12 面向对象编程IV：继承与多态

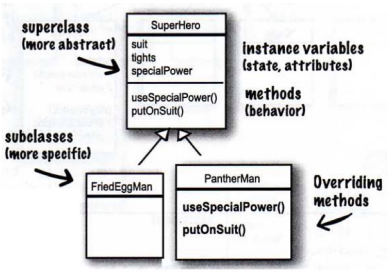
一. 继承

1. 继承的由来

(1)重复代码.

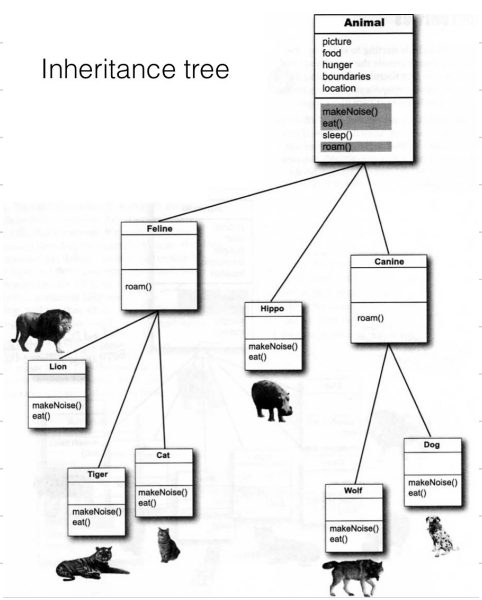
(2)分类、抽象的概念.

类图：



- 子类继承了父类所有的成员变量和成员方法
- 可以增加成员变量和成员方法
- 可以覆盖父类的成员方法
- 不可以覆盖父类的成员变量

继承树：



2. is-a 继承测试

- When one class inherits from another, we say that the subclass extends the superclass.
- When you want to know if one thing should extend another, apply the IS-A test.

Triangle IS-A Shape, yeah, that works.

Cat IS-A Feline, that works too.

Surgeon IS-A Doctor, still good.

is-like-a：重写了一定的方法

二. 抽象类和抽象方法

抽象类：不可被实例化.

An abstract method has no body!

Because you've already decided there isn't any code that would make sense in the abstract method, you won't put in a method body. So no curly braces—just end the declaration with a semicolon.

```
public abstract void eat();
```

No method body!
End it with a semicolon.

If you declare an abstract *method*, you MUST mark the *class* abstract as well. You can't have an abstract method in a non-abstract class.

非抽象类中不可以有抽象方法，抽象类中可以有非抽象方法.

- 抽象方法的存在就是为了多态
- 具体的子类必须实现所有父类的抽象方法
- 实现抽象方法就像子类覆盖父类方法一样

三. 多态.

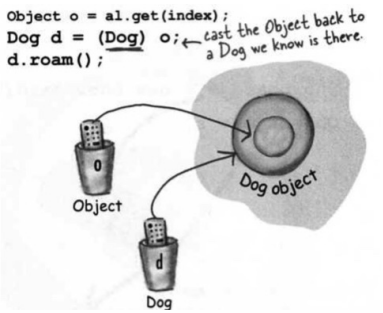
Java.lang.Object：所有类的父类

- 使用Object List
  - ArrayList<Object> myDogArrayList = new ArrayList<Object>(); 存放Object引用类型
  - Dog aDog = new Dog();
  - myDogArrayList.add(aDog);
  - Dog d = myDogArrayList.get(0); 无法通过编译！

- 编译时，编译器决定你是否能调用某个方法
  - 依据引用变量的类型，而不是引用变量指向的对象的类型
- 执行时，JVM虚拟机决定实际哪个方法被调用
  - 依据实际引用变量指向的对象的类型

静态检查

强制类型转换：



多态的思想

- 多态通过分离“做什么”和“怎么做”，从另一角度将接口和实现分离开来。
- 而多态的作用则是消除类型之间的耦合关系多态方法调用允许一种类型表现出与其他相似类型之间的区别，只要它们都是从同一基类导出而来的。这种区别是根据方法行为的不同来而表示出来的，虽然这些方法都可以通过同一个基类来调用。
- 多态可以表达不同的计算类型，并且在运行的时候动态的确定正确的计算。
- 多态是指多个方法使用同一个名字有多种解释，当使用这个名字去调用方法时，系统将选择重载自动的选择其中的一个方法。在多态中只关心一个对象做什么，而不关心如何去做。

方法覆盖 Overriding

- 条件：
  - 在不同类（父类和子类）、同一个个方法、方法名字相同
- 参数必须一致，返回值必须兼容
  - 例如：父类返回Animal，子类返回dog
- 方法的可达性不能降低。
  - 例如：父类是缺省的，子类是public

overloading是同一个类内两个方法
overriding是子类与父类间的方法.

子类不可以 overriding 父类的私有方法
父类的私有方法被编译器认为是final方法

四. 方法调用的字节码