LAB 2: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

I. MỤC TIÊU:

- Tìm hiểu các khái niệm trong lập trình hướng đối tượng:
  - Thuộc tính: nội tại (instant variable), class (class variable), tham chiếu (reference variable)
  - Phương thức: Hàm dựng, Get/Set, hành vi
  - Phương thức: toString(), hashCode() & equals()
- Tính đóng gói, tính kế thừa, tính đa hình trong java
- Quy tắc đặt tên trong Java
- Cấu trúc tập: ArrayList, HashSet, HashMap
- Mô hình quan hệ

II. TÓM TẮT LÝ THUYẾT:

A. Cái khái niệm trong lập trình hướng đối tượng

1. **Lớp và đối tượng:**

a. Khai báo lớp:

```
class <TenLop> {
        Thuộc tính 1
        Thuộc tính 2
        …
        Hàm dựng 1
        Hàm dựng 2
        …
        Hành động 1
        Hành động 2
        …
}
```

b. Tạo đối tượng:

```
<TenLop> a = new TenLop();
```

2. **Thuộc tính:**

a. Nội tại:

- Khai báo:

```
class <TenLop> {

    //Khai báo thuộc tính đối tượng

    <kieu_du_lieu>   ten_bien1;
    <kieu_du_lieu>   ten_bien2;
            …

}
```

- Truy cập:

```
<TenLop>  a = new <TenLop>();

a.ten_bien1;

a.ten_bien2;
```

b. Lớp:

- Khai báo:

```
class  <TenLop> {
        …
        static <kieu_du_lieu>   ten_bien_lop;
}
```

- Truy cập:

```
<TenLop>.ten_bien_lop;
```

c. Tham chiếu:

- Khai báo:

```
class <TenLop> {
        //Khai báo thuộc tính đối tượng
        <TenLopThamChieu>   doi_tuong_1;
        <TenLopThamChieu>   doi_tuong_2;
          …
}
```

- Truy cập:

```
<TenLop>  b = new <TenLop>();
b.doi_tuong_1.phuongthuc();
b.doi_tuong_2.phuongthuc();
```

3. **Phương thức:**

a. *Phương thức của đối tượng:*

- Khai báo:

```
class <TenLop> {
        Thuộc tính 1
        Thuộc tính 2
        …
        Hàm dựng 1
        Hàm dựng 2
        …
        //Khai báo các phương thức của đối tượng
        <kieu_du_lieu> TenPhuongThuc1(tham số) {
                //Định nghĩa
        }
        <kieu_du_lieu> TenPhuongThuc1(tham số) {
                //Định nghĩa
        }
        …
}
```

- Truy cập:

```
<TenLop>  a = new <TenLop>();
a.TenPhuongThuc1(tham số);
a.TenPhuongThuc2(tham số);
```

b. Phương thức lớp:

- Khai báo:

```
class <TenLop> {
        Thuộc tính 1
        Thuộc tính 2
        …
        Hàm dựng 1
        Hàm dựng 2
        …
        //Khai báo các phương thức của lớp
        static <kieu_du_lieu> TenPhuongThuc1(tham số) {
                //Định nghĩa
        }
        static <kieu_du_lieu> TenPhuongThuc1(tham số) {
                //Định nghĩa
        }
        …
}
```

- Truy cập:

```
<TenLop>.TenPhuongThuc1(tham số);
<TenLop>.TenPhuongThuc2(tham số);
```

4. **Hàm dựng:**
   - Dùng để khởi gán giá trị ban đầu cho đối tượng, cùng tên với tên lớp, không có kiểu trả về (kể cả kiểu void)
   - Khai báo:

```
class <TenLop> {
        Thuộc tính 1
        Thuộc tính 2
        …
        <TenLop>() {
                //Định nghĩa
        }
        <TenLop>(tham số) {
                //Định nghĩa
        }
        …
}
```

   - Gọi hàm dựng:

```
<TenLop> a = new <TenLop>();
<TenLop> b = new <TenLop>(tham số)
```

5. **Phương thức Get/Set:**
   - **Bao đóng** : là khả năng che dấu thông tin của đối tượng.
   - Sử dụng các quyền truy cập trong java để che dấu thông tin:

Modifier	Class	Package	Subclass	World
public	✔	✔	✔	✔
protected	✔	✔	✔	✘
*no modifier**	✔	✔	✘	✘
private	✔	✘	✘	✘

   - **private**: dùng để che dấu
   - **public**: để bên ngoài sử dụng
   - Cung cấp phương thức **Get**(để đọc) / **Set**(để ghi) cho các thuộc tính private.

6. **Phương thức toString(), hashCode() & equals():**
   - Được thừa kế từ lớp Object.
   - *Phương thức toString()*: chuyển đối tượng thành đối tượng String, thường được dùng để mô tả đối tượng
   - *Phương thức hashCode() & equals():* dùng để định danh đối tượng, mặc định java dùng địa chỉ bộ nhớ của đối tượng để định danh, trên thực tế ta dùng thuộc tính khóa để định danh đối tượng

7. **Code conventions trong java**

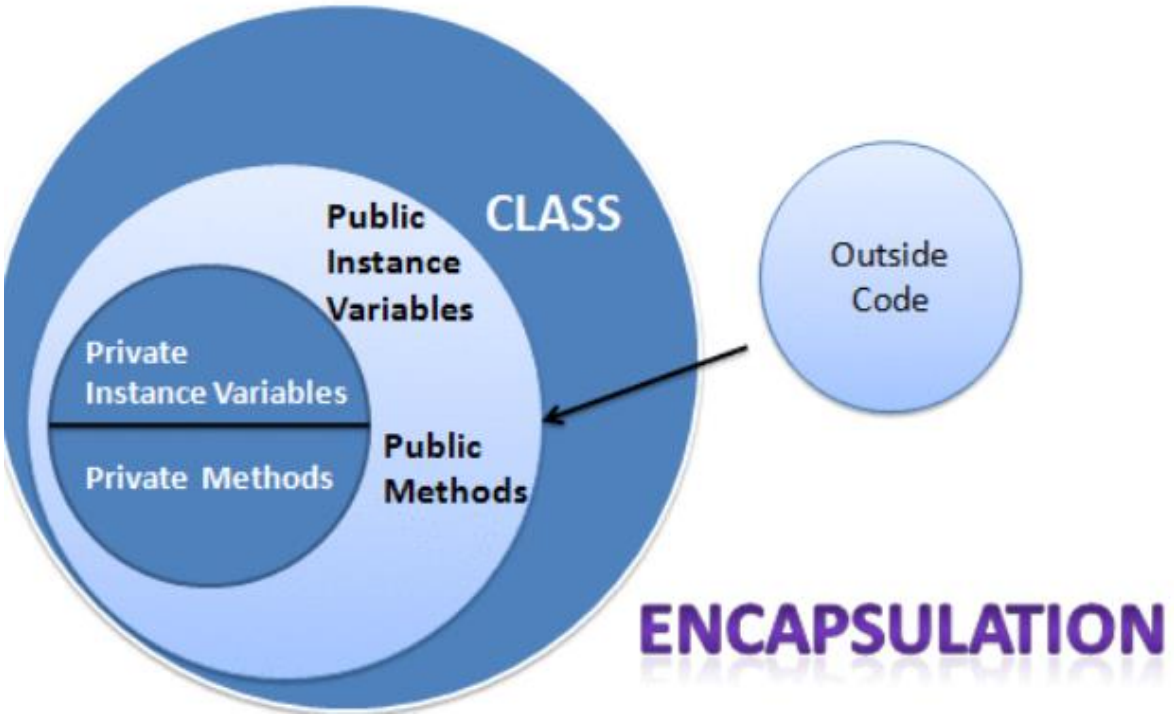
a) Tầm quan trọng của coding convention:
   - 80% chi phí của phần mềm nằm ở khâu bảo trì
   - Sự thay đổi nhân sự liên tục của các công ty phần mềm nên các phần mềm là thành quả của nhiều nhân viên. Không có phần mềm nào được duy trì bởi một người
   - Coding convention giúp nâng cao khả năng đọc hiểu code của phần mềm giúp nhân viên mới tang thời gian hiểu được code của dự án nhanh hơn và kỹ lưỡng hơn
   - Khi giao sản phẩm phần mềm, thì bạn cần chắc chắn source code của phần mềm cần được đóng gói một cách sạch sẽ nhất.

b) Các coding convention cơ bản trong java:

- Đặt tên Class hoặc interface phải là danh từ và mỗi chữ cái đầu tiên của từ thành phần phải là chữ viết hoa ví dụ: class Raster, class ImageSprite, interface Storing
- Phương thức phải là động từ với từ thành phần đầu tiên viết thường và chữ cái đầu tiên của từ tiếp theo phải viết hoa. Ví dụ: run(), runFast(),getBackGround();
- Các biến phải được đặt tên chữ thường cho chữ đầu tiên và viết hoa chữ cái đầu tiên của chữ tiếp theo. Tên của biến phải ngắn gọn nhất và ý nghĩa nhất ví dụ: int i; int myWidth; float min;
- Những biến hằng số được khai báo phải viết hoa toàn bộ. ví dụ: int MIN_WIDTH = 90; int MAX_WIDTH = 280;
- Tên package phải được đặt chữ cái thường toàn bộ. ví du: com.sun.eng, java.lang

8. Tính chất của lập trình hướng đối tượng trong Java

a) Tính đóng gói: là đặt tất cả các biến (thuộc tính) và các phương thức(method) vào trong một đơn vị được gọi là Object



Bạn có thể tạo lớp **read-only** hoặc **write-only** bằng việc cài đặt phương thức setter hoặc getter.

Với cách cài đặt này không thể tùy tiện gán giá trị cho các thuộc tính trong lớp, muốn gán các thuộc tính này phải thông qua phương thức set, chính điều này lập trình viên có thể kiểm soát dữ liệu(ví dụ: kiểm soát tham số đưa vào phải khác không, phải là chuỗi,…)

Với cách cài đặt này thì không thể tùy tiện lấy ra thuộc tính trong lớp mà phải thông qua hàm get, lập trình viên có thể rà soát đối tượng lấy(ví dụ: chỉ cho lấy khi thỏa mãn quyền hạn-> bảo mật)

Ví dụ:

```
package oopsconcept;
public class Mobile {
    private String manufacturer;
    private String operating_system;
    public String model;
    private int cost;
    //Constructor to set properties/characteristics of object
    Mobile(String man, String o,String m, int c){
        this.manufacturer = man;
        this.operating_system=o;
        this.model=m;
        this.cost=c;
    }
    //Method to get access Model property of Object
    public String getModel(){
        return this.model;
    }
    // We can add other method to get access to other properties
}
```

b) Tính kế thừa: là một tính chất quan trọng trong lập trình hướng đối tượng. Tính chất này giúp tạo một lớp từ một lớp sẵn có. Đặc tính này vô cùng quan trọng trong tái sử dụng mã ví dụ về tính kế thừa

```
package oopsconcept;
public class Android extends Mobile{
        //Constructor to set properties/characteristics of object
        Android(String man, String o,String m, int c){
                super(man, o, m, c);
            }
        //Method to get access Model property of Object
        public String getModel(){
            return "This is Android Mobile- " + model;
        }
}
```

```
package oopsconcept;
public class Blackberry extends Mobile{
    //Constructor to set properties/characteristics of object
    Blackberry(String man, String o,String m, int c){
                    super(man, o, m, c);
                }
    public String getModel(){
        return "This is Blackberry-"+ model;
    }
}
```

c) Tính đa hình là tính năng quan trọng của Java cho phép định nghĩa một từ, một được thực hiện tùy theo những ngữ cảnh khác nhau. Ví dụ như trong tiếng anh cùng một từ Run nhưng khi được sử dụng cho một người, một doanh nghiệp, một chiếc máy tính thì ý nghĩa hoàn toàn khác nhau

Trong java có 2 kiểu đa hình

- Static (compile time polymorphism/ Method overloading): loại này thực thi method khác nhau dựa vào tham số đầu vào khác nhau

```
package oopsconcept;
class Overloadsample {
    public void print(String s){
        System.out.println("First Method with only String- "+ s);
    }
    public void print (int i){
        System.out.println("Second Method with only int- "+ i);
    }
    public void print (String s, int i){
        System.out.println("Third Method with both- "+ s + "--" + i);
    }
}
public class PolymDemo {
    public static void main(String[] args) {
        Overloadsample obj = new Overloadsample();
        obj.print(10);
        obj.print("Amit");
        obj.print("Hello", 100);
    }
}
```

- Dynamic (run time polymorphism/ Method Overriding) loại này thực thi method khác nhau dựa vào cách ghi đè vào hàm của lớp cha

```java
package oopsconcept;
public class OverridingDemo {
    public static void main(String[] args) {
        //Creating Object of SuperClass and calling getModel Method
        Mobile m = new Mobile("Nokia", "Win8", "Lumia",10000);
        System.out.println(m.getModel());
        //Creating Object of Sublcass and calling getModel Method
        Android a = new Android("Samsung", "Android", "Grand",30000);
        System.out.println(a.getModel());
        //Creating Object of Sublcass and calling getModel Method
        Blackberry b = new Blackberry("BlackB", "RIM", "Curve",20000);
        System.out.println(b.getModel());
    }
}
```

d) Lớp trừu tượng trong Java

- **Tính trừu tượng** là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng. Nói cách khác, nó chỉ hiển thị các thứ quan trọng tới người dùng và ẩn các chi tiết nội tại, ví dụ: để gửi tin nhắn, người dùng chỉ cần soạn text và gửi tin. Bạn không biết tiến trình xử lý nội tại về phân phối tin nhắn.
- Tính trừu tượng giúp bạn trọng tâm hơn vào đối tượng thay vì quan tâm đến cách nó thực hiện.

Trong Java có 2 cách để thực hiện lớp trừu tượng là: Abstraction và interface

e) Sự khác nhau của Abstraction và interface:

- Interface chỉ có các phương thức trừu tượng còn Abstraction có thể có phương thức trừu tượng hoặc không trừu tượng. trong Java 8 Abstraction còn có thể có phương thức default và static
- Các biến được khai báo trong interface mặc định là final, còn khai báo trong Abstraction thì non-final
- Lớp Abstraction có thể có biến final, non final, static, non static nhưng interface thì chỉ có biến final và static
- Để kế thừa interface thì sử dụng từ khóa implements còn kế thừa Abstraction thì sử dụng extends
- Thành phần của interface mặc định là public còn Abstraction class là private hoặc protected

B. Cấu trúc tập:

1. **ArrayList:** lưu danh sách các phần tử có thứ tự, được phép trùng nhau.

8. *Khai báo:*

List<Kiểu dữ liệu> ds = new ArrayList<>();

9. *Các phương thức thường dùng:*

add(): thêm phần tử

get(): lấy phần tử

remove(): xóa phần tử

size(): trả về số phần tử trong danh sách

indexOf(): trả về vị trí đầu tiên của phần tử

lastIndexOf(): trả về vị trí cuối cùng của phần tử

2. **HashSet:** lưu danh sách các phần tử không có thứ tự, không được phép trùng nhau.

10. *Khai báo:*

Set<Kiểu dữ liệu> ds = new HashSet<>();

11. *Các phương thức thường dùng:*

add(): thêm phần tử

remove(): xóa phần tử

size(): trả về số phần tử trong danh sách

toArray(): trả về mảng các phần tử trong danh sách

3. **HashMap**: lưu danh sách các phần tử dưới dạng bảng băm (key → value)

12. *Khai báo:*

Map<K, V> ds = new HashMap<>();

K: kiểu dữ liệu dùng làm khóa

V: kiểu dữ liệu giá trị cần lưu trữ

13. *Các phương thức thường dùng:*

put(): thêm phần tử

get(): lấy phần tử

remove: xóa phần tử

keySet(): trả về tập khóa của bảng băm

size(): trả về số phần tử

C. Mô hình quan hệ:

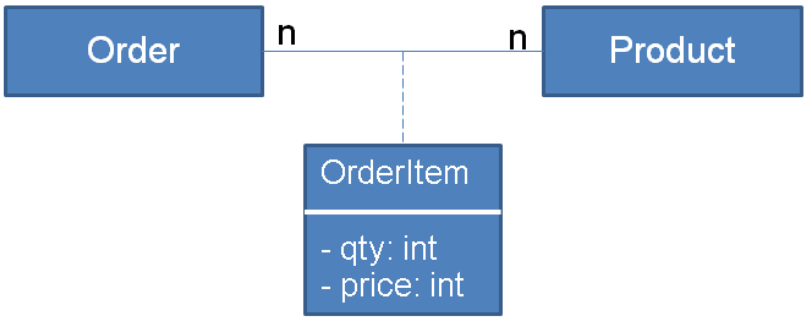
1. Quan hệ 1-n:

```
class Customer {
        List<AccountBank> accs;
}
class AccountBank {
        Customer cus;
}
```

2. Quan hệ n-n (không có thuộc tính kết hợp):


```
class Employee {
        List<Project> pros;
}
class Project {
        List<Employee> ems;
}
```

3. Quan hệ n-n (có thuộc tính kết hợp)

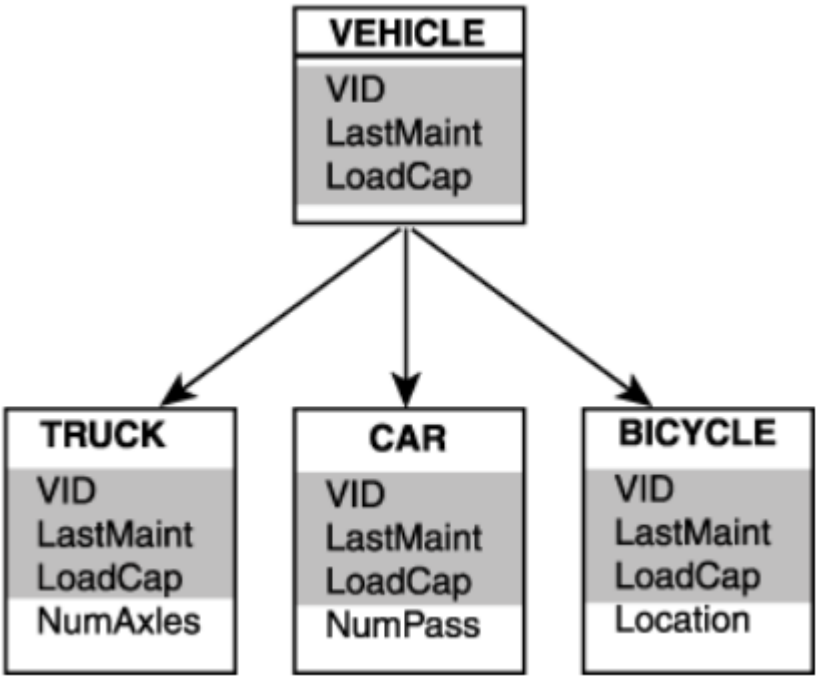



```
class Order {
        int orderId;
        List<OrderItem> items;
}
class Product {
        int productId;
        List<OrderItem> items;
}
```

```
class OrderItemId {
        int orderId;
        int productId;
        ...
}
```

```
class OrderItem {
        OrderItemId id;
        Order order;
        Product product;
        int qty;
        int price;
}
```

4. Quan hệ kế thừa



```java
public class Verhicle{
    protected int id;
    protected int lastMain;
    protected int lastCap;
    public Verhicle(){

    }

    public Verhicle(int id,int lastMain,int lastCap){
        this.id = id;
        this.lastMain = lastMain;
        this.lastCap = lastCap;
    }

    public String toString(){
        return "verhicle";
    }

}
public class Car extends Verhicle {
    private int numberCar;

    public int getNumberCar() {
        return numberCar;
    }

    public void setNumberCar(int numberCar) {
        this.numberCar = numberCar;
    }

    public Car(){}

    public Car(int numberCar,int id,int lastMain,int lastCap){
        super(id,lastMain,lastCap);
        this.numberCar = numberCar;
    }

    @Override
    public String toString(){
        return "car is numberCar="+ this.numberCar;
    }
}
```

```java
public class Truck extends Verhicle {

    private int numAxles;

    public int getNumAxles() {
        return numAxles;
    }

    public void setNumAxles(int numAxles) {
        this.numAxles = numAxles;
    }

    public Truck(int id,int lastMain,int lastCap,int numAxles){
        super(id, lastMain, lastCap);
        this.numAxles = numAxles;
    }

    @Override
    public String toString(){
        return "car is Truck numAxles="+ this.numAxles;
    }

}
```

III. NỘI DUNG THỰC HÀNH:

Bài tập 1: Mở **Lab2_1.java**: Khai báo lớp hình tròn gồm 1 thuộc tính r, có 2 hàm dựng không đối số và 1 đối số, phương thức Get/Set, phương thức hành vi **tinhCV,** phương thức **toString** in thông tin đối tượng.

```java
public class lab02_1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        HinhTron ht = new HinhTron(10.0);
        System.out.println(ht);
    }
    public static class HinhTron {
        private double banKinh;

        // Constructor không đối số
        public HinhTron() {
        }

        // Constructor có một đối số
        public HinhTron(double banKinh) {
            this.banKinh = banKinh;
        }

        // Phương thức Get/Set cho thuộc tính banKinh
        public double getBanKinh() {
            return banKinh;
        }

        public void setBanKinh(double banKinh) {
            this.banKinh = banKinh;
        }

        // Phương thức tính chu vi
        public double tinhChuVi() {
            return 2 * Math.PI * banKinh;
        }

        // Phương thức toString
        @Override
        public String toString() {
            return "Bán kính: " + banKinh + "\nChu vi: " + tinhChuVi();
        }
    }

}
```

Yêu cầu:

1. Chạy chương trình, quan sát kết quả.
2. Hãy viết thêm phương thức hành vi **tinhDT** tính diện tích hình tròn, chỉnh sửa code trong phương thức **toString** để hiển thị thêm diện tích.

Bài tập 2:

Viết chương trình khai báo lớp **HinhChuNhat** gồm 2 thuộc tính dai, rong, có 2 hàm dựng không đối số và 2 đối số, phương thức Get/Set, phương thức hành vi **tinhCV, tinhDT** phương thức **toString** in thông tin đối tượng.

```java
public static void main(String[] args) {
    // TODO Auto-generated method stub
    HinhChuNhat hcn = new HinhChuNhat(10, 12);
    System.out.println(hcn);
}
public static class HinhChuNhat {
    private double dai;
    private double rong;

    // Constructor không đối số
    public HinhChuNhat() {
    }

    // Constructor có hai đối số
    public HinhChuNhat(double dai, double rong) {
        this.dai = dai;
        this.rong = rong;
    }

    // Phương thức Get/Set cho thuộc tính dai
    public double getDai() {
        return dai;
    }

    public void setDai(double dai) {
        this.dai = dai;
    }

    // Phương thức Get/Set cho thuộc tính rong
    public double getRong() {
        return rong;
    }

    public void setRong(double rong) {
        this.rong = rong;
    }

    // Phương thức tính chu vi
    public double tinhChuVi() {
        return 2 * (dai + rong);
    }

    // Phương thức tính diện tích
    public double tinhDienTich() {
        return dai * rong;
    }

    // Phương thức toString
    @Override
    public String toString() {
        return "Chiều dài: " + dai + "\nChiều rộng: " + rong + "\nChu vi: " + tinhChuVi() + "\nDiện tích: " + tinhDienTich();
    }
}
```

Bài tập 3:

Viết chương trình khai báo lớp SinhVien gồm các thuộc tính: mssv, hoTen, diemTB. Nhập vào danh sách sinh viên gồm 5 phần tử bao gồm: Nguyễn Văn A, Nguyễn Văn B, Nguyễn Văn C, Nguyễn Văn D, Nguyễn Văn E

```java
public class lab02_3 {
    public static class SinhVien {
        private String mssv;
        private String hoTen;
        private double diemTB;

        // Constructor
        public SinhVien(String mssv, String hoTen, double diemTB) {
            this.mssv = mssv;
            this.hoTen = hoTen;
            this.diemTB = diemTB;
        }

        // Phương thức toString
        @Override
        public String toString() {
            return "MSSV: " + mssv + "\nHọ tên: " + hoTen + "\nĐiểm trung bình: " + diemTB;
        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);
        ArrayList<SinhVien> danhSachSV = new ArrayList<>();

        // Nhập thông tin của 5 sinh viên
        for (int i = 1; i <= 5; i++) {
            System.out.println("Nhập thông tin sinh viên thứ " + i + ":");
            System.out.println("MSSV:");
            String mssv = scanner.nextLine();
            System.out.println("Họ và tên:");
            String hoTen = scanner.nextLine();
            System.out.println("Điểm trung bình:");
            double diemTB = scanner.nextDouble();
            scanner.nextLine(); // Đọc bỏ dòng mới
            danhSachSV.add(new SinhVien(mssv, hoTen, diemTB));
        }

        // In danh sách sinh viên đã nhập
        System.out.println("\nDanh sách sinh viên:");
        for (SinhVien sv : danhSachSV) {
            System.out.println(sv);
            System.out.println();
        }
    }

}
```

Bài tập 4:

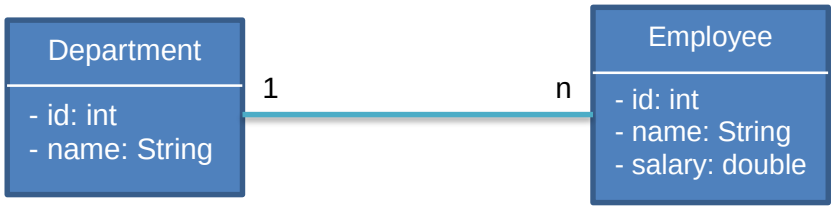
Khai báo lớp **NhanVien** gồm 3 thuộc tính: hoten, ngaycong, luongcb. Hai hàm dựng không đối số và 3 đối số, phương thức Get/Set, phương thức hành vi **getLuong()** = ngaycong x luongcb, phương thức toString in thông tin nhân viên.

Trong chương trình chính, khai báo danh sách các đối tượng NhanVien, in danh sách nhân viên, tính trung bình lương toàn nhân viên.

Bài tập 5: cho sơ đồ lớp như hình dưới đây, xây dựng 2 class tương ứng và viết hàm nhập và xuất cho từng class.

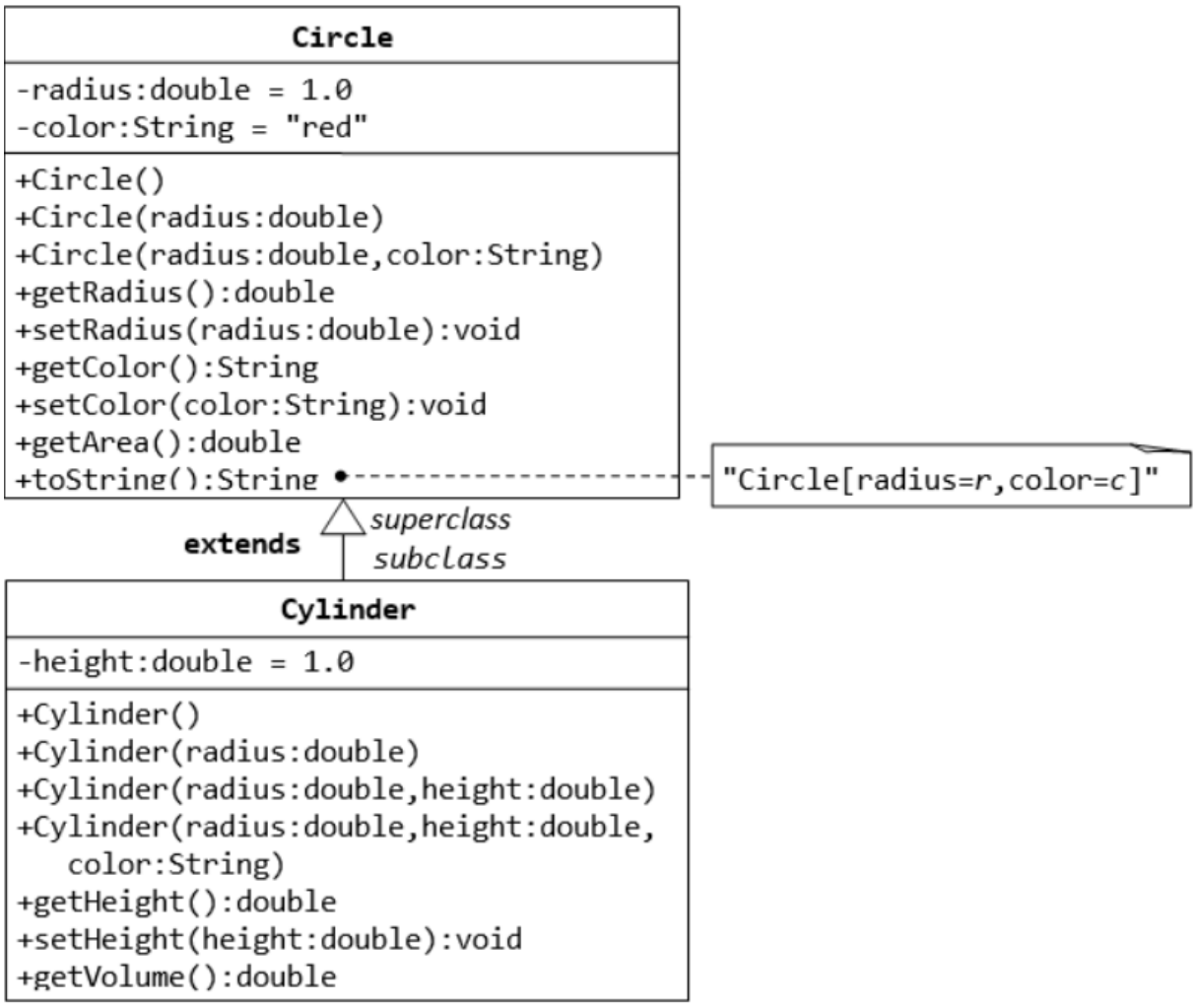
Bài tập 6: Cài đặt bài tập 2 bằng Set và Map.

Bài tập 7: Cho sơ đồ lớp như hình

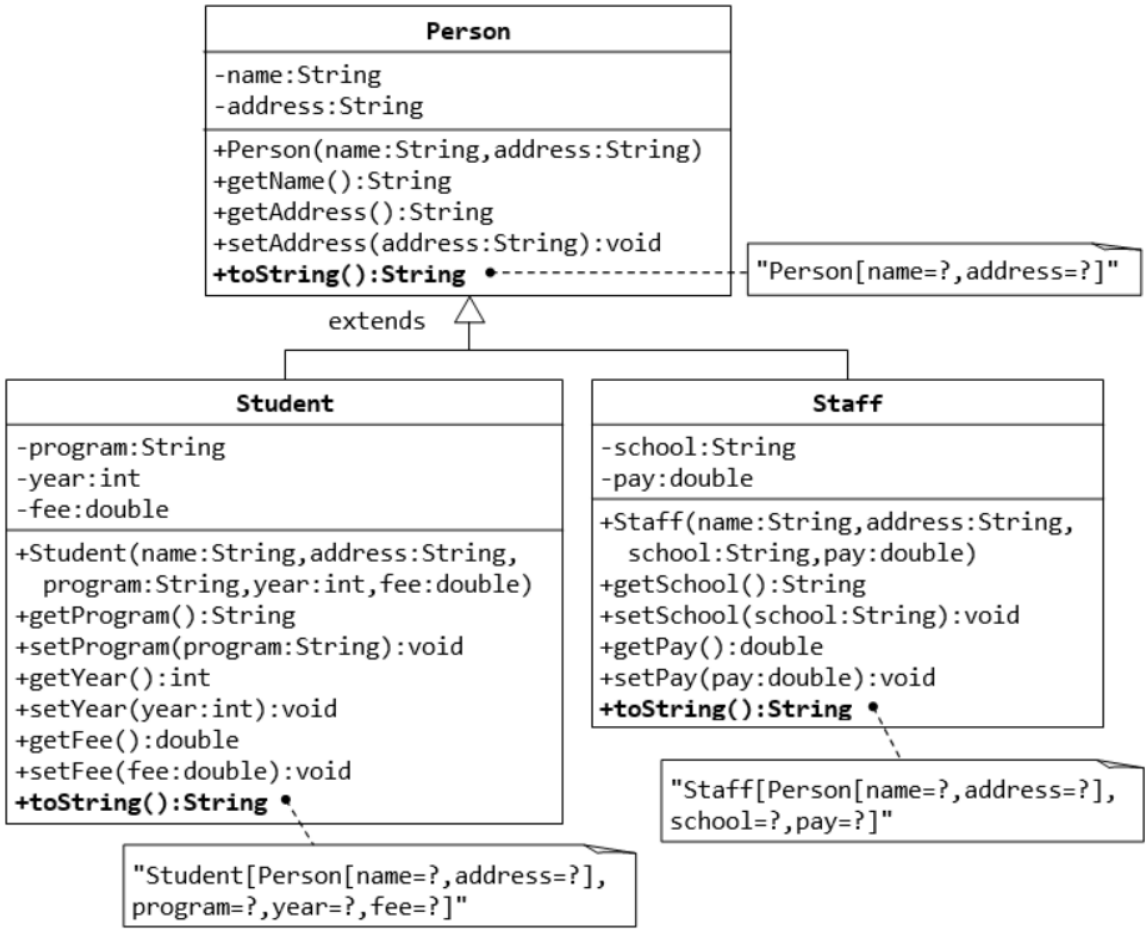


Khai báo sơ đồ lớp trên. Viết chương trình tạo và in ra danh sách **Department** và danh sách **Employee** tương ứng.

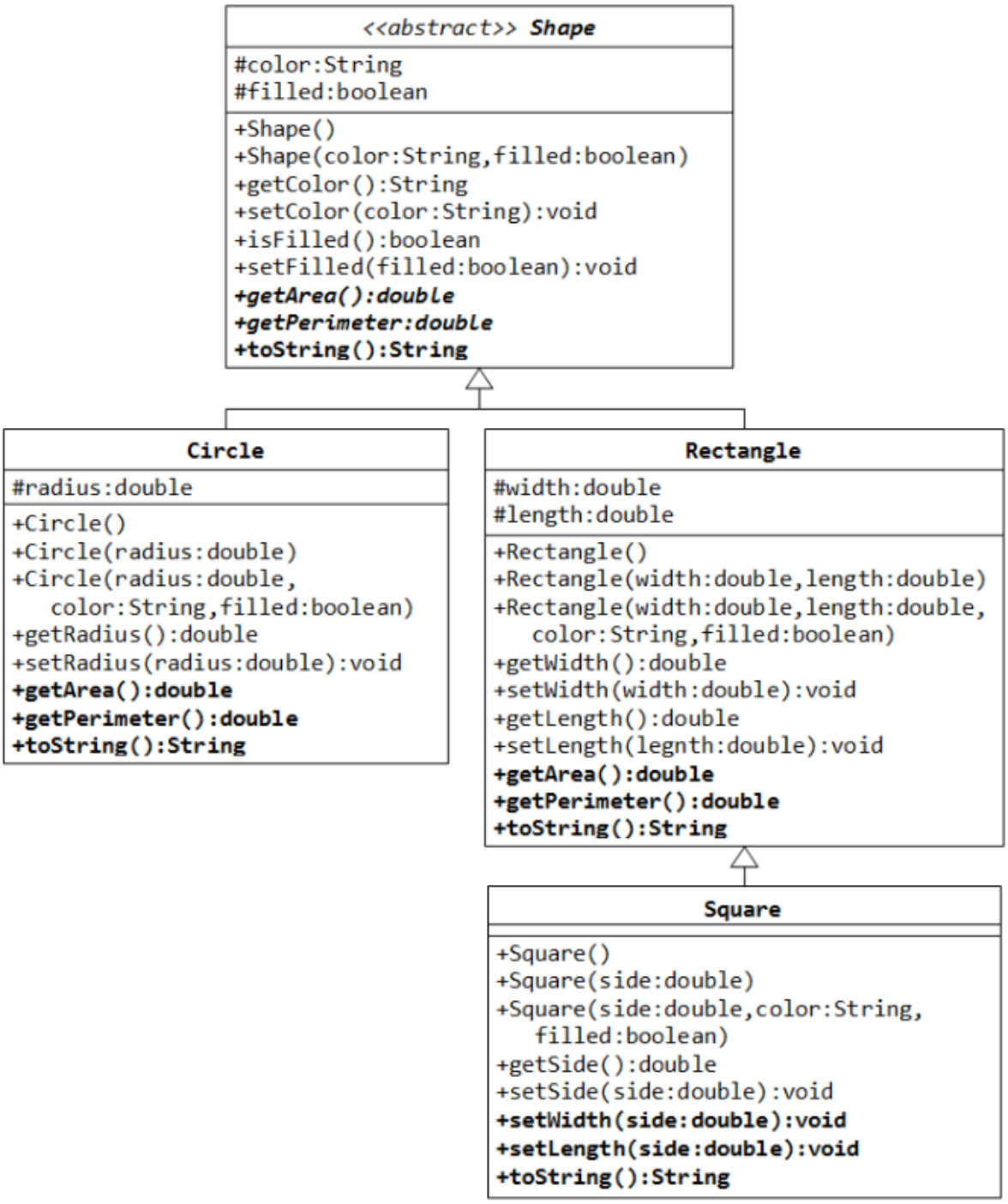
Bài tập 8: viết các class theo sơ đồ như sau:



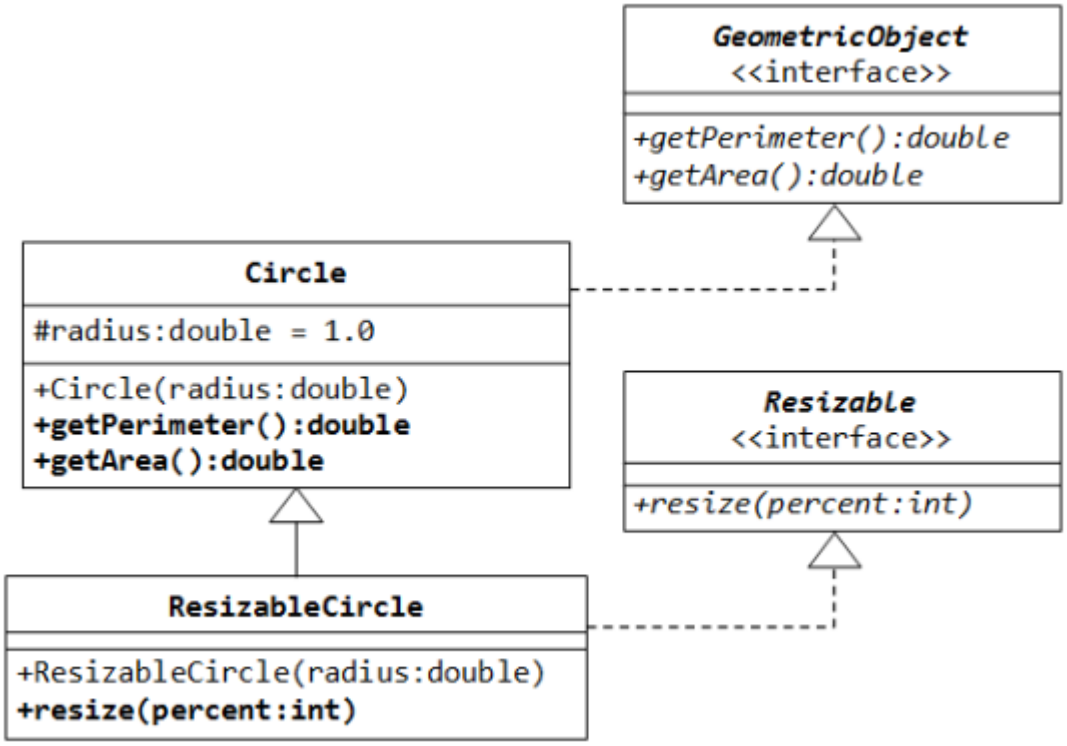
Bài tập 9: viết class có sơ đồ như sau


Person
-name:String
-address:String
+Person(name:String,address:String)
+getName():String
+getAddress():String
+setAddress(address:String):void
+toString():String
"Person[name=?,address=?]"
extends
Student
-program:String
-year:int
-fee:double
+Student(name:String,address:String,
program:String,year:int,fee:double)
+getProgram():String
+setProgram(program:String):void
+getYear():int
+setYear(year:int):void
+getFee():double
+setFee(fee:double):void
+toString():String
"Student[Person[name=?,address=?],
program=?,year=?,fee=?]"
Staff
-school:String
-pay:double
+Staff(name:String,address:String,
school:String,pay:double)
+getSchool():String
+setSchool(school:String):void
+getPay():double
+setPay(pay:double):void
+toString():String
"Staff[Person[name=?,address=?],
school=?,pay=?]"

Bài tập 10: viết các lớp có sơ đồ như sau:



Bài tập 11: Thiết kế các class có sơ đồ như sau sử dụng interface


GeometricObject
<<interface>>
+getPerimeter():double
+getArea():double
Circle
#radius:double = 1.0
+Circle(radius:double)
+getPerimeter():double
+getArea():double
Resizable
<<interface>>
+resize(percent:int)
ResizableCircle
+ResizableCircle(radius:double)
+resize(percent:int)