第4章-类的重用

基类（base class）也称超类（superclass），是被直接或间接继承的类。

派生类（derived-class）也称子类（subclass），继承其他类而得到的类，继承所有祖先的状态和行为，派生类可以增加变量和方法，派生类也可以覆盖（override）继承的方法。

子类不能直接访问从父类中继承的私有属性及方法，但可使用公有（及保护）方法进行访问。

隐藏和覆盖：子类对从父类继承来的数形变量及方法可以重新定义。

如何访问被隐藏的父类属性?

1.调用从父类继承的方法，则操作的是从父类继承的属性。

2.使用super.属性

子类能继承父类中的静态属性，但可以对父类中的静态属性进行操作。

父类private属性的方法不能被继承，因此也就不能被覆盖。

运行时多态是通过动态联编实现的。就是在运行时根据对象的实体类型确定对象的动作。

Object类：Java程序中所有类的直接或间接父类，类库中所有类的父类，处在类层次最高点，包含了所有Java类的公共属性，其构造方法是Object().

相等和同一的区别

比较运算符“==”判断的是这两个对象是否同一

Object类中定义了equals()方法，其定义如下，即判断两个对象是否同一。

```
public boolean equals(Object x){
    reutrn this == x;
}
```

equals()方法的重写：要判断两个对象各个属性域的值是否相同，则不能使用从Object类继承来的equals方法，而需要在类声明中对equals方法进行重写。

String类中已经重写了Object类的equals方法，可以判别两个字符串是否内容相同。

```
class Apple{
    private String color;
    private boolean ripe;
    public Apple(String aColor, boolean isRipe){
        color = aColor;
        ripe = isRipe;
    }
    public void setColor(String aColor){color = aColor;}
    public void setRipe(boolean isRipe){ripe = isRipe;}
    public String getColor(){return color;}
    public boolean getRipe(){return ripe;}
    public String toString(){
        if(ripe)return("A ripe " + color + " apple");
        else return("A not so ripe " + color + " apple");
    }
    public boolean equals(Object obj){
        if(obj instanceof Apple){
```

```
            Apple a = (Apple)obj;
            return (color.equals(a.getColor()) && (ripe == a.getRipe()));
        }
        else return false;
    }
}

public class AppleTester{
    public static void main(String[] args){
        Apple A = new Apple("red", true);
        Apple B = new Apple("red", true);
        if(A.equals(B)){
            System.out.println("A == B");
        }else System.out.println("A != B");
    }
}
```

clone方法

finalize()方法

getClass方法

终结类与终结方法：

被final修饰符修饰的类和方法

终结类不能被继承

终结方法不能被当前类的子类重写

终结类的特点：不能有派生类。

终结方法的特点：不能被派生类覆盖。

抽象类代表一个抽象概念的类，没有具体事例对象的类，不能使用new方法进行实例化，类前需加修饰符abstract。

抽象方法，仅有方法头，而没有方法体和操作实现。

```
class GeneralType <Type> {
    Type object;
    public GeneralType(Type object) {
        this.object = object;
    }
    public Type getObj() {
        return object;
    }
}

public class TestGeneralType{
    public static void main(String[] args){
        GeneralType<Integer> i = new GeneralType<Integer>(1);
        GeneralType<Double> d = new GeneralType<Double>(0.33);
        System.out.println("i.object = " + (Integer)i.getObj());
        System.out.println("d.object = " + (Double)d.getObj());
    }
}

```

```java
class GeneralMethod{
    <Type> void printClassName(Type object) {
        System.out.println(object.getClass().getName());
    }
}

public class TestGeneralMethod{
    public static void main(String[] args){
        GeneralMethod gm = new GeneralMethod();
        gm.printClassName(1);
        gm.printClassName(0.33);
        gm.printClassName("Hello");
    }
}
```

String的charAt方法

StringBuffer类：其对象是可以修改的字符串，该类的方法不能被用于String类的对象。

```java
class StringEditor{
    public static String getLetter(String str){
        int len = str.length();
        StringBuffer sb = new StringBuffer(len);
        for(int i = 0; i < len;i++){
            char ch = str.charAt(i);
            if(Character.isLetter(ch))
                sb.append(ch);
        }
        return new String(sb);
    }
}

public class TestStringBuffer{
    public static void main(String[] args){
        String s = "asb23123asd";
        String res = StringEditor.getLetter(s);
        System.out.println(res);
    }
}
```