Practice OCP:

Chapter 1: Advanced Class Design:

**Java Class Design:**

* Triển khai kế thừa bao gồm công cụ sửa đổi hiển thị và thành phần.
* Triển khai tính đa hình
* Override hashCode , equals , toString từ Object class
* Phát triển code sử dụng từ khóa static trên block thành phần , biến , phương thức và classes

**Advanced Java Class Design:**

* Phát triển code sử dụng class abstract và phương thức abstract.
* Phát triển code sử dụng từ khóa final
* Tạo inner classes bao gồm static inner class , local class , nested class, và anonymous inner class.
* Sử dụng kiểu Enumerated bao gồm phương thức , Constructors trong một kiểu enum.
* Phát triển code khai báo , implements và extends interface và sử dụng annotation @Override.

**Access Modifiers**

Đầu tiên , đánh giá về access modifiers public , protected , default và private. Giả sử phương thức dưới đây tồn tại , bây giờ xem xét các biến thành phần mà nó cố gắng truy cập sau đây:

public static void main(String[] args) {  
 BigCat cat = new BigCat();  
 System.out.println(cat.name); System.out.println(cat.hasFur);  
 System.out.println(cat.hasPaws);  
 System.out.println(cat.id); }

Bây giờ , giả sử mỗi class có phương thức main và các thành phần của một BigCat cố gắng in ra toàn bộ 4 biến thành phần. Biến thành phần nào sẽ được chấp nhận trong mỗi trường hợp ?

package cat;  
public class BigCat {  
public String name = "cat";  
protected boolean hasFur = true;  
boolean hasPaws = true;  
private int id;  
}

package cat.species;  
public class Lynx extends BigCat { }

package cat;  
public class CatAdmirer { }

package mouse;  
public class Mouse { }

Mặc dù code biên dịch cho BigCat nhưng nó ko phải cho toàn bộ các class.

Dòng cat.name biên dịch cho toàn bộ các class bởi vì bất kì code nào cũng có thể truy cập thành phần Public. Dòng cat.id biên dịch trong BigCat bởi vì chỉ có code trong cùng 1 class có thể truy cập thành phần private. Dòng cat.hasPaws biên dịch trong BigCat và CatAdmirer bởi vì chỉ có code trong cùng một package có thể truy cập thành phần default.

Cuối cùng dòng cat.hasFur cũng biên dịch trong BigCat và CatAdmirer. Protected cho phép các subclass và code trong cùng package truy cập thành phần protected. Vì code đang được truy cập thông qua biến thay vì kế thừa, vì thế nó không phải lợi ích từ protected. Tuy nhiên , nếu code trong main là Lynx cat = new Lynx(); Lynx sẽ được cho phép truy cập vào cat.hasFur sử dụng protected bởi vì nó có thể được nhìn bởi một subclass.

**Overloading và Overriding**

Đánh giá về sự khác nhau giữa overloading và overriding.

Để đánh giá vấn đề Overloading và Overriding chỉ khi tên phương thức giống nhau, thực thi Overriding chỉ khi chữ kí của phương thức giống nhau. Chữ kí phương thức là tên phương thức , danh sách tham số . Với Overloading, tham số phương thức phải khác nhau về kiểu hoặc số lượng tham số.

Khi thực hiện nhiều phương thức được overload , Java sẽ tìm kiếm kết quả phù hợp nhất đầu tiên, nó cố gắng tìm như sau:

* So sánh chính xác về kiểu
* So sánh với kiểu Superclass
* Chuyển đổi độ lớn của kiểu nguyên thủy
* Chuyển đổi kiểu Autoboxing
* Var-args

Với việc Overriding , phương thức được override phải tuân thủ luật sau đây:

* Access modifier phải giống hoặc phạm vi truy cập lớn hơn
* Kiểu trả về phải giống nhau trong trường hợp nguyên thủy hoặc hạn chế hơn được gọi là covariant.
* Nếu bất kì checked exception nào được throw ra , chỉ cùng exceptions hoặc subclasses của exceptions đó được cho phép throw hoặc ko throw ra exception.

Phương thức không được là static. (Nếu có , phương thức bị ẩn hoặc ko được override)