# MỤC LỤC

**MỤC LỤC** ..... **I**
**HƯỚNG DẪN** ..... **III**
**BÀI 1: LÀM QUEN VỚI NGÔN NGỮ JAVA** ..... **1**
**1.1 TÓM TẮT LÝ THUYẾT** ..... **1**
*1.1.1 Cài đặt môi trường Java* ..... *1*
*1.1.2 Chương trình Java đầu tiên* ..... *1*
*1.1.3 Ngôn ngữ lập trình Java* ..... *2*
**THỰC HÀNH CƠ BẢN** ..... **5**
**THỰC HÀNH NÂNG CAO** ..... **10**
**BÀI TẬP VỀ NHÀ** ..... **10**
**BÀI 2: LỚP VÀ ĐỐI TƯỢNG** ..... **11**
**2.1 TÓM TẮT LÝ THUYẾT** ..... **11**
*2.1.1 Lớp và đối tượng* ..... *11*
*2.1.2 Định nghĩa truy cập* ..... *11*
*2.1.3 Phương thức khởi tạo* ..... *12*
*2.1.4 Thành phần static* ..... *12*
**THỰC HÀNH CƠ BẢN** ..... **13**
**THỰC HÀNH NÂNG CAO** ..... **22**
**BÀI TẬP VỀ NHÀ** ..... **24**
**BÀI 3: MẢNG VÀ CHUỖI** ..... **26**
**3.1 TÓM TẮT LÝ THUYẾT** ..... **26**
*3.1.1 Mảng một chiều* ..... *26*
*3.1.2 Mảng hai chiều* ..... *27*
*3.1.3 Chuỗi* ..... *27*
**THỰC HÀNH CƠ BẢN** ..... **29**
**THỰC HÀNH NÂNG CAO** ..... **32**
**BÀI TẬP VỀ NHÀ** ..... **32**
**BÀI 4: KẾ THỪA VÀ ĐA HÌNH** ..... **34**
**4.1 TÓM TẮT LÝ THUYẾT** ..... **34**
*4.1.1 Hiện thực lớp con trong Java* ..... *34*
*4.1.2 Kỹ thuật phân cấp kế thừa* ..... *34*
*4.1.3 Ghi đè phương thức* ..... *35*
**THỰC HÀNH CƠ BẢN** ..... **36**
**THỰC HÀNH NÂNG CAO** ..... **41**
**BÀI TẬP VỀ NHÀ** ..... **43**
**BÀI 5: LỚP TRỪU TƯỢNG VÀ GIAO DIỆN** ..... **44**
**5.1 TÓM TẮT LÝ THUYẾT** ..... **44**

*5.1.1 Lớp trừu tượng* ..... *44*
*5.1.2 Giao diện* ..... *45*
**THỰC HÀNH CƠ BẢN** ..... **47**
**THỰC HÀNH NÂNG CAO** ..... **51**
**BÀI TẬP VỀ NHÀ** ..... **52**
**BÀI 6: XỬ LÝ NGOẠI LỆ** ..... **53**
**6.1 TÓM TẮT LÝ THUYẾT** ..... **53**
*6.1.1 Khối lệnh try – catch* ..... *53*
*6.1.2 Khối lệnh try-finally* ..... *54*
*6.1.3 Khối lệnh try – catch – finally* ..... *54*
*6.1.4 Từ khoá throw và throws* ..... *55*
**THỰC HÀNH CƠ BẢN** ..... **57**
**THỰC HÀNH NÂNG CAO** ..... **58**
**BÀI 7: LỚP TỔNG QUÁT VÀ COLLECTION** ..... **59**
**7.1 TÓM TẮT LÝ THUYẾT** ..... **59**
*7.1.1 Lớp tổng quát* ..... *59*
*7.1.2 Java Collections* ..... *59*
*7.1.3 Các thuật toán cơ bản trên Java Collections* ..... *60*
**THỰC HÀNH CƠ BẢN** ..... **61**
**THỰC HÀNH NÂNG CAO** ..... **63**
**BÀI TẬP VỀ NHÀ** ..... **64**
**TÀI LIỆU THAM KHẢO** ..... **66**

# HƯỚNG DẪN

## MÔ TẢ HỌC PHẦN

Học phần **Thực hành Lập trình hướng đối tượng** cung cấp cho sinh viên những kiến thức cơ bản về lập trình hướng đối tượng thông qua ngôn ngữ lập trình Java. Học phần này là nền tảng để tiếp thu các ngôn ngữ lập trình cấp cao trong chương trình đào tạo. Mặt khác, học phần cung cấp nền tảng kiến thức để tư duy lập trình hướng đối tượng nhằm giải các bài toán ứng dụng trong thực tế.

Học xong học phần này, sinh viên phải nắm được các vấn đề sau:

- Làm quen với ngôn ngữ lập trình Java.
- Các khái niệm trong lập trình hướng đối tượng.
- Xây dựng lớp và đối tượng.
- Tính đóng gói, kế thừa và đa hình.
- Lớp trừu tượng và giao diện.
- Xử lý ngoại lệ và lập trình tổng quát trong Java.

## NỘI DUNG HỌC PHẦN

- Bài 1. LÀM QUEN VỚI NGÔN NGỮ JAVA
- Bài 2. LỚP VÀ TẠO ĐỐI TƯỢNG
- Bài 3. MẢNG VÀ CHUỖI
- Bài 4. KẾ THỪA VÀ ĐA HÌNH
- Bài 5. LỚP TRỪU TƯỢNG VÀ GIAO DIỆN
- Bài 6. XỬ LÝ NGOẠI LỆ
- Bài 7. LẬP TRÌNH TỔNG QUÁT VÀ COLLECTION

## YÊU CẦU HỌC PHẦN

Có tư duy tốt về logic và kiến thức toán học cơ bản.

# CÁCH TIẾP NHẬN NỘI DUNG HỌC PHẦN

Học phần này trang bị cho sinh viên các kiến thức cơ bản về lập trình hướng đối tượng như lớp, đối tượng, tính kế thừa, lớp trừu tượng và giao diện; đồng thời sinh viên có thể sử dụng các cấu trúc điều khiển, các kiểu dữ liệu cơ sở cùng các phép toán của ngôn ngữ lập trình Java để hiện thực các chương trình. Ngoài ra, học phần này còn định hướng phong cách lập trình, kỹ năng lập trình hướng đối tượng trong ngôn ngữ lập trình cấp cao.

# PHƯƠNG PHÁP ĐÁNH GIÁ HỌC PHẦN

Để học tốt học phần này, người học cần ôn tập các kiến thức đã học, thực hành đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

Điểm đánh giá: gồm 2 phần

1. Điểm quá trình (chuyên cần + bài tập)
2. Điểm kiểm tra (trung bình cộng các bài kiểm tra)

# BÀI 1: LÀM QUEN VỚI NGÔN NGỮ JAVA

Bài này giúp người học nắm được các nội dung sau:

- *Biết cách thiết lập môi trường Java;*
- *Sử dụng được các lệnh điều khiển chương trình;*
- *Hiểu được các phương thức dùng trong Java;*
- *Sử dụng được các hàm nhập xuất dữ liệu cơ bản;*

## 1.1 TÓM TẮT LÝ THUYẾT

### 1.1.1 Cài đặt môi trường Java

- **Cài đặt JDK** (Java Development Kit). Tải phiên bản mới nhất tại website của Oracle: ***www.oracle.com***/technetwork/java/javase/downloads/index.html
- **Cài đặt IDE** (Integrated Development Environment) hỗ trợ lập trình Java:
  - Apache NetBeans IDE (https://netbeans.apache.org)
  - Eclipse (http://www.eclipse.org)
  - IntelliJ IDEA (https://www.jetbrains.com/idea/)

### 1.1.2 Chương trình Java đầu tiên

- Tạo project mới trong Apache Netbeans IDE:

File → New Project … → Java → Java Application: Project Name (tên project), Project Location (chọn nơi lưu project)

- Mã nguồn chương trình Java đầu tiên:

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

- Thực thi chương trình: Run → Run File (Shift + F6)
- Xem kết quả thực thi tại output: Window → output (Ctrl + 4)

## 1.1.3 Ngôn ngữ lập trình Java

❖ **Các phần tử cơ sở:**

- Định danh (indentifer): là tên biến, tên hằng, tên lớp và tên phương thức trong Java.
- Từ khóa (keywords): là những định danh định sẵn của Java như: package, import, class, public, private, …
- Chú thích (comment): // (dòng), /* … */ (nhiều dòng), /** … / (tài liệu)
- Kiểu dữ liệu nguyên thủy (primitive data type): **char**, **byte**, **short**, **int**, **long**, **float**, **double**, **boolean**.
- Kiểu dữ liệu tham chiếu (reference type): mảng (array), lớp (class), giao diện (interface)
- Các loại biến: cục bộ (local), thực thể (instance), tĩnh (static)
- Toán tử: phép toán số học **(+ - * / % ++ --**), phép toán so sánh**(< <= > >= == !=**), phép toán luận lý (**&& ||**), phép gán **(= += -= *= /= %=**)

❖ **Câu lệnh điều khiển chương trình:**

- **Câu lệnh if – else**

```
if (expression1) {
   // codes
}
else if (expression2) {
   // codes
```

```
}
else if (expression3) {
   // codes
}
// ...
else {
   // codes
}
```

❖ **Câu lệnh switch – case**

```
switch (expression) {
   case value_1:
      // codes
      break; // optional
   case value_2:
      // codes
      break; // optional
   // ...
   default: // Optional
      // codes
}
```

❖ **Vòng lặp while**

```
while (condition) {
   //statement or code to be executed
}
```

❖ **Vòng lặp for**

```
for(initialization; condition; increment/decrement){
  //statement or code to be executed
}
```

❖ **Vòng lặp do – while**

```
do {
   //statement or code to be executed
} while (condition);
```

- Lệnh nhảy: break, continue

### ❖ Phương thức (method)

- Định nghĩa phương thức:

```
[modifiers] returnType methodName([parameterList]) {
     //method body
}
```

- Nạp chồng phương thức (overloading method) cho phép định nghĩa các phương thức trùng tên nhưng khác nhau về kiểu dữ liệu hoặc số lượng các tham số.

### ❖ Nhập xuất dữ liệu

- Nhập dữ liệu:

```
1. import java.util.Scanner; //thư viện của lớp Scanner
2. Scanner input = new Scanner(System.in); //tạo đối tượng của lớp
   Scanner
```

- Xuất dữ liệu:

```
1. System.out.print()
2. System.out.println()
3. System.out.printf()
```

- Các phương thức nhập xuất: **next(), nextInt(), nextLong(), nextDouble(), nextLine(), nextByte(), nextBoolean()**

### ❖ Cấu trúc file chương trình Java:

```
package packagename; // Định nghĩa gói
import java.io.* ; // Các gói cần sử dụng
//Định nghĩa các lớp và các interface
 public class ClassName{
      public static void main(String[] args){
          //codes
      }
 }
```

# THỰC HÀNH CƠ BẢN

**Câu 1:** Viết chương trình xuất ra màn hình các thông tin sau.

```
Hello! I'm <your name>.
I'm <your age> years old.
This is my first java program.
```

**Hướng dẫn:**

- Mở NetBeans: File → New Project … → Java → Java Application
- Project Name: HoTenSV_Bai1

```
//File Cau1.java
public class Cau1 {
    public static void main (String[] agrs) {
         int age = 20;
         String name = "Nguyen Thanh Hai";
         System.out.println("Helo! I'm " + name);
         System.out.println("I'm " + age + " years old");
         System.out.println("This is my first java program.");
    }
}
```

- Thực thi chương trình: Run → Run File (Shift + F6)
- Xem kết quả tại output: Window → output (Ctrl + 4)

**Câu 2:** Viết chương trình nhập các kiểu dữ liệu nguyên thủy: int, float, double, chuỗi ký tự. Xuất các giá trị vừa nhập ra màn hình.

**Hướng dẫn:**

```
//Cau2.java
import java.util.Scanner;
public class Cau2 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int number = input.nextInt(); // nhập số kiểu int
        System.out.println("You entered " + number);
```

```
        System.out.print("Enter float: ");
        float myFloat = input.nextFloat(); // nhập số kiểu float
        System.out.println("Float entered = " + myFloat);

        System.out.print("Enter double: ");
        double myDouble = input.nextDouble(); // nhập số kiểu double
        System.out.println("Double entered = " + myDouble);

        System.out.print("Enter text: ");
        String myString = input.next(); // nhập chuỗi ký tự
        System.out.println("Text entered = " + myString);

        // closing the scanner object
        input.close();
    }
}
```

**Câu 3:** Nhập vào ba số nguyên dương a, b, c. Kiểm tra xem ba số đó có lập thành tam giác không? Nếu có hãy cho biết tam giác đó thuộc loại nào? (cân, vuông, đều)

- Điều kiện để b số lập thành tam giác: tổng hai cạnh phải lớn hơn cạnh còn lại. Vậy a, b, c lập thành ba cạnh của tam giác thì (a + b) > c và (a + c) > b và (b + c) > a.
- Xét loại tam giác:
  - Tam giác cân: a = b hoặc b = c hoặc c = a
  - Tam giác đều: a = b = c
  - Tam giác vuông: $a^2 = b^2 + c^2$ hoặc $b^2 = a^2 + c^2$ hoặc $c^2 = a^2 + b^2$

**Hướng dẫn:**

```
//File Cau3.java
import java.util.Scanner;
public class Cau3 {
  public static void main(String[] args) {
    int a, b, c; // Khai báo ba biến int
    Scanner sc = new Scanner(System.in); // Tạo đối tượng Scanner
    System.out.print("Nhập a: ");
    a = sc.nextInt();
```

```
    System.out.print("Nhập b: ");
    b = sc.nextInt();
    System.out.print("Nhập c: ");
    c = sc.nextInt();

    if (laTamgiac(a,b,c)){
      System.out.println("Là tam giác!");
      if(laTamgiacDeu(a,b,c)){
        System.out.println("Và còn đều nữa!");
      }else {
        if (laTamgiacVuong(a, b, c)) {
          System.out.println("Và còn vuông nữa!");
        }
        if (laTamgiacCan(a, b, c)) {
          System.out.println("Và còn cân nữa!");
        }
      }
    }else{
      System.out.println("Không phải là tam giác!");
    }
  }
// phương thức kiểm tra tam giác
  public static boolean laTamgiac(int a, int b, int c) {
    if ((a + b) > c && (a + c) > b && (b + c) > a)
        return true;
    return false;
  }
  // phương thức kiểm tra tam giác vuông
  public static boolean laTamgiacVuong(int a, int b, int c){
    if ((a*a + b*b) == c*c || (a*a + c*c) == b*b || (b*b + c*c) == a*a)
      return true;
    return false;
  }
  // phương thức kiểm tra tam giác cân
  public static boolean laTamgiacCan(int a, int b, int c){
    if (a == b || b == c || c == a)
      return true;
    return false;
  }
  // phương thức kiểm tra tam giác đều
  public static boolean laTamgiacDeu(int a, int b, int c){
    if (a == b && b == c)
      return true;
```

```
    return false;
  }
}
```

**Câu 4:** Nhập vào một ký tự, nếu là chữ hoa xuất: CHUHOA, nếu là chữ thường xuất CHUTHUONG, nếu là số chẵn xuất SOCHAN và là số lẽ xuất SOLE

**Câu 5:** Giải và biện luận phương trình bậc nhất ax + b = 0.

Hướng dẫn:

- Khai báo hai biến a, b kiểu số thực để lưu hệ số của phương trình do người dùng nhập tùy ý từ bàn phím. Nhập *a, b.*
- Giải thuật: Xét đủ hai trường hợp a = 0 và a ≠ 0.

**Câu 6:** Giải và biện luận phương trình bậc hai: $ax^2 + bx + c = 0$.

Hướng dẫn:

- Khai báo ba biến a, b, c kiểu số thực. Nhập các hệ số a, b, c.
- Xét a = 0. Phương trình trở thành bậc nhất bx + c = 0, giải và biện luận theo b, c (tương tự bài 1).
- Xét a ≠ 0. Tính $d = b^2 - 4ac$. Xét các trường hợp d = 0, d < 0 và d > 0.

**Câu 7:** Viết chương trình nhập vào số nguyên dương n. Kiểm tra xem n có phải là số chính phương hay không? (số chính phương là số khi lấy căn bậc hai có kết quả là nguyên)

**Câu 8:** Viết chương trình nhập vào tháng của một năm, cho biết số ngày của tháng đó. Nếu nhập vào <1 hoặc >12 thì thông báo "This month is invalid."

**Hướng dẫn:**

- Khai báo biến và nhập vào tháng.
- Các tháng 1, 3, 5, 7, 8, 10, 12 có 31 ngày.
- Các tháng 4, 6, 9, 11 có 30 ngày
- Nếu là tháng 2 thì yêu cầu nhập thêm năm, nếu là năm nhuận thì tháng 2 có 29 ngày, còn lại 28 ngày. Năm nhuận là năm chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100.

- Nếu tháng nhập vào không thuộc các tháng trên thì thông báo "This month is invalid"

**Câu 9:** Viết chương trình tính tiền cước TAXI. Biết rằng:

- KM đầu tiên là 5000đ.
- KM tiếp theo là 4000đ.
- Nếu lớn hơn 30km thì mỗi km thêm sẽ là 3000đ.
- Hãy nhập số km sau đó in ra số tiền phải trả.

**Câu 10:** Viết chương trình kiểm tra năm nhập từ bàn phím có phải là năm nhuận không? Năm nhuận là năm chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100.

**Câu 11:** Viết chương trình nhập hai số a và b vào từ bàn phím. Nhập kí tự thể hiện một trong bốn phép toán: cộng (+), trừ (-), nhân (*), chia (:). In ra kết quả thực hiện phép toán đó trên hai số a, b.

**Câu 12:** Viết chương trình đảo ngược một số nguyên dương. Ví dụ: n = 1234 → 4321

**Câu 13:** Viết chương trình nhập một số nguyên có hai chữ số. Hãy in ra cách đọc của nó. Ví dụ: 95 → Chín mươi lăm; 11 → Mười một; 31 → Ba mươi mốt

**Câu 14:** Viết chương trình tính tổng sau: $S_n = 1 + 1.2 + 1.2.3 + \ldots + 1.2.3\ldots n$, với n nhập từ bàn phím.

**Câu 15:** Viết chương trình in ra tam giác vuông cân, tam giác cân với chiều cao h (h nhập từ bàn phím). Giả sử h = 5, kết quả minh họa như sau:

```
*                    *
* *                  * * *
* * *                * * * * *
* * * *              * * * * * * *
* * * * *            * * * * * * * * *
```

# THỰC HÀNH NÂNG CAO

**Câu 16:** Viết chương trình nhập vào một số nguyên, 0 < n < 100. Nếu n không thỏa điều kiện thì yêu cầu nhập lại.

a. Kiểm tra n có phải là số nguyên tố không?

b. Kiểm tra n có phải là số chính phương không?

**Câu 17:** Viết chương trình chuyển đổi số thập phân sang nhị phân và ngược lại.

**Câu 18:** Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không? Nếu ngày hợp lệ in ra màn hình ngày tiếp theo.

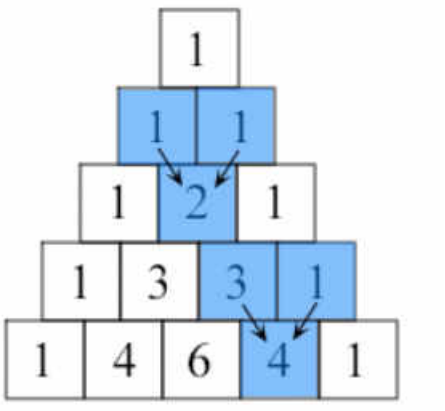
# BÀI TẬP VỀ NHÀ

**Câu 19:** Viết chương trình xuất ra màn hình n phần từ đầu tiên của dãy Fibonacci với dãy bắt đầu là $F_0 = F_1 = 1$, phần tử tiếp theo là tổng hai phần tử trước đó. Với n nhập từ bàn phím (n > 1).

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 …

**Câu 20:** Viết phương thức in ra màn hình n dòng đầu tiên của tam giác Pascal. Tam giác Pascal là một hình số học và hình học được Blaise Pascal nghĩ ra.

Ví dụ tam giác Pascal:



# BÀI 2: LỚP VÀ ĐỐI TƯỢNG

Bài này giúp người học nắm được các nội dung sau:

- *Hiểu cấu trúc và cách xây dựng một lớp;*
- *Hiểu và sử dụng được các định nghĩa truy cập;*
- *Biết cách viết các phương thức khởi tạo;*
- *Biết cách sử dụng từ khoá **this**;*
- *Biết cách sử dụng từ khóa **static**.*

## 2.1 TÓM TẮT LÝ THUYẾT

### 2.1.1 Lớp và đối tượng

- Lớp (class) là một nhóm các đối tượng có chung các thuộc tính và hành vi.
- Khai báo lớp:

```
[modifiers] class ClassName {
    // Các thuộc tính
    // Các phương thức
  }
```

- Đối tượng (object) là  thực thể của một lớp.

```
ClassName objectName = new ClassName();
```

### 2.1.2 Định nghĩa truy cập

- Định nghĩa truy cập (access modifiers) là xác định độ truy cập phạm vi vào dữ liệu của các thuộc tính, phương thức hoặc class.

| Modifiers | **private** | frienly (default) | **protected** | **public** |
|---|---|---|---|---|
| Cùng class | Yes | Yes | Yes | Yes |

| Modifiers | **private** | frienly (default) | **protected** | **public** |
|---|---|---|---|---|
| Cùng gói, khác class | No | Yes | Yes | Yes |
| Lớp con trong cùng gói với lớp cha | No | Yes | Yes | Yes |
| Khác gói, khác lớp | No | No | No | Yes |
| Lớp con khác gói với lớp cha | No | No | Yes | Yes |

## 2.1.3 Phương thức khởi tạo

- Phương thức khởi tạo (constructor) là phương thức được thực thi ngay vào lúc khởi tạo đối tượng. Một lớp có thể có 0/1/n  contructor.
- Phương thức khởi tạo có tên phải trùng với tên lớp và không có kiểu trả về (kể cả trả về **void**).

## 2.1.4 Thành phần static

- Ta có thể truy xuất thành phần static (dữ liệu, phương thức) thông qua tên lớp.

```
ClassName.staticMemberName
```

# THỰC HÀNH CƠ BẢN

**Câu 1:** Xây dựng lớp học sinh (HocSinh), biết rằng mỗi học sinh gồm các thuộc tính

- Mã số, họ tên, điểm trung bình.
- Xây dựng phương thức get/set cho mỗi thuộc tính của lớp học sinh.
- Xây dựng phương thức khởi tạo cho lớp học sinh bằng cách sử dụng ba loại phương thức khởi tạo khác nhau.
- Xây dựng phương thức nhập học sinh, xuất học sinh
- Viết lớp Demo1 chứa phương thức main() để thực thi các chức năng:
  - Khởi tạo để tạo ra ba đối tượng học sinh bằng ba cách khác nhau.
  - Với cách tạo đối tượng học sinh bằng phương thức khởi tạo không có tham số, hãy gọi phương thức nhập và xuất thông tin học sinh ở lớp học sinh.
  - Thực hiện đổi tên của học sinh thành một tên mới.
  - Tìm tên của học sinh có điểm trung bình lớn nhất.
- Xây dựng lớp danh sách học sinh gồm các phương thức:
  - Nhập danh sách.
  - In danh sách.
  - Sắp xếp danh sách giảm dần theo điểm trung bình của học sinh
- Viết lớp Demo2 chứa phương thức main():
  - Tạo một đối tượng danh sách học sinh.
  - Nhập thông tin cho danh sách học sinh.
  - In danh sách học sinh đã được sắp thứ tự.

**Hướng dẫn:**

- Xây dựng lớp **HocSinh**

```
//File HocSinh.java
import java.util.Scanner;
public class HocSinh{
```

```
    private int maSo;
    private String hoTen;
    private float dtb;

    //phương thức get/set cho từng thuộc tính
    public int getMaso() {
        return maSo;
    }
    public void setMaso(int maSo) {
        this.maSo = maSo;
    }
    public String getHoten() {
        return hoTen;
    }
    public void setHoten(String hoTen) {
        this.hoTen = hoTen;
    }
    public float getDtb() {
        return dtb;
    }
    public void setDtb(float dtb) {
        this.dtb = dtb;
    }
    //phương thức khởi tạo không có tham số
    public HocSinh() {
        this.maSo = 0;
        this.hoTen = null;
        this.dtb = 0f;
    }
    //phương thức khởi tạo có tham số
    public HocSinh(int maSo, String hoTen, float dtb) {
        this.maSo = maSo;
        this.hoTen = hoTen;
        this.dtb = dtb;
    }
    //phương thức khởi tạo sao chép
    public HocSinh(HocSinh tmp) {
        this.maSo = tmp.maSo;
```

```
        this.hoTen = tmp.hoTen;
        this.dtb = tmp.dtb;
    }
    //phương thức nhập thông tin cho học sinh
    public void input(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Nhap ma so: ");
        maSo = sc.nextInt();
        System.out.println("Nhap ho ten: ");
        sc.nextLine();
        hoTen = sc.nextLine();
        System.out.println("Nhap diem trung binh: ");
        dtb = sc.nextFloat();
    }
    //phương thức xuất thông tin
    public void output(){
        System.out.println(maSo + " - " + hoTen + " - " + dtb);
    }
    //phương thức xếp loại
    public void rank(){
        if(dtb < 5)
            System.out.println("Xep loai yeu");
        else{
            if(dtb >=5 && dtb < 7)
                System.out.println("Xep loai trung binh");
            else {
                //tu viet code
            }
        }
    }
}
```

- Lớp **Demo1** chứa phương thức main() để thực thi các chức năng theo yêu cầu:

```
//File Demo1.java
public class Demo1 {
     public static void main(String[] args) {
       //tạo học sinh (hs1) sử dụng phương thức khởi tạo không có tham số
          HocSinh hs1 = new HocSinh();
          hs1.input(); // gọi phương thức nhập cho hs1
```

```
        hs1.output(); // xuất thông tin của học sinh hs1
        // tạo học sinh thứ 2 sử dụng phương thức khởi tạo có tham số
        HocSinh hs2 = new HocSinh(111, "Nguyen Ngoc Lan", 8.5f);
        hs2.output(); // xuất thông tin của học sinh hs2
        // tạo học sinh thứ 2 sử dụng phương thức khởi tạo sao chép từ hs2
        HocSinh hs3 = new HocSinh(hs2);
        hs3.output(); // xuất thông tin của học sinh hs3
        // đổi tên học sinh 3 thành tên mới
        hs3.setHoten("Phan Chau Tuan");
        // kiểm tra thông tin học sinh đã thay đổi
        hs3.output();
        // tìm tên của hs có đtb lớn nhất
        float max = hs1.getDtb();
        String ht = hs1.getHoten();
        if (max < hs2.getDtb()) {
            max = hs2.getDtb();
            ht = hs2.getHoten();
        }
        if (max < hs3.getDtb()) {
            max = hs3.getDtb();
            ht = hs3.getHoten();
        }
        System.out.println("Hoc sinh" + ht + "co DTB con nhat la: " + max);
    }
}
```

- Xây dựng lớp Danh sách học sinh (**DSHocSinh**) gồm các phương thức:

```
public class DSHocSinh {
   //các thuộc tính
   private HocSinh ds[]; //ds là mảng một chiều, mỗi phần tử lưu một học sinh
   private int soLuong;
   //các phương thức
   public void nhapDS(){
        //nhập số lượng học sinh
        //... tự code
        ds = new HocSinh[soLuong]; //cấp phát bộ nhớ cho ds
        //nhập thông tin cho từng học sinh trong ds
        for(int i = 0; i < soLuong; i++){
            //mỗi phần tử trong ds là 1 đối tượng hs
```

```
            //phải khởi tạo đối tượng trước khi thì mới nhập thông tin
            ds[i] = new HocSinh();
            ds[i].input();
        }
    }
    public void xuatDS(){
        System.out.println("Danh sach hoc sinh la: \n");
        for(int i = 0; i < soLuong; i++)
            ds[i].output();
    }
    public void sapXep(){
        //tự code
    }
}
```

- Viết lớp **Demo3** chứa phương thức main():

```
class Demo3{
    public static void main(String[] args){
        DSHocSinh danhsach = new DSHocSinh(); //tạo danh sách học sinh
        danhsach.nhapDS();
        danhsach.xuatDS();
        danhsach.sapXep();
    }
}
```

**Câu 2:** Viết chương trình thực hiện công việc sau:

1. Xây dựng lớp phân số bao gồm:

   Thành phần dữ liệu: tử số, mẫu số

   Xây dựng các phương thức cho lớp phân số:

- Phương thức khởi tạo không có tham số
- Phương thức khởi tạo hai tham số.
- Nhập phân số
- In phân số
- Tính ước số chung lớn nhất
- Rút gọn phân số

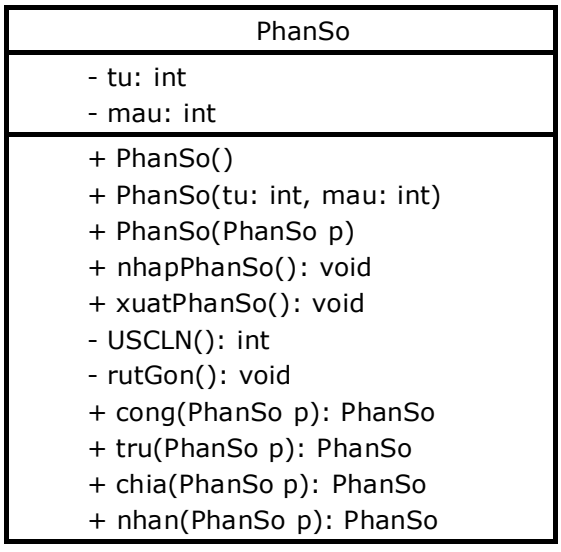
- Cộng hai phân số
- Trừ hai phân số
- Nhân hai phân số
- Chia hai phân số

2. Xây dựng lớp Demo chứa phương thức main() để thực hiện các chức năng sau:

- Tạo đối tượng phân số p1 dùng phương thức khởi tạo mặc định, xuất ra màn hình, quan sát kết quả. Sau đó gọi phương thức nhapPhanSo() cho phân số p1 và xuất kết quả ra màn hình.
- Tạo đối tượng phân số p2 = 4/16 dùng phương thức khởi tạo hai tham số, xuất ra màn hình, quan sát kết quả.
- Tạo đối tượng phân số p3 dùng phương thức khởi tạo hai tham số, với tham số là giá trị được nhập từ bàn phím.
- Cộng phân số p1 và phân số p3, xuất ra màn hình phân số kết quả.
- Tạo đối tượng phân số p4 dùng phương thức khởi tạo sao chép từ phân số kết quả tính được ở trên.
- Nhân p4 với p2, xuất ra màn hình phân số kết quả.

**Hướng dẫn:**

1. Xây dựng lớp **PhanSo** theo yêu cầu:



- Cách viết các phương thức khởi tạo (constructor):
  - Phương thức khởi tạo mặc định: Khi gọi phương thức này, chương trình sẽ tạo một phân số mặc định có tử số là 0 và mẫu số là 1.

```
// phương thức khởi tạo mặc định
public PhanSo() {
    tu = 0;
    mau = 1;
}
```

  - Phương thức khởi tạo phân số khi biết tử và mẫu, tham số truyền vào là hai giá trị tương ứng cho tử và mẫu. Khi gọi phương thức này, chương trình sẽ tạo ra một phân số có tử số và mẫu số bằng giá trị của tham số thứ nhất và thứ hai truyền vào.

```
// phương thức khởi tạo có tham số
public PhanSo(int tu, int mau) {
    this.tu = tu;
    this.mau = mau;
}
```

  - Phương thức khởi tạo sao chép: tạo phân số từ một phân số khác.

```
// phương thức khởi tạo sao chép
public PhanSo(PhanSo) {
    tu = p.tu;
    mau = p.mau;
}
```

- Viết phương thức nhập và xuất phân số:

```
//nhập phân số
public void nhapPhanSo(){
 // nhập tử số --- tự viết
 // nhập mẫu số
}
//xuất phân số
public void xuatPhanSo(){
 // xuất ra màn hình dưới dạng tu/mau
 // ví dụ: 3/4 -- tự viết
}
```

- Viết phương thức tìm ước chung lớn nhất của hai số nguyên dương để dùng cho việc rút gọn phân số:

$$UCLN(a,b) = \begin{cases} a & \text{nếu } a = b \\ UCLN(a-b,b) & \text{nếu } a > b \\ UCLN(a,b-a) & \text{nếu } a < b \end{cases}$$

```
// tìm ước chung lớn nhất của hai số nguyên
// trả về: giá trị ucln tìm được
private int UCLN(int a, int b){
 // tự viết
}
```

- Viết phương thức rút gọn hai phân số:

```
// rút gọn phân số
private void rutGon(){
 int ucln = UCLN(tu, mau);
 // tự viết tiếp ...
}
```

- Viết phương thức cộng hai phân số: cộng phân số là đối tượng của lớp đang xét với phân số p là tham số truyền vào. Kết quả trả về là phân số tổng.
  - Bước 1: tìm mẫu số chung bằng cách lấy hai mẫu số của hai phân số nhân nhau (mẫu số của phân số kết quả bằng mẫu số chung).
  - Bước 2: tìm tử số của phân số kết quả bằng cách cộng tử của hai phân số sau khi quy đồng.
  - Bước 3: rút gọn phân số kết quả và trả về

```
// cộng hai phân số
public PhanSo cong(PhanSo p) {
   PhanSo kq = new PhanSo();

   kq.mau = mau * p.mau;
   kq.tu = tu * p.mau + p.tu * mau;

   // sau khi cộng xong, rút gọn phân số kết quả
   kq.rutGon() ;
```

```
        return kq;
    }
```

- *Các phương thức trừ, nhân, chia làm tương tự.*

2. Xây dựng lớp Demo chứa phương thức main() thực hiện:

- Tạo đối tượng phân số p1 dùng phương thức khởi tạo mặc định, xuất ra màn hình, quan sát kết quả.
  - Gọi phương thức nhapPhanSo() cho phân số p1 và xuất ra màn hình.
  - **Lưu ý:** Với cách khởi tạo này thì phân số tạo ra luôn có tử = 0 và mẫu = 1. Nếu muốn phân số có tử mẫu tùy muốn thì phải sử dụng phương thức nhapPhanSo()
- Tạo đối tượng phân số p2 = 4/16 dùng phương thức khởi tạo hai tham số, xuất ra màn hình, quan sát kết quả.

```
PhanSo p2 = new PhanSo(4, 16);
p2.xuatPhanSo();
```

- Tạo đối tượng phân số p3 dùng phương thức khởi tạo hai tham số, với tham số là giá trị được nhập từ bàn phím

```
int t, m;
//nhập giá trị cho 2 biến t, m
//... tự nhập
//khởi tạo phân số p3 có tử mẫu theo tham số truyền vào là t, m
PhanSo p3 = new PhanSo(t, m);
p3.xuatPhanSo();
```

  - **Lưu ý:** Với cách khởi tạo này thì phương thức nhapPhanSo() không được dùng đến → không phải viết phương thức nhapPhanSo() khi sử dụng cách khởi tạo này.
- Cộng phân số p1 và phân số p3, xuất ra màn hình phân số kết quả.
- Tạo đối tượng phân số p4 dùng phương thức khởi tạo sao chép từ phân số kết quả tính được ở trên.
- Nhân p4 với p2, xuất ra màn hình phân số kết quả.

**Câu 3:** Xây dựng lớp Coodinate (tọa độ của điểm trong không gian hai chiều)

- Thành phần dữ liệu: hoành độ x và tung độ y.

- Xây dựng các phương thức gồm:
  - Phương thức khởi tạo không tham số: khởi tạo với hai giá trị x = 0, y = 0
  - Phương thức khởi tạo hai tham số.
  - Phương thức tính tổng các thành phần x và y của hai điểm.
  - Phương thức tìm điểm đối xứng qua trục hoành của một điểm.
  - Phương thức in tọa độ của một điểm.

# THỰC HÀNH NÂNG CAO

**Câu 4:** Xây dựng lớp NhanVien (Nhân Viên) gồm:

- Thành phần dữ liệu các thuộc tính: mã số, họ tên, lương cơ bản, hệ số lương, số lượng nhân viên
- Xây dựng các phương thức:
  - Phương thức khởi tạo mặc định, có tham số, sao chép
  - Các phương thức: set/get cho mã số, họ tên, hệ số lương
  - Các phương thức: nhập dữ liệu cho nhân viên từ bàn phím, xuất đối tượng nhân viên, tính lương cho nhân viên, in số lượng nhân viên.
- Xây dựng lớp danh sách nhân viên gồm các phương thức: nhập danh sách nhân viên, xuất danh sách nhân viên.
- Xây dựng lớp Demo chứa phương thức main() thực hiện các việc sau:
  - Tạo ba đối tượng Nhân viên bằng ba cách khác nhau
  - Nhập dữ liệu cho một nhân viên từ bàn phím, xuất kết quả ra màn hình
  - Thay đổi họ tên cho nhân viên và xuất ra màn hình.
  - In ra thông tin của nhân viên có hệ số lương cao nhất trong 3 nhân viên.
  - Nhập danh sách nhân viên và xuất ra màn hình cùng với lương của mỗi nhân viên.
  - In số lượng nhân viên trong danh sách.

**Hướng dẫn:**

Xây dựng lớp NhanVien tương tự các bài trên. Điểm khác biệt là lớp này có các thành phần tĩnh (static): dữ liệu tĩnh và phương thức tĩnh.

1. Xây dựng lớp NhanVien có các dữ liệu sau:

```
String maSo, hoTen;
static float luongCoBan;
static int soLuongNhanVien = 0;
float heSoLuong;
```

- Tự xây dựng các phương thức: các constructor mặc định, có tham số và sao chép; các phương thức: set/get cho mã số, họ tên, hệ số lương; các phương thức: nhập dữ liệu cho nhân viên từ bàn phím.
- Phương thức tính lương cho nhân viên: tự viết
- Phương thức in số lượng nhân viên: là phương thức tĩnh

```
public static void inSoLuongNhanVien(){
     System.out.print ("So luong nhan vien: " + soLuongNhanVien);
}
```

2. Xây dựng lớp Demo chứa phương thức main() thực hiện các việc sau:

- Tạo ba đối tượng Nhân viên bằng ba cách khác nhau
- Nhập dữ liệu cho một nhân viên từ bàn phím, xuất ra màn hình
- Thay đổi họ tên cho nhân viên và xuất ra màn hình
- In ra thông tin của nhân viên có hệ số lương cao nhất trong ba nhân viên.

```
float max = nv1.getHeSoLuong();
NhanVien nvt = new NhanVien(nv1);
if(nv2.getHeSoLuong() > max){
     max = nv2.getHeSoLuong();
     nvt = nv2;
}
if(nv3.getHeSoLuong() > max){
     max = nv3.getHeSoLuong();
     nvt = nv3;
}
System.out.println("Nhan vien co he so luong lon nhat: ");
System.out.println(nvt);
```

- Nhập danh sách nhân viên và xuất ra màn hình cùng với lương của mỗi nhân viên.
- In số lượng nhân viên có trong danh sách.

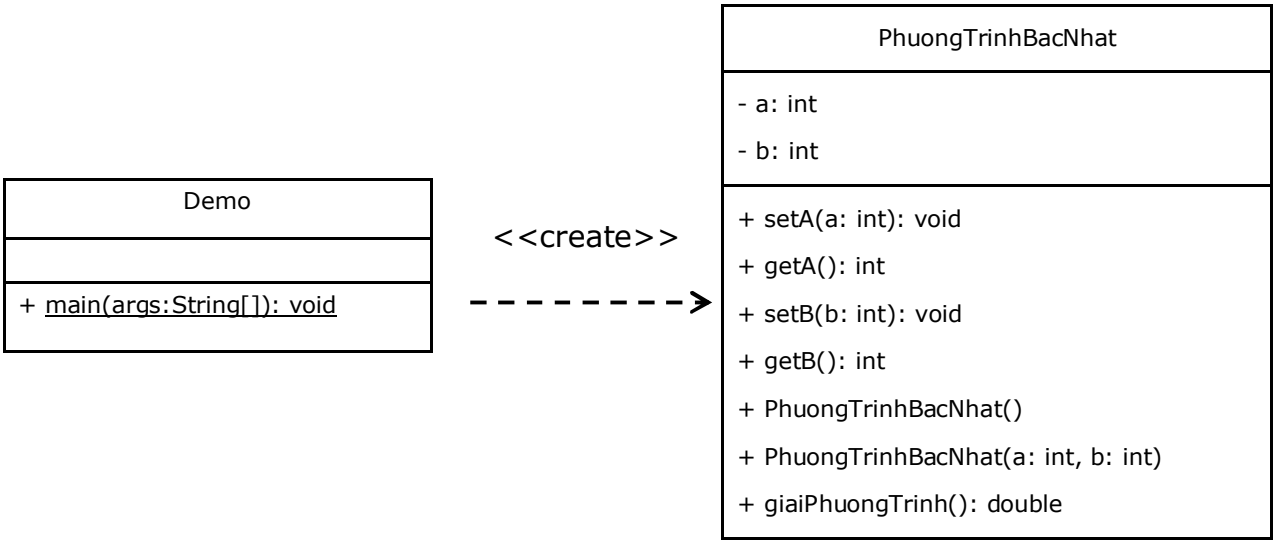
**Câu 5:** Xây dựng lớp TamGiac (Tam giác) gồm các thành phần dữ liệu: Độ dài cạnh thứ nhất, độ dài cạnh thứ hai, độ dài cạnh thứ ba của tam giác.

- Viết phương thức khởi tạo (constructor) với ba tham số kiểu int tương ứng là ba cạnh của tam giác
- Viết các phương thức của đối tượng tam giác: Tính chu vi tam giác, tính diện tích tam giác và xác định loại tam giác.
- Viết lớp thử nghiệm (lớp Demo) cho lớp tam giác vừa tạo (trong lớp này chứa phương thức main() để mô tả một vài đối tượng được tạo từ lớp tam giác. Cho biết diện tích và chu vi của mỗi tam giác, đồng thời cho biết loại tam giác.

# BÀI TẬP VỀ NHÀ

**Câu 6:** Tạo lớp PhuongTrinhBacNhat có hai biến thực thể a và b là hai số nguyên.

- Định nghĩa các phương thức set/get cho các biến thực thể.
- Tạo hai phương thức khởi tạo (constructor) cho đối tượng:
  - Phương thức khởi tạo mặc định: khởi gán các giá trị bằng 0 cho các dữ liệu của đối tượng.
  - Phương thức khởi tạo có tham số: constructor có đầy đủ tham số (số tham số của constructor này bằng với số thành phần dữ liệu của đối tượng). Constructor này thường dùng để khởi tạo đối tượng đầy đủ.
- Viết một phương thức giaiPT dùng để giải phương trình bậc nhất ax + b = 0
- Viết lớp thử nghiệm (lớp Demo) của lớp PhuongTrinhBacNhat vừa tạo.
- UML class diagram:


Demo
+ main(args:String[]): void
<<create>>
PhuongTrinhBacNhat
- a: int
- b: int
+ setA(a: int): void
+ getA(): int
+ setB(b: int): void
+ getB(): int
+ PhuongTrinhBacNhat()
+ PhuongTrinhBacNhat(a: int, b: int)
+ giaiPhuongTrinh(): double

# BÀI 3: MẢNG VÀ CHUỖI

*Bài này giúp người học nắm được các nội dung sau:*

- *Sử dụng được mảng một chiều trong Java;*
- *Hiểu và ứng dụng mảng đối tượng;*
- *Sử dụng được mảng hai chiều trong các ứng dụng;*
- *Hiểu và sử dụng chuỗi trong Java*

## 3.1 TÓM TẮT LÝ THUYẾT

### 3.1.1 Mảng một chiều

Mảng (array) là một tập hợp các phần tử có cùng kiểu được lưu trữ gần nhau trong bộ nhớ.

Khai báo mảng một chiều (kiểu nguyên thủy):

```
dataType[] arrayName; //  dataType arrayName[];
arrayName = new dataType[size];
Hoặc
dataType[] arrayName = new dataType[size];
```

Khai báo mảng một chiều (kiểu tham chiếu):

```
ClassName[] objArray; //hoặc ClassName objArray[];
objArray = new ClassName[size];
Hoặc
ClassName objArray[] = new ClassName[size];
```

Vòng lặp for each:

```
for (dataType element : arrayName) {
  // code block to be executed
}
```

## 3.1.2 Mảng hai chiều

Dữ liệu mảng hai chiều được lưu trữ theo hàng và cột (hay còn gọi là dạng ma trận).

Khai báo mảng hai chiều:

```
dataType[][] arrayName; // dataType arrayName[][];
arrayName = new dataType[size1][size2];
hoặc
dataType[][] arrayName = new dataType[size1][size2];
```

Khai báo mảng hai chiều trong đó mỗi dòng có số cột khác nhau:

```
int[][] myArray = new int[2][];
myArray[0] = new int[2];
myArray[1] = new int[3];
hoặc
int[][] myArray = { {1, 2}, {3, 4, 5} };
```

## 3.1.3 Chuỗi

Trong Java, chuỗi là dãy các ký tự và thuộc kiểu tham chiếu. Gồm:

- **Lớp String**: là một lớp mà các cá thể hoặc đối tượng của nó có thể chứa chuỗi không thay đổi hoặc không đổi (chuỗi bất biến).

  Có hai cách để tạo chuỗi trong Java:

```
String line = "Hutech University";
String line = new String("Hutech University");
```

- **Lớp StringBuffer và StringBuilder**: StringBuilder và StringBuffer là rất giống nhau, điều khác biệt là tất cả các phương thức của StringBuffer đã được đồng bộ.

```
// Constructor của StringBuffer
StringBuffer() // an initially-empty StringBuffer
StringBuffer(int size) // with the specified initial size
StringBuffer(String s) // with the specified initial content
// Độ dài
```

```
int length()

// Các method xây dựng nội dung
StringBuffer append(type arg) //type là kiểu dữ liệu
StringBuffer insert(int offset, type arg)

// Các method thao tác trên nội dung
StringBuffer delete(int start, int end)
StringBuffer deleteCharAt(int index)
void setLength(int newSize)
void setCharAt(int index, char newChar)
StringBuffer replace(int start, int end, String s)
StringBuffer reverse()

// Các method trích ra toàn bộ hoặc một phần dữ liệu.
char charAt(int index)
String substring(int start)
String substring(int start, int end)
String toString()

// Các method tìm kiếm vị trí.
int indexOf(String searchKey)
int indexOf(String searchKey, int fromIndex)
int lastIndexOf(String searchKey)
int lastIndexOf(String searchKey, int fromIndex)
```

# THỰC HÀNH CƠ BẢN

**Câu 1:** Viết chương trình nhập vào một mảng gồm n số nguyên và thực hiện các công việc sau:

- Xuất giá trị các phần tử mảng ra màn hình.
- Tìm giá trị phần tử lớn nhất (max), nhỏ nhất (min) trong mảng.
- Tìm các số nguyên tố có trong mảng.
- Sắp xếp mảng theo thứ tự tăng dần.

**Hướng dẫn:**

```
//File Cau1.java
import java.util.Scanner;
public class Cau1 {
    public static void main(String[] args) {
        int[] a;
        int n;
        Scanner scanner = new Scanner(System.in);
        System.out.print("Nhap n: ");
        n = scanner.nextInt();
        //Khởi tạo mảng
        a = new int[n];
        nhap(a, n);
        xuat(a, n);
        //In ra min max
        System.out.println("Min: " + min(a,n));
        System.out.println("Max: " + max(a,n));
        //in số nguyên tố trong mảng
        System.out.print("Cac so nguyen to la: ");
        for (int i = 0; i < n; i++) {
            if (check(a[i]))
                System.out.print(a[i] + " ");
        }
        System.out.println("");
        //gọi phương thức sắp xếp và xuất mảng
        sapXep(a, n);
        xuat(a, n);
    }
    //phương thức nhập mảng
    public static void nhap(int a[], int n){
```

```
       //sinh viên tự viết
    }
    //phương thức xuất mảng
    public static void xuat(int a[], int n){
       //sinh viên tự viết
    }
    //phương thức tìm phần tử min
    public static int min(int a[], int n){
         //sinh viên tự viết
    }
    //phương thức tìm phần tử max
    public static int max(int a[], int n){
         //sinh viên tự viết
    }
    //phương thức kiểm tra số nguyên tố
    public static boolean check(int n){
         //sinh viên tự viết
    }
    //phương thức sắp xếp mảng tăng
    public static void sapXep(int a[], int n){
         //sinh viên tự viết
    }
}
```

**Câu 2:** Viết chương trình thực hiện công việc sau:

- Tạo lớp Account gồm hai thuộc tính a, b (số nguyên) và hai phương thức: setData(int, int) để gán giá trị cho hai thuộc tính a, b; showData() dùng để hiển thị giá trị của hai thuộc tính a và b.
- Tạo lớp Demo (chứa phương thức main()): tạo mảng đối tượng Account gồm hai phần tử, gọi phương thức setData và showData cho từng đối tượng.

**Hướng dẫn:**

- Lớp Account:

```
class Account{
    int a;
    int b;
        //Gán giá trị cho thuộc tính
    public void setData(int c, int d){
        a = c;
```

```
        b = d;
    }

    // Hàm in thông tin đối tượng
    public void showData(){
        System.out.println("Giá trị của a = " + a);
        System.out.println("Giá trị của b = " + b);
    }
}
```

- Lớp Demo:

```
public class Demo {
    public static void main(String args[]){
        //Khởi tạo mảng đối tượng
        Account obj[] = new Account[2] ;
        obj[0] = new Account();
        obj[1] = new Account();
        // Gán dữ liệu
        obj[0].setData(1, 2);
        obj[1].setData(3, 4);
        // In đối tượng ra màn hình
        System.out.println("Phần tử mảng thứ 0");
        obj[0].showData();
        System.out.println("Phần tử mảng thứ 1");
        obj[1].showData();
    }
}
```

**Câu 3:** Viết chương trình thực hiện các công việc sau:

a. Nhập ma trận vuông (n×n) gồm các số nguyên.

b. Xuất ma trận ra màn hình.

c. Tìm phần tử lớn nhất trong ma trận.

d. Tính tổng các phần tử trên đường biên ma trận.

e. Sắp xếp các phần tử trên ma trận tăng dần theo hình zigzag.

f. Kiểm tra ma trận có đối xứng qua đường chéo chính không?

**Câu 4:** Viết chương trình nhập vào một chuỗi và thực hiện các công việc sau:

a. Đếm khoảng trắng có trong chuỗi.

b. Đếm số từ có trong chuỗi.

c. Thống kê số lần xuất hiện của mỗi ký tự trong chuỗi.

# THỰC HÀNH NÂNG CAO

**Câu 5:** Viết chương trình thực hiện các công việc sau:

a. Tạo lớp Sach gồm:

- Thuộc tính: mã sách, tiêu đề, tác giả
- Phương thức: khởi tạo (mặc định và có tham số), hiển thị thông tin sách.

b. Tạo lớp Demo chứa phương thức main() dùng để:

- Tạo danh sách các đối tượng của lớp Sach với n quyển sách.
- Nhập danh sách các đối tượng của lớp Sach
- Xuất thông tin danh sách các đối tượng của lớp Sach ra màn hình.

# BÀI TẬP VỀ NHÀ

**Câu 6:** Bạn được yêu cầu xây dựng chương trình gồm có lớp Order và lớp Item thỏa mãn yêu cầu sau:

a. Mỗi Order (đơn hàng) đều có một ID (mã đơn)

b. Mỗi Order (đơn hàng) có một danh sách Item (mặt hàng)

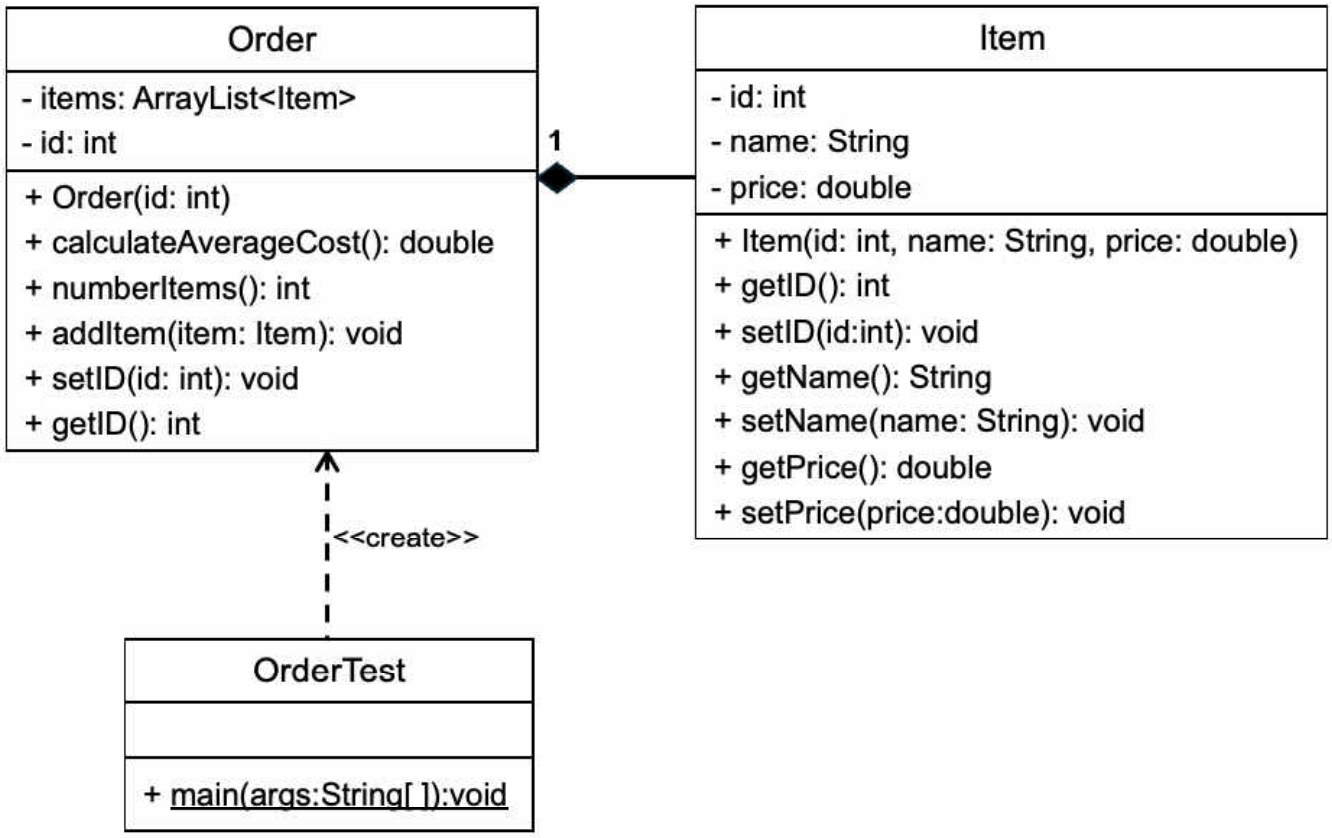
c. Mỗi Item (mặt hàng) đều có ID, tên và giá

d. Mỗi lớp có các phương thức khởi tạo, phương thức get và set phù hợp.

e. Lớp Order có phương thức double calculateAverageCost() để tính giá trị trung bình của chi phí của tất cả các mặt hàng trong một đơn đặt hàng.

f. Tạo một lớp có phương thức main() để nhận đầu vào của các mặt hàng của một đơn hàng từ bàn phím và hiển thị chi phí trung bình.

**Hướng dẫn:** Minh họa sơ đồ lớp như sau:



# BÀI 4: KẾ THỪA VÀ ĐA HÌNH

Bài này giúp người học nắm được các nội dung sau:

- *Tính kế thừa là gì trong LTHĐT, các loại kế thừa trong các ngôn ngữ LTHĐT;*
- *Đa hình là gì trong LTHĐT;*
- *Hiện thực được đặc điểm kế thừa trong LTHĐT với Java;*
- *Hiện thực được đặc điểm đa hình trong LTHĐT với Java.*

## 4.1 TÓM TẮT LÝ THUYẾT

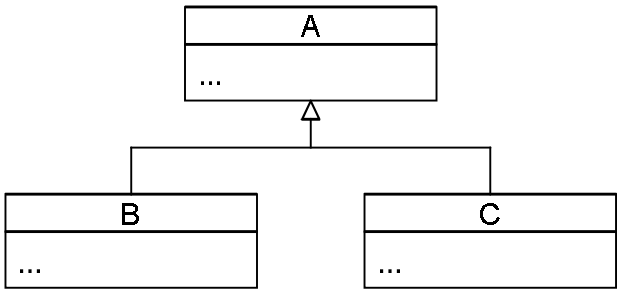
### 4.1.1 Hiện thực lớp con trong Java

Khai báo lớp kế thừa từ lớp khác:

```
class SubClassName extends SuperClassName {
     // ...
}
```

### 4.1.2 Kỹ thuật phân cấp kế thừa

- Liệt kê đặc điểm của các đối tượng cần quan tâm.
- Tìm tập giao của các tính chất giữa các lớp, tách tập giao này để xây dựng lớp cha.
- Đặt một tên gợi có ý nghĩa cho lớp cha.
- Phần còn lại sau khi tách tập giao là các lớp con.




**Hình 3.1: Minh họa phân cấp kế thừa**

## 4.1.3 Ghi đè phương thức

Ghi đè (Override) là phương thức thuộc lớp cha được định nghĩa lại ở lớp con. Trong Java, ghi đè phương thức (overriding method) là phương pháp "viết lại" cách thức hoạt động của phương thức của lớp cha (hay lớp dẫn xuất).

**Ví dụ 4.1:** Sử dụng ghi đè phương thức trong Java

```
class BaseClass {
    public void doSomething(String str){
        System.out.println("Base impl:" + str);
    }
}
class ChildClass extends BaseClass{
   public void doSomething(String str){
       System.out.println("Child impl:" + str);
   }
}
public class OverrideTest {
    public static void main(String[] args) {
        BaseClass bc = new ChildClass();
        bc.doSomething("override");
    }
}
```

# THỰC HÀNH CƠ BẢN

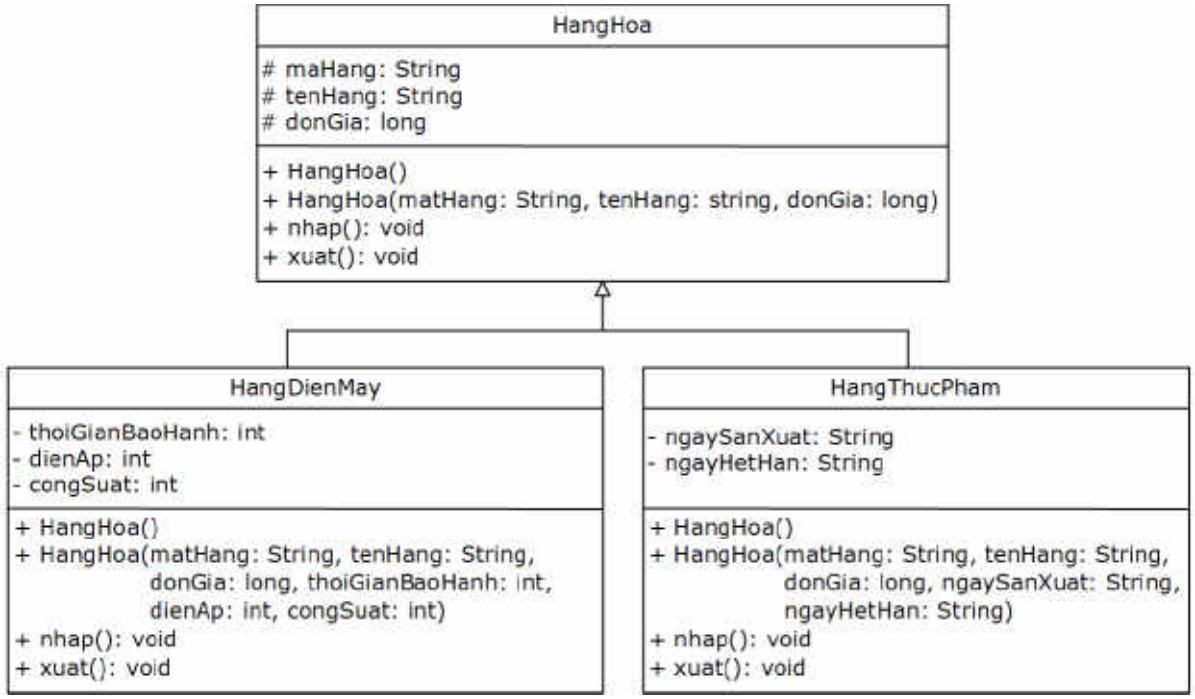
**Câu 1:** Phân tích phân cấp kế thừa cho các lớp:

- HangDienMay (Hàng Điện Máy) gồm các thuộc tính: Mã hàng, tên hàng, giá, thời gian bảo hành, điện áp, công suất.
- HangThucPham (Hàng Thực Phẩm) gồm các thuộc tính: Mã hàng, tên hàng, giá, ngày sản xuất, ngày hết hạn dùng.

Xây dựng các lớp theo sơ đồ phân cấp kế thừa và lớp thử nghiệm chứa phương thức main() cho phép tạo mỗi loại một mặt hàng cụ thể, sau đó xuất thông tin về các mặt hàng này.

**Hướng dẫn:**

1. Sơ đồ lớp phân cấp kế thừa như sau:




- Đặc điểm chung giữa hai lớp HangDienMay và HangThucPham: mã hàng, tên hàng và đơn giá.
- Xây dựng lớp cha gồm những thành phần chung của hai lớp, giả sử đặt tên lớp cha là HangHoa (Hàng Hoá), lớp cha có những thành phần dữ liệu là: maHang, tenHang và donGia. Các phương thức chung là: nhap() và xuat().

2. Cài đặt các lớp dựa trên sơ đồ lớp:

- Cài đặt lớp HangHoa, trong đó các thành phần dữ liệu: maHang, tenHang và donGia được khai báo với phạm vi **protected** *(cho phép các lớp con kế thừa nhưng không được truy xuất bên ngoài lớp cha và bên ngoài lớp con)*.

```
class HangHoa {
   protected string maHang;
   protected string tenHang;
   protected long donGia;

   // phương thức khởi tạo mặc định
   public HangHoa() {
     maHang = "";
     tenHang = "";
     donGia = 0;
   }
   // phương thức khởi tạo có tham số
   public HangHoa(String matHang, string tenHang, long donGia){
     // tự viết ...
   }
   // phương thức nhập
   public void nhap(){
     // tự viết ...
   }
   // phương thức xuất
   public void xuat(){
     // tự viết ...
   }
}
```

- Cài đặt lớp HangDienMay kế thừa lớp HangHoa, HangDienMay sẽ có tất cả các thành phần của lớp HangHoa, ngoài ra nó có thêm một số thành phần dữ liệu riêng: thoiGianBaoHanh (thời gian bảo hành), dienAp (điện áp) và congSuat (công suất)

  - Cú pháp khai báo lớp kế thừa:

    **class** *SubClass* **extends** *SuperClass* **{**

      **// ...**

    **}**

```
class HangDienMay extends HangHoa
   private int thoiGianBaoHanh;
   private int dienAp;
   private int congSuat;
   // ...
}
```

- Viết các phương thức khởi tạo cho HangDienMay.
  - Một mặt hàng điện máy có bao nhiêu thuộc tính (properties)? → có sáu thuộc tính, ba thuộc tính kế thừa từ lớp cha (maHang, tenHang, donGia) và ba thuộc tính (thoiGianBaoHanh, dienAp, congSuat) của riêng nó.
  - Phương thức khởi tạo mặc định

```
public HangDienMay()
   super(); // Gọi phương thức mặc định của lớp cha
   thoiGianBaoHanh = 0;
   dienAp = 0;
   congSuat = 0;
}
```

  - Phương thức khởi tạo có tham số: có sáu tham số gồm ba tham số khởi tạo các thuộc tính của lớp HangHoa (maHang, tenHang, donGia) và ba tham số còn lại để khởi tạo các thuộc tính: thoiGianBaoHanh, dienAp, congSuat.

```
public HangDienMay(String matHang, string tenHang, long donGia,
                   int thoiGianBaoHanh, int dienAp, int congSuat)
   // Gọi phương thức có tham số của lớp cha
   // để khởi tạo dữ liệu cho các thuộc tính của lớp cha
   super(matHang, tenHang, donGia);
   this.thoiGianBaoHanh = thoiGianBaoHanh;
   this.dienAp = dienAp;
   this.congSuat = congSuat;
}
```

- Viết phương thức nhập cho HangDienMay (tự viết)
  - Nhập giá trị cho các thuộc tính kế thừa, gọi phương thức nhập của lớp cha.

Cú pháp gọi phương thức của lớp cha:

**super.***methodName***(**[*parameterList*]**);**

- Nhập giá trị cho các thuộc tính của riêng nó (nhập như bình thường)

- Viết phương thức xuất cho HangDienMay (tự viết)
  - Xuất giá trị của các thuộc tính kế thừa, gọi phương thức xuất của lớp cha.
  - Xuất giá trị của các thuộc tính của riêng nó.
- Cài đặt lớp HangThucPham kế thừa lớp HangHoa tương tự như lớp HangDienMay.
- Cài đặt lớp Demo chứa phương thức main() cho phép tạo mỗi loại một mặt hàng cụ thể (sử dụng các cách khởi tạo khác nhau), sau đó xuất thông tin về các mặt hàng này.
  - Biến đối tượng có thể khai báo biến là lớp cha nhưng khởi tạo biến là đối tượng thuộc lớp con.
  - Ví dụ: HangHoa h = new HangDienMay();

**Câu 2:** Sử dụng Câu 1, tiếp tục xây dựng lớp DSHangHoa (Danh Sách Hàng Hoá)

1. Để lưu trữ danh sách hàng hóa, ta sử dụng mảng một chiều. Vì mảng chứa nhiều mặt hàng, vừa có hàng điện máy vừa có hàng thực phẩm nên kiểu dữ liệu của mảng không thể là kiểu HangDienMay hay HangThucPham được.

(*) Trước tiên, khai báo kiểu của mảng là kiểu HangHoa. Khi tạo mặt hàng cụ thể cho từng phần tử của mảng ta mới khởi tạo theo loại mặt hàng mong muốn.

2. Xây dựng lớp DSHangHoa có thuộc tính: số lượng mặt hàng (soLuong), danh sách các mặt hàng (danhSach) và phương thức:

- Khởi tạo danh sách ban đầu chưa có mặt hàng nào (số lượng là 0)
- Thêm một mặt hàng vào danh sách

```
public void themMatHang(HangHoa h){
        danhSach[soLuong++] = h;
}
```

- Xuất mặt hàng theo loại

```
public void xuatDanhSachTheoLoai (byte loai) {
    for(int i = 0; i < soLuong; i++){
      if(loai == 1 && danhSach[i] instanceof HangDienMay)
        danhSach[i].xuat();
      if(loai != 1 && danhSach[i] instanceof HangThucPham)
        danhSach[i].xuat();
    }
}
```

3. Xây dựng phương thức main() để chương trình thực hiện theo chức năng như sau:

```
byte chon, loai;
Scanner doc = new Scanner(System.in);
DSHangHoa dshh = new DSHangHoa();
HangHoa h;
do {
  System.out.println("1: Them mot mat hang vao dang sach");
  System.out.println("2: Xuat mat hang theo loai");
  System.out.println("0: Thoat");
  System.out.println("Chon chuc nang:");
  chon = doc.nextByte();
  switch(chon){
    case 1:
      System.out.println("1: hang dien may, 2: hang thuc pham. " +
                         "Hay chon loai mat hang:");
      loai = doc.nextByte();
      if (loai == 1)
        h = new HangDienMay();
      else
        h = new HangThucPham();
      h.nhap();
      dshh.themMatHang(h);
      break;
    case 2:
      System.out.println("Xuat danh sach mat hang nao" +
                          "(1.Dien may/2.Thuc pham):");
      loai = doc.nextByte();
      dshh.xuatDanhSachTheoLoai(loai);
      break;
    default:
      chon = 0;
      break;
  }
} while (chon != 0);
```

# THỰC HÀNH NÂNG CAO

**Câu 3:** Để quản lý phân loại sách tại cửa hàng sách với các thông tin sau:

- Sách văn học: mã sách, tên sách, tác giả, giá và lần xuất bản.
- Sách giáo khao: mã sách, tên sách, tác giả, giá, lớp và nhà xuất bản.

a. Hãy phân tích phân cấp kế thừa cho các lớp.

b. Xây dựng các lớp theo sơ đồ phân cấp kế thừa và lớp thử nghiệm chứa phương thức main() cho phép tạo mỗi loại sách cụ thể, sau đó xuất thông tin về các loại sách này.

**Câu 4:** Sử dụng Câu 3 thực hiện các chức năng sau:

- Nhập thông tin các quyển sách trên cùng một danh sách chứa cả sách văn học và sách giáo khoa.
- Xuất thông tin các quyển sách và giá sau khi giảm, biết rằng:
  - Sách văn học được giảm 20% nếu giá sách > 300.000 đồng
  - Sách giáo khoa được giảm giá 10% nếu do nhà xuất bản "Giáo Dục" xuất bản
- Tìm sách có giá lớn nhất sau khi được giảm.

**Hướng dẫn:**

- Xây dựng lớp **Sach**:

```
class Sach {
    //các thuộc tính
    String maSach;
    String tenSach;
    String tacGia;
    double gia;
    //phương thức khởi tạo mặc định
    public Sach() {
        maSach = null;
        tenSach = null;
        tacGia = null;
        gia = 0.0;
    }
    //phương thức khởi tạo có tham số
```

```
    public Sach(String ms, String tens, String tg, double g) {
        maSach = ms;
        tenSach = tens;
        tacGia = tg;
        gia = g;
    }
    //phương thức set tên sách
    public void setTenSach(String ts)   {
        tenSach = ts ;
    }
    //phương thức set tên sách
    public String getTenSach() {
        return tenSach ;
    }
    public void input()   {
        //tu viet nhap thong tin sach
    }
    public void output() {
        //tu viet xuat thong tin sach
    }
    //phương thức tỉ lệ phần trăm giảm giá để lớp con override
    public double reducePrice();
}
```

- Xây dựng lớp **VanHoc**:

```
class VanHoc extends Sach {
    int lanXuatBan;
    VanHoc() {
        //tự viết
    }
    VanHoc(String ms, String tens, String tg, double g, int lxb) {
        // tự viết
    }
    //// tự viết các phương thức cần thiết
}
```

- Xây dựng lớp GiaoKhoa:

```
class GiaoKhoa extends Sach {
    int lop;
    String nhaXuatBan;
    GiaoKhoa() {
        // tự viết
    }
```

```
    GiaoKhao(String ms, String tens, String tg, double g, int l, String nxb){
        // tự viết
    }
    //// tự viết các phương thức cần thiết
}
```

# BÀI TẬP VỀ NHÀ

**Câu 5:** Để quản lý các phương tiện cá nhân chính chủ đối với các phương tiện Xe máy, xe đạp, xe hơi gồm các thuộc tính chung: Mã nhân viên, họ tên nhân viên, giá xe. Đối với mỗi phương tiện khác nhau gồm các thông tin riêng cần quản lý:

- Xe đạp: quản lý thông tin xuất xứ
- Xe máy: quản lý biển số xe và loại xe
- Xe hơi: quản lý biển số xe, thời gian đăng kiểm gần nhất

Xây dựng các lớp theo sơ đồ phân cấp kế thừa và lớp thử nghiệm chứa phương thức main() cho phép tạo mỗi loại xe cụ thể, sau đó xuất thông tin về các loại xe này.

**Câu 6:** Sử dụng Câu 5, tiếp tục xây dựng lớp chứa một danh sách gồm nhiều loại xe như xe đạp, xe máy, xe hơi. Xây dựng phương thức main() sao cho chương trình có các chức năng sau:

- Tạo một danh sách gồm nhiều loại xe khác nhau.
- Xuất danh sách xe theo từng loại xe.
- Tìm các xe có giá cao nhất.
- Sắp xếp danh sách xe theo thứ tự giá xe từ thấp đến cao.

# BÀI 5: LỚP TRỪU TƯỢNG VÀ GIAO DIỆN

Bài này giúp người học nắm được các nội dung sau:

- *Hiểu cách xây dựng lớp trừu tượng;*
- *Sử dụng được lớp trừu tượng trong lập trình hướng đối tượng;*
- *Biết cách xây dựng giao diện;*
- *Áp dụng được giao diện cho các yêu cầu thực tế.*

## 5.1 TÓM TẮT LÝ THUYẾT

### 5.1.1 Lớp trừu tượng

❖ **Đặc điểm lớp trừu tượng**

- Một lớp được khai báo với từ khóa **abstract** là lớp trừu tượng.
- Lớp trừu tượng có thể có các phương thức trừu tượng hoặc phương thức thông thường.
- Lớp trừu tượng có thể khai báo 0, 1 hoặc nhiều phương thức trừu tượng.
- Không thể khởi tạo một đối tượng trực tiếp từ một lớp trừu tượng.

❖ **Khai báo lớp trừu tượng:**

```
[public] abstract class ClassName {
    // các thuộc tính
    // các phương thức
}
```

❖ **Phương thức trừu tượng:**

```
[modifiers] abstract returnType methodName([parameterList]);
```

**Ví dụ 5.1:** Minh họa lớp trừu tượng

```
abstract class Animal {
   abstract void makeSound();
}
class Dog extends Animal {
   public void makeSound() {
      System.out.println("Bark bark.");
   }
}
class Cat extends Animal {
   public void makeSound() {
      System.out.println("Meows ");
   }
}
class Main {
   public static void main(String[] args) {
      Dog d1 = new Dog();
      d1.makeSound();
      Cat c1 = new Cat();
      c1.makeSound();
   }
}
```

## 5.1.2 Giao diện

Giao diện (interface) là một kiểu dữ liệu tham chiếu trong Java. Nó là tập hợp các phương thức trừu tượng. Khi một lớp hiện thực (implements) giao diện, thì nó sẽ kế thừa những phương thức trừu tượng của giao diện đó.

- Một số đặc điểm của giao diện:
  - Không thể khởi tạo, nên không có phương thức khởi tạo.
  - Tất cả các phương thức trong giao diện luôn ở dạng **public abstract** mà không cần khai báo.
  - Các thuộc tính trong giao diện luôn ở dạng **public static final** mà không cần khai báo, yêu cầu phải có giá trị.

- Mục đích của giao diện là để thay thế đa kế thừa của những ngôn ngữ khác (ví dụ như C++, Python…).

❖ **Khai báo interface:**

```
[public] interface interfaceName {
        // Khai báo các hằng
    // Khai báo các phương thức trừu tượng
    // Khai báo các phương thức mặc định
}
```

**Ví dụ 5.2:** Minh họa sử dụng interface

```
// File Mail.java
public interface Mail {
    void sendMail();
    void log();
}
// File MailImpl.java
public class MailImpl implements Mail {
    @Override
    public void sendMail() {
         System.out.println("Sending email");
    }
    @Override
    public void log() {
         System.out.println("Log!!!");
    }
}
// File Main.java
public class Main {
    public static void main(String[] args) {
         Mail mail = new MailImpl();
         mail.sendMail();
         mail.log();
    }
}
```

# THỰC HÀNH CƠ BẢN

**Câu 1:** Viết chương trình quản lý học viện như sau:

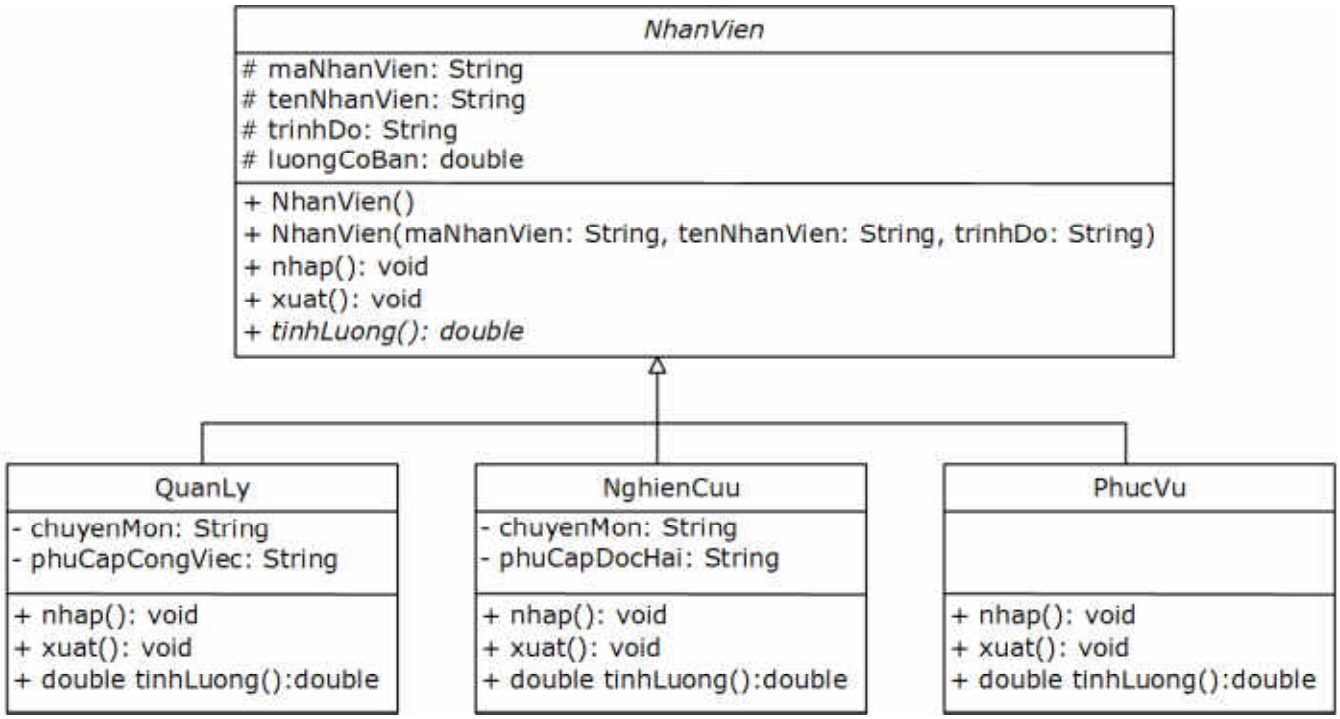
- Nhân viên quản lý (mã nhân viên, tên nhân viên, trình độ, chuyên môn, lương cơ bản, phụ cấp chức vụ). Lương = lương cơ bản + phụ cấp chức vụ
- Nhân viên nghiên cứu (mã nhân viên, tên nhân viên, trình độ, chuyên môn, lương cơ bản, phụ cấp độc hại). Lương = lương cơ bản + phụ cấp độc hại
- Nhân viên phục vụ (mã nhân viên, tên nhân viên, trình độ, lương cơ bản). Lương = lương cơ bản

Yêu cầu:

- Khái quát hóa các lớp được mô tả theo sơ đồ phân cấp kế thừa.
- Xây dựng lớp thử nghiệm chứa phương thức main(), trong đó khai báo các đối tượng nhân viên, nhập dữ liệu và tính lương cho từng nhân viên.

**Hướng dẫn:**

1. Sơ đồ lớp kế thừa trong chương trình:



Trong các phương thức của lớp NhanVien, phương thức nào là phương thức trừu tượng?

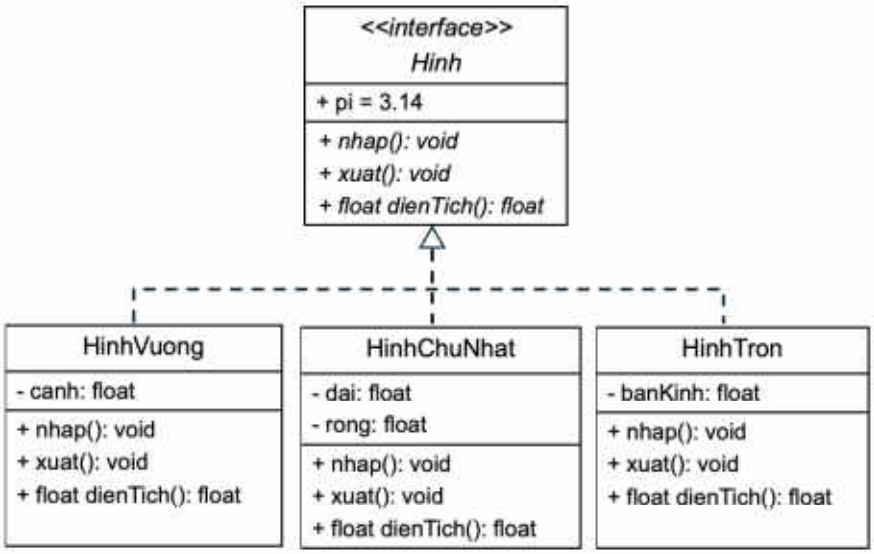
- Vì một nhân viên nói chung, ta chưa biết cách tính lương như thế nào nên phương thức tính lương là phương thức trừu tượng, do đó lớp *NhanVien* là lớp trừu tượng.
- Tùy thuộc vào từng loại nhân viên cụ thể mà có cách tính lương khác nhau (tường minh phương thức tính lương cho từng lớp con).

2. Xây dựng các lớp:

- Lớp trừu tượng *NhanVien*
  - Lớp *NhanVien* là lớp trừu tượng
  - Phương thức nhap() dùng để nhập các thuộc tính: maNhanVien, tenNhanVien, trinhDo, luongCoBan.
  - Phương thức Xuat() dùng để xuất các thuộc tính: maNhanVien, tenNhanVien, trinhDo, luongCoBan.
  - Phương thức *TinhLuong()* là phương thức trừu tượng nên chỉ khai báo mà không cài đặt chi tiết.
- Lớp QuanLy
  - Lớp QuanLy kế thừa từ lớp *NhanVien*
  - Trong phương thức nhap(), gọi phương thức nhập của lớp cha và nhập cho các thuộc tính chuyên môn, phụ cấp chức vụ
  - Phương thức Xuat() cũng gọi phương thức xuất của lớp cha và xuất các thuộc tính chuyên môn, phụ cấp chức vụ
  - Phương thức TinhLuong(): Lương = Lương cơ bản + phụ cấp chức vụ
- Lớp NghienCuu
  - Tương tự lớp QuanLy, nhưng Lương = Lương cơ bản + phụ cấp độc hại
- Lớp PhucVu
  - Lớp PhucVu kế thừa lớp NhanVien, ví không có thêm thuộc tính nào khác so với lớp *NhanVien* nên không cần viết lại phương thức nhap(), Xuat()
  - Chỉ viết lại phương thức TinhLuong(): Lương = Lương cơ bản

3. Cài đặt lớp thử nghiệm: Tạo các đối tượng nhân viên khác nhau, nhập xuất dữ liệu cho các nhân viên này và tính lương cho từng nhân viên.

**Câu 2:** Cho sơ đồ phân cấp như hình sau:




- Xây dựng giao diện *Hinh* và các lớp HinhVuong, HinhChuNhat, HinhTron.
- Xây dựng lớp thử nghiệm chứa phương thức main(), trong đó khai báo một mảng các đối tượng hình, thể hiện tính đa hình bằng cách cho phép lựa chọn nhập thông tin cho mỗi phần tử trong mảng là hình vuông, hình chữ nhật hay hình tròn, tính diện tích của mỗi hình trong mảng.

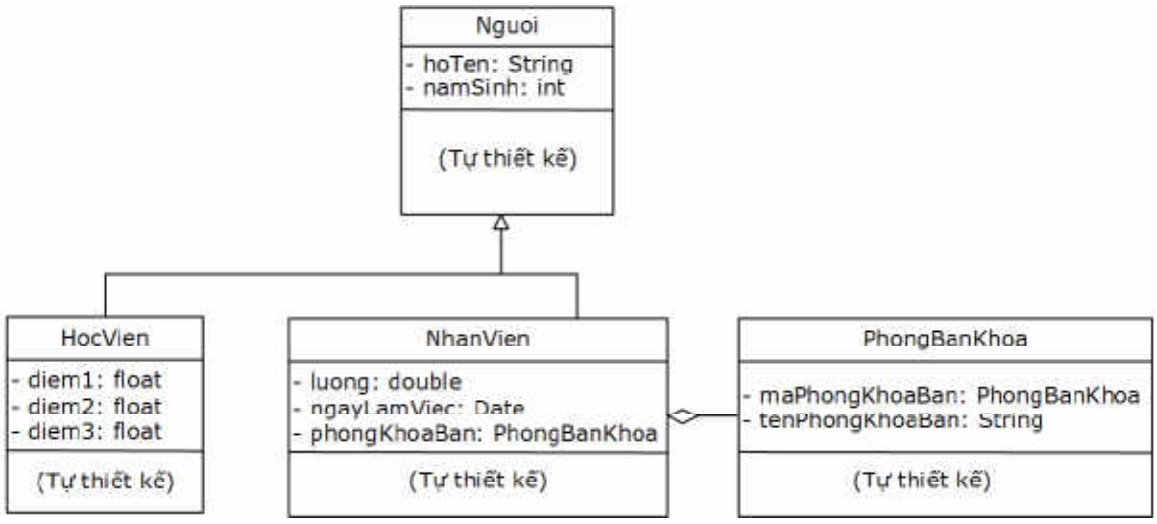
**Hướng dẫn:**

1. Vì mỗi loại hình có thành phần dữ liệu riêng, có diện tích được tính theo công thức khác nhau nên trong sơ đồ trên *Hinh* **là một giao diện** chứa:
   - Thành phần dữ liệu là hằng số pi: **public float pi = 3.14;**
   - Các phương thức chưa được cài đặt chi tiết. *Chi tiết các phương thức sẽ được cài ở từng lớp con cụ thể.*
2. Xây dựng lớp thử nghiệm chứa phương thức main():
   - Khai báo mảng các đối tượng hình.
   - Cho phép người dùng lựa chọn sẽ nhập hình nào (vuông, chữ nhật, tròn).
   - Viết chương trình sử dụng hai lớp trong hai gói này

```
//Nhập số lượng hình: Nhập n
//...
Hinh ds[] = new Hinh[n]; //Khai báo mảng
//Nhập danh sách hình
for (int i = 0 ; i < n ; i++) {
     System.out.println ("Chon loai hinh se nhap: ");
     System.out.println ("1: hinh vuong");
     System.out.println ("2: hinh chu nhat");
     System.out.println ("3: hinh tron");

       int chon = 0;
       //code phần nhập lựa chọn của người dùng
       switch (){
       case 1:
              ds[i] = new HinhVuong();
              ds[i].nhap();
              break;
        case 2:
              ds[i] = new HinhChuNhat();
              ds[i].nhap();
              break;
        case 3:
              ds[i] = new HinhTron();
              ds[i].nhap();
              break;
        default:
              System.out.println ("Ban phai chon 1 trong 3 loai tren");
       }
}
for (int k = 0 ; k < n ; k++) {
         ds[k].xuat();
         System.out.println ("Dien tich: " + ds[k].DienTich());
}
```

**Câu 3:** Viết chương trình mô tả minh họa thiết kế sau:



- Chương trình có giao diện như sau:

```
1.Nhập một học viên
2.Nhập một nhân viên
3.Xuất thông tin một học viên
4.Xuất thông tin một nhân viên
5.Thoát
```

**Hướng dẫn:**

1. Xây dựng lớp ConNguoi (Con Người)
2. Xây dựng lớp HocVien (Học Viên) kế thừa lớp ConNguoi
3. Xây dựng lớp PhongKhoaBan (Phòng Ban Khoa)
4. Xây dựng lớp NhanVien (Nhân Viên) kế thừa lớp ConNguoi, và có dữ liệu phongBanKhoa là một đối tượng của lớp PhongKhoaBan.

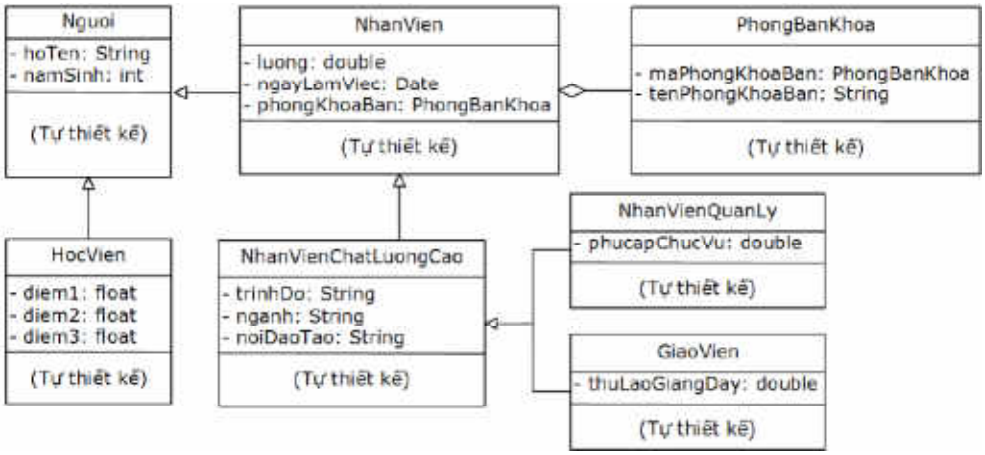
- Trong phương thức nhập dữ liệu cho một đối tượng NhanVien
  - Gọi phương thức nhập của lớp cha để nhập *Họ tên* và *Năm sinh* của nhân viên.
  - Nhập thông tin *Lương* và *Ngày nhận việc* của nhân viên như bình thường
  - Để nhập *phòng ban khoa* của nhân viên, cần khởi tạo đối tượng

    pbk = new PhongKhoaBan();

    pbk.nhap(); //gọi phương thức nhap() của đối tượng pbk
- Tương tự với phương thức xuất dữ liệu cho một đối tượng NhanVien

# THỰC HÀNH NÂNG CAO

**Câu 4:** Viết chương trình minh họa thiết kế sau:



Chương trình có giao diện như sau:

1. Nhập một học viên
2. Nhập một nhân viên quản lý
3. Nhập một giáo viên
4. Xuất thông tin danh sách học viên
5. Xuất thông tin danh sách nhân viên quản lý
6. Xuất thông tin danh sách giáo viên
7. Thoát

# BÀI TẬP VỀ NHÀ

**Câu 5:** Viết chương trình quản lý thông tin Giáo Viên gồm các lớp:

1. Xây dựng lớp MonHoc (Môn Học) gồm các thuộc tính: tên môn học, số tín chỉ. Viết các phương thức cần thiết sau:

- Viết các phương thức get/set cho tên môn học và số tín chỉ
- Viết các phương thức khởi tạo mặc định và phương thức khởi tạo đầy đủ các tham số
- Viết các phương thức nhập – xuất cho lớp MonHoc.

2. Xây dựng lớp Nguoi (Người) gồm họ tên và tuổi. Viết các phương thức cần thiết sau:

- Viết các phương thức get/set cho họ tên và tuổi.
- Viết các phương thức khởi tạo mặc định và phương thức khởi tạo đầy đủ các tham số.
- Viết các phương thức nhập – xuất cho lớp Nguoi.

3. Xây dựng lớp GiaoVien (Giáo Viên) "kế thừa lớp Người" và bao gộp danh sách môn học. Lớp GiaoVien gồm các thuộc tính: danh sách môn học và lương (một giáo viên dạy nhiều môn học). Viết các phương thức cần thiết sau:

- Viết các phương thức khởi tạo mặc định và phương thức khởi tạo đầy đủ các tham số
- Viết các phương thức nhập – xuất cho lớp GiaoVien.

# BÀI 6: XỬ LÝ NGOẠI LỆ

Bài này giúp người học nắm được các nội dung sau:

- *Biết cách xử lý ngoại lệ trong Java;*
- *Hiểu ý nghĩa và sử dụng các khối try, catch và finally;*
- *Biết cách thức lan truyền ngoại lệ;*
- *Hiểu và sử dụng được cơ chế lan truyền ngoại lệ.*

## 6.1 TÓM TẮT LÝ THUYẾT

Ngoại lệ (exception) là một tình trạng bất thường. Trong Java, ngoại lệ là một sự kiện mà phá vỡ luồng chuẩn của chương trình. Nó là một đối tượng mà được ném tại thời gian chạy. Ngoại lệ được triển khai bằng cách sử dụng các lớp như ***Throwable***, **Exception**, **RuntimeException** và các từ khóa như **throw, throws, try, catch và finally**.

### 6.1.1 Khối lệnh try – catch

```
try {
    // code có thể ném ra ngoại lệ
}
catch (ExceptionClassName ex) {
    // code xử lý ngoại lệ
}
```

**Ví dụ 6.1:** Sử dụng khối lệnh try – catch

```
public class Example1 {
    public static void main(String[] args) {
        try {
            int zero = 0;
            int average = 10 / zero;
            System.out.println("Average = " + average);
        }
        catch (ArithmeticException ex) {
```

```
            System.out.println(ex);
        }
        System.out.println("Finished!");
    }
}
```

## 6.1.2 Khối lệnh try-finally

```
try {
    // code có thể ném ra ngoại lệ
}
finally {
    // code trong khối finally sẽ luôn thực thi
}
```

**Ví dụ 6.2:** Sử dụng khối lệnh try-finally

```
public class Example2 {
    public static void main(String[] args) {
        try {
            int res = 10/2;
            System.out.println("10/2 = " + res);
        }
        finally {
            System.out.println("Khối lệnh luôn được thực thi");
        }
        System.out.println("Finished!");
    }
}
```

## 6.1.3 Khối lệnh try – catch – finally

```
try {
    // code có thể ném ra ngoại lệ
}
catch (ExceptionClassName_1 ex) {
    // code xử lý ngoại lệ 1
}
catch (ExceptionClassName_2 ex) {
    // code xử lý ngoại lệ 2
}
// ...
catch (ExceptionClassName_n ex) {
    // code xử lý ngoại lệ n
}
```

```
finally {
    // code trong khối này luôn được thực thi
}
```

**Ví dụ 6.3:** Sử dụng khối lệnh try – catch – finally

```
public class Example3 {
    public static void main(String[] args) {
        try {
            int arr[] = new int[5];
            arr[5] = 4;
            System.out.println("arr[5] = " + arr[5]);
        }
        catch (NullPointerException ex) {
            System.out.println(ex);
        }
        finally {
            System.out.println("Khối lệnh luôn được thực thi");
        }
        System.out.println("Finished!");
    }
}
```

## 6.1.4 Từ khoá throw và throws

- Từ khóa **throw** được sử dụng để ném ra một ngoại lệ cụ thể.

**Ví dụ 6.4:** Sử dụng từ khóa **throw**

```
public class Example4 {
    public static void main(String[] args) {
        ageValid(19);
        System.out.println("------");
        ageValid(17);
    }
    public static void ageValid(int age){
        if(age < 18){
            throw new ArithmeticException("Age not valid!");
        }
        else{
            System.out.println("Welcome!");
        }
    }
}
```

- Từ khóa **throws** được sử dụng để khai báo một ngoại lệ.

**Ví dụ 6.5:** Sử dụng từ khóa **throws**

```
public class Example5 {
    void aMethod() throws IOException {
        throw new IOException("Device error");
    }
    //
    public static void main(String args[]) {
        try {
            TestException obj = new TestException();
            obj.aMethod();
        }
        catch (Exception e) {
            System.out.println("Exception handled!");
        }
        System.out.println("Hello Codelearn!");
    }
}
```

# THỰC HÀNH CƠ BẢN

**Câu 1:** Giả sử cho chương trình sẽ tạo ra 10 số nguyên ngẫu nhiên và lưu vào mảng 10 phần tử. Người dùng nhập vào một chỉ số mảng (vị trí phần tử) để xuất ra nàm hình.

```
// Tạo một mảng 10 các số nguyên ngẫu nhiên
int randomIntNumbers[] = new int[10];
Random rand = new Random();
for(int i = 0; i < 10; i++) {
    randomIntNumbers[i] = rand.nextInt(100);
}
// Người dùng nhập vào vị trí phần tử cần xuất ra màn hình
Scanner input = new Scanner(System.in);
System.out.println("Bạn muốn in ra phần tử mảng thứ mấy? ");
int index = input.nextInt();
System.out.println("Phần tử mảng thứ " + index + " mang giá trị " +
                   randomIntNumbers[index]);
```

Dùng ngoại lệ tương ứng để khi có *Exception* xảy ra thì chương trình có thể thông báo lỗi kịp thời đến với người dùng.

```
// Tạo một mảng 10 các số nguyên ngẫu nhiên
int randomIntNumbers[] = new int[10];
Random rand = new Random();
for(int i = 0; i < 10; i++) {
    randomIntNumbers[i] = rand.nextInt(100);
}

try {
    // // Người dùng nhập vào vị trí phần tử cần xuất ra màn hình
    Scanner input = new Scanner(System.in);
    System.out.println("Bạn muốn in ra phần tử mảng thứ mấy? ");
    int index = input.nextInt();
    System.out.println("Phần tử mảng thứ " + index + " mang giá trị " +
                       randomIntNumbers[index]);
}
catch (InputMismatchException e) {
    System.out.println("Phần tử mảng chưa hợp lệ," +
```

```
                        " vui lòng nhập vào một số nguyên!");
}
catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Phần tử mảng chưa hợp lệ," +
                        " vui lòng nhập vào giá trị từ 0 đến 9!");
}
```

**Câu 2:** Viết chương trình cho phép tính giá trị của biểu thức sau. Yêu cầu xử lý ngoại lệ có thể xảy ra.

$$A = \sqrt{\frac{5x - y}{2x + 7y}}$$

**Câu 3:** Viết chương trình cho phép tạo một mảng 2 chiều cỡ m x n với m, n nhập từ bàn phím. Cài đặt các xử lý ngoại lệ cần thiết.

# THỰC HÀNH NÂNG CAO

**Câu 4:** Xây dựng lớp ngoại lệ DateException cho các lỗi về ngày tháng.

**Câu 5:** Viết chương trình cho phép người dùng nhập vào ngày, tháng năm, nếu thông tin này không hợp lệ sẽ tung ra một ngoại lệ DateException, sau đó thông báo cho người nhập biết và cho phép người dùng nhập lại.

# BÀI 7: LỚP TỔNG QUÁT VÀ COLLECTION

Bài này giúp người học nắm được các nội dung sau:

- *Hiểu được lớp tổng quát trong lập trình Java;*
- *Biết cách sử dụng các kiểu dữ liệu tổng quát;*
- *Hiểu và vận dụng được các Collection cho các ứng dụng trong Java;*

## 7.1 TÓM TẮT LÝ THUYẾT

### 7.1.1 Lớp tổng quát

Một lớp tổng quát trong Java được định nghĩa với một hoặc nhiều tham số kiểu dữ liệu. Tham số kiểu dữ liệu này được sử dụng để chỉ định kiểu dữ liệu cho các thuộc tính và phương thức của lớp đó.

Khai báo, định nghĩa lớp tổng quát:

```
[public] class ClassName<T1, T2, ...> {
  // liệt kê các thuộc tính và phương thức sử dụng kiểu dữ liệu T1, T2,
...
}
```

Tạo đối tượng sử dụng lớp tổng quát:

```
ClassName<Type1, Type2, ...> obj = new ClassName<>([parameterList]);
```

### 7.1.2 Java Collections

Java Collections cung cấp các cấu trúc dữ liệu và cả thuật toán đi kèm để lưu trữ, quản lý, xử lý dữ liệu theo nhiều cách khác nhau.

Có hai loại cơ bản của Java Collections:

- **Collection Interface**: được sử dụng để đại diện cho các nhóm phần tử và cung cấp các phương thức để quản lý.
  - ***List*: ArrayList, LinkedList, Vector**
  - ***Set*: HashSet, LinkedHashSet, TreeSet**
  - ***Queue*: ArrayDueue, PriorityQueue**
- **Map Interface**: hỗ trợ các tính năng như lưu trữ, truy xuất, xóa và tìm kiếm các phần tử theo key:
  - **HashMap**
  - **LinkedHashMap**
  - **TreeMap**

## 7.1.3 Các thuật toán cơ bản trên Java Collections

- Sắp xếp: **`sort()`**
- Tìm kiếm:

```
find(Object o)
findIndexOf(List<? extends Comparable<? super T>> list, T key)
getOrDefault(Object key, V defaultValue)
```

- Xóa: `remove()` và `clear()`
- Thêm: `add(element)`

# THỰC HÀNH CƠ BẢN

**Câu 1:** Ứng dụng lớp Pair<T> để chứa giá trị lớn nhất và nhỏ nhất trong mảng.

Lớp Pair<T> được cho như sau:

```
public class Pair<T> {
    private T first;
    private T second;
    public Pair() {
        first = null;
        second = null;
    }
    public Pair(T first, T second) {
        this.first = first;
        this.second = second;
    }
    public T getFirst() {
        return first;
    }
    public void setFirst(T first) {
        this.first = first;
    }
    public T getSecond() {
        return second;
    }
    public void setSecond(T second) {
        this.second = second;
    }
}
```

**Hướng dẫn:**

- Xây dựng một phương thức *minMaxFromArray()* để lưu hai giá trị min và max này vào trong một lớp *Pair*.
- Phương thức *minMaxFromArray():*

```
public static <T extends Comparable> Pair<T> minMaxFromArray(T[] arr) {
    T min = arr[0];
    T max = arr[0];
    for (int i = 0; i < arr.length; i++) {
        if (min.compareTo(arr[i]) > 0)
            min = arr[i];
        if (max.compareTo(arr[i]) < 0)
            max = arr[i];
    }
    return new Pair<>(min, max);
}
```

- Phương thức **main()** của ứng dụng:

```
public static void main(String[] args) {
    String[] arrStr = new String[] { "Một", "Hai", "Ba",
                                "Bốn", "Năm", "Sáu", "Bảy" };
    Pair<String> pairStr = minMaxFromArray(arrStr);
    System.out.println("Min của màng String là " + pairStr.getFirst());
       System.out.println("Max của màng String là " + pairStr.getSecond());
    Integer[] arrInt = new Integer[] { 5, -19, 40, 33, 25, -7 };
    Pair<Integer> pairInt = minMaxFromArray(arrInt);
    System.out.println("Min của màng Integer là " + pairStr.getFirst());
       System.out.println("Max của màng Integer là " + pairStr.getSecond());
}
```

**Câu 2:** Xây dựng ứng dụng từ điển.

- Từ vựng (Word): từ vựng và nghĩa của nó.

- Từ điển: tổ chức lưu từ vựng và phương thức tra từ.

**Hướng dẫn:**

- Xây dựng lớp từ vựng (Word):

```
public class Word<K, V> {
    private K key;
    private V value;
    public Word(K key, V value) {
        this.key = key;
        this.value = value;
    }
    public K getKey() {
        return key;
    }
    public void setKey(K key) {
        this.key = key;
    }
    public V getValue() {
        return value;
    }
    public void setValue(V value) {
        this.value = value;
    }
    // Phương thức giúp so sánh key này với key nào đó khác
    public boolean isKeyEquals(K anotherKey) {
        return (this.key.equals(anotherKey));
    }
}
```

- Xây dựng lớp ***Dictionary*** để tổ chức lưu trữ các *từ vựng* và giải thuật tra từ:

```
public class Dictionary<K, V> {
    private Word<K, V>[] words;
    public Dictionary(Word<K, V>[] words) {
        this.words = words;
    }
    public V findWord(K keySearch) {
        for (Word<K, V> word : words) {
            if (word.isKeyEquals(keySearch)) {
                return word.getValue();
            }
        }
        return null;
    }
}
```

# THỰC HÀNH NÂNG CAO

**Câu 3:** Một đơn vị sản xuất gồm có các cán bộ là công nhân, kỹ sư, nhân viên. Mỗi cán bộ cần quản lý các dữ liệu: Họ tên, tuổi, giới tính(name, nữ, khác), địa chỉ.

- Cấp công nhân sẽ có thêm các thuộc tính riêng: Bậc (1 đến 10).
- Cấp kỹ sư có thuộc tính riêng: Nghành đào tạo.
- Các nhân viên có thuộc tính riêng: Công việc.

Yêu cầu 1: Xây dựng các lớp CongNhan, KySu, NhanVien kế thừa từ lớp CanBo.

Yêu cầu 2: Xây dựng lớp QLCB (quản lý cán bộ) cài đặt các phương thức thực hiện các chức năng sau:

- Thêm mới cán bộ.
- Tìm kiếm theo họ tên.
- Hiện thị thông tin về danh sách các cán bộ.
- Thoát khỏi chương trình.

**Câu 4:** Một thư viện cần quản lý các tài liệu bao gồm Sách, Tạp chí, Báo. Mỗi tài liệu gồm có các thuộc tính sau: Mã tài liệu(Mã tài liệu là duy nhất), Tên nhà xuất bản, số bản phát hành.

- Các loại sách cần quản lý thêm các thuộc tính: tên tác giả, số trang.
- Các tạp chí cần quản lý thêm: Số phát hành, tháng phát hành.
- Các báo cần quản lý thêm: Ngày phát hành.

Yêu cầu 1: Xây dựng các lớp để quản lý tài liệu cho thư viện một cách hiệu quả.

Yêu cầu 2: Xây dựng lớp QuanLySach có các chức năng sau:

- Thêm mới tài liêu: Sách, tạp chí, báo.
- Xoá tài liệu theo mã tài liệu.
- Hiện thị thông tin về tài liệu.
- Tìm kiếm tài liệu theo loại: Sách, tạp chí, báo.
- Thoát khỏi chương trình.

# BÀI TẬP VỀ NHÀ

**Câu 5:** Các thí sinh dự thi đại học bao gồm các thí sinh thi khối A, B, và khối C. Các thí sinh cần quản lý các thông tin sau: Số báo danh, họ tên, địa chỉ, mức ưu tiên.

- Thí sinh thi khối A thi các môn: Toán, Lý, Hoá.
- Thí sinh thi khối B thi các môn: Toán, Hoá, Sinh.
- Thí sinh thi khối C thi các môn: Văn, Sử, Địa.
  - Yêu cầu 1: Xây dựng các lớp để quản lý các thi sinh dự thi đại học.
  - Yêu cầu 2: Xây dựng lớp TuyenSinh có các chức năng:
- Thêm mới thí sinh.
- Hiện thị thông tin của thí sinh và khối thi của thí sinh.
- Tìm kiếm theo số báo danh.
- Thoát khỏi chương trình.

**Câu 6:** Để quản lý các hộ dân cư trong một khu phố, người ta cần các thông tin sau: Số thành viên trong gia đình, Số nhà, thông tin mỗi cá nhân trong gia đình. Với mỗi

cá nhân, người ta quản lý các thông tin sau: Họ tên, Tuổi, Nghề nghiệp, Số căn cước công dân (duy nhất cho mỗi người).

- Yêu cầu 1: Hãy xây dựng lớp Nguoi để quản lý thông tin của mỗi cá nhân.
- Yêu cầu 2: Xây dựng lớp HoGiaDinh để quản lý thông tin của từng hộ gia đình.
- Yêu cầu 2: Xây dựng lớp KhuPho để quản lý các thông tin của từng hộ gia đình.
- Yêu cầu 3: Nhập n hộ dân (n nhập từ bàn phím), hiển thị thông tin của các hộ trong khu phố.

# TÀI LIỆU THAM KHẢO

[1]Khoa Công nghệ Thông tin (2024). Tài liệu học tập học phần "Lập trình hướng đối tượng". Trường đại học Công nghệ Tp.HCM.

[2]Khoa Công nghệ Thông tin (2021). Tài liệu học tập học phần "Thực hành lập trình hướng đối tượng". Trường đại học Công nghệ Tp.HCM.

[3]Y. Daniel Liang (2018). Introduction to Java Programming, Brief Version, Global. Pearson.

[4]Cay S. Horstmann (2022). Core Java, Volume I: Fundamentals (Twelfth Edition). Addison-Wesley.

[5]Harvey, M.D & Paul, J.D. (2018), Java: How to Program (11th ed.), Prentice Hall.