

LUYỆN TẬP



Một số câu hỏi lý thuyết

- **Câu 1:** Hãy nêu điểm khác nhau giữa lớp trừu tượng và lớp trong java?
- **Câu 2:** Nếu lớp cha chỉ có phương thức khởi dựng mặc định thì lớp con không cần dùng từ khóa super trong phương thức khởi dựng của nó. Nhưng tại sao nếu lớp cha chỉ có một phương thức khởi dựng có tham số (và không có phương thức khởi tạo mặc định) thì trong phương thức khởi dựng của lớp con, ta vẫn phải dùng từ khóa super?
- **Câu 3:** Đặc tính sử dụng lại mã được thể hiện ở các nguyên lý nào của lập trình hướng đối tượng. Hãy phân tích sự khác nhau trong việc sử dụng lại các nguyên lý này và cho ví dụ minh họa.
- **Câu 4:** So sánh sự khác nhau giữa phương thức khởi tạo và phương thức thông thường. Tại sao khi xây dựng lớp trong java lại nên xây dựng phương thức khởi tạo mặc định?
- **Câu 5:** So sánh điểm khác biệt giữa this() và super(). Cho ví dụ cụ thể?

Một số câu hỏi lý thuyết

- **Câu 6:** Trừu tượng hóa là gì? So sánh đối tượng và lớp?
- **Câu 7:** So sánh chồng phương thức và ghi đè lại phương thức.
- **Câu 8:** Phương thức khởi tạo là gì? Chúng có những tác dụng gì? Lấy ví dụ về chồng phương thức khởi tạo?
- **Câu 9:** Cần chú ý gì khi thực hiện Upcasting và Downcasting các đối tượng của lớp trong lập trình hướng đối tượng?
- **Câu 10:** Hãy nêu sự giống nhau và khác nhau giữa giao diện và lớp trừu tượng trong java.
- **Câu 11:** Hãy nêu điều kiện cần và đủ để lập trình viên có thể tiến hành việc: Khai báo đối tượng là đối tượng của lớp A nhưng lại khởi tạo đối tượng của lớp B như sau: `A obj = new B();`

Câu hỏi lý thuyết

- Câu 1: Hãy nêu điểm khác nhau giữa lớp trừu tượng và lớp trong java?
- Lớp và lớp trừu tượng đều thể hiện các khái niệm trừu tượng hoá, đóng gói các thông tin về thuộc tính và phương thức

Lớp (class)	Lớp trừu tượng (abstract class)
- Tạo các đối tượng từ lớp thông qua khởi tạo	- Không tạo các thể hiện của lớp do lớp chưa hoàn thiện. Khai báo lớp với từ khoá abstract
- Tất cả các phương thức đều có phần cài đặt đầy đủ	- Chứa các phương thức trừu tượng, chỉ có chữ kí chứ không có phần thân
- Lớp con khi kế thừa có thể sử dụng các dữ liệu của lớp cha theo nguyên lý kế thừa	- Khi một lớp con kế thừa từ 1 lớp trừu tượng có 2 trường hợp xảy ra: + Lớp con sẽ là lớp thông thường nếu như các phương thức trừu tượng chưa hoàn chỉnh trong lớp cha được ghi đè và hoàn thiện. + Lớp con vẫn sẽ là lớp trừu tượng nếu nó chưa hoàn thiện các phương thức trừu tượng của lớp cha.

Câu hỏi lý thuyết

- Câu 2: Nếu lớp cha chỉ có phương thức khởi dựng mặc định thì lớp con không cần dùng từ khóa `super` trong phương thức khởi dựng của nó. Nhưng tại sao nếu lớp cha chỉ có một phương thức khởi dựng có tham số (và không có phương thức khởi tạo mặc định) thì trong phương thức khởi dựng của lớp con, ta vẫn phải dùng từ khóa `super`?
- Trong quá trình khởi tạo đối tượng thì **phần dữ liệu của lớp cha sẽ được khởi tạo trước các phần dữ liệu của lớp con** → thể hiện qua: câu lệnh đầu tiên trong constructor của lớp con sẽ là lời gọi đến constructor của lớp cha với cú pháp: **`super()`** hoặc **`super(danh sách tham số)`**
- Lời gọi này có thể được cung cấp theo 2 cách:
 - Khi lớp cha có constructor mặc định (default constructor – phương thức khởi tạo không tham số) → nếu không có lời gọi `super` tường minh → Java sẽ TỰ ĐỘNG bổ sung lời gọi `super()`.
 - Khi lớp cha không xây dựng constructor mặc định mà chỉ có 1 constructor chứa tham số → Java sẽ ko tự động bổ sung lời gọi trong lớp con mà lập trình viên sẽ phải tự viết gọi nó một cách tường minh.

Câu hỏi lý thuyết

- Câu 3: Đặc tính sử dụng lại mã được thể hiện ở các nguyên lí nào của lập trình hướng đối tượng. Hãy phân tích sự khác nhau trong việc sử dụng lại các nguyên lí này và cho ví dụ minh họa.
- Đặc tính tái sử dụng mã nguồn được thể hiện ở 2 nguyên lí của lập trình hướng đối tượng là: Kết tập và Kế thừa.

Kết tập	Kế thừa
<p>Kết tập tái sử dụng thông qua đối tượng.</p> <p>Tạo ra lớp mới là tập hợp các đối tượng của các lớp đã có</p> <p>Lớp toàn thể có thể sử dụng dữ liệu và hành vi thông qua các đối tượng thành phần</p>	<p>Kế thừa tái sử dụng thông qua lớp</p> <p>Tạo lớp mới bằng cách phát triển lớp đã có</p> <p>Lớp con kế thừa dữ liệu và hành vi của lớp cha</p>
Quan hệ "là một phần" ("is a part of")	Quan hệ "là một loại" ("is a kind of")
Không sử dụng từ khoá	Sử dụng từ khoá extends
<p>Biểu diễn mối quan hệ giữa 2 lớp trong UML: hình thoi tại phía lớp tổng thể, bội số quan hệ cung cấp thông tin về số lượng các đối tượng của 2 lớp tham gia trong quan hệ</p>	<p>Biểu diễn mối quan hệ giữa 2 lớp trong UML: mũi tên tam giác rỗng với đầu mũi tên đặt ở phía lớp cha</p>
Ví dụ: Bánh xe là một phần của Ô tô Viết code minh họa	Ví dụ: Ô tô là một loại phương tiện vận tải Viết code minh họa

Câu hỏi lý thuyết

- Câu 4: So sánh sự khác nhau giữa phương thức khởi tạo và phương thức thông thường. Tại sao khi xây dựng lớp trong java lại nên xây dựng phương thức khởi tạo mặc định?

Phương thức (method)	Phương thức khởi tạo (constructor)
Phương thức: là thành viên của lớp, mô hình hoá các hoạt động của các đối tượng (đáp ứng cho các thông điệp gửi đến đối tượng)	Constructor: Không được xem là thành viên của lớp, nhiệm vụ khởi tạo giá trị ban đầu cho các thuộc tính khi tạo mới đối tượng
Sử dụng: gọi phương thức thông qua đối tượng và toán tử "." : Tên đối tượng.phương thức(danh sách tham số)	Sử dụng: kết hợp với toán tử new để khởi tạo đối tượng: new Constructor(danh sách tham số)
Cú pháp: có thể đặt tên bất kỳ, khai báo phương thức gồm đầy đủ các thông tin: kiểu trả về, tên phương thức, danh sách tham số và kiểu dữ liệu của tham số	Cú pháp: tên của constructor trùng với tên của lớp, không có kiểu trả về (trả về tham chiếu đến đối tượng được tạo mới) Java: constructor không thể dùng các từ khóa abstract, static, final, native, synchronized.

- Khi xây dựng các lớp trong Java thì chúng ta nên xây dựng phương thức khởi tạo mặc định vì nó là phương thức khởi tạo không tham số, thiết lập các giá trị mặc định cho các thuộc tính của đối tượng → cung cấp cho người lập trình một cách khởi tạo đơn giản, dễ dàng cho các đối tượng

Câu hỏi lý thuyết

- Câu 5: So sánh điểm khác biệt giữa `this()` và `super()`. Cho ví dụ cụ thể?

- **`this()`**: Gọi đến các phương thức khởi tạo khác của lớp.

Ví dụ:

```
public class Ship {
    private double x = 0.0, y = 0.0;
    public String name;

    public Ship (String name) {
        this.name = name;
    }

    public Ship (String name, double x, double y){
        this (name); this.x = x; this.y = y;
    }
}
```

- **`super()`**: Gọi đến các phương thức khởi tạo của lớp cha (được sử dụng trong các lớp con)

Ví dụ:

```
public class Vehicle {
    protect String name;
    protect int count;

    public Vehicle (String name, int count) {
        this.name = name; this.count = count;
    }
}

public class Moto extends Vehicle {
    private String type;

    public Moto (String name, int count, String type){
        super(name, count); this.type = type;
    }
}
```


Câu hỏi lý thuyết

- Câu 6: Trừu tượng hóa là gì? So sánh đối tượng và lớp?
- Trừu tượng hóa là 1 trong 4 nguyên lí của lập trình hướng đối tượng, là quá trình loại bỏ các thông tin ít quan trọng, chỉ giữ lại những thông tin có nhiều ý nghĩa, quan trọng trong hoàn cảnh cụ thể.

Lớp	Đối tượng
Lớp là mô hình hoá, mô tả các khái niệm.	Đối tượng là các sự vật thật, là thực thể thực sự.
Lớp như 1 bản mẫu, định nghĩa các phương thức và thuộc tính của các đối tượng	Mỗi đối tượng được tạo ra từ một lớp có các dữ liệu cụ thể (thuộc tính) và hành vi (phương thức) của nó.
Một lớp là một sự trừu tượng hóa của các đối tượng	Một đối tượng là một thể hiện (instance) của một lớp

Câu hỏi lý thuyết

- Câu 7: So sánh chồng phương thức và ghi đè lại phương thức.
- Nạp chồng và ghi đè phương thức là các kỹ thuật thể hiện cho nguyên lý đa hình trong lập trình hướng đối tượng

Chồng phương thức (Overloading)	Ghi đè phương thức (Override)
Phạm vi: các phương thức nạp chồng trong cùng lớp	Phạm vi: ghi đè phương thức trong mối quan hệ kế thừa giữa lớp cha và lớp con. Phương thức trong lớp con ghi đè / thay thế / định nghĩa lại phương thức của lớp cha
Cú pháp: các phương thức nạp chồng trùng tên và khác chữ kí (số lượng hay kiểu dữ liệu của đối số)	Cú pháp: phương thức ghi đè hoàn toàn giống về giao diện (chữ kí), giới hạn truy nhập không chặt hơn phương thức của lớp cha, không ném ra các ngoại lệ mới so với phương thức của lớp cha
Các phương thức có thể khác về kiểu dữ liệu trả về	Các phương thức ghi đè phải có cùng kiểu dữ liệu trả về
	Không được ghi đè các phương thức static, private và final trong lớp cha

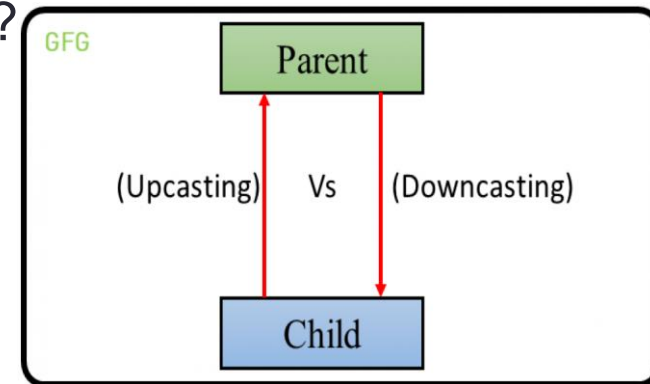
Câu hỏi lý thuyết

- Câu 8: Phương thức khởi tạo là gì? Chúng có những tác dụng gì? Lấy ví dụ về chồng phương thức khởi tạo?
- Phương thức khởi tạo là phương thức có nhiệm vụ khởi tạo giá trị ban đầu cho các thuộc tính khi tạo mới đối tượng. Tên của constructor trùng với tên của lớp, không có kiểu trả về (trả về tham chiếu đến đối tượng được tạo mới). Sử dụng: kết hợp với toán tử new để khởi tạo đối tượng.
- Ví dụ về chồng phương thức khởi tạo: cung cấp các cách khởi tạo khác nhau cho các đối tượng

```
public class Student {  
    private String name;  
    private int tuoi;  
    public Student() { name = "An"; tuoi = 18; }  
    public Student(String name, int tuoi) { this.name = name; this.tuoi = tuoi; }  
}
```

Câu hỏi lý thuyết

- Câu 9: Cần chú ý gì khi thực hiện Upcasting và Downcasting các đối tượng của lớp trong lập trình hướng đối tượng?
- Chuyển đổi kiểu của các đối tượng thuộc các lớp có mối quan hệ kế thừa.
- Thể hiện tính đa hình trong lập trình hướng đối tượng.



Upcasting	Downcasting
Đối tượng của lớp cha được gán tham chiếu tới đối tượng của lớp con	Chuyển kiểu đối tượng là một thể hiện của lớp cha xuống thành đối tượng là thể hiện của lớp con
Một biến kiểu dữ liệu superclass có thể tham chiếu đến tất cả các đối tượng subclass → sử dụng như một kiểu tổng quát	Sử dụng downcasting khi cần sử dụng các tính chất riêng của subclass mà ở superclass không có
Tự động chuyển đổi kiểu	Phải ép kiểu
Có thể xảy ra lỗi Compile error (lỗi biên dịch)	Có thể xảy ra lỗi Runtime error (lỗi khi chạy chương trình – ngoại lệ ClassCastException) Kiểm tra kiểu trước khi chuyển đổi: toán tử instanceof

Câu hỏi lý thuyết

- Câu 10: Hãy nêu sự giống nhau và khác nhau giữa giao diện và lớp trừu tượng trong java.
- Giao diện và lớp trừu tượng: đều chứa các phương thức chưa hoàn thiện (chỉ có phần khai báo, không có phần thân phương thức)
- **Abstract class (Lớp trừu tượng):** có thể hiểu là một class cha cho tất cả các Class có cùng bản chất.
 - Mỗi lớp con chỉ có thể kế thừa từ một lớp trừu tượng (đơn kế thừa).
 - Abstract class không cho phép tạo các thể hiện (instance), vì vậy không thể khởi tạo một đối tượng trực tiếp từ một Abstract class.
- **Interface (Giao diện):** có thể được xem như một mặt nạ cho tất cả các lớp cùng cách thức hoạt động nhưng có thể khác nhau về bản chất.
 - Lớp con có thể thực thi nhiều Interface để bổ sung đầy đủ cách thức hoạt động của mình (đa kế thừa).

Câu hỏi lý thuyết

- Câu 10: Hãy nêu sự giống nhau và khác nhau giữa giao diện và lớp trừu tượng trong java.

Lớp trừu tượng	Giao diện
Abstract Class là "khuôn mẫu" cho các lớp con kế thừa (quan hệ "là một loại" → cùng bản chất)	Interface chứa các "khuôn mẫu" cho phương thức → các lớp con khi thực thi giao diện phải cài đặt cụ thể (các lớp này có thể khác bản chất)
Có thể chứa các phương thức instance (các phương thức đã được triển khai với phần thân cụ thể)	Chỉ có thể chứa danh sách chữ ký phương thức
Phương thức có thể sử dụng những access modifiers như private, protected, default, public, static	Phương thức của interface không cần khai báo access modifiers vì chúng chỉ sử dụng public, abstract và đã được mặc định
Có thể chứa các thuộc tính final và non-final	Chỉ có thể chứa các thuộc tính hằng
Lớp con phải định nghĩa lại (override) các phương thức trừu tượng	Các lớp thực thi Interface sẽ phải định nghĩa lại toàn bộ các phương thức có trong Interface
Một lớp con chỉ được thừa kế 1 lớp cha là lớp trừu tượng (Dùng từ khóa extends)	Một lớp có thể thực thi nhiều Interface cùng một lúc (Dùng từ khóa implements)

Câu hỏi lý thuyết

- Câu 11: Hãy nêu điều kiện cần và đủ để lập trình viên có thể tiến hành việc: Khai báo đối tượng là đối tượng của lớp A nhưng lại khởi tạo đối tượng của lớp B như sau: `A obj = new B();`
- Lớp B là con của lớp A
- Lớp B có phương thức khởi tạo mặc định

Một số câu hỏi trắc nghiệm



Câu 1

- Lựa chọn nào sau đây là tên biến không hợp lệ trong Java?
 1. \$char
 2. 1MyNumber ✓
 3. case ✓
 4. _int



Câu 2

- Câu lệnh nào dưới đây không sinh ra lỗi biên dịch?
 1. `char my_char = 'c';` ✓
 2. `char your_char = 'int';`
 3. `char what = 'Hello';`
 4. `char what_char = "L";`



Câu 3

- Cho đoạn mã nguồn dưới đây:

```
public class Unary{  
    public static void main(String[] args) {  
        int x = 7;  
        int y = 6*x++;  
        System.out.println (" y= " + y);  
        int a = 7;  
        int b = 6*++a;  
        System.out.println (" b= " + b);  
    }  
}
```

- Kết quả nhận được khi thực thi đoạn mã này?

1. y= 48 b= 48
2. y= 42 b= 48
3. y= 48 b= 42
4. y= 42 b= 42





Câu 4

- Cho đoạn mã nguồn dưới đây:

```
byte a = 7;
```

```
short b = 3;
```

```
c = a * ++b;
```

- Giả sử đoạn mã hoạt động chính xác, hãy cho biết kiểu dữ liệu có thể của biến c?
 1. short, int, long, float, double
 2. short, char, int, float, double
 3. byte, short, int, long, float, double
 4. int, long, float, double ✓



Câu 5

- Cho đoạn mã nguồn dưới đây:

```
int x = 9;
```

```
int y = -2;
```

```
System.out.println("output: " + x%y);
```

- Kết quả nhận được khi thực thi đoạn mã này?

1. output: -1

2. output: 1 ✓

3. output: 4.5

4. output: 4



Câu 6

- Cho biết kết quả thực hiện của đoạn mã sau?

```
public class LastLaugh {  
    . . . public static void main(String args[]) {  
    . . .     System.out.print("H" + "a");  
    . . .     System.out.print('H' + 'a');  
    . . . }  
}
```

Kết quả: hiển thị trên màn hình
Ha169



Câu 7

- Cho biết kết quả thực hiện của đoạn mã sau?

```
public class MyClass {  
    . . . public static void main(String[] args) {  
    . . .     System.out.println(1 + 2 + "3");  
    . . .     System.out.println("1" + 2 + 3);  
    . . . }  
}
```

Kết quả: hiển thị trên màn hình

33

123

Câu 8

- Viết chương trình tính **HarmonicSum** $Harmonic(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$
- Đoạn chương trình chưa hoàn thiện, hãy bổ sung những chỗ còn trống:

```
public 1 HarmonicSum {  
    public static 2 main (String[] args) {  
        int maxDenominator = 50000;  
        3 sum = 0.0;  
        for (int denominator = 1; denominator <= maxDenominator; ++denominator) {  
            4  
        }  
        5.out.println("The sum is: " + sum);  
    }  
}
```

1. class
2. void
3. double
4. sum += (double) 1/denominator;
5. System



Câu 9

- Trong đoạn mã này giá trị của X là bao nhiêu để in ra tất cả các phần tử trong mảng?

```
int values[] = {1,2,3,4,5,6,7,8};  
for(int i=0; i< X; ++i)  
System.out.println(values[i]);
```

Giá trị của X là:
8



Câu 10

- Cho biết kết quả khi thực thi đoạn chương trình sau:

```
int x = 3;  
int y = 8;  
System.out.println(y / x);  
System.out.println((double)y / x);  
System.out.println((double)(y / x));  
System.out.println(y / (double)x);
```

Kết quả: hiển thị trên màn hình

2

2.6666666666666665

2.0

2.6666666666666665



Câu 11

- Chú trọng đến các tính chất quan trọng trong khi bỏ qua các chi tiết ít quan trọng được gọi là:
 1. Trừu tượng hóa ✓
 2. Đa hình
 3. Đóng gói
 4. Che giấu thông tin



Câu 12

- "Che đi các chi tiết cài đặt và chỉ cho thấy giao diện của mô-đun" là đặc điểm của khái niệm nào?
 1. Trừu tượng hóa
 2. Đa hình
 3. Đóng gói ✓
 4. Tái sử dụng



Câu 13

■ Phương án nào là khai báo phương thức hợp lệ?

1. `void method1 { /* ... */ }`
2. `void method2() { /* ... */ }` ✓
3. `void method3(void) { /* ... */ }`
4. `method4() { /* ... */ }`
5. `method5(void) { /* ... */ }`



Câu 14

- Cho định nghĩa lớp như sau, thuộc tính nào có thể truy nhập bên ngoài gói my.project:

```
package my.project;  
public class MyClass {  
    int i;  
    public int j;  
    private int l;  
}
```

Biến j



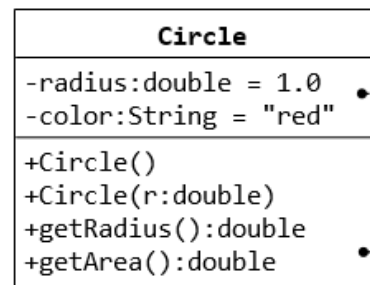
Câu 15

- Cho biết đoạn mã sau biên dịch có thành công không?

```
import java.util.*;  
package com.cnpm.hust;  
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Hello world");  
    }  
}
```

Đoạn mã có lỗi biên dịch
Câu lệnh khai báo package phải để trên đầu tiên trong file mã nguồn

Câu 16



\pm varName:type = default-value

- denotes private access
+ denotes public access

\pm methodName(parmName:type,...):retunType

- Đoạn mã xây dựng lớp Circle theo sơ đồ lớp UML tương ứng, tuy nhiên còn một số chỗ trống chưa hoàn thành, hãy điền vào chỗ trống các nội dung thích hợp?

```
public A Circle {  
    B double radius;  
    private C color;  
    D Circle() {  
        radius = 1.0;  
        color = "red";  
    }  
    public E (double r) {  
        radius = r;  
        color = "red";  
    }  
}
```

```
public double F {  
    return radius;  
}  
public double getArea() {  
    return G;  
}  
}
```

A. class
B. private
C. String
D. public

E. Circle
F. getRadius()
G. radius*radius*3.14

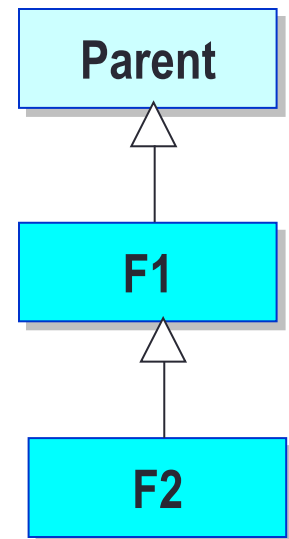
Một số bài tập

Bài 1. Constructor chaining

```
public class Parent
{
    public Parent()
    {
        System.out.println("Invoke parent default constructor");
    }
}
```

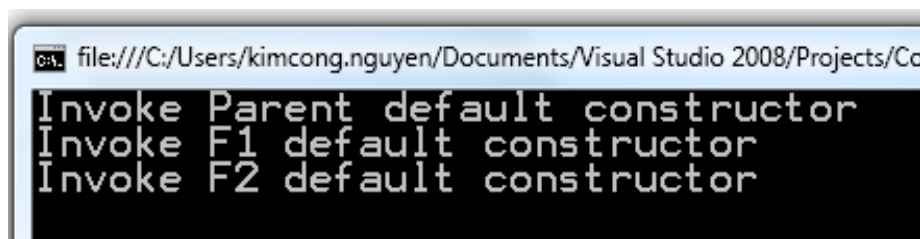
```
public class F1 extends Parent
{
    public F1()
    {
        System.out.println("Invoke F1 default constructor");
    }
}
```

```
public class F2 extends F1
{
    public F2()
    {
        System.out.println("Invoke F2 default constructor");
    }
}
```



Bài 1. Constructor chaining (2)

```
public static void main(String [] args)
{
    new F2 () ;
}
```



file:///C:/Users/kimcong.nguyen/Documents/Visual Studio 2008/Projects/Cc

```
Invoke Parent default constructor
Invoke F1 default constructor
Invoke F2 default constructor
```

1. Object

2. Parent() call **super()**

3. F1() call **super ()**

4. F2() call **super()**

5. Main() call **new F2()**

Bài 2.

- Đoạn mã dưới đây có lỗi gì không?

```
public class A {  
    public A() {  
        super();  
        this(10);  
    }  
  
    public A(int i) {  
        System.out.println(i);  
    }  
}
```

Bài 3.

- Kết quả khi thực thi chương trình dưới đây?

```
class M {  
    int i;  
    public M(int i) {  
        this.i = i--;  
    }  
}  
  
class N extends M {  
    public N(int i) {  
        super(++i);  
        System.out.println(i);  
    }  
}  
  
public class MainClass {  
    public static void main(String[] args) {  
        N n = new N(26);  
    }  
}
```

Bài 4.

- Cho biết kết quả khi thực hiện đoạn mã sau:

```
public class MyClass {  
    public static void main(String[] args) {  
        B b = new B("Test");  
    }  
}  
  
class A {  
    A() { this("1", "2"); }  
    A(String s, String t) { this(s + t); }  
    A(String s) { System.out.println(s); }  
}  
  
class B extends A {  
    B(String s) { System.out.println(s); }  
    B(String s, String t) { this(t + s + "3"); }  
    B() { super("4"); };  
}
```

Bài 5.

- Cho biết kết quả biên dịch và thực hiện đoạn mã sau:

```
class Base extends Exception {}
class Derived extends Base {}

public class Main {
    public static void main(String args[]) {

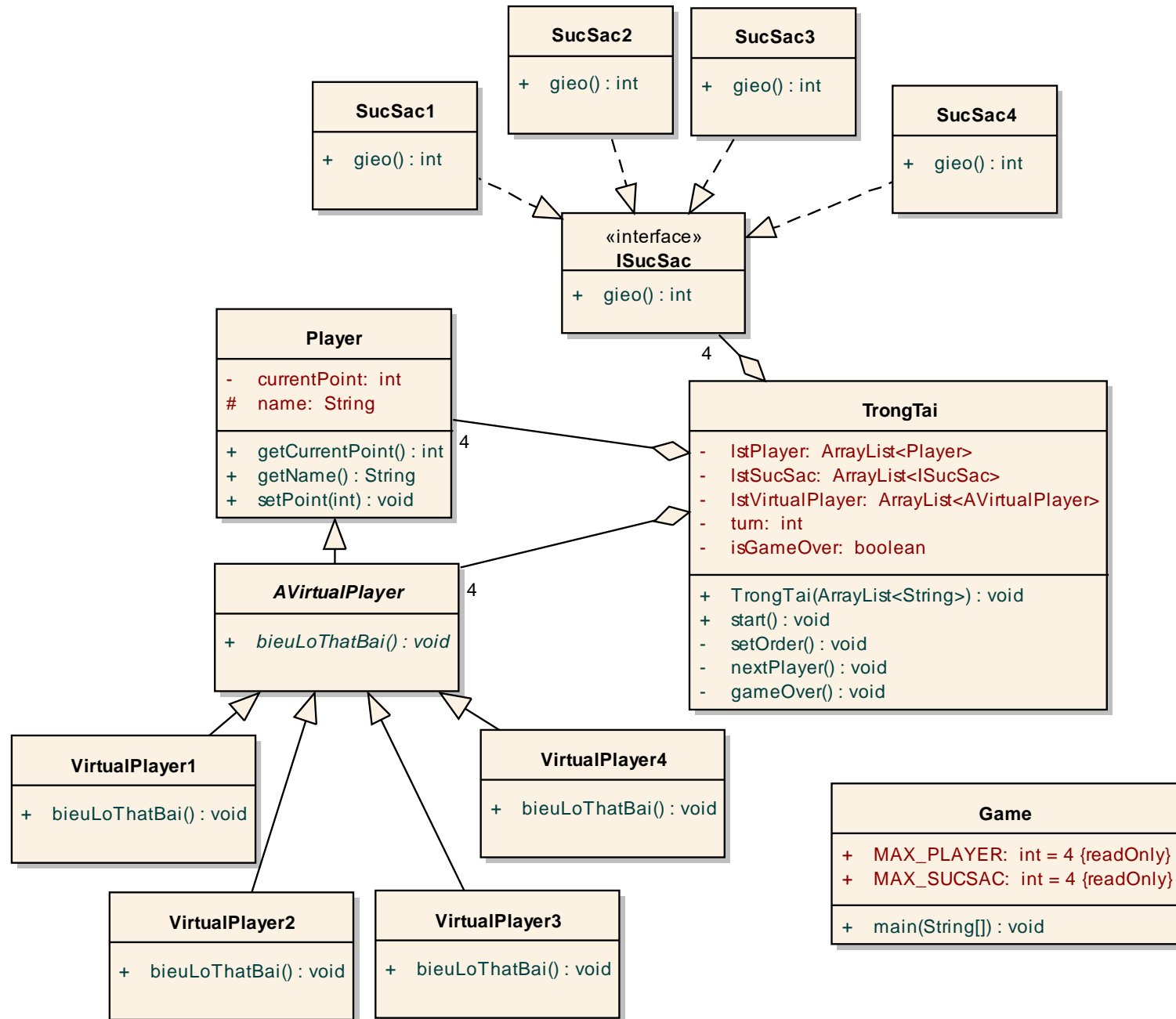
        try {

            throw new Derived();
        }
        catch(Base b)      {
            System.out.println("Caught base class exception");
        }
        catch(Derived d)   {
            System.out.println("Caught derived class exception");
        }
    }
}
```

Bài 6.

Cần lập trình 1 trò chơi như sau (trên 1 máy tính duy nhất):

- Số lượng người chơi: **luôn là 4**. Nếu có < 4 người chơi thật tham gia, máy tính sẽ tự động thêm 1 số người chơi ảo cho đủ 4 người.
- Có **4 quân súc sắc**. Mỗi quân súc sắc, xác suất cho 1 mặt là 20%, các mặt còn lại là 16% (lần lượt 4 quân cho các mặt 1, 2, 3, 4 chấm)
- Mỗi người chơi ít nhất có các thuộc tính và hành vi cơ bản sau:
 - **Tên**
 - **Số điểm** đang có
 - Riêng người chơi ảo có thêm cách **thức biểu lộ thất bại khác nhau**. (Lưu ý có tối đa 4 người chơi ảo)
- **Có 1 trọng tài**, điều khiển cuộc chơi. Trọng tài có nhiệm vụ:
 - **Chỉ định người chơi tiếp theo**.
 - **Tính điểm** cho người chơi. Nếu tổng điểm cũ và điểm vừa gieo của 1 người chơi là 21 \rightarrow thắng cuộc, kết thúc cuộc chơi. Nếu điểm cũ + điểm vừa gieo lớn hơn 21 \rightarrow tính điểm của người chơi là 0. Trường hợp còn lại, cộng điểm bình thường.
 - Tuyên bố người thắng cuộc. Những người chơi ảo thua cuộc sẽ lần lượt thực hiện cách thức biểu lộ thất bại của mình.
- Mỗi người chơi, khi đến lượt, sẽ nhận ngẫu nhiên 1 quân súc sắc và gieo



main() lớp Game

```
ArrayList<String> lst =  
    new ArrayList<String>() ;  
lst.add("Dat") ;  
TrongTai t = new TrongTai(lst) ;  
t.start() ;
```

Phương thức khởi dựng

```
public TrongTai(ArrayList<String> lstName){
    //Khoi tao cac suc sac
    lstSucSac = new ArrayList<ISucSac>();
    lstSucSac.add(new SucSac1());lstSucSac.add(new SucSac2());
    lstSucSac.add(new SucSac3());lstSucSac.add(new SucSac4());

    //Khoi tao cac player ao
    lstVirtualPlayer = new ArrayList<AVirtualPlayer>();
    lstVirtualPlayer.add(new VirtualPlayer1());
    lstVirtualPlayer.add(new VirtualPlayer2());
    lstVirtualPlayer.add(new VirtualPlayer3());
    lstVirtualPlayer.add(new VirtualPlayer4());

    //Khoi tao cac player that
    lstPlayer = new ArrayList<Player>();
    for (String name:lstName){
        lstPlayer.add(new Player(name));
    }

    //Them vao cac player ao cho du nguoi choi
    if (lstPlayer.size()<Game.MAX_PLAYER){
        int boSung = Game.MAX - lstPlayer.size();
        for (int i =0; i<boSung; ++i)
            lstPlayer.add(lstVirtualPlayer.get(i));
    }
}
```

```
public TrongTai(ArrayList<String> lstName) {  
    //Khoi tao cac suc sac  
    lstSucSac = new ArrayList<ISucSac>();  
    lstSucSac.add(new SucSac1());lstSucSac.add(new  
    SucSac2());  
    lstSucSac.add(new SucSac3());lstSucSac.add(new  
    SucSac4());  
  
    //Khoi tao cac player ao  
    lstVirtualPlayer = new  
    ArrayList<AVirtualPlayer>();  
    lstVirtualPlayer.add(new VirtualPlayer1());  
    lstVirtualPlayer.add(new VirtualPlayer2());  
    lstVirtualPlayer.add(new VirtualPlayer3());  
    lstVirtualPlayer.add(new VirtualPlayer4());  
}
```

//Khoi tao cac player that

```
lstPlayer = new ArrayList<Player>();  
for (String name:lstName){  
    lstPlayer.add(new Player(name));  
}
```

//Them vao cac player ao cho du nguoi choi

```
if (lstPlayer.size()<Game.MAX_PLAYER){  
    int boSung = Game.MAX - lstPlayer.size();  
    for (int i =0; i<boSung; ++i)  
        lstPlayer.add(lstVirtualPlayer.get(i));  
}  
}
```

start()

```
public void start() {  
    setOrder();  
    turn = 0;  
    isGameOver = false;  
    while (!isGameOver) {  
        nextPlayer();  
    }  
    gameOver();  
}
```

nextPlayer()

```
private void nextPlayer(){
    //Dua vào turn, tính ra người chơi tiếp theo là ai
    if (turn >= Game.MAX) turn = 0;
    Player nxtPlayer = lstPlayer.get(turn);

    //Lấy súc sắc, gieo và tính điểm
    int random = (int)Math.floor(Math.random() * (Game.MAX_SUCSAC));
    int diemGieo = lstSucSac.get(random).gieo();
    int diemMoi = diemGieo + nxtPlayer.getCurrentPoint();
    nxtPlayer.setCurrentPoint(diemMoi <= 21 ? diemMoi : 0);

    //Kiểm tra kết thúc trò chơi
    if (nxtPlayer.getCurrentPoint() == 21){
        isGameOver = true;
        return;
    }
    turn ++;
}
```

Lớp VirtualPlayer1

```
public class VirtualPlayer1 extends
    AVirtualPlayer{

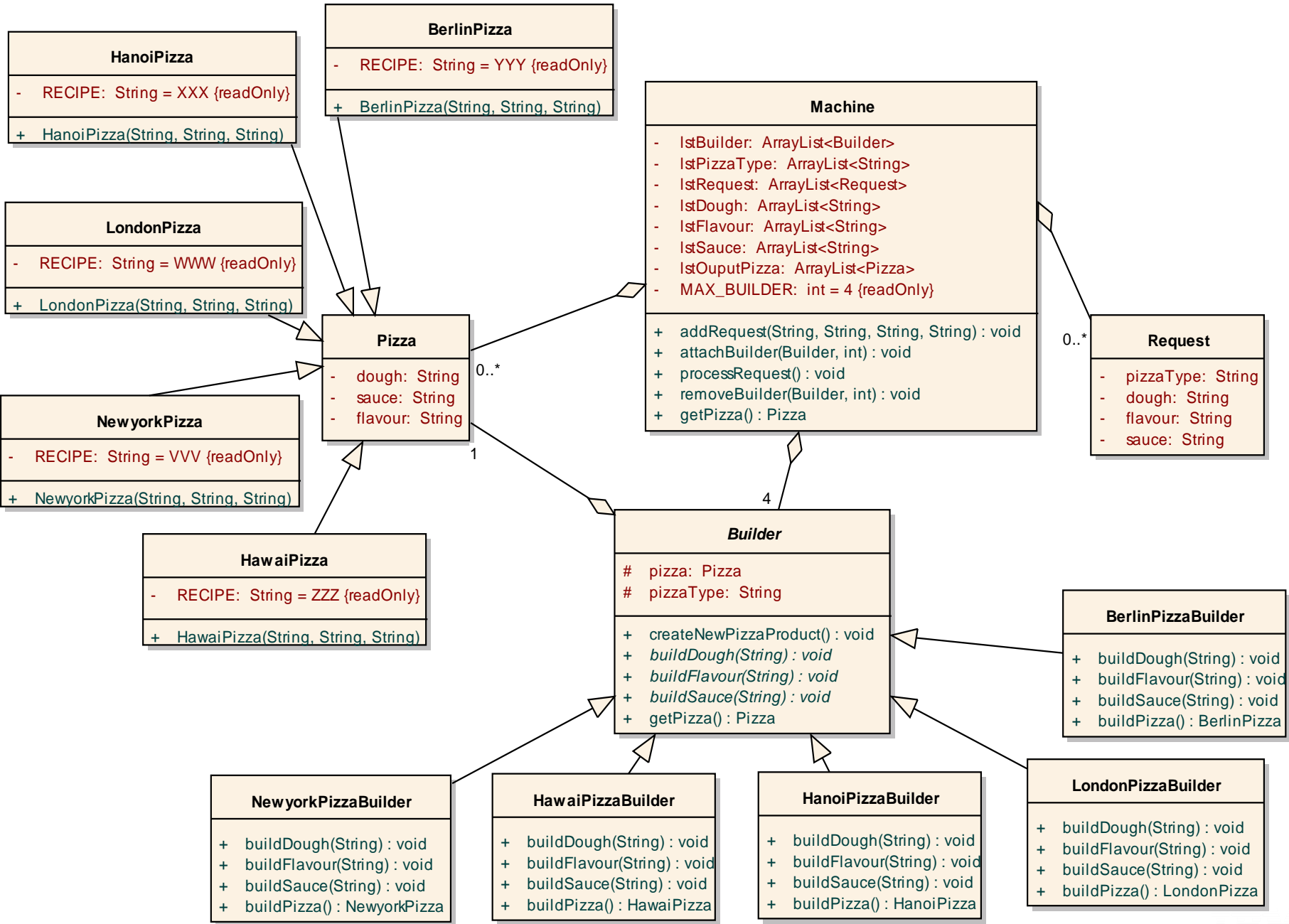
    public VirtualPlayer1() {
        this.name = "VirtualPlayer1";
    }

    @Override
    public void bieuLoThatBai() {
        System.out.println("Hic, thua roi 1");
    }

}
```


Bài 7.

- Cần xây dựng HT (máy) làm bánh Pizza tự động ở 1 công ty. Pizza có các thuộc tính: tên loại bột (dough), tên loại nước sốt (sauce), tên loại hương vị (flavour).
- Có 5 loại bánh Pizza (Hawai Pizza, NewYork Pizza, Berlin Pizza, London Pizza, Hanoi Pizza). Mỗi loại có công thức riêng và có thùng chế biến tương ứng (Hawai Pizza Builder, NewYork Pizza Builder, Berlin Pizza Builder, London Pizza Builder, Hanoi Pizza Builder). Mỗi thùng chế biến ít nhất cung cấp các phương thức tạo bột, tạo hương vị, tạo nước sốt cho bên ngoài gọi đến.
- Mỗi máy làm bánh được cài đặt tối đa 4 thùng chế biến (có thể ít hơn, và có thể tháo bớt, thêm các thùng dễ dàng). Tại 1 thời điểm, mỗi thùng chế biến chỉ tạo được 1 bánh Pizza, nhưng 4 thùng có thể cùng lúc tạo 4 bánh Pizza)
- Nhân viên trong công ty chỉ cần chọn 1 loại bánh trong danh sách (tùy cài đặt các thùng chế biến), chọn loại nguyên liệu làm bánh, đợi, nhận bánh ở khay ra (OutputTray) & thưởng thức. Khi nhận yêu cầu từ nhân viên, máy sẽ kích hoạt thùng chế biến tương ứng, thực hiện lần lượt tạo bột, tạo hương vị và tạo nước sốt, trả lại bánh Pizza mong muốn cho nhân viên.
- Lưu ý: do việc làm bánh mất thời gian, nên phải giải quyết được nhiều yêu cầu làm bánh gửi liên tục (ví dụ một nhân viên đặt bánh, chưa nhận được, một nhân viên khác lại đặt bánh tiếp).



Chú thích

- Nhân viên được chọn
 - Loại bánh
 - Nguyên liệu làm bánh với loại bánh trên
- Mỗi xxxBuilder như 1 hộp đen, 1 đơn đặt hàng từ Machine

Chú thích

- Trong xxxPizzaBuilder:
 - Có chứa pizza (nếu chưa chế biến → null).
 - Bắt đầu việc làm bánh bằng cách gọi phương thức `createNewPizzaProduct()`
 - Các phương thức `buildDough()`, `buildSauce()`, `buildFlavour()` lần lượt tạo các thành phần tương ứng cho pizza
 - Phương thức `getPizza()` sẽ trả về pizza, đồng thời set lại `pizza = null`, chuẩn bị sẵn sàng cho yêu cầu mới
 - Cần viết phương thức khởi tạo cho xxxPizzaBuilder, trong đó thiết lập giá trị `pizzaType` tương ứng

Chú thích

- Machine:

- Phương thức processRequest có thể cài đặt là 1 vòng lặp, luôn kiểm tra xem list request có rỗng không, nếu không thì gọi đến các Builder phù hợp
 - Cách khác?
- Khi attach/remove 1 builder, cần cập nhật IstPizzaType tương ứng
- IstBuilder luôn có 4 phần tử (nên có thể cài đặt tách list thành 4 thuộc tính cho Machine)
 - Mỗi phần tử ứng với 1 khay Builder

Chú thích

- Trong ví dụ này
 - Có thể không cần tách các HanoiPizza, LondonPizza, ... thành các lớp
 - Không cần mô hình hóa chi tiết công thức, cách chế biến (Do các Builder lo). Chỉ cần biết thứ tự gồm 3 bước chế biến

Bài 8.

Cần XD hệ thống ĐHND cho một tòa nhà 10 tầng (80 phòng):

- Có hệ thống điều hòa tổng, cho mọi hành lang, phòng, ...
- 10 phòng đặc biệt có hệ thống điều hòa riêng
- Điều hòa tổng và điều hòa riêng đều có chức năng:
 - Tăng/Giảm nhiệt độ, Bật/Tắt, Thông báo về nhiệt độ hiện thời, Tăng/Giảm tốc độ quạt gió, Bật/Tắt chế độ tuyết.
- Có máy điều khiển trung tâm cho người điều hành.
 - Thực hiện các chức năng kể trên cho hệ thống điều hòa tổng
 - Khóa/mở khóa từng điều hòa riêng

Bài 8. (2)

- Tại 10 phòng đặc biệt:
 - có điều khiển riêng cho hệ thống điều hòa của phòng đó.
 - Người dùng có thể thực hiện các chức năng nói trên.
 - Tuy nhiên, khi điều hòa riêng của phòng bị khóa, bất kỳ chức năng nào được yêu cầu đều không được thực hiện, (thông báo là đang bị khóa)
 - Nếu hệ thống điều hòa riêng được mở khóa, cửa chắn thông gió (của phòng này với hệ thống điều hòa tổng) sẽ tự động được đóng vào. Ngược lại, nếu khóa, cửa chắn thông gió lại mở ra.
- Người điều hành được HT hỗ trợ dự toán tiền điện cho từng tháng:
 - Giá tiền điện cho điều hòa tổng = $\sum (30 - \text{Nhiệt độ điều hòa}) \cdot \Delta t \cdot 1000$ (nghìn VNĐ)
 - Giá tiền điện cho điều hòa riêng = $\sum (30 - \text{Nhiệt độ điều hòa}) \cdot \Delta t \cdot 20$ (nghìn VNĐ)
 - Trong đó, Δt tính theo đơn vị giờ, là thời gian **Nhiệt độ điều hòa** được duy trì.

Bài 8. (3)

Yêu cầu:

- Thiết kế biểu đồ lớp để xây dựng hệ thống. Giải thích
- Viết code Java cho lớp ứng với đối tượng Điều hòa tổng. Nếu lớp này kế thừa/Thực thi 1 lớp/giao diện khác, viết code đầy đủ cho lớp/giao diện đó.
- Viết code cho phương thức dự toán tiền điện của toàn bộ hệ thống

