

Java面向对象编程

本节概要



类和对象



继承



多态

类和对象

编程范式

编程是 程序员 用特定的语法+数据结构+算法 组成的代码来告诉计算机如何执行任务的过程。

一个程序是程序员为了得到一个任务结果而编写的一组指令的集合，实现一个任务的方式有很多种不同的方式，对这些不同的编程方式的特点进行归纳总结得出来的编程方式类别，即为编程范式。不同的编程范式本质上代表对各种类型的任务采取的不同的解决问题的思路，大多数语言只支持一种编程范式，当然也有些语言可以同时支持多种编程范式。两种最重要的编程范式分别是面向过程编程和面向对象编程。

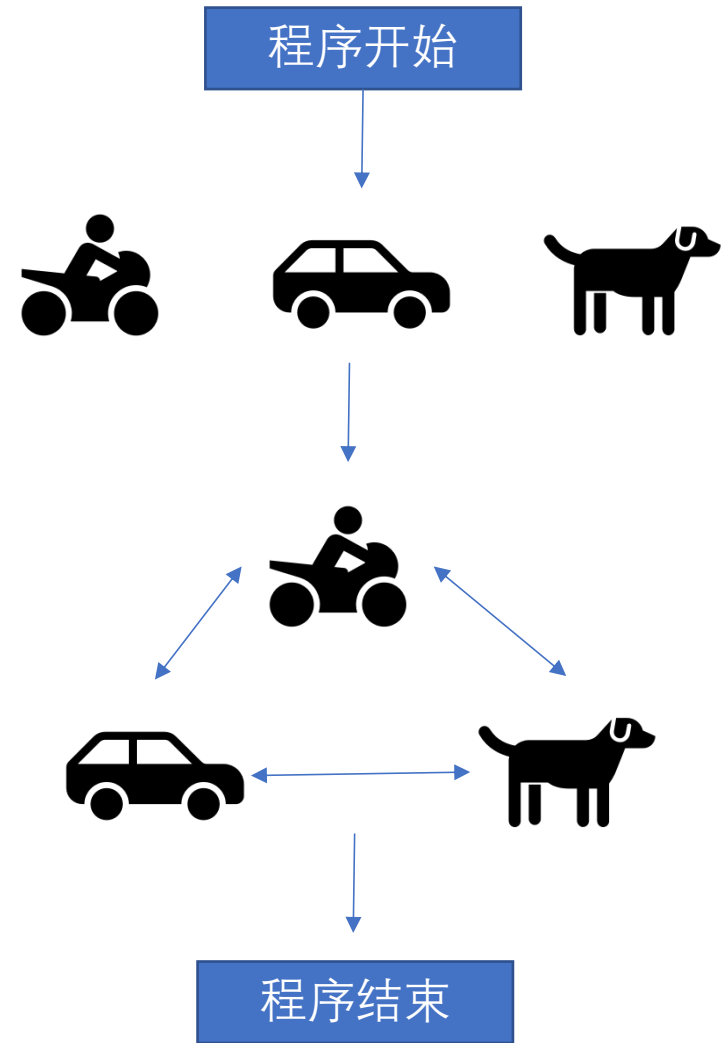
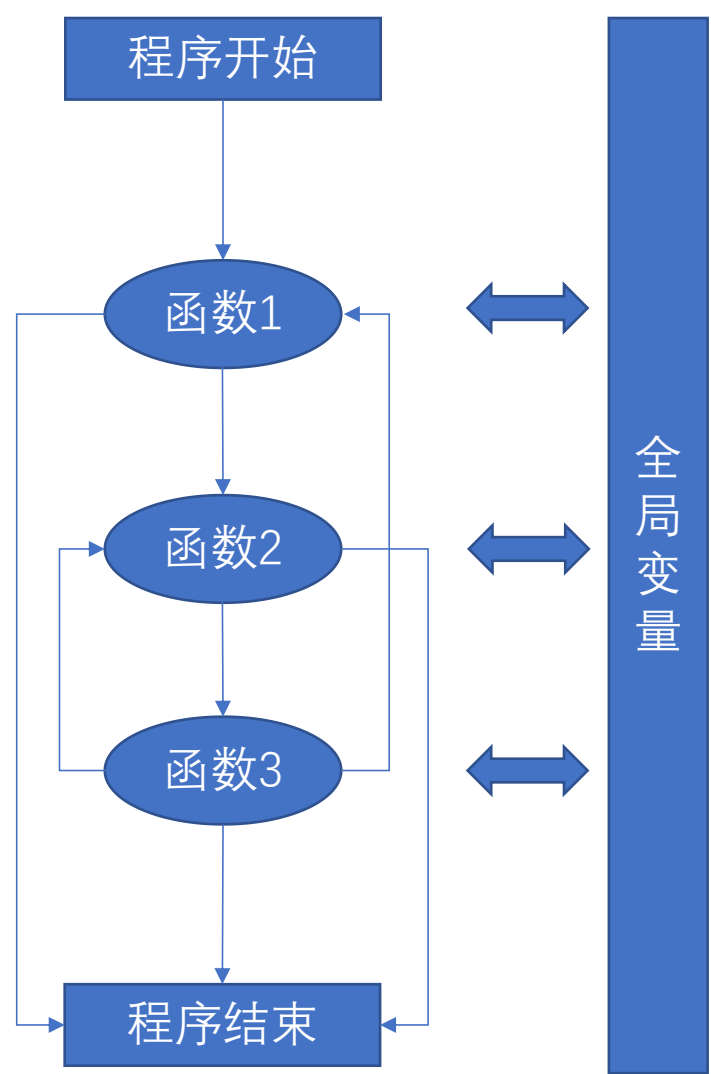
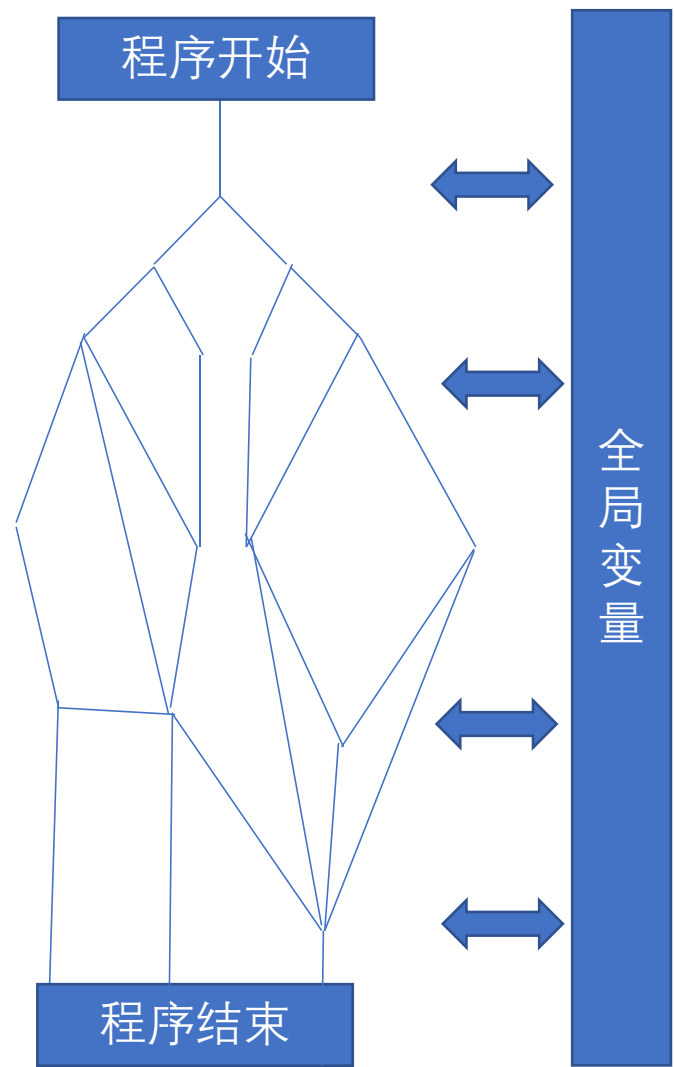
面向过程编程(Procedural Programming)

- 适用于处理简单的一次性任务，一个大问题分解成很多个小问题或子过程，这些子过程再执行的过程再继续分解直到小问题足够简单到可以在一个小步骤范围内解决。
- 典型代表：工厂流水线。
- 变量与数据分离。
- 个人视角，想做某件事件，每一步都要通过程序定义出来，写死了，在这个程序里，你只被设定做一事件，新增功能，或修改原有功能，将导致整个程序的逻辑都得更改，用面向过程的方式写代码，那你care的就是整个事情的执行过程。

面向对象编程(Object Oriented Programing)

- 处理需要不断迭代、维护或交互的复杂任务。先把世界按物种、样貌、有无生命等各种维度分类，然后给每类东西建模型，再让其在不脱离你模型定义的框架下，自我繁衍、交互、发展。
- 典型代表：大型网游。
- 变量与方法结合。
- 上帝视角，你现在要创世纪，把这么多人、动物、山河造出来，先造模子（定义类），再一个个复制（类实例化）模子定义了人这个物种所具备的所有特征（属性、行为）。

编程范式



一切都是对象

- 现实世界中万物皆对象
- 现实中的对象都具有外观长相（状态）和**行为举措**

对象是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位。一个对象由一组属性和对这组属性进行操作的一组服务组成。从更抽象的角度来说，对象是问题域或实现域中某些事物的一个抽象，它反映该事物在系统中需要保存的信息和发挥的作用；它是一组属性和有权对这些属性进行操作的一组服务的封装体。**客观世界是由对象和对象之间的联系组成的。**



FrameWork 框架

一切都是对象

◆ 面向对象编程与面向过程编程

- 面向对象是相对面向过程而言
- 面向对象和面向过程都是一种思想
- 面向过程
 - 强调的是功能行为
- 面向对象
 - 将功能封装进对象，强调具备了功能的对象。
- 面向对象是基于面向过程的。

面向对象编程

- 是一种符合人们思考习惯的思想
- 可以将复杂的事情简单化
- 将程序员从执行者转换成了指挥者
- 完成需求时：
 - 先要去找具有所需的功能的对象来用。
 - 如果该对象不存在，那么创建一个具有所需功能的对象。
 - 这样简化开发并提高复用。

面向过程与面向对象的区别

面向过程	面向对象
数据与功能是分离。 会大量使用全局变量。 ■ 命名冲突问题。 ■ 并发冲突问题。	将数据与专门操作该数据的方法整合到一起。
简单易懂，适用于处理简单逻辑	各种封装继承，需要通盘阅读，支持处理复杂逻辑
可扩展性弱	可扩展性强

面向对象编程

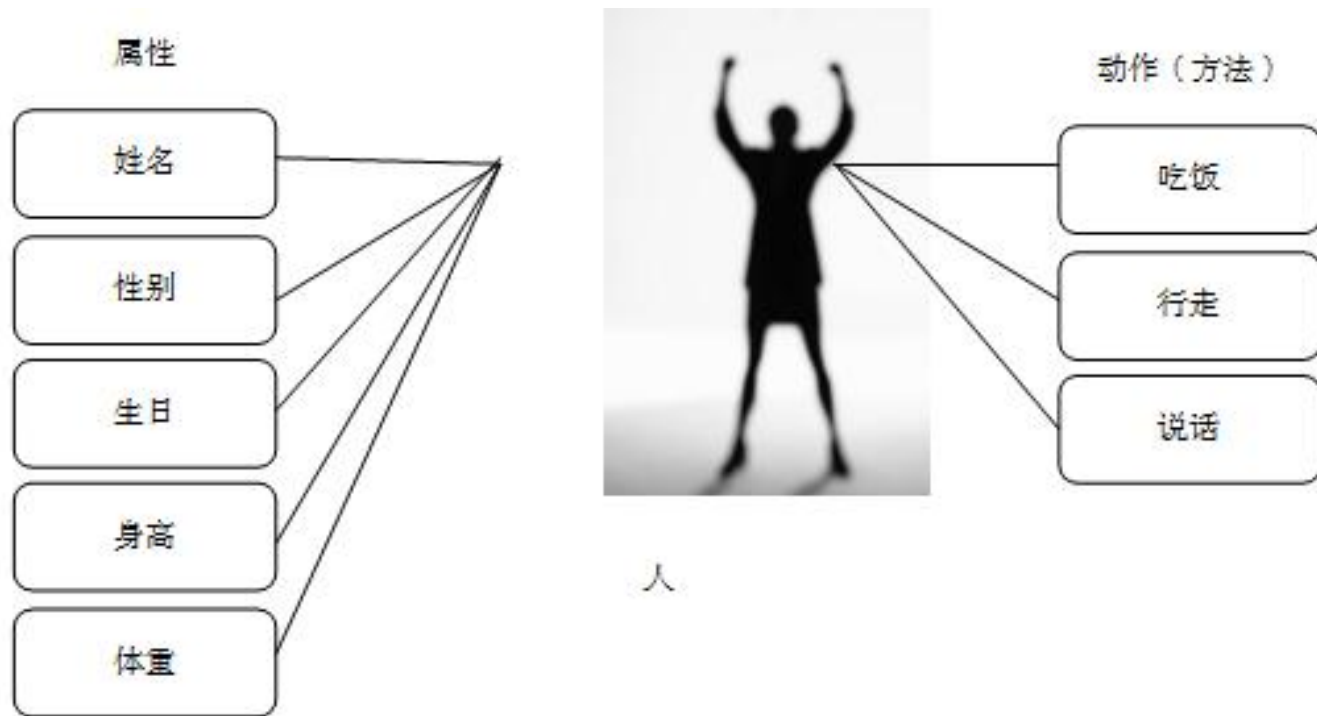
- 开发的过程：其实就是不断的创建对象，使用对象，指挥对象做事情。
- 设计的过程：其实就是在管理和维护对象之间的关系。
- 面向对象的特征：
 - 封装(encapsulation)
 - 继承(inheritance)
 - 多态(polymorphism)

面向对象编程

名词	解释
类	对一类拥有相同属性的对象的抽象、蓝图、原型、模板。在类中定义了这些对象的都具备的属性（variable）、共同的方法(melody)
属性	把属于同一类对象的特征用程序来描述，叫做属性，以人类为例，年龄、身高、性别、姓名等都叫做属性，一个类中，可以有多个属性。
方法	人类还能做好多事情，比如说话、走路、吃饭等，相比较于属性是名词，说话、走路是动词，这些动词用程序来描述就叫做方法。
实例(对象)	一个对象即是一个类的实例化后实例，一个类必须经过实例化后方可在程序中调用，一个类可以实例化多个对象，每个对象亦可以有不同的属性，就像人类是指所有人，每个人是指具体的对象，人与人之前有共性，亦有不同
实例化	把一个类转变为一个对象的过程就叫实例化

面向对象编程-基本理念

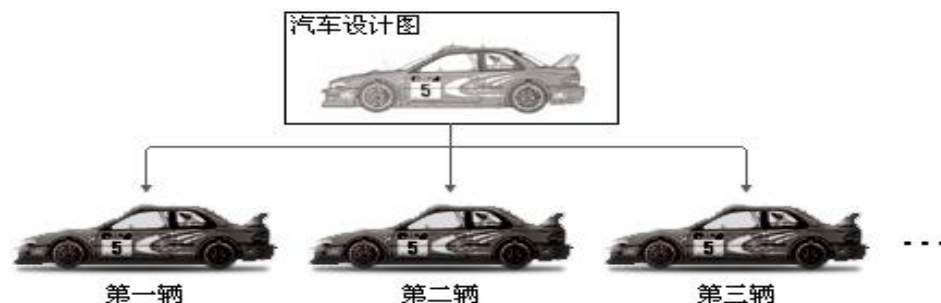
面向对象编程是Java采用的基本编程思想，它可以将属性和代码集成在一起，定义为类，从而使程序设计更加简单、规范、有条理。



在日常生活中，要描述一个事务，既要说明它的属性，也要说明它所能进行的操作。

类与对象的关系

- 使用计算机语言就是不断的在描述现实生活中的事物。
- java中描述事物通过类的形式体现，类是具体事物的抽象，概念上的定义。
- 对象即是该类事物实实在在存在的个体。



类与对象的关系就如设计图纸和产品的关系，
类的实例化结果就是对象，而对一类对象的抽象就是类

类的定义

■类的定义一般包含两个方面： 属性和方法

- 生活中描述事物无非就是描述事物的属性和行为。
 - 如：人有身高，体重等属性，有说话，打球等行为。
- Java中用类class来描述事物也是如此
 - 属性：对应类中的成员变量。
 - 行为：对应类中的成员函数。
- 定义类其实在定义类中的成员(成员变量和成员函数)。

■ 类的例子： 钟表

- 数据（状态）
int hour; int minute; int second;
- 函数（行为）
setTime(); showTime();

■ 类的例子： 人

数据

char *name; char *gender; int age; int id;

行为

生物行为

eat(), step(),...

社会行为

work(), study()14

类的定义

◆ 类定义的格式

```
[public | protected | private] [abstract | final] class 类名称  
    [extends 父类名称] [implements 接口名称列表]  
{  
    数据成员定义及初始化;  
    方法定义;  
}
```

public: 访问控制

abstract : 抽象类 (abstract class)

不完全的类。含有抽象函数 (abstract method) 的类必须是抽象类。

final: 终结类 (final class)

终结类的定义是完全的, 这种类不能生成子类。

类的成员变量

◆ 类的成员变量和程序中的局部变量不同

■ 成员变量：

- 成员变量定义在类中，在整个类中都可以被访问。
- 成员变量随着对象的建立而建立，存在于对象所在的堆内存中。
- 成员变量有默认初始化值。

■ 局部变量：

- 局部变量只定义在局部范围内，如：函数内，语句内等。
- 局部变量存在于栈内存中。
- 作用的范围结束，变量空间会自动释放。
- 局部变量没有默认初始化值。

类的声明和定义

- ◆ 以下代码定义了一个Student类，每个Student对象具备三个属性（成员变量）和一个行为（成员方法）

```
public class Student {  
    // 成员变量  
    public String name;    // 姓名  
    public String address; // 地址  
    public int age;        // 年龄  
    // 成员方法：显示学生名牌数据  
    public void printNameCard() {  
        System.out.println("姓名: " + name);  
        System.out.println("地址: " + address);  
        System.out.println("年龄: " + age);  
        System.out.println("-----");  
    }  
}
```

声明对象变量和对象的实例化

◆ 声明类之后，我们可以在程序中将这个类当作数据类型，来声明对象变量

```
Student joe, jane, current;
```

- 上述代码中，我们声明了三个Student类的joe、jane和current，称为对象变量。
- Java 对象变量的内容是对象的“引用”（Reference），可以大概类比为C语言中的指针（但实际不一样）
- 创建对象是分配一块内存空间来保存对象内容，而创建对象变量是创建一个能指向这个对象内容的引用，使得程序能够通过地址来访问该对象。
- 在我们上述的这句代码中，我们只是声明了三个对象变量，并没有创建对象。



声明对象变量和对象的实例化

◆ 声明对象变量之后，我们可以实例化一个对象，并将其引用保存在该对象变量中

```
Student joe = new Student();
```

声明对象变量

实例化对象

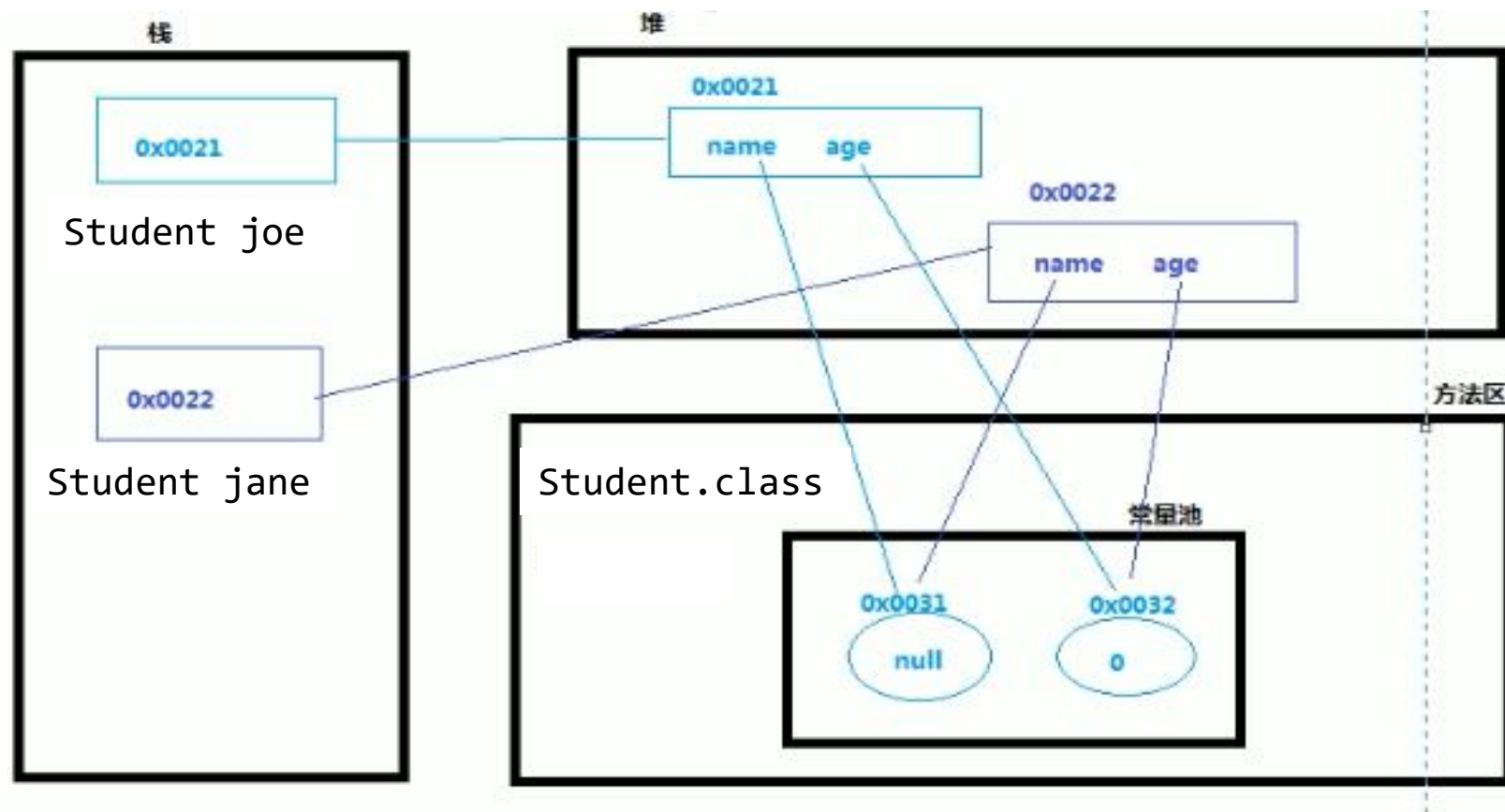
← 使用“类”这个模板来创建一个对象

在这句代码中，我们调用了默认的构造函数来实例化一个Student对象

◆ 然后，我们就可以指挥这个对象执行某个行动

```
joe.printNameCard();
```

声明对象变量和对象的实例化



类的构造函数

- ◆ 在前面的代码中，我们调用了构造函数Student()来实例化对象

我们写了这个函数吗？

Java编译器自动给没有构造函数的类生成一个构造函数

```
public Student(){ super(); }
```

调用父类的默认
构造函数

- ◆ 如果我们想自己定义类的实例化方法，该怎么办？（比如在初始化的时候就定义对象的属性）

类的构造函数

◆可以自己定义类的构造函数

■ 特点：

1. 函数名与类名相同
2. 不用定义返回值类型
3. 不可以写return语句


■ 作用：

给对象进行初始化。

■ 注意：

1. 默认构造函数的特点。
2. 多个构造函数是以重载的形式存在的。

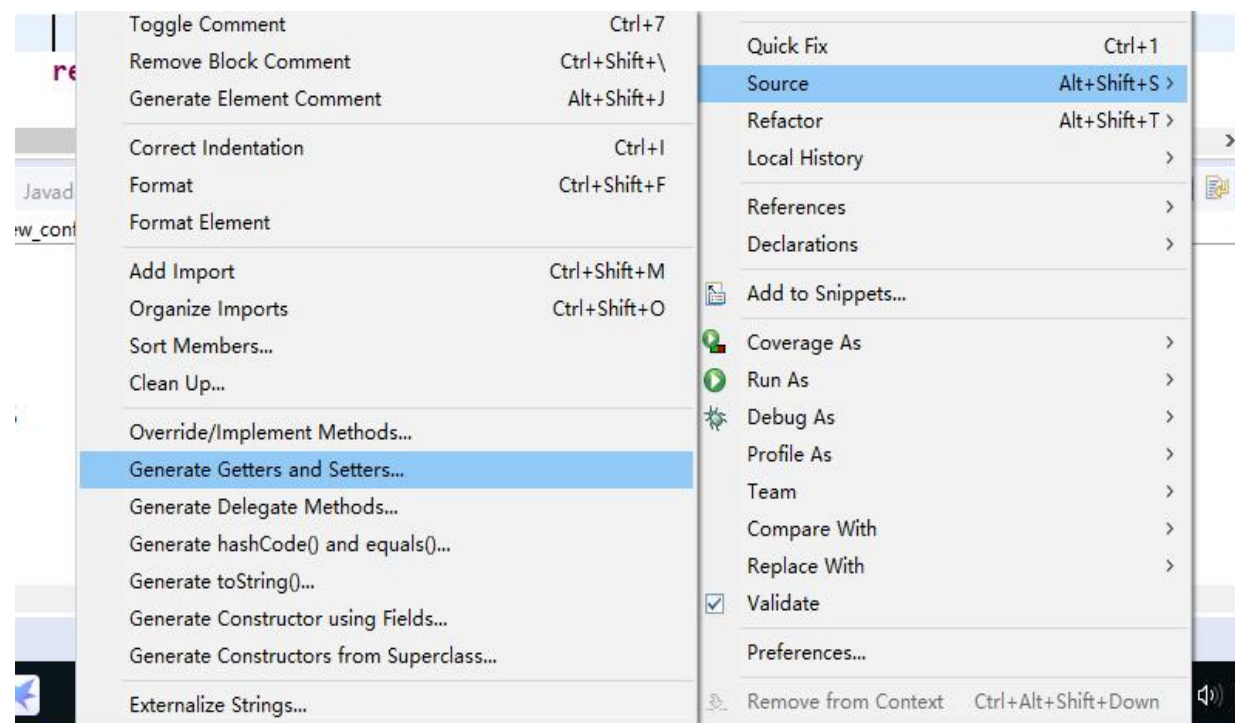
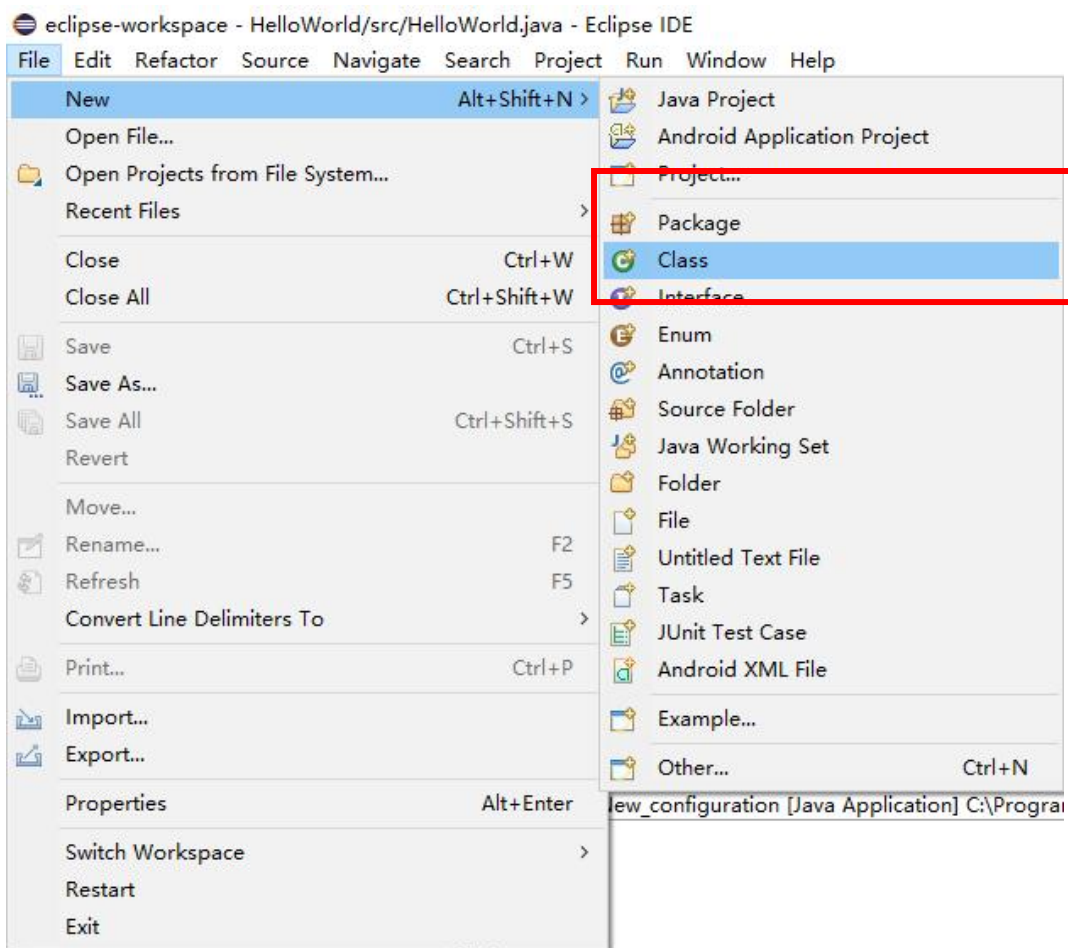
```
public class Student {  
    Student(String n, String add, int a){  
        this.name = n;  
        this.add = address;  
        this.age = a;  
    }  
    // 成员变量  
    .....  
    // 成员方法：显示学生名牌数据  
    .....  
}
```



- 特点：this代表其所在函数所属对象的引用。
- 换言之：this代本类对象的引用。

练一练

在Eclipse中如何快速生成类、属性读写

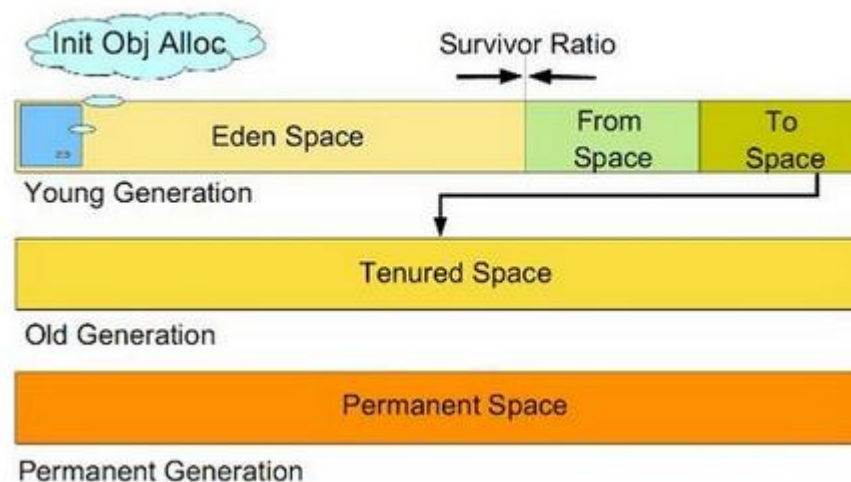


对象初始化和回收

- 系统在生成对象时，首先为对象开辟内存空间，并自动调用构造方法对实例变量进行初始化
- 对象不再使用时，系统会调用垃圾回收程序将其占用的内存回收
- 手动实现垃圾回收：

`System.gc();`

堆结构图如下：



类的访问控制

- 缺省类 (default class)

缺省类仅能被同一个包内的其他所有类访问。

- 公共类 (public class)

可以被随意地访问

类型	无修饰	public
同一包中的类	yes	yes
不同包中的类	no	yes

类成员的访问控制

◆ 类成员的访问控制包括类成员变量和成员函数（方法）

- 公有（public）
 - 允许所有访问。
- 保护（protected）
 - 允许以下的访问：
 1. 同一个包内的访问
 2. 其他包中的子类的访问
- 私有（private）
 - 仅允许所在类成员的访问
- 缺省（default）
 - 仅允许同一个包内的访问；又被称为“包（package）访问权限”

类成员的访问控制

类型	private	无修饰	protected	public
同一类	yes	yes	yes	yes
同一包中的子类	no	yes	yes	yes
同一包中的非子类	no	yes	no	yes
不同包中的子类	no	no	yes	yes
不同包中的非子类	no	no	no	yes

例程

◆ 类的声明

```
public class Student { // Student类声明
    // 成员变量
    private String name; // 姓名
    private String address; // 地址
    private int age; // 年龄
    // 成员方法：显示学生名牌数据
    public void printNameCard() {
        System.out.println("姓名: " + name);
        System.out.println("地址: " + address);
        System.out.println("年龄: " + age);
        System.out.println("-----");
    }
    // 成员方法：设置姓名数据
    public void setName(String n){ name = n; }
    // 成员方法：设置地址数据
    public void setAddress(String a){ address = a; }
    // 成员方法：设置年龄数据
    public void setAge(int v) { age = v; }
    // 成员方法：返回姓名
    public String getName(){ return name; }
    // 成员方法：返回月份
    public String getAddress(){ return address; }
    // 成员方法：返回年龄
    public int getAge(){ return age; }
}
```

思考：为什么要将属性
设置为private，然后定义
public的访问函数？

↓
封装的好处

例程

◆ 类的实例化和使用

```
public class Ch7_3_2 {  
    // 主程序  
    public static void main(String[] args) {  
        // 声明Student对象变量且创建对象  
        Student joe = new Student();  
        Student jane = new Student();  
  
        joe.setName("陈会安"); // 调用方法设置joe数据  
        joe.setAddress("朝阳区");  
        joe.setAge(37);  
        jane.setName("江小鱼"); // 调用方法设置jane数据  
        jane.setAddress("北京市");  
        jane.setAge(30);  
        jane.printNameCard(); // 调用方法显示学生数据  
        // 取得学生数据  
        String name = joe.getName();  
        String address = joe.getAddress();  
        int age = joe.getAge();  
        // 显示学生数据  
        System.out.println("[姓名]: " + name);  
        System.out.println("[地址]: " + address);  
        System.out.println("[年龄]: " + age);  
    }  
}
```

static(静态)关键字

◆ static关键字

用于修饰成员（成员变量和成员函数）

■ 被修饰后的成员具备以下特点：

- 随着类的加载而加载
- 优先于对象存在
- 被所有对象所共享
- 可以直接被类名调用

■ 使用注意

- 静态方法只能访问静态成员
- 静态方法中不可以写this，super关键字
- 主函数是静态的

└──────────→ 思考：为什么？？

方法的重载

◆Java的类允许拥有多个同名的方法，只需要参数不同即可，这种称为“重载”

含义：传递消息的差异即可让对象识别并执行不同的行为

```
class Date {    // Date类声明
    private int day;
    private int month;
    private int year;
    // 成员方法(1): 设置日期数据
    public void setDate(int d, int m, int y) {
        day = d;           // 设置日期
        month = m;         // 设置月份
        year = y;          // 设置年份
    }
    // 成员方法(2): 设置日期数据
    public void setDate(int d, int m) {
        day = d;           // 设置日期
        month = m;         // 设置月份
        year = 1975;       // 设置年份
    }
    // 成员方法(3): 设置日期数据
    public void setDate(long d, long m, long y) {
        day = (int) d;      // 设置日期
        month = (int) m;    // 设置月份
        year = (int) y;     // 设置年份
    }
}
```

```
    // 成员方法: 显示日期数据
    public void printDate() {
        System.out.println(month+"/"+day+"/"+year);
    }
}
// 主程序类
public class Ch7_3_4 {
    // 主程序
    public static void main(String[] args) {
        // 声明Date类类型的变量，并且创建对象
        Date myBirthday = new Date();
        Date oneBirthday = new Date();
        Date today = new Date();
        // 指定日期数据
        myBirthday.setDate(13, 9, 1970);
        oneBirthday.setDate(12, 6);
        today.setDate(5L, 10L, 2011L);
        System.out.print("我的生日: ");
        myBirthday.printDate(); // 调用对象的方法
        System.out.print("女友生日: ");
        oneBirthday.printDate();
        System.out.print("今日: ");
        today.printDate();
    }
}
```

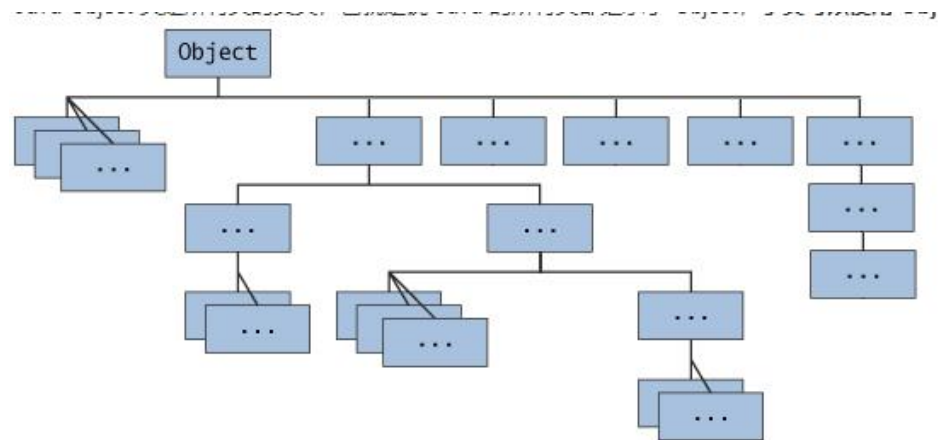
类的序列化

◆ toString()

Object类的toString()方法默认返回 类名@hsahcode

重载Object类的toString()方法

```
public String toString()  
{  
    return "这是Student类的对象, 姓名: "+this.name+", 年龄是: "+this.age+", 地址是: "+this.address;  
}
```



课堂练习 (1)

- 在同一包中新建Circle类及对应的java文件:
 - 包含成员变量：常量PI，私有变量x、y、r，分别为圆心的x、y坐标、圆半径r；
 - 包含至少4种构造函数Circle()、Circle(double r)、Circle(double x, double y)、Circle(double x, double y, double r)，实现重构函数的重载，注意为未出现在形参中的变量制定缺省值。
 - 包含成员函数setX()、setY()、setR(),分别用于设置圆属性，calArea()、calPerimeter()，分别用于计算圆的面积和周长；display()，用于打印该对象的x, y, r的值
 - 包含静态成员变量num，用于记录实例化的数量
 - 包含静态类函数printNum，用于打印已实例化的Circle对象数量。

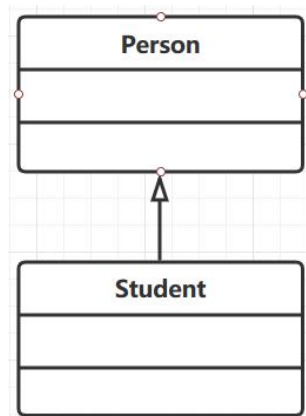
继承

类的层次结构

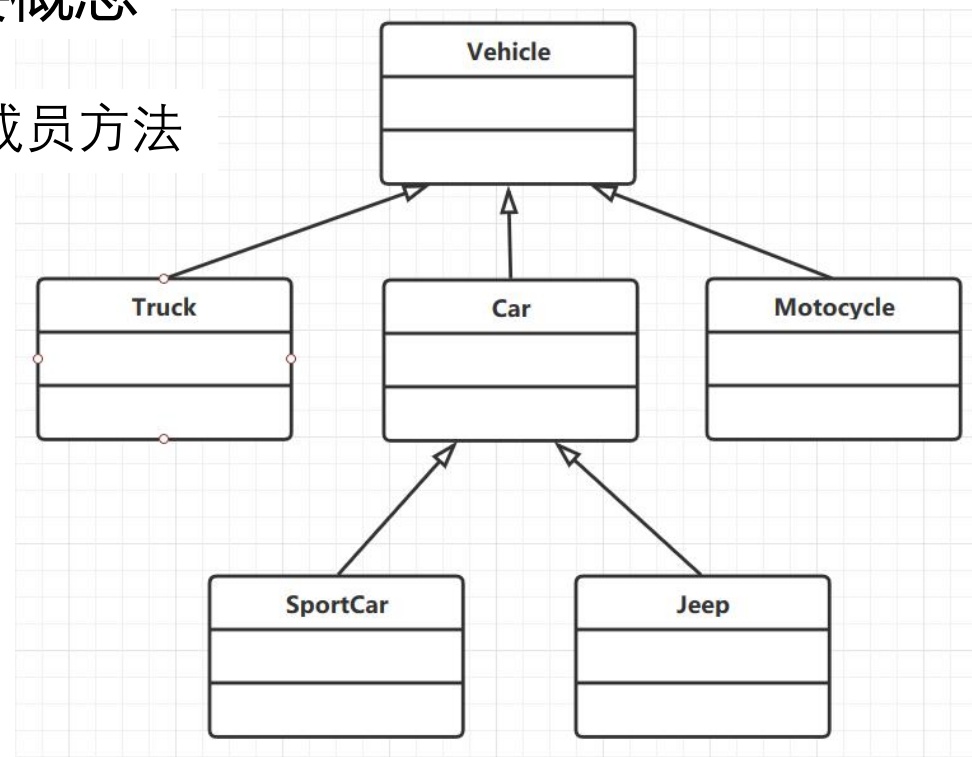
◆ “继承” 是面向对象程序设计的一个重要概念

声明类(子类)可以继承现存类(父类)的部分或全部成员方法

子类(subClass)对象
is a
父类(superClass)对象



例1



例2

◆ 在JAVA的世界里，一切类都是Object类的子类

类的继承

- ◆ 在声明子类之前，我们首先需要有一个父类可以继承。以Person和Student的继承关系为例，首先我们定义父类Person如下

```
class Person {  
    private int id;  
    private String name;  
    private double height;  
    public void setID(int id){}  
    public void setName(String n){}  
    public void setHeight(double h){}  
    public void personInfo(){}  
}
```

Person 类具备身份证号（id）、姓名（name）、身高（height）的成员变量和访问的成员方法

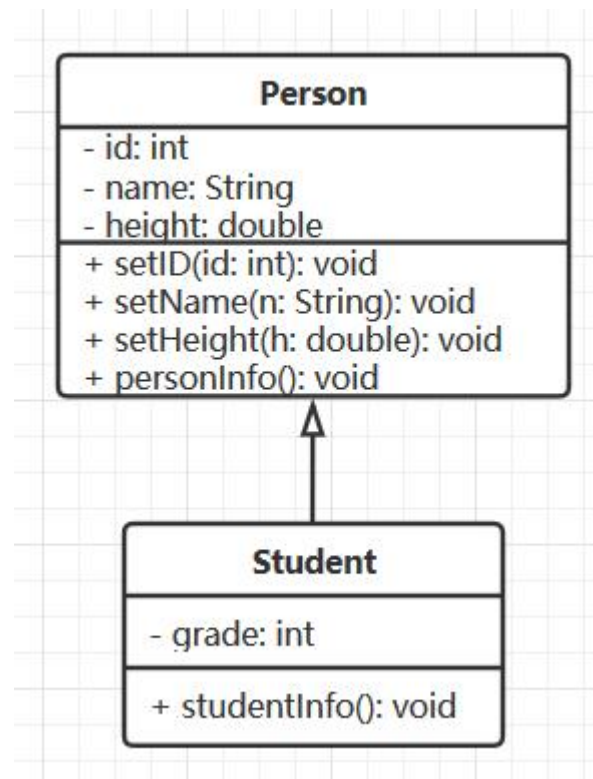
类的继承

◆ 然后，我们可以使用继承的语法（关键字 **extends**）来声明子类

```
class 子类名称 extends 父类名称 {  
    ..... // 额外的成员变量和方法  
}
```

■ Student 子类

```
class Student extends Person {  
    // 额外的成员变量  
    private int grade;  
    // 额外的方法  
    public Student(int id, String n,  
double h, int score){}  
    public void studentInfo(){  
}  
}
```



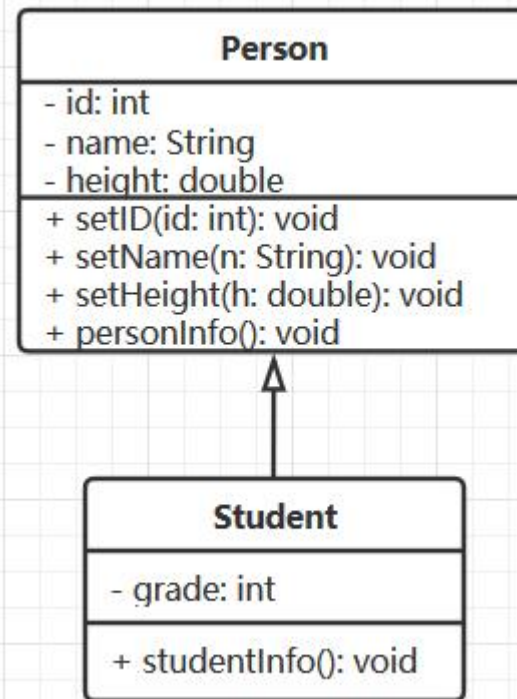
Student 子类中新增了一个成员变量`grade`，用于保存学生成绩；
一个成员方法，用于显示学生数据

类的继承

◆继承的访问限制

- 子类可以继承父类的所有成员变量和方法，但是具备访问限制：
 - 子类不能访问父类中声明为private的成员变量和方法
 - 父类的构造函数不属于类的成员，所以子类不能继承父类的构造函数，只能调用

在前面的例子中，Person类的成员变量id、name和height都是private的，子类Student只能通过父类的public方法对它们进行修改
例如：super.setID()



重写和隐藏父类的方法/成员变量

- ◆ 如果继承的父类方法不符合需求，在子类可以声明同名、同参数和返回值类型的方法来取代父类的方法，称为“重写” (override)
- ◆ 父类中的static方法在子类中需要用static方法来取代，称为“隐藏”，否则会报错

```
class Person {  
    .....  
    public static void printClassName() {}  
    .....  
    public void personInfo() {}  
}
```

```
class Student extends Person {  
    .....  
    public static void printClassName() {}  
    .....  
    public void personInfo() {}  
}
```

- ◆ 在上面的代码中，子类Student重写了父类的成员方法personInfo和隐藏了父类的static方法

类的继承与构造函数

◆ 观察下面的代码，运行会发生什么？

```
class Person {  
    public Person() {  
        System.out.println("A Person has been created!");  
    }  
}  
public class Student extends Person {  
    public Student() {  
        System.out.println("A Student has been created!");  
    }  
    public static void main(String[] args) {  
        Student student = new Student();  
    }  
}
```

◆ 为什么会这样？到底几个对象被创建了？

类的继承与构造函数

- ◆ Java 在创建类的对象时，会显式或隐式追溯调用父类的构造方法(一直到Object类为止，Object是所有Java类的最顶层父类)
- ◆ 可以使用super关键字显式地调用父类构造函数，或者使用this关键字显式调用子类构造函数

```
class Student
{
    public int num;//学号
    public int age;//年龄
    public Student(int num1,int age1)
    {
        num=num1;
        age=age1;
    }
    Student()
    {
        this(num1,age1);//第一行
    }
}
```

◆ 思考

- 为什么super(...)和this(...)调用语句不能同时在一个构造函数中出现？
- 为什么super(...)或this(...)调用语句只能作为构造函数中的第一句出现？

Object 类

◆hashCode ()

◆toString ()

◆对象的复制

```
Student student3=student2;  
student3.setID(2020010005);
```

◆对象间的比较

== 与 equals () 的差异

== 比较对象的内存地址

equals () 如果没有被重载就是比较内存地址， 如果有重载则根据具体实现来判断。

➤ 课后思考：如何实现深拷贝？

课堂练习 (2)

1. 设计以下的类并实现：

- **Employee**: 这是所有员工总的父类，
 - 属性：员工的姓名(String name)，员工的生日(Date birthday)。
 - 方法：double getSalary(int month) 根据参数月份来确定工资，如果该月员工过生日，则公司会额外奖励100元。
- **SalariedEmployee**: Employee的子类，拿固定工资的员工。
 - 属性：月薪(float monthSalary)
- **HourlyEmployee**: Employee的子类，按小时拿工资的员工，每月工作超出160小时的部分按照1.5倍工资发放。
 - 属性：每小时的工资(float hourSalary)、每月工作的小时数(float monthHours)
- **SalesEmployee**: Employee的子类，销售人员，工资由月销售额和提成率决定。
 - 属性：月销售额(float monthSales)、提成率(float saleRate)
- **BasePlusSalesEmployee**: SalesEmployee的子类，有固定底薪的销售人员，工资由底薪加上销售提成部分。
 - 属性：底薪(float baseSalary)。

2. 写测试的main函数，把若干各种类型的员工放在一个Employee数组里，并打印出某月每个员工的工资数额。

- 注意：要求把每个类都做成完全封装，不允许非私有化属性。

多态

重写和隐藏父类的方法/成员变量

◆ 请看下面的代码

```
public class Main {  
    public static void main(String[] args) {  
        Person p = new Student();  
        p.run(); // 应该打印Person.run还是Student.run?  
    }  
}  
class Person {  
    public void run() {  
        System.out.println("Person.run");  
    }  
}  
class Student extends Person {  
    public void run() {  
        System.out.println("Student.run");  
    }  
}
```

多态

- 多态是指，针对某个类型的方法调用，其真正执行的方法取决于运行时期实际类型的方法。对某个类型调用某个方法，执行的实际方法可能是某个子类的覆写方法。
- 假设我们定义一种收入，需要给它报税，那么先定义一个Income类：

```
class Income {  
    protected double income;  
    public double getTax() {  
        return income * 0.1; // 税率10%  
    }  
}
```

多态

- 对于工资收入，可以减去一个基数，那么我们可以从Income派生出SalaryIncome，并覆写getTax():

```
class Salary extends Income {  
    @Override  
    public double getTax() {  
        if (income <= 5000) {  
            return 0;  
        }  
        return (income - 5000) * 0.2;  
    }  
}
```

多态

- 如果你享受国务院特殊津贴，那么按照规定，可以全部免税：

```
class StateCouncilSpecialAllowance extends Income {  
    @Override  
    public double getTax() {  
        return 0;  
    }  
}
```


多态

- 如果你享受国务院特殊津贴，那么按照规定，可以全部免税：

```
class StateCouncilSpecialAllowance extends Income {  
    @Override  
    public double getTax() {  
        return 0;  
    }  
}
```

多态

- 现在，我们要编写一个报税的财务软件，对于一个人的3种收入进行报税，可以这么写：

```
public double totalTax(Income... incomes) {  
    double total = 0;  
    for (Income income: incomes) {  
        //依次累加三种收入的税  
        total = total + income.getTax();  
    }  
    return total;  
}
```

利用多态，totalTax()方法只需要和Income打交道，它完全不需要知道Salary和StateCouncilSpecialAllowance的存在

覆写Object方法

- 因为所有的class最终都继承自Object，而Object定义了几个重要的方法：
 - toString(): 把instance输出为String;
 - equals(): 判断两个instance是否逻辑相等;
 - hashCode(): 计算一个instance的哈希值。
- 在必要的情况下，我们可以覆写Object的这几个方法。

覆写Object方法

```
class Person {  
    ...  
    // 显示更有意义的字符串:  
    public String toString() {  
        return "Person:name=" + name;  
    }  
    // 比较是否相等:  
    public boolean equals(Object o) {  
        // 当且仅当o为Person类型:  
        if (o instanceof Person) {  
            Person p = (Person) o;  
            // 并且name字段相同时, 返回true:  
            return this.name.equals(p.name);  
        }  
        return false;  
    }  
    // 计算hash:  
    public int hashCode() {  
        return this.name.hashCode();  
    }  
}
```

覆写Object方法

```
class Person {  
    ...  
    // 显示更有意义的字符串:  
    public String toString() {  
        return "Person:name=" + name;  
    }  
    // 比较是否相等:  
    public boolean equals(Object o) {  
        // 当且仅当o为Person类型:  
        if (o instanceof Person) {  
            Person p = (Person) o;  
            // 并且name字段相同时, 返回true:  
            return this.name.equals(p.name);  
        }  
        return false;  
    }  
    // 计算hash:  
    public int hashCode() {  
        return this.name.hashCode();  
    }  
}
```

调用super

- 在子类的覆写方法中，如果要调用父类的被覆写的方法，可以通过super来调用。

```
class Person {
    protected String name;
    public String hello() {
        return "Hello, " + name;
    }
}

Student extends Person {
    @Override
    public String hello() {
        // 调用父类的hello()方法:
        return super.hello() + "!";
    }
}
```

面向对象编程-基本概念

在面向对象程序设计中，将事务的属性和方法都包含在类中，而对象则是类的一个实例。如果将人定义为类的话，那么某个具体的人就是一个对象。不同的对象拥有不同的属性值。

(1) 类 (class)：具有相同或相似性质的对象的抽象就是类。因此，对象的抽象是类，类的具体化就是对象。例如，如果人类是一个类，则一个具体的人就是一个对象。

(2) 对象 (Object)：面向对象程序设计思想可以将一组数据和与这组数据有关操作组装在一起，形成一个实体，这个实体就是对象。

(3) 封装：将数据和操作捆绑在一起，定义一个新类的过程就是封装。

(4) 属性：也称为成员变量，把属于同一类对象的特征用程序来描述，叫做属性，以人类为例，年龄、身高、性别、姓名等都叫做属性，一个类中，可以有多个属性。

(4) 方法：也称为成员函数，是指对象上的操作，作为类声明的一部分来定义。方法定义了对一个对象可以执行的操作。

(5) 构造函数：一种成员函数，用来在创建对象时初始化对象。构造函数一般与它所属的类完全同名。

(6) 析构函数：析构函数与构造函数相反，当对象脱离其作用域时（例如对象所在的函数已调用完毕），系统自动执行析构函数。析构函数往往用来做“清理善后”的工作。

课堂练习 (3)

1. Shape类为父类:

- 包含成员变量x, y, 分别为原点的x和y坐标;
- 包含静态变量num用来记录Shape对象的实例化数量;
- 包含构造函数, Shape()、Shape(double x, double y);
- 包含成员函数, public void display(), 打印原点的x, y坐标;
- 包含静态函数, public void printNum()用来打印num的值。
- 包含抽象函数, public abstract double calArea()。
- 包含抽象函数, public abstract double calPerimeter()函数。

2. 重载Shape, 实现两个子类: Rect, Circle

课堂练习 (4)

四、人狗大战：

1、定义2个类，Member、Game

- Member的属性 生命值(life), 最大攻击力(ability), 抽象行为函数(act)
- Member扩展出两个子类Dog、Man, Man的行为是 attack, Dog的行为是 bite, 各定义一个静态变量, 描述存货的成员数量。

2、在游戏一开始, 各生成 10个Dog对象和Man对象

- 人的生命为10、最大攻击力是1
- 狗的生命力是5、最大攻击力是2

3、在游戏的每个轮次里：

- 从人和狗中各随机选出 1个活着的对象进行PK
 - 利用Member提供的act函数
 - 伤害值为随机数, 取值范围: 0-该对象的最大攻击力。
- 一直到某一方的所有对象均死亡, 则输出剩余方信息, 游戏结束。
 - 根据Dog类和Man类的静态变量, 去判断是否均死亡。