giảng viên) là hai lớp dẫn xuất từ lớp này.

Mỗi học phần nhiều sinh viên đăng ký tham gia học (**Enrolment**). Sinh viên tham gia học phần sẽ có một trạng thái (*status*), một điểm chữ (*grade*) và một điểm số (*numGrade*).

Trong đó, phương thức **addSection** dùng để thêm một học phần cho khóa học:

*addSection(String sectionNbr, String semester, String room, Faculty facultyMember, Set<Enrolment> listEnrolls): void*

HD: Mapping to java code

```
public class Person {
    private String lastName;
    private String firstName;
}
```

```
public class Student extends Person {
    private int yearMatriculated;
    private String studentID;
}
```

<pre> public class Faculty extends Person {     private String <u>office</u>;     private String <u>facultyID</u>; } </pre>	<pre> public class Section {     private String <u>semester</u>;     private String <u>sectionNbr</u>;     private String <u>room</u>;     private Faculty <u>facultyMember</u>;     private Set&lt;Enrolment&gt; <u>listEnrolls</u>; } </pre>
<pre> public class Enrolment {     private String <u>status</u>;     private String <u>grade</u>;     private float <u>numGrade</u>;     private Student <u>student</u>; } </pre>	<pre> public class Course {     private String <u>courseNbr</u>;     private String <u>courseTitle</u>;     private int <u>credits</u>;     private Set&lt;Section&gt; <u>sections</u>;      public int getNbrStudents(String sectionNbr){}      public void addSection(String sectionNbr,         String semester, String room, Faculty         facultyMember, Set&lt;Enrolment&gt; listEnrolls){} } </pre>

Viết lớp cho phần thử nghiệm:

1. Tạo danh sách giảng viên.
2. Tạo danh sách sinh viên.
3. Tạo khóa học.
4. Lập danh sách sinh viên tham gia vào 1 học phần của khóa học do 1 giảng viên giảng dạy.
5. In ra màn hình theo mẫu:

```

===== Thông tin học phần =====
Khóa học : [OOP - Lập trình hướng đối tượng (4TC)]
Mã học phần: 0602
Học kỳ: I (2015 - 2016)
Phòng học: H5.01
Giảng viên: Lê Kim Khánh (Khoa : CNTT)
===== Danh sách sinh viên =====
Mã SV   Họ tên      Khóa năm   Điểm
140211   Hoàng Dũng    2014       8.5
140511   Trần Bình     2014       9.5
140811   Lê Huỳnh     2014       7.0
140611   Hồ Huyền     2013       5.5

Tổng số sinh viên : 4

```

## Bài 9.

Viết lớp mô tả các toán tử trên danh sách liên kết sử dụng LinkedList. Chương trình tạo hai LinkedList chứa thông tin là các chuỗi String. Các phần tử của danh sách này được đưa vào danh sách kia. Các chuỗi trong danh sách được chuyển sang chữ hoa, xóa các phần tử.

## Bài 10.

Viết chương trình tra cứu danh bạ điện thoại, sử dụng cấu trúc collection cho phù hợp để lưu trữ thông tin của danh bạ và dễ dàng thực hiện công việc:

- Tra cứu theo địa chỉ, một địa chỉ có thể đăng kí nhiều số điện thoại cố định.
- Tra cứu theo số điện thoại.

**Bài 11.**

Quản lý khách hàng xếp hàng mua vé tại nhà ga. Thông tin lưu trữ cho khách hàng gồm: *số CMND (String), tên khách hàng, ga đến, giá tiền (double)*.

Tạo hệ thống menu gồm các mục:

- Thêm một khách hàng mới vào hàng đợi mua vé.
- Bán một vé cho khách hàng. Chỉ bán cho người đăng ký trước.
- Hiện thị danh sách khách hàng.
- Hủy một khách hàng ra khỏi danh sách (khách hàng không mua vé nữa).
- Thống kê tình hình bán vé.
- Lưu danh sách vào file.
- Hiện thị danh sách các ga đang chờ mua vé.
- Hiện thị danh sách các ga đang chờ mua vé và số vé tương ứng cho ga.

Lưu ý:

- Số khách hàng trong danh sách hiện tại là số khách đang chờ, nhưng chưa có vé. Khi một khách hàng đã mua vé, thì loại khách hàng này ra khỏi danh sách chờ mua vé.
- Việc mua vé phải có thứ tự: ai vào trước thì mua vé trước (FIFO).
- Mỗi khi khách hàng mua được vé phải lưu lại khách hàng này để dùng cho việc thống kê.
- Mỗi khi thêm một khách hàng mới, nếu số CMND khách hàng đã có thì không tạo phần tử mới mà chỉ cập nhật lại ga và giá tiền đến cho khách hàng đó.
- Mục thống kê tình hình: cho biết còn bao nhiêu khách hàng chờ nhận vé, bao nhiêu khách hàng đã nhận vé, tổng số tiền đã thu về là bao nhiêu.
- Việc lưu danh sách: chỉ lưu các khách hàng chờ mua vé. Các khách hàng đã nhận vé xem như kết sổ trong ngày không cần lưu lại.
- Khi chương trình vừa được chạy, lập tức tự động nạp toàn bộ danh sách khách hàng từ file (cách khách hàng chưa có vé).
- Khi hiện thị danh sách các ga đến đang chờ mua vé, chỉ hiện thị tên ga đó một lần. (Ví dụ: giả sử 10 khách hàng nhưng đăng ký đi đến 2 ga, thì chỉ hiện thị 2 hàng).