传智播客

《Java 基础入门》

教学设计

课程名称：Java 基础入门

授课年级：2014 年级

授课学期：2014 学年第一学期

教师姓名：某某老师

2014 年 02 月 09 日

<table>
<tr><td>课题名称</td><td>第 3 章 面向对象上</td><td>计划学时</td><td>6 课时</td></tr>
<tr><td>内容分析</td><td colspan="3">Java 是一种面向对象的语言，认识面向对象的编程思想对于 Java 学习至关重要。在面向对象中，有两个重要的概念，分别是类和对象，本课程将对类和对象的基本知识进行详细讲解，并结合程序学习如何使用面向对象的思想开发 Java 应用</td></tr>
<tr><td>教学目标及基本要求</td><td colspan="3">要求学生理解面向对象的程序设计思想，掌握类的设计、对象的创建、类的封装、构造方法的定义及其重载、this 和 static 关键字的使用以及单例设计模式，了解垃圾回收机制、内部类和文档注释的的使用</td></tr>
<tr><td>重点及措施</td><td colspan="3">教学重点：类的定义、对象的创建、类的封装、构造方法的定义和重载、this 和 static 关键字的使用、单例设计模式</td></tr>
<tr><td>难点及措施</td><td colspan="3">教学难点：类与对象的创建和使用、构造方法重载的规则、this 关键字和 static 关键字的使用、单例设计模式</td></tr>
<tr><td>教学方式</td><td colspan="3">教学采用教师课堂讲授为主，使用教学 PPT 讲解</td></tr>
<tr><td>教学过程</td><td colspan="3">
第一课时
（面向对象的概念、类与对象、类的定义、对象的创建与使用）
面向对象的概念
✧ 什么是面向对象
面向对象是一种符合人类思维习惯的编程思想。现实生活中存在各种形态不同的事物，这些事物之间存在着各种各样的联系。在程序中使用对象来映射现实中的事物，使用对象的关系来描述事物之间的联系，这种思想就是面向对象。
✧ 面向对象的好处
同面向过程进行比较，面向对象的好处是：
• 代码的复用性提高
• 使用者无须关心具体的实现细节
• 转变程序员的角色，更加符合人的思维习惯
✧ 面向对象的特征
• 封装性
• 继承性
• 多态性
类与对象的关系
✧ 用具体的例子引导。
例如：玩具和玩具模型的关系。
</td></tr>
</table>

在上面的图中，玩具模型可看作是一个类，一个个玩具可看作对象。玩具是由玩具模型创建出来的，同理，对象是根据类创建出来的，并且一个类可以创建多个对象。

✧ **总结**

类用于描述多个对象的共同特征，它是对象的模板。对象用于描述现实中的个体，它是类的实例。

类的定义

✧ **为什么定义类**

由于对象是虚拟出来的东西，是看不见摸不着的，我们需要在程序中使用对象，就必须找到描述对象的方式，定义一个类就可以解决这个问题。

✧ **类的定义**

将一系列特征相似的对象中的共同属性和方法抽象出来用一段特殊的代码来进行描述，这段特殊的代码我们就称之为一个类。类使用 class 关键字来进行定义，后面跟上类的名称。

✧ **定义一个类，并针对这个类进行讲解。**

```
class Person {
    int age;        // 定义 int 类型的变量 age
    // 定义 speak() 方法
    void speak() {
        System.out.println("大家好，我今年" + age + "岁!");
    }
}
```

其中，Person 是类名，age 是成员变量，speak()是成员方法。在成员方法 speak()中可以直接访问成员变量 age。

对象的创建与使用

✧ **对象的产生**

应用程序想要完成具体的功能，仅有类是远远不够的，还需要根据类创建实例对象。在 Java 程序中可以使用 new 关键字来创建对象，具体格式如下：

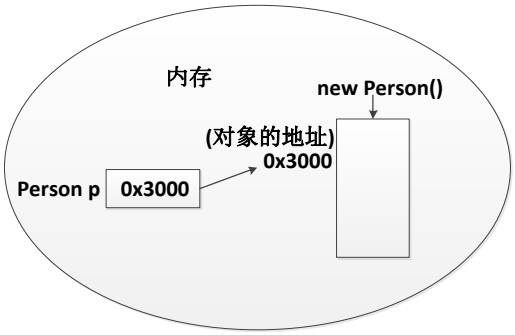
```
类名 对象名称 = new 类名();
```

例如，创建 Person 类的实例对象代码如下：

```
Person p = new Person();
```

其中“new Person()”用于创建 Person 类的一个实例对象，“Person p”则

是声明了一个 Person 类型的变量 p。中间的等号用于将 Person 对象在内存中的地址赋值给变量 p，这样变量 p 便持有了对象的引用。在内存中变量 p 和对象之间的引用关系如下图所示。



✧ **对象的使用**

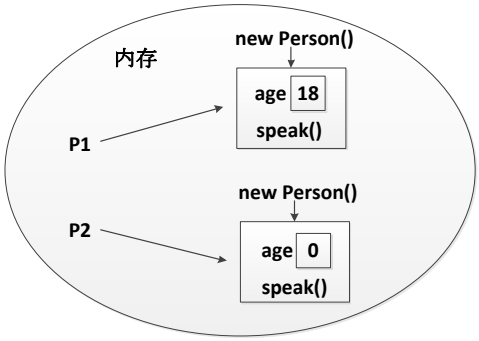
在创建 Person 对象后，可以通过对象的引用来访问对象所有的成员，具体格式如下：

```
对象引用.对象成员
```

通过一段代码来演示对象的使用情况。

```
Person p1 = new Person();
Person p2 =new Person();
p1.age = 18;
p1.speak();
p2.speak();
```

执行完毕后，对象在内存中的状态如图所示。



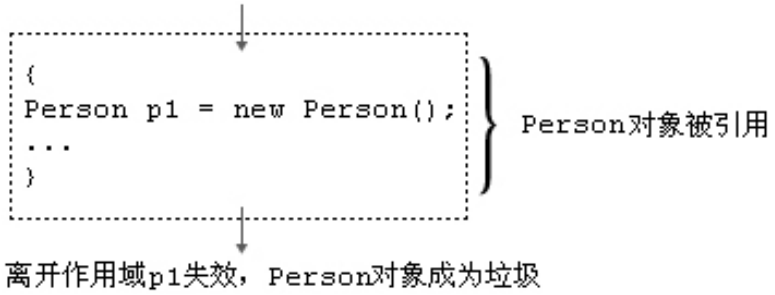
✧ **成员变量的初始化**

当一个对象被创建时，Java 虚拟机会对其中各种类型的成员变量自动进行初始化赋值。基本数据类型初始化值为 0，引用数据类型初始化值为 null，具体如下表所示。

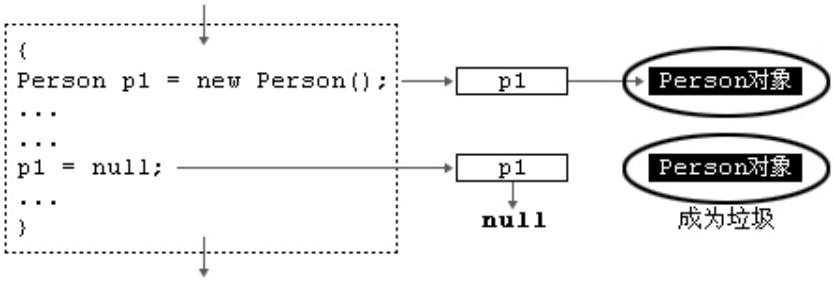
成员变量类型	初始值
byte	0
short	0
int	0
long	0L
float double	0.0F 0.0D
char boolean	'\u0000'（表示为空） False
All reference type	Null

✧ **对象的生命周期**

每个创建的对象都有自己的生命周期，对象只能在其有效的生命周期被使用。当没有任何变量引用这个对象时，它将成为垃圾对象，不能再被使用。对象变成垃圾的第一种情况：



对象变成垃圾的第二种情况：



第二课时

（类的设计、类的封装、构造方法的定义和重载、this 关键字）

类的设计

在 Java 中，对象是通过类创建出来的。因此，在程序设计时，最重要的就是类的设计。例如，要在程序中描述一个学校所有学生的信息，可以先设计一个学生类（Student），在这个类中定义两个属性 name、age 分别表示学生的姓名和年龄，定义一个方法 introduce()表示学生做自我介绍，具体代码如下：

```
public class Student {
    String name;
    int age;
    public void introduce() {
        // 方法中打印属性 name 和 age 的值
    System.out.println("大家好，我叫" + name + ",我今年" + age + "岁!");
    }
}
```

类的封装

✧ **为什么要封装类**

根据上述定义好的 Student 类创建对象，并访问对象成员，如下所示：

```
public class Example03 {
    public static void main(String[] args) {
        Student stu = new Student(); // 创建学生对象
```

```
            stu.name = "李芳";                    // 为对象的 name 属性赋值
            stu.age = -30;                        // 为对象的 age 属性赋值
            stu.introduce();                      // 调用对象的方法
        }
    }
```

程序的运行结果如下：

当把程序中的年龄赋值为一个负数-30 时，虽然程序不会报错，但在现实生活中明显是不合理的。为了解决年龄不能为负数的问题，在设计一个类时，应该对成员变量的访问作出一些限定，不允许外界随意访问。这就需要实现类的封装。

✧ **什么是类的封装**

所谓类的封装是指在定义一个类时，将类中的属性私有化，即使用 private 关键字来修饰，私有属性只能在它所在类中被访问，为了能让外界访问私有属性，需要提供一些使用 public 修饰的公有方法，其中包括用于获取属性值的 getXxx 方法和设置属性值的 setXxx 方法。

构造方法的定义

✧ **为什么需要构造方法**

实例化一个类的对象后，如果要为这个对象中的属性赋值，则必须要通过直接访问对象的属性或调用 setXxx 方法的方式才可以。如果需要在实例化对象的同时就为这个对象的属性进行赋值，可以通过构造方法来实现。

✧ **构造方法的定义**

- 函数的名称与类相同
- 没有返回值类型声明
- 不能在方法中使用 return 语句返回一个值

注意：没有返回值类型声明不等同于“void”，void 也是一种返回值类型声明，那就是没有返回值。

构造方法的重载

与普通方法一样，构造方法也可以重载，在一个类中可以定义多个构造方法，只要每个构造方法的参数类型或参数个数不同即可。

构造方法的一些细节

- 每一个类都至少有一个构造函数，如果在定义类时，没有显式地声明任何构造函数，系统会自动为这个类创建一个无参的构造函数，里面没有任何代码。
- 在定义构造方法时，如果没有特殊需要，都应该使用 public 关键字修饰。

this 关键字

✧ **this 关键字的作用**

可以解决成员变量与局部变量名称冲突的问题。

✧ **this 关键字的三种用法**

- 通过 this 关键字可以明确地去访问一个类的成员变量
- 通过 this 关键字调用成员方法。
- 构造方法是在实例化对象时被 Java 虚拟机自动调用的，在程序中不能像调用其它方法一样去调用构造方法，但可以在一个构造方法中使用“this([参数 1,参数 2…])”的形式来调用其它的构造方法。

✧ **使用 this 时需要注意的问题**

- 只能在构造方法中使用 this 调用其它的构造方法，不能在成员方法中使用。
- 在构造方法中，使用 this 调用构造方法的语句必须位于第一行，且只能出现一次。
- 不能在一个类的两个构造方法中使用 this 互相调用。

第三课时

（垃圾回收、静态变量、静态方法、静态代码块、单例模式）

垃圾回收机制

对象在没有任何引用时，它将成为垃圾。垃圾对象是不会被马上回收的，只有 JVM 检测到内存中的垃圾堆积到一定程度时才会回收。如果我们不希望等到这个时候回收，可以使用 System.gc()来人工回收垃圾。当一个对象在内存中被释放时，它的 finalize()方法会被自动调用。

static 关键字

static 关键字可以修饰类的成员，如成员变量、成员方法以及代码块等

静态变量

✧ **定义**

被 static 修饰的变量称为静态变量。

✧ **引用方式**

静态变量可以使用“类名.变量名”的方式访问

✧ **特点**

静态变量在类加载的时候就完成了初始化，它可以被所有实例所共享。

✧ **注意的问题**

static 关键字只能用于修饰成员变量，不能用于修饰局部变量。

静态方法

✧ **定义**

被 static 修饰的方法称为静态方法。

✧ **引用方式**

静态方法可以使用“类名.方法名”的方式访问。

✧ **注意的问题**

- 静态方法内部不能直接访问外部非静态的成员。
- 在静态方法内部，只能通过创建该类的对象来访问外部的非 static 的方

法。

- 在静态方法中，不能使用 this 关键字。

静态代码块

✧ **定义**

被 static 修饰的代码块称为静态代码块。

✧ **特点**

静态代码块在类加载的时候就执行了，它一般用于初始化类的成员变量。

单例模式

✧ **什么是设计模式**

针对某一问题的最佳解决方案，我们在程序中称之为设计模式。

设计模式是在大量的实践中总结和理论化之后优选的代码结构、编程风格、以及解决问题的思考方式。设计模式就像是经典的棋谱，不同的棋局，我们用不同的棋谱，免得我们自己再去思考和摸索。

✧ **单例设计模式的定义**

所谓类的单例设计模式，就是采取一定的方法保证在整个软件系统中，某个类只能存在一个对象实例，并且该类只提供一个取得其对象实例的方法。

✧ **单例设计模式的特征**

- 将构造方法私有化。
- 对外提供一个公有的 get 方法，让别人通过此方法来获得实例。
- 由于不能创建对象，所以 get 方法必须静态，这样别人才能使用“类名.方法名”的方式访问。
- 需要提供一个静态变量记住一个实例，并且将该实例返回。
- 该实例应该私有，禁止外界进行修改。

✧ **单例模式的示例代码**

下面是一个实现了单例设计模式的程序。

```
class Single {
    // 自己创建一个对象
    private static Single INSTANCE = new Single();
    private Single() {}                          // 私有化构造方法
     // 提供返回该对象的静态方法
    public static Single getInstance() {
        return INSTANCE;
    }
}
```

第四课时

（成员内部类、静态内部类、方法内部类、Java 的帮助文档）

成员内部类

✧ **定义**

在类中定义的类称为成员内部类。在 Java 中，允许在一个类的内部定义类，这样的类称作内部类

✧ **引用方式**

外部类名.内部类名 变量名 =new 外部类名().new 内部类名();

✧ **成员内部类特点**

内部类可以直接访问外部类的成员，而外部类不能直接访问内部类的成员。

静态内部类

✧ **定义**

被 static 修饰的内部类称为静态内部类。

✧ **引用方式**

外部类名.内部类名 变量名 =new 外部类名.内部类名();

✧ **注意的问题**

- 非静态的内部类中不能声明静态的成员
- static 修饰的内部类中可以定义非 static 修饰的成员
- static 修饰的内部类中不能访问外部非 static 的成员

方法内部类

✧ **定义**

在成员方法中定义的类称为方法内部类。

✧ **引用方式**

内部类名 变量名 =new 内部类名();

✧ **注意的问题**

- 方法内部类只能在当前方法中使用
- 方法内部类不能访问方法中定义的局部变量，除非这个局部变量被声明为 final

Java 帮助文档

✧ **文档注释的格式**

文档注释以“/**”开始，以“*/”标志结束。

✧ **生成文档的命令**

javadoc -d . -version -author Person.java

✧ **文档注释中某些特殊的标记说明**

@author：用于对类的说明，表示这个程序的作者

@version：用于对类说明，表示这个程序的开发版本号

@param：用于对方法的说明，表示方法上定义的参数以及参数对应的说明

@return：用于对方法的说明，表示方法的返回值代表的意义

✧ **JDK 帮助文档的分类**

- Oracle 公司官方发布的 HTML 格式的 JDK 帮助文档，可以从 Oracle 公司的官方网站 http://www.oracle.com 下载。
- Java 爱好者根据官方文档制作而成的 CHM 格式的 JDK 帮助文档，它具有独特的搜索功能和不同的语言版本，被许多开发者所钟爱。

第五课时

上机练习（总结，测试题）

1、 总结本章内容

2、 通过题库发放相关测试题，检查学生掌握情况。

上机练习主要针对本章中需要重点掌握的知识点，以及在程序中容易出错的内容进行练习，通过上机练习可以考察同学对知识点的掌握情况，对代码的熟练程度。

上机一：（考察知识点为类的定义）

请按照以下要求设计一个 Student 类。

要求如下：

1） Student 类中定义两个成员变量 name 和 age，分别表示学生的姓名和年龄，其中，变量 name 是 String 类型，初始值为“张三”，变量 age 是 int 类型，初始值为 19。

2） Student 类中定义一个成员方法 speak()，表示学生说话的行为，在方法中访问 name 和 age 两个成员变量，输出学生的姓名和年龄。

上机二：（考察知识点为类的封装）

请按照以下要求设计一个 Student 类。

要求如下：

1） 针对上机一中的 Student 类进行修改，使用封装的方式，将 name 属性和 age 属性使用 private 关键字修饰为私有属性，并对外提供公有的 getName()、setName(String n)、getAge()和 setAge(int a)方法。

2） 在 setAge(int a)方法中对传入的参数进行检查，如果传入的参数为负数，则输出“设置的年龄不合法”，如果不为负数，则输出设置的 age 值。

3） 定义一个测试类，在 main()方法中创建 Student 对象，并调用对象的 setName(String n)和 setAge(int a)方法来设置的 name 属性和 age 属性值，然后并调用 speak()方法输出相应信息。

上机三：（考察知识点为构造方法重载）

请按照以下要求设计一个 Student 类。

要求如下：

1） 在 Student 类中定义三个重载的构造方法，一是无参的构造方法，二是接收一个 String 类型的构造方法，该方法用于为 name 属性赋值，三是接收两个参数的构造方法，该方法用于为 name、age 属性赋值。

2） 定义一个测试类，在 mian()方法中创建三个 Student 对象，并在创建对象时为 name、age 属性赋值。

第六课时
上机练习（测试题）

上机一：（考察知识点为单例模式）

请按照以下要求设计一个单例模式的 Singleton 类。

要求如下：

1） 为了确保外界不能创建该类的实例对象，需要将该类的构造方法设置私有，即使用 private 关键字修饰。

2） 由于外界不能创建实例对象，我们只有在定义的类中创建该类的实例对象，并且需要定义一个静态变量 INSTANCE 引用此实例对象，为了让外界不能通过 Singleton.INSTANCE 的方式访问该变量，还需要将该变量使用 private 关键字修饰。

	3） 为了让外界使用 INSTANCE 实例对象，还需提供一个静态方法将这个实例对象返回。 **上机二：（考察知识点为方法内部类）** 请按照以下要求设计一个 Outer 类。 要求如下： 1） 定义一个外部类 Outer，并在该类中定义一个静态内部类 Inner。 2） 在内部类中定义一个静态变量 staticField 和一个静态方法 staticMethod()，并将该变量的值设置为"静态内部类的静态变量"，在该方法中输出"静态内部类的静态方法"。 3） 定义一个测试类，在 main()方法中输出 staticField 的值，并且调用静态方法 staticMethod()。
思考题和习题	见教材第三章后的习题
教学后记	