编程入门基础

——指针泛化与多态

主讲教师：耿宇航

泛化

- 泛化就是抽象化
  - 把具体的东西抽象地看待
  - 把 “猫” 看为 “动物”
  - 把 “苹果” 看待为 “水果”
- 泛化也叫向上塑型，是通往多态的路口

指针的泛化

- 泛化是指针（引用）的泛化
- 并不是“对象”的泛化
- 并不是“类”的泛化

```
class 水果
{
...
}
```

```
class 苹果 extends 水果
{
...
}
```

```
水果 x;
x = new 苹果();

// x 的身份是指针
// 赋值动作不会引起对象的变化
// 赋值动作当然也不会改变那两个类
```

泛化的动作

- 除了像其它语句一样的赋值动作，泛化并不会在内存中作出什么特殊的处理来。它仅仅是让编译器把一种变量的类型当成是另一种类型来处理。
- 这么做，可以把许多本来不同的类型一视同仁地对待
- 这么做既是可行的，也是合理的。

把子类对象看作父类对象

- 这样可行吗？
  - 可行，我们自愿放弃了更丰富的信息
- 这样有用吗？
  - 有用，我们可以用同样的类型处理不同的东西。为了发现共性

泛化的对立面

- 泛 = 广泛，宽泛
- 泛化的对立面就是“特化”或者“具体化”
- 已知 a 是猫，我们也可以说 a 是动物
- 如果已知 a 是动物，我们不能断定 a 一定是猫
- 动物 a = new 猫();
- 猫 b = a;
- 虽然客观上上面的操作无可厚非，但实际编译不通！因为第二句话是“特化”，编译器不能任之。

指针与它指向对象的类型

- 指针的类型与它指向的对象的类型可以不同
- 指针的类型可以更“宽泛”，这样就放弃了对象的独特动作，只做一般的操作。
- 如何知道指针所指向对象的真实类型？
  - 用 instanceof 测试
  - 用 反射 机制

instanceof

- 这不是一个方法
- 这是一个java关键字，如同 while, if 等
- 用法： 指针 instanceof 类型　返回一个boolean
- if(a instanceof 猫) ….
- if(a instanceof Object) …

对象的类型

- 对象总是具体的类创建的，因而有一个真实类型
- 操作对象的时候总是要通过一个指针，指针的类型有可能与对象的真实类型不同，叫做：表面类型
- 表面类型 = 编译类型，语法类型，形式类型，…
- 真实类型 = 运行时类型，实质类型，实际类型，…

怎样的现象叫多态？

- 通过父类的引用能调用子类中的方法
- 在程序运行前(编译时)，无法确定调用哪个方法的现象

```
动物 a;
if(Math.random() > 0.5)
  a = new 猫();
else
  a = new 狗();

a.eat();  // 编译时的迷茫....
```

多态带来了什么好处？

- 可以对多种有共性的事物统一（不是分别）地处理
- 站在更高的层次管理对象的行为
- 软件的所谓蓝图化设计

另一个角度找多态

- 过去写好的代码未经修改，可以调用到刚刚写好的代码！

```
class 猫
{
public String toString()
{
        return “I am Cat!”;
}
}

猫 a = new 猫();
System.out.println(a);
```

println 是很早的代码，从未修改。但它却调用了新写的 **toString()**方法。

覆盖是纽带

出现泛化

```
public void println(Object x)
{
   println(x.toString());
}
```

```
猫 a = new 猫();
System.out.println(a);
```

编译时瞄准

```
class 猫
{
public String toString()
{
        return “I am Cat!” ;
}
}
```

```
class Object
{
…
public String toString()
…
}
```

运行时寻找到

多态的应用

- 对象间的比较问题
  - 直接对对象变量去比较，比较的是对象的存储地址。
  - 要比较对象的内容是否相同，可以使用对象提供的方法来完成

谢　谢！