面向对象的陷阱

本书笔记知识点来自于李刚《Java程序员的基本修养》

- instanceof运算符
- 重载方法
- static关键字
- 构造器陷阱

1: instanceof运算符

instanceof运算符有三个使用规则限制：

- 要判定的对象的类与后面的类相同
- 要判定的对象的类和是后面的类的子类
- 要判定的对象的类和是后面的类的父类

2: 构造器陷阱

1. 构造器创造对象么？
   构造器并不会创建对象，构造器的作用是初始化，new关键字的作用才是在内存中为对象申请空间。除了new关键字可以创建一个对象之外，还存在两种方法：**反序列化恢复Java对象和clone方法复制Java对象**。但值得注意的是通过反复序列化恢复的Java和之前的对象在内存地址不一样，但是具有相同的变量值(一个特例是singleton模式，它所恢复的对象和原来的对象是同一个)。
   如何让一个Java对象可以复制自己？
   可以让Java类实现Cloneable()接口，实现clone()方法即可。但是和反序列化一样，clone方法复制的对象也是在堆中的一个新对象，这一点可以理解，虽然是复制但毕竟是新对象。下面例子实现一个类对Cloneable接口的实现。

```
class Dog implements Cloneable {
    int age;
    String name;
    public Dog(String name, int age) {
        this.age = age;
        this.name = name;
    }
    public Object clone() {
        Dog dog;
        try {
            dog = (Dog) super.clone();
        }
        catch(CloneNotupportedException e) {
            e.printStackTrace();
        }
        return dog;
    }
}
这样子就可以为对象类实现了Clonenable接口，实例对象可以复制。
```

3: 方法重写的陷阱

当一个子类继承了父类时，如果子类想改变父类的行为，就要重写父类的方法。

1. 重写private方法
   当一个类中的方法的修饰符是private时，他只能被自己内部方法以及自己实例对象访问，尽管子类继承了父类的一些方法，但是子类还是无法访问父类的private方法，如果在子类里面定义了和父类中方法名以及参数名相同的方法，这只是一个新方法，并不是新方法。**因此，如果一个类就是作为基类来使用的，期待子类重写他的方法，那么方法修饰符不能使用private。**
2. 重写其他权限的父类方法
   上面提到父类中private修饰的方法不能被子类重写。但是是不是对于其他情况下，父类中的方法就能被子类重写了呢？也不一定！有一个特例。
   当一个类中的方法没有权限修饰符修饰时，默认的权限就是包访问权限，也就是在同一个包内的子类继承可以重写父类方法，如果继承的子类和父类不在一个包内，那么即使子类中的方法返回类型和方法签名和父类一致，也只是定义一个新的方法，并不是重写父类方法。

4: 非静态内部类陷阱

非静态内部类可以直接访问外部类的方法和变量，因此在一些场合经常使用。

1. 非静态内部类的构造器不存在没有参数的构造器，因为内部类必须依赖于外部类而存在。因此尽管我们在使用是，好像还是像一般类的实例化那样`Inner inner = new Inner()`来进行类的实例化，但是系统在编译时会自动的为内部类构造器中添加一个参数，这个参数就是外部类的实例。

5: static关键字

static关键字所修饰的，甭管原本是什么属性，修饰之后属于类本身，也就是类成员。

1. 静态方法属于类
   因为static关键字修饰的方法属于类本身，而不是属于类的对象，因此在实际上调用静态方法时，调用的其实是声明变量时所指定的那个类，而不是在堆中实际创建的对象那个类。看下面例子：

```
public Base {
    public static void info() {
        System.out.println("This is the base method");
    }
}
public Sub extends Base {
    public static void info() {
        System.out.println("This is the Sub method");
    }
}
public class Test {
    public static void main(String[] args) {
        Base a1 = new Base();
        a1.info();
        Base a2 = new Sub();
        a2.info();
    }
}
两次测试结果都是This is the base method.
```

   看上面那个例子，表面上好像子类重写了父类的info()方法，其实不然，被static关键字修饰的方法不能被子类重写，即使有一个相同的方法，也只是一个新方法，而非重写。
2. 静态内部类的限制
   在上面我们曾经写过非静态内部类的一些陷阱，因此如有可能，最好采用static的内部类，当采用静态内部类时，外部类相当于静态类运行的一个包，这样操作就很方便。但是内部类有一个缺陷，**就是它无法访问外部类的非静态成员。**