JAVA WEB开发基础

第3讲：面向对象编程基础

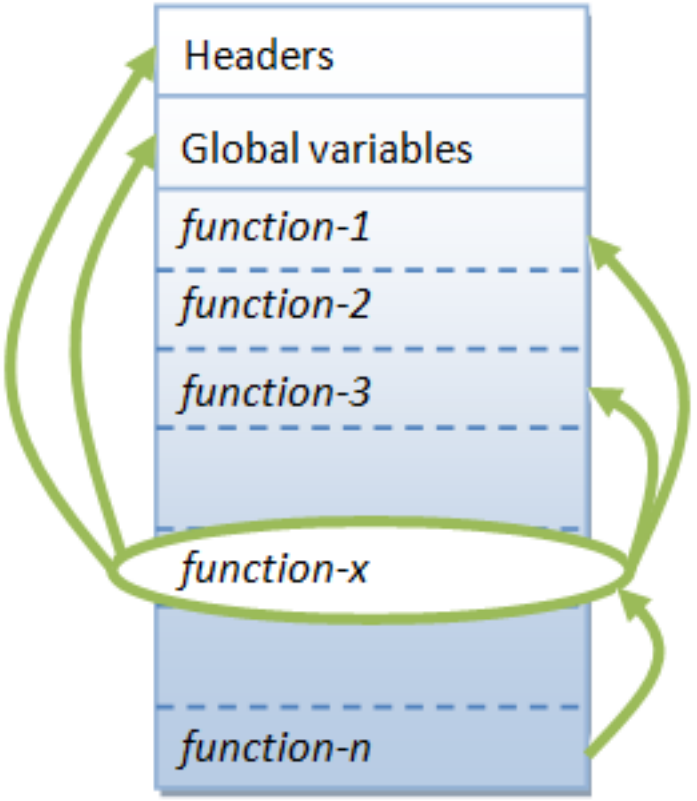
主讲人：康育哲

本讲内容

- 面向对象概述
- 类和实例
- 类成员
- 方法种种
- 关键字this

面向对象概述

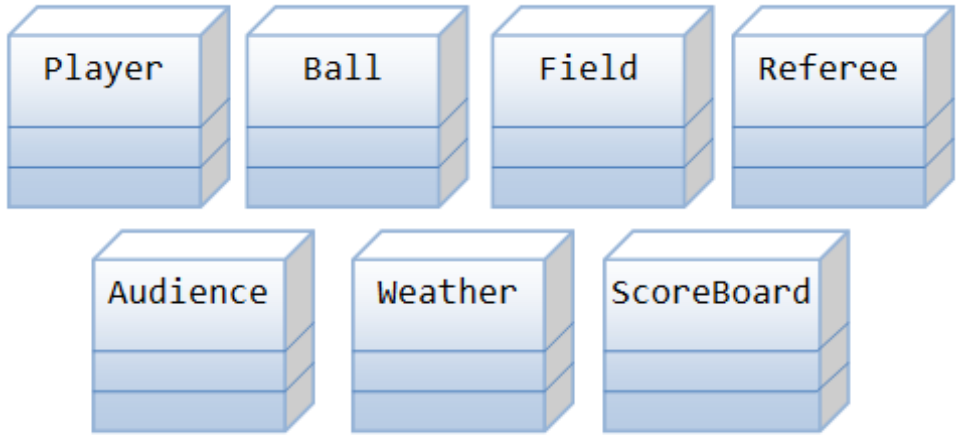
- 传统的面向过程编程
  - 代码复用性较差
    - 组件为函数，有可能引用全局变量和其他函数
  - 语言抽象度较低
    - 程序员不但需要关注业务，更需要关注底层计算机技术



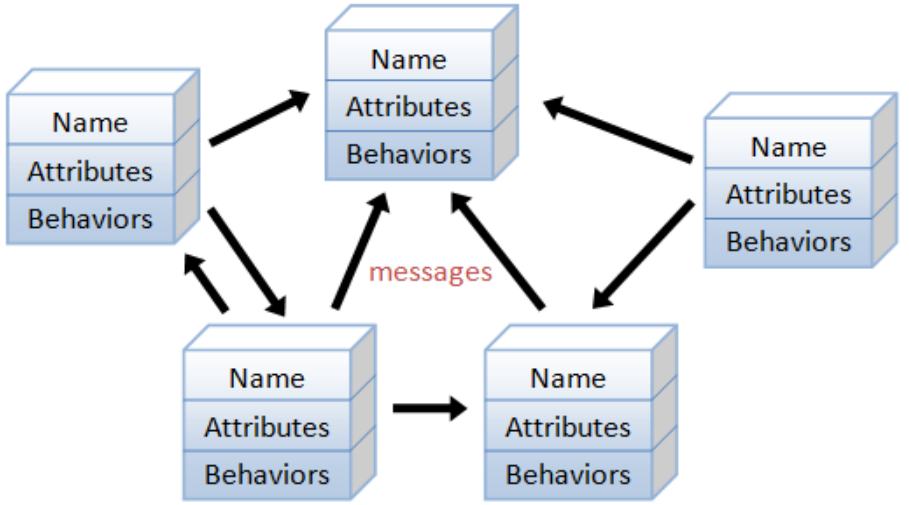
A function (in C) is not well-encapsulated

面向对象概述

- 面向对象编程
  - 类是基本单元，封装了静态属性和动态操作
  - 抽象度较高，封装了底层计算机技术
  - 示例：足球游戏



Classes (Entities) in a Computer Soccer Game



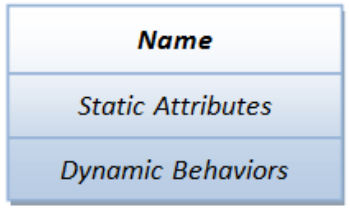
An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

面向对象概述

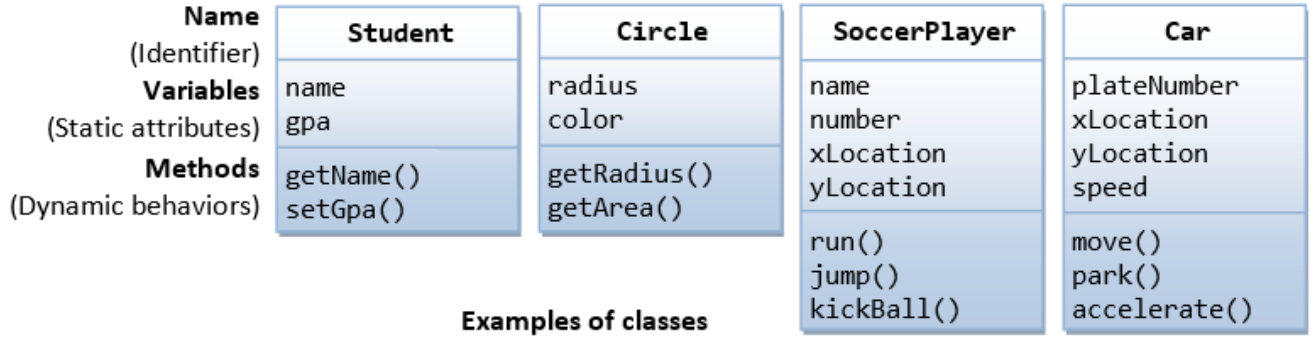
- 面向对象编程的优势
  - 软件设计复杂度降低：程序员关注业务多于关注技术
  - 软件维护成本降低：面向对象的代码易于理解、修改、测试
  - 软件复用：避免“再造轮子”

类和实例

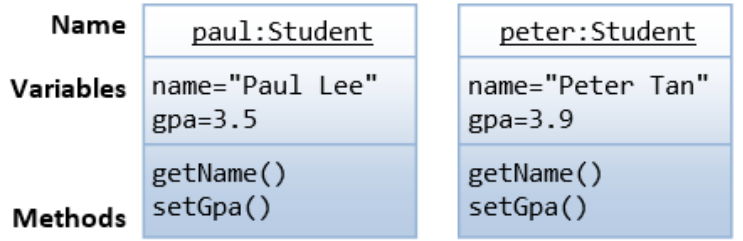
- 概念
  - 类（Class）：一组相同种类对象的定义
    - 图示：三要素
      - 类名（标识）
      - 变量（属性、字段、状态）
      - 方法（行为、操作、函数）
  - 实例（Instance）：类的一个特定对象的实现
  - 类与对象的UML图示
  - 类的小结
    - 类是由程序员定义的抽象的、可复用的软件实体，是对现实世界的模拟
    - 类的组成三要素：类名、变量、方法
    - 类是对数据结构和算法的封装



A class is a 3-compartment box



Examples of classes



Two instances - paul and peter - of the class Student

类和实例

- Java中的类定义
  - 关键字：class
  - 类命名规范
    - 单数名词或单数名词词组
    - 驼峰式命名
    - 选择有意义的名称
  - 示例

```
[AccessControlModifier] class ClassName {
   // Class body contains members (variables and methods)
   ......
}
```

```
public class Circle {          // class name
   double radius;              // variables
   String color;

   double getRadius() { ..... }  // methods
   double getArea() { ..... }
}
```

类和实例

- Java中的实例化
  - 用类名声明类型
  - 用new构造
  - 示例

```
// Declare 3 instances of the class Circle, c1, c2, and c3
Circle c1, c2, c3;  // They hold a special value called null
// Construct the instances via new operator
c1 = new Circle();
c2 = new Circle(2.0);
c3 = new Circle(3.0, "red");

// You can Declare and Construct in the same statement
Circle c4 = new Circle();
```

类和实例

- Java中的成员调用
  - 用实例名指定对象
  - 用.运算符调用成员（变量或方法）
  - 示例

```
// Suppose that the class Circle has variables radius and color,
//  and methods getArea() and getRadius().
// Declare and construct instances c1 and c2 of the class Circle
Circle c1 = new Circle ();
Circle c2 = new Circle ();
// Invoke member methods for the instance c1 via dot operator
System.out.println(c1.getArea());
System.out.println(c1.getRadius());
// Reference member variables for instance c2 via dot operator
c2.radius = 5.0;
c2.color = "blue";
```

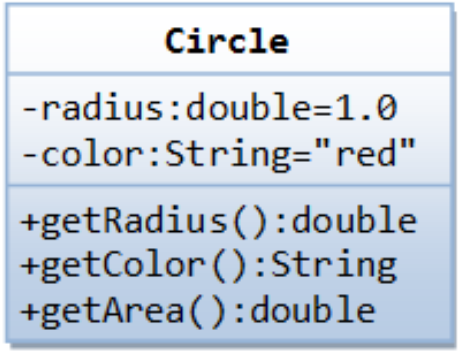
类成员

- 成员变量（Member Variables）
  - 三要素：名称、类型、初值
  - 命名规范
    - 名词或名词词组
    - 驼峰式命名
- 成员方法（Member Methods）
  - 四要素：名称、参数列表、返回值、函数体
  - 命名规范
    - 动词或动词词组
    - 驼峰式命名

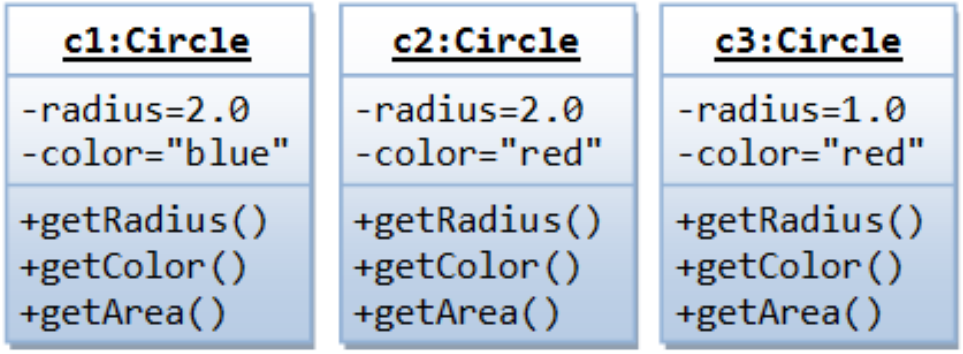
类成员

- 示例：Circle Class

Class Definition



Instances




```java
/*
 * The Circle class models a circle with a radius and color.
 */
public class Circle {    // Save as "Circle.java"
   // Private instance variables
   private double radius;
   private String color;

   // Constructors (overloaded)
   public Circle() {                    // 1st Constructor
      radius = 1.0;
      color = "red";
   }
   public Circle(double r) {            // 2nd Constructor
      radius = r;
      color = "red";
   }
   public Circle(double r, String c) { // 3rd Constructor
      radius = r;
      color = c;
   }

   // Public methods
   public double getRadius() {
      return radius;
   }
   public String getColor() {
      return color;
   }
   public double getArea() {
      return radius * radius * Math.PI;
   }
}
```

方法种种

- 构造方法（Constructor）
  - 用于构造类的实例
  - 方法名和类名相同
  - 无返回值，函数体内不能写return
  - 必须通过new调用
  - 可重载，不可继承
- 默认构造方法
  - 无参数的构造方法

方法种种

- 访问控制修饰符（Access Control Modifier）
  - 用于控制类或类成员的可见性
  - public：类/变量/方法对整个系统可见
  - private：类/变量/方法仅对类成员可见
  - UML标识：public用+表示，private用-表示
- 数据的封装和隐藏
  - 类数据通过public成员进行封装
  - 类数据通过private成员对外隐藏
- Getter/Setter
  - 对private成员变量的读写操作封装成public方法
  - 作用：数据封装

```
// Setter for color
public void setColor(String newColor) {
   color = newColor;
}

// Setter for radius
public void setRadius(double newRadius) {
   radius = newRadius;
}
```

方法种种

- toString()方法
  - 一个设计良好的Java类应该具有一个toString()方法
  - 返回值：String类型
  - 作用：把对象的关键数据以字符串形式展现出来，以便显示和调试
  - System.out.println()或String类的+运算符会显示调用对象的toString()方法

```
public String toString() { ..... }
```

```
Circle c1 = new Circle();
System.out.println(c1.toString());   // Explicitly calling toString()
System.out.println(c1);              // Implicit call to c1.toString()
System.out.println("c1 is: " + c1);  // '+' invokes toString() to get a String before concatenation
```

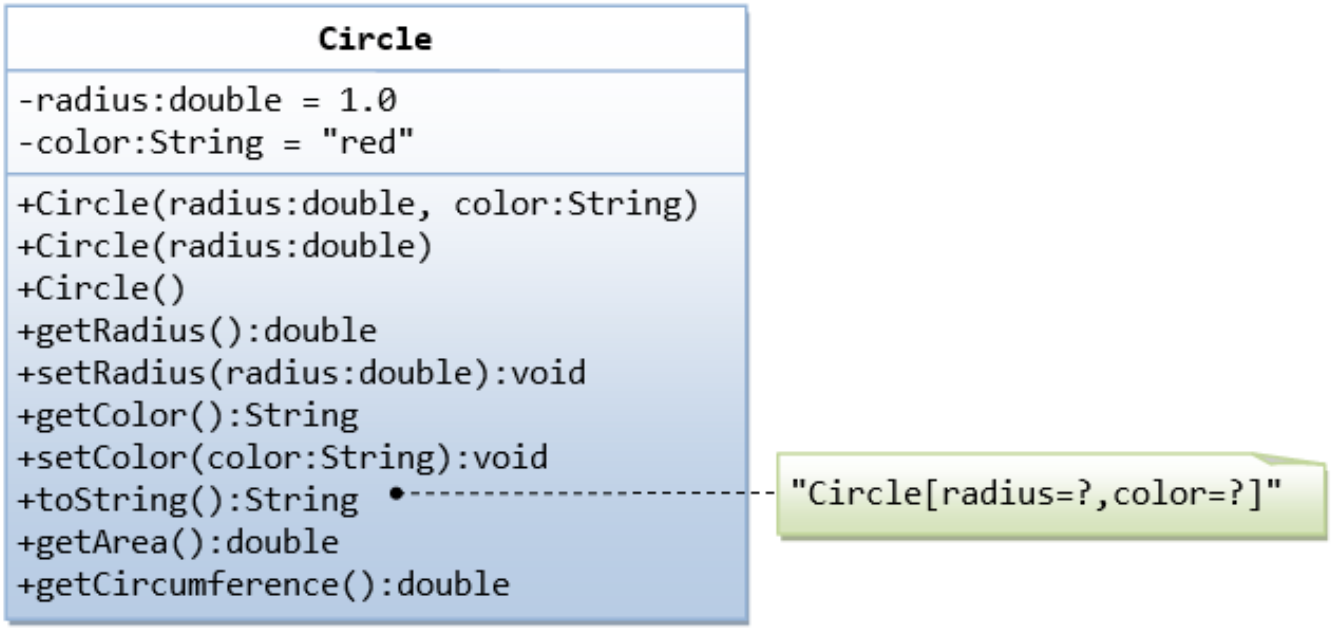
关键字THIS

- Java关键字this用来指代当前实例
- 作用：参数和成员变量同名时，避免歧义
- 在构造方法里，可以通过this调用另一个构造方法
- 在普通方法里，可以返回this，把实例自身返回给调用者

```
public class Circle {
   double radius;                  // Member variable called "radius"
   public Circle(double radius) { // Method's argument also called "radius"
      this.radius = radius;
         // "radius = radius" does not make sense!
         // "this.radius" refers to this instance's member variable
         // "radius" resolved to the method's argument.
   }
   ...
}
```

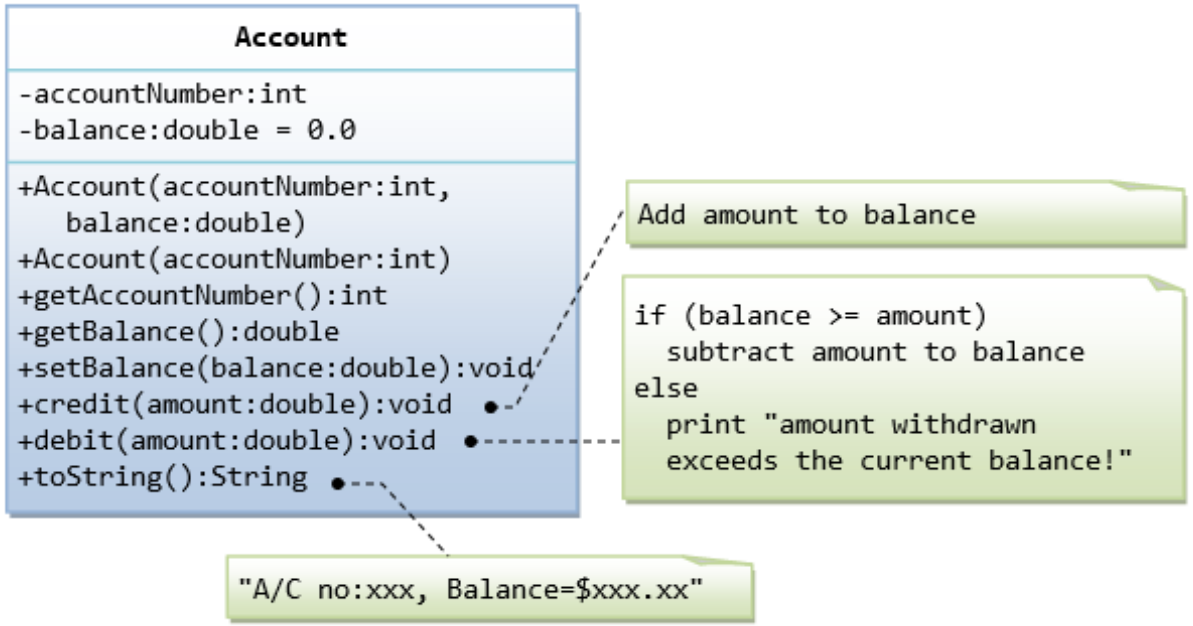
练习

- 根据下面的Circle类图完成该类的实现，并另外编写一个包含main方法的测试类TestCircle，通过打印的方式对Circle类的public成员进行输出测试。



练习

- 下面的Account类图表示银行账户的类模型，请完成类的实现，并另外编写一个含main方法测试类TestAccount，通过打印的方式对Account类的public成员进行输出测试。
  - 两个成员变量：accountNumber表示账号，balance表示余额
  - Getter&Setter：账号只可读不可改，故只提供getter方法；余额可读也可改
  - public方法credit()表示存钱，debit()表示取钱
  - toString()方法按格式把账户信息转为字符串



练习

- Student类图表示学生信息的类模型，请完成类的实现，并另外编写一个含main方法测试类TestStudent，通过打印的方式对Student类的public成员进行输出测试。
  - private成员变量：numCourses表示这个学生当前已修课程数量，courses和grades是两个并列的数组，前者表示课程代码，后者表示成绩，两个数组中的元素一一对应
  - name有Getter方法，address有Getter和Setter方法
  - 方法addCourseGrade(course, grade)表示添加一门已修的课程和成绩，已添加的课程不能重复添加，不必考虑数组超限的情况
  - 方法printGrades()按格式打印全部已修课程的成绩
  - 方法getAverageGrade()返回所有已修课程的平均成绩