Báo cáo bài tập nhóm Java

1. Abstract

- **Tính trừu tượng (Abstraction)** là **che giấu chi tiết cài đặt bên trong** và **chỉ hiển thị các hành vi (method) cần thiết** cho người dùng.
- Người dùng chỉ cần biết **"làm gì"** (what) chứ không cần biết **"làm như thế nào"** (how).
- Tính trừu tượng thường thể hiện thông qua **abstract class, abstract method** hoặc **interface.**
- Từ khóa **abstract** là **non-access modifier**, được sử dụng cho **classes** và **methods.**
  - **Abstract class:** là 1 lớp được khai báo nhưng ko tạo object (phải được kế thừa bởi lớp khác)
  - **Abstract method:** được sử dụng trong Abstract class, không có body (body chỉ được khai báo trong lớp kế thừa)
- Ví dụ:

```
// Abstract class
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}
// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}
class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

- Để khai báo 1 abstract class/method, ta khai báo như sau:

```
<access_modifier> abstract class <TênLớp> {
    // fields (biến)
    // constructors
    // abstract methods (phải override ở lớp con)
    // normal methods (có thân hàm, dùng được luôn)
}
```

+ public → lớp có thể được truy cập từ bất kỳ đâu.
+ protected (không áp dụng cho class top-level, chỉ dùng cho inner class).
+ default (không ghi gì cả) → lớp chỉ được truy cập trong **cùng package.**

2. Interface

- Interface là 1 cách khác để thực hiện tính kế thừa của class trong Java
- Là 1 tập hợp các methods (không có body)
- Ví dụ

```
// interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void run(); // interface method (does not have a body)
}
```

- Để truy cập hay sử dụng, ta cần sử dụng từ khóa "**implements**" (thay vì **extends**), và các class sử dụng **interface** bắt buộc phải ghi đè các methods có trong **interface** (ngoại trừ **default** method)
- Ví dụ:

```
// Interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void sleep(); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
class Pig implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
```

```
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig();  // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

3. Interface vs Abstract class

- **Methods**: ***Class abstract*** có các phương thức *abstract* và *non-abstract*. Trong khi ***Interface*** chỉ có phương thức *abstract*, từ Java 8, thì ***Interface*** có thêm 2 loại phương thức là *default* và *static*.
- **Variables**: ***Class abstract*** có thể có các biến *final*, *non-final*, *static* và *non-static*. Trong khi ***Interface*** chỉ có các biến *static* và *final*.
- **Implementation**: ***Class abstract*** có thể *implement* các ***Interface***. Trong khi ***Interface*** thì không thể *implement* ***class abstract***.
- **Inheritance**: ***Class abstract*** có thể kế thừa được một class khác. Trong khi ***Interface*** có thể kế thừa được nhiều ***Interface*** khác.

```
interface A {
    void methodA();
}

interface B {
    void methodB();
}

// Interface C kế thừa nhiều interface
interface C extends A, B {
    void methodC();
}

class Demo implements C {
    public void methodA() { System.out.println("A"); }
    public void methodB() { System.out.println("B"); }
    public void methodC() { System.out.println("C"); }
}
```

- **Accessibility**: các thành viên trong ***Interface*** kiểu mặc định là *public*. Trong khi ***class abstract*** thì lại có thể là *private*, *protected*, ..

4. **Khi nào nên dùng *Abstract class*, khi nào nên dùng *Interface*?**
- ***Class abstract*** đại diện cho mối quan hệ "IS - A" (Ôtô là Xe)
- ***Interface*** đại diện cho mối quan hệ "like - A" (Ô tô có thể chuyển động).
- Tạo một ***class abstract*** khi bạn đang cung cấp các hướng dẫn cho một class cụ thể.
- Tạo ***Interface*** khi chúng ta cung cấp các hành vi bổ sung cho class cụ thể và những hành vi này không bắt buộc đối với class đó.

5. **Danh sách thành viên**
- Nguyễn Tự Kiên - B23DCCN465
- Nguyễn Thành Đạt - B23DCCN136
- Nguyễn Văn Đoan - B23DCKH023
- Đào Bình Minh - B23DCKH076
- Nguyễn Minh Hiếu - B23DCKH040