Java4:类和对象

授课教师：邱元杰　电子邮箱：yuanjiq@126.com，微信电话: 13679081552

第4章 类和对象

面向对象程序设计

类的创建

方法过载

类的构造方法

类成员和实例成员

对象创建与使用

面向对象程序设计

- 面向对象程序（OOP）依照现实世界的实体的特点，把复杂的现实的事物按它们所共有的状态和行为抽象并封装。
- 面向对象程序语言有三个特征:
  - 封装性
  - 多态性
  - 继承性

面向对象程序设计

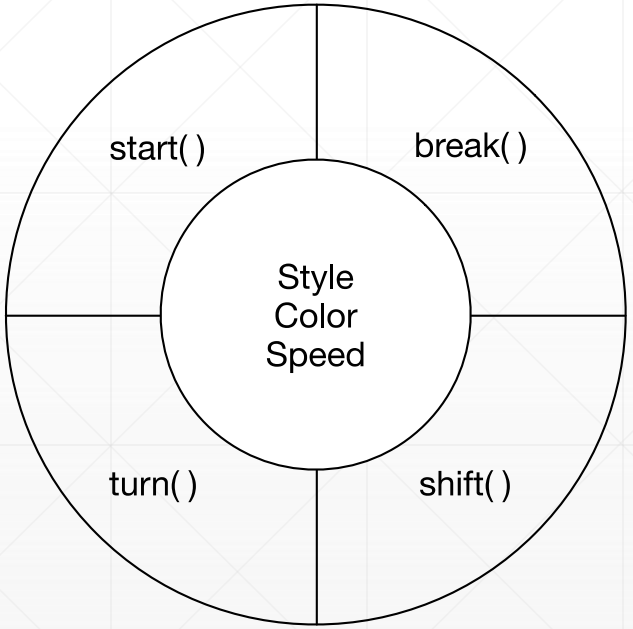
- 对基于“数据”本身，而非“功能”的面向对象程序设计，从抽象的概念上分析所要处理的数据是哪些类(class)，到各类数据可进行的操作方法，都是通过各个封装好的数据模块组合起来解决问题。
- 一个数据模块称为一个对象(object)，它是抽象出来的某个类(class)的一个实例。类就是一个引用数据类型，对象则是这个引用数据类型的变量。对象就是经过实例化的实体的引用。

面向对象程序设计

- 类的变量描述对象的属性，类的方法体现对象的行为。封装在类中的这些数据的变量和方法，为对象的创建提供了模板。
- 对象之间的交互作用是通过对象的消息机制实现的。所谓消息是对象对其他或者自身方法的访问，一个消息的产生就是一个对象方法的调用。
- 类和对象是面向对象程序设计中的基本概念，也是编程方法的基础。类就是Java的执行单位。Java运行的就是Java类本身

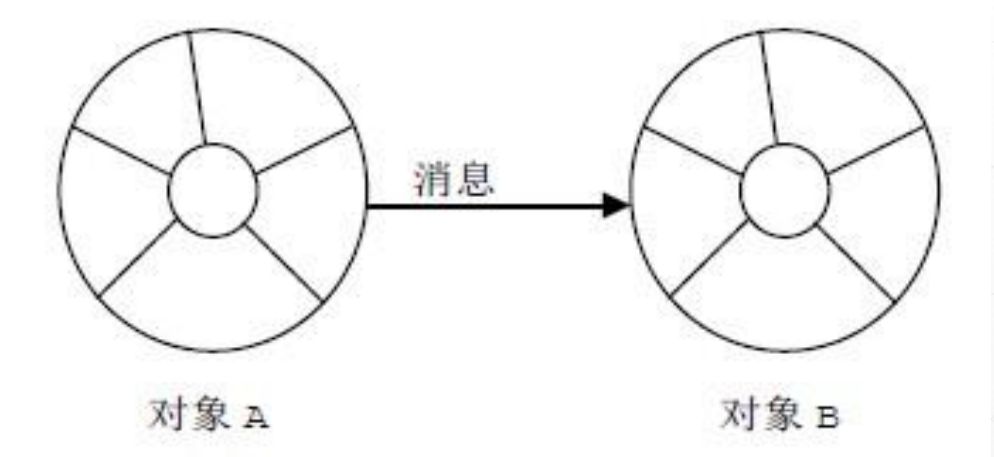
面向对象程序设计：对象

- 对象是现实世界事物的模型化。用变量来表示状态，用方法来实现行为。
- 对象的变量组成了对象的中心或核，而方法是围绕在变量四周的。
- 封装：某一对象的变量对程序中的其它的对象而言是隐藏的，它是在这个对象方法的保护监理之下



面向对象程序设计：消息

- 消息是软件对象之间相互作用和互相联系。
- 对象的行为由方法来实现，消息传递是对象之间进行交互的主要方式。
- 消息由三个部分组成：
  - 接收消息的对象(目标对象)；
  - 接收者采用的方法(目标方法)；
  - 方法所需传递的参数。



面向对象程序设计：类

- 类定义了对象的类型。包括对象内部的数据，它是对象的特性，和对象对自己的数据上的操作功能。因此类就是对象的模板。
- 类是某种对象共有的状态和行为的原型。
- 一个类可以有许多的对象，每一个对象都是这个类的一个实例，它们也就都具有相同的特征。

第4章 类和对象

面向对象程序设计

类的创建

方法过载

类的构造方法

类成员和实例成员

对象创建与使用

类的创建

- 类是Java语言的最基本概念，组成Java程序的基本要素。类是Java的执行单位，Java运行的就是Java类本身。它封装了该类对象的变量和方法。
- 一个类包括有三个部分：
  - 类声明；
  - 类成员；
  - 类的构造器(方法)。
- 类的一般格式是：
  - classDeclaration{
  - classBody
  - }

类的创建

- 类的一般声明形式：
- [public] class <clsName> extends <supCls> implements <intf>
  - class是表示创建类的关键字；
  - <clsName>是Java合法标识符；
  - [public]是可选项，表示该类是public类；类的可选项还有abstract、final等等；
  - extends <superCls>则是继承性表示，该类继承了类<superCls>
  - implements <intf>则是对接口实现表示，该类实现了接口<intf>

类的创建：类体定义

- 类的体包含有：
  - 成员变量。在类中创建的变量，表示对象属性；
  - 成员方法。类的方法表示对象的行为或能力；
  - 类的构造器(方法)。为创建类的实例所使用。
- 类的一般格式是：

```
classDeclaration{
        memberVariableDeclaration
        memberMethodDeclaration
        classStructorDeclaration
}
```

类的创建：成员变量

- 成员变量
  - 在类中创建的变量，表示对象属性；
  - 作用域在类中是全局的，能被类中所有方法所访问；
  - 在创建时有初始化值。
  - <type> variableList = initialValue;
  - 例： int a, b; //成员变量a和b都有一个初始值为0。
- 成员变量的定义还有许多可加的修饰符，用于声明成员变量的访问控制权限和使用限制。
  - 访问控制权限的修饰符有public、protected、private等；
  - 使用限制的修饰符有final、static、transient、volatile等。

类的创建：成员变量初始化

- 创建的变量时初始化值。
  - 例： int a, b; //成员变量a和b都有一个初始值为0。
- 创建时赋初值；
  - 例： int x = 10, y = 20; //x,y分别赋初值10,20。
- 在类的构造器中对成员变量的赋初值。

例：Point(){

x = 10; //x赋初值10

y = 20; //y赋初值20

}

类的创建：常量

- 使用修饰符final修饰的变量就像常量一样地使用，称其为常量符号。
- 常量符号数据只能读取，不能改变。通常常量符号标识符全用大写字母，单词间用“_”分隔。
  - 例：

```
final int MAX_MONTH = 12;
final int MAX_DAY = 31;
final int MAX_WEEK = 7;
```

类的创建：成员方法

- 方法的创建分为二个部分，一个是方法声明，另一个是方法体。

```
methodDeclaration{
        methodBody
}
```

- 方法声明格式：

[accessControl] returnType methodName(paraList)

- 方法体格式：

语句及语句块组成；

类的创建：成员方法

- 方法声明
- [accessControl] returnType methodName(paraList)

  [accessControl]是可选项，为访问控制修饰符，限定访问权限.

  returnType是方法返回数据类型，它表示方法返回时返回数据的类型.

  methodName是方法名，它是Java合法标识符.

  paraList是方法参数列表，表示方法调用时所带参数。称为形式参数。
- 例如：
  - int getMax(int a, int b)
  - void setData(int x, int y)

类的创建：成员方法

- 方法体由语句及语句块组成，包含有：
  - 局部变量声明；
  - 流程控制语句；
  - 语句块；
  - return语句*；
- 当方法返回类型不是void时，方法中必须有带表达式return语句：

  return expression;
- 当方法返回类型是void时，方法中可以有return语句，但是不能带表达式：

  return;

VariableCovert.java

定义一个汽车类

有状态	有功能
• 速度 • 方向 • 油量	• 启动 • 停止 • 加速 • 减速

类的创建：成员方法

- 方法调用时的参数传递
  - 方法在被调用时，其参数的数据传递是值传递，即实际参数传值给形式参数。
  - 形式参数是简单类型

    在方法调用时，实际参数将其存储单元的数据赋值给形式参数
  - 形式参数是引用类型

    在方法中，引用类型的参数没有发生引用的改变，则形式参数对引用中的变量值改变自然会影响到实际参数引用中变量的值

类的成员方法

- 形式参数是简单类型示例

```
x=1; y=2
void setData(int a, int b){
    a = 10;
    b = 20;
  }
setData(x,y)
```

- 参数a,b是简单类型，在函数中a 和b的值发生改变，但是不会影响调用的实参x和y，函数调用完成后，x和y的值仍然是1和2。

类的成员方法

- 形式参数是引用类型示例1

```
Point pi=new Point(1,2);
void setRefer1(Point p){
  p.x = 10;
  p.y = 20;
}
setRefer1(pi)
```

- 参数p是引用类型，其成员的x,y值发生改变，会影响调用的实参，即函数结束后成员x和y的值改变。

类的成员方法

- 形式参数是引用类型示例2

```
Point pi=new Point(1, 2)
void setRefer2(Point p){
  p = new Point();
  p.x = 10;
  p.y = 20;
}
setRefer2(pi)
```

- 参数p是引用类型，改变引用，其变量x,y值发生改变，不会影响调用的实参，函数调用结束后pi的成员值不变。

第4章 类和对象


面向对象程序设计


类的创建


方法过载


类的构造方法


类成员和实例成员


对象创建与使用

方法过载

- **方法过载**
  - **在同一个类中创建的具有相同方法名，但是参数不同的方法。**
  - **参数不同：**
    - **数量不同；**
    - **数量相同，但是对应的类型不同。**
  - **方法过载也是多态性表现之一。**
  - **方法过载中的方法由调用时的实参决定调用的方法是那个。**

- 示例：

```
void setData(int a, int b)

void setData(Point p)

//个数不同

void setData(float a, float b)

//类型不同

void setData()

//个数不同
```

MethodOverLoadTest.java

区分过载方法

- 过载的方法都必须采取独一无二的自变量类型列表，即使自变量的顺序也足够我们区分两个方法
  - 主类型的过载
    - 主（数据）类型能从一个“较小”的类型自动转变成一个“较大”的类型
  - 返回值过载?
    - 为什么只有类名和方法自变量列出?
    - 为什么不根据返回值对方法加以区分?

OverloadingOrder.java PrimitiveOverloading.java

第4章 类和对象

面向对象程序设计

类的创建

方法过载

类的构造方法

类成员和实例成员

对象创建与使用

类的构造器

- 每一类都有自己的构造方法，或者称为类的构造器。构造方法是用来创建一个类的实例的。
  - 构造方法是用类名作构造方法名；
  - 构造方法具有参数和语句体，但没有返回类型的声明。如果有返回类型声明，则此方法就再不是构造方法，而成为一个一般的成员方法；
  - 构造方法不是类的成员方法，所以不能用对象调用它。
  - 构造方法的调用是由new运算符实现；

类的构造器

- 每一类都有自己的构造方法，或者称为类的构造器。构造方法是用来创建一个类的实例的。
  - 构造方法返回的是这个类的实例的引用；
  - 构造方法中的语句实现对成员变量的初始化；
  - 构造方法的方法过载。一个类可以有多个构造方法；
  - 构造方法之间通过this()形式相互调用。

类的构造器

- 类构造器(方法)
  - 类的构造方法又可以分为两种
    - 默认构造方法
    - 非默认构造方法。
- 默认构造方法是指不带参数的构造方法，该方法可以有语句，也可以没有语句；
- 非默认构造方法是指带参数的构造方法。

类的构造器

- 类创建时没有创建构造器，则在编译时编译器自动为该类添加一个默认构造器，即没有创建构造器的类可以直接使用默认构造器。
- 类创建时如果有任何构造器被创建，则在编译时编译器不再为该类创建默认构造器。即创建了构造方法的类，不再自动有一个默认的构造方法，或者说默认构造器失效，除非也创建一个不带参数的构造器。

ConstructionMethos.java

成员变量初始化

- 类的成员变量初始化
- 成员变量初始化三个步：
  - 创建时分配存储空间赋上缺省的初值，如左表所示;
  - int a;　　//a有初始值
  - 在成员变量声明时赋的初值;
  - int a = 10;　　//a赋初值10
  - 在创建实例时，由构造器的赋初值。
  - Point(int a, int b){
  - x = a;
  - y = b;
  - }

类型	值
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
boolean	false
引用类型	null

VariableInitTest.java

第4章 类和对象

面向对象程序设计

类的创建

方法过载

类的构造方法

类成员和实例成员

对象创建与使用

static 关键字

- 一旦将成员设为static，数据或方法就不会同那个类的任何对象实例联系到一起
- 即使从未创建那个类的一个对象，仍能调用一个static方法，或访问一些static数据
- 由于static 方法不需要创建任何对象，所以它们不可简单地调用其他那些成员

类成员

- 有static修饰的成员，也可以称为静态成员；

  例：static int a;

- 类成员：可以用类名直接访问：

  <类名>.<成员名>

  例：Point.a = 10;

- 类成员也可以用对象名来访问，但是所该类的所有对象都共享类成员变量。

```
Point p = new Point();
Point.a = 10;
p.a = 100;
```

- 类成员提供了事实上的全局变量和全局方法

StaticDemo.java

实例成员

- 通过创建实例才能访问和使用的成员；
  - 创建对象，由对象访问。例：p.x = 10;
- 定义时无static修饰的成员；

例：class Point{

int x, y; //x,y为实例成员变量

}

- 实例成员不能用类名直接访问。
  - 例：Point.x 访问是不合法的。

ClassInstMember.java StaticTest.java

实例成员和类成员

- 声明为static的方法有以下几条限制：
  - 它们仅能调用其他的static方法。
  - 它们只能访问static数据。
  - 它们不能以任何方式引用this或super
- static块*
  - 如果你需要通过计算来初始化你的static变量，你可以声明一个static块
  - static块仅在该类被加载时执行一次

UseStatic.java

第4章 类和对象

面向对象程序设计

类的创建

方法过载

类的构造方法

类成员和实例成员

对象创建与使用

对象、实例和引用

对象：

- 是一个变量，它的存储开销是一个地址存储单元。

实例：

- 是在存储空间分配的存储堆。是垃圾回收的目标。

引用：

- 对象通过它存储的实例的起始地址对实例实现访问。

对象创建与使用

- 对象的生命周期：
  - 创建、使用和销毁三个阶段。
- 创建：分两步。
  - 声明：

    Point p;
  - 实例化：

    通过new运算符调用构造器，通过赋值=对这个实例引用。

    p = new Point();

对象创建与使用

- 对象的生命周期：
  - 对象使用
  - 通过对象名对成员的访问：

    <对象名>•<成员>

    p.x = 100;

    int temp = p.getData();
  - 销毁：
    - 垃圾回收：实例开销的回收。
    - 由JVM自动完成。
    - 调用finalize()方法处理。

对象创建与使用举例

- 类Point有成员变量x和y，成员方法setData()

```
class Point{
   int x,y;
   Point(int a, int b){
     x = a;
     y = b;
   }
   Point(){
     this(0,0);
   }
   void setData(int x, int y){
     this.x = x;
     this.y = y;
   }
}
```

对象创建与使用举例

➢有类TestPoint：

```
public class TestPoint{
 public static void main(String[] arg){
     Point p1 = new Point();//对象p1创建
     Point p2 = new Point();//对象p2创建
     p1.setData(10,20);
     p2.setData(11,22);
     p2 = p1;
     p1 = null;
  }
}
```

MakeRectangle.java

对象创建与使用

- Java程序运行时系统通过垃圾收集，周期性地释放不再被对象引用所占用的内存，完成对象的清除
- finalize()方法
  - Java在对象作为垃圾被收集前，自动调用对象从Object类继承来的finalize()方法，清除自己所占用的资源。该方法声明的格式以下：

```
protected void finalize() throws throwable{
        ...              //clean up code for this class
      super.finalize();
}
```

Garbage.java

再论‘封装’

- OOP的最重要的好处之一是对数据和操作该数据的代码的封装，通过类来完成封装性
- 在创建一个类时，你正在创建一种新的数据类型，不但要定义数据的属性，也要定义操作数据的代码。
- 方法定义了对该类数据相一致的控制接口。
- 通过类的方法来使用类，而没有必要担心它的实现细节或在类的内部数据实际上是如何被管理的
- 既然细节被隐蔽，当需要时，它的内部工作可以被改变。只要你的代码通过类的方法来使用它，内部的细节可以改变而不会对类的外部带来负面影响。

TestStack.java

思考

- 面向对象程序设计的三个特性是什么？
- 如何用类的成员来表示对象的属性和行为的？举例说明。
- 类的成员变量是如何获得初始化的值的？
- 方法的数据传递方式有几种，它们各有什么特点？举例说明。
- 方法过载是指什么，创建方法时如何才能方法过载？举例说明。
- 实例成员和类成员有什么不同，它们如何使用？

作业1

- 创建一个类Point
  - 有成员变量x, y，它们都是int类型。
  - 有四个成员方法setX(int)、setY(int)、getPoint()和movePoint(int,int)。
    - setX(int)和setY(int)方法是设置成员变量x和y的值；
    - getPoint()方法则获得由x, y构成的坐标点；
    - movePoint(int,int)带两个int参数，用来修改x，y构成的坐标点。

作业2

- Point类有一个构造方法，不带参数，为x, y设置原点值。
- 另一个类为TestPoint，有一个main()方法，用来对Point类的实例进行测试。要求为其实例设置(0,0)坐标点，再移动到(10, 20)坐标点上，并输出实例调用相应的方法的结果