Chapter 9 对象和类

9.1 对象和类的定义

对象：现实世界中可识别的**实体**，具有一定的状态和行为

类：是对同种对象（对象具有相同的属性或者方法）的**抽象**

举例：我是一个人，你也是一个人，那么人就是一个类，而我和你就是这个类的实例，是对象。

9.2 定义类、实例化类

当创建对象数组时，数组元素的缺省初值为null。

```
Circle[] circleArray = new Circle[10]; //这时没有构造Circle对象，只是构造数组
for(int i = 0; i < circleArray.length; i++) {
    circleArray[i] = new Circle( );    //这时才构造Circle对象，可使用有参构造函数
}
```

9.3 理解构造函数

构造函数特征：

- 无返回类型
- 名字同类名
- 用于初始化对象
- 只在new时被自动执行

构造函数限制：

- 必须是实例方法（无static），可以为共有、保护、私有和包级权限
- 如果没有定义任何构造函数，编译器会自动提供一个不带参数的默认构造函数
- 如果已自定义构造函数，则不会提供默认构造函数

9.4 理解对象访问

9.5 对象的变量、常量和方法

9.5.1 实例变量和静态变量

- 实例变量：未用static修饰的成员变量，属于类的具体实例，只能通过对象访问
- 静态变量：用static修饰的变量，被类的所有实例所共享，可以通过实例访问，也可以直接通过类访问

9.5.2 实例常量和静态常量

- 实例常量：没有用static修饰的final变量
- 静态常量：用static修饰的final变量
- 所有常量可以按需指定访问权限，但由于他们不能被修改，所以通常定义为public

9.5.3 类和实例的方法

静态方法

- 用static修饰的方法
- 构造函数不能用static修饰
- 静态方法没有this引用
- 静态方法可以通过对象和类名调用
- **静态方法内部只能访问类的静态成员 (因为实例成员必须有实例才存在，当通过类名调用静态方法时，可能该类还没有一个实例)**
- 静态方法没有多态性

final修饰

- final修饰实例方法时，表示该方法不能被子类覆盖（Override）。非final实例方法可以被子类覆盖
- final修饰静态方法时，表示该方法不能被隐藏（hiding）。非final静态方法可以被子类隐藏
- 构造函数不能为final

方法重载（overload）

同一个类中、或者父类子类中的多个方法具有相同的名字，但这些方法具有不同的参数列表(不含返回类型，即无法以返回类型作为方法重载的区分标准)

方法覆盖（override）和 方法隐藏（hiding）

发生在父类和子类之间，前提是继承。子类中定义的方法与父类中的方法具有相同的方法名字、相同的参数列表、相同的返回类型（也允许子类中方法的返回类型是父类中方法返回类型的子类），对于实例方法来说这就是方法覆盖，对于静态方法来说这就是方法隐藏。

```
public class A {
    public void m(int x, int y){
        // do something
    }
    // m方法的重载
    public void m(double x, double y){
        // do something
    }
}

class B extends A{
    // m方法的重载
    public void m(float x, float y){
        // do something
    }
    // m方法的覆盖
    public void m(int x, int y){
        // do something else
```

```
    }
    // 下面这个方法既不是重载也不是覆盖
    public int m(int x, int y){

    }
}
```

9.6 可见性修饰符

类访问控制符：

- public
- 包级（默认）

类成员访问控制符以及作用：

- private：只能被当前类定义的函数访问
- protected：子类、同一包中的类的函数可以访问
- public：所有类的函数可以访问
- 包级（默认）：无修饰符的成员，只能被同一包中的类访问

访问权限	本类	本包	子类	它包
public	√	√	√	√
protected	√	√	√	X
包级（默认）	√	√	X	X
private	√	X	X	X

Java继承时无继承控制(见继承，即都是公有继承，和C++不同)，故父类成员继承到派生类时访问权限保持不变（除了私有）。

9.7 变量的作用域和访问优先级

- 类的成员变量额作用域是整个类，和声明位置无关
- 如果一个成员变量的初始化依赖于另一个变量，则另一个便令必须在前面声明
- 如函数的局部变量i与类的成员变量i名称相同，那么优先访问局部变量i，成员变量i被隐藏(可通过this.i或类名.i访问)。

9.8 this引用

this引用指向调用某个方法的当前对象。

```
public class Foo {
    protected int i = 5;
    protected static double k = 1.0;

    void setI(int i){
```

```
        this.i = i;
    }
    static void setK(double k){
        Foo.k = k;
    }

    public static void main(String[] args) {
        Foo foo1 = new Foo();
        Foo foo2 = new Foo();

        foo1.setI(19);      // this指向f1
        foo2.setI(2);       // this指向f2
    }
}
```