

```

4   public C() {
5       System.out.println("C's no-arg constructor invoked");
6       this(0);
7   }
8
9   public C(int p) {
10      p = p;
11  }
12
13  public void setP(int p) {
14      p = p;
15  }
16 }

```

9.34 下面代码中哪里有错误?

```

public class Test {
    private int id;

    public void m1() {
        this.id = 45;
    }

    public void m2() {
        Test.id = 45;
    }
}

```

## 关键术语

action (动作)

anonymous object (匿名对象)

attribute (属性)

behavior (行为)

class (类)

class's variable (类变量)

client (客户)

constructor (构造方法)

date field (数据域)

data field encapsulation (数据域封装)

default constructor (默认构造方法)

dot operator (.) (点操作符)

getter (or accessor) (访问器 (读取器))

instance (实例)

instance method (实例方法)

instance variable (实例变量)

instantiation (实例化)

immutable class (不可变类)

immutable object (不可变对象)

no-arg constructor (无参构造方法)

null value (空值)

object (对象)

object-oriented programming (OOP) (面向对象程序设计)

package-private (or package-access) (包私有 (或包访问))

private constructor (私有的构造方法)

property (属性)

public class (公共类)

reference type (引用类型)

reference variable (引用变量)

setter (or mutator) (设置方法 (修改器))

state (状态)

static method (静态方法)

static variable (静态变量)

this keyword (this 关键字)

Unified Modeling Language (UML) (统一建模语言)

## 本章小结

1. 类是对象的模板。它定义对象的属性, 并提供用于以创建对象的构造方法以及操作对象的普通方法。

2. 类也是一种数据类型。可以用它声明对象引用变量。对象引用变量中似乎存放了一个对象，但事实上，它包含的只是对该对象的引用。严格地讲，对象引用变量和对象是不同的，但是大多数情况下，它们的区别是可以忽略的。
3. 对象是类的实例。可以使用 `new` 操作符创建对象，使用点操作符 (.) 通过对象的引用变量来访问该对象的成员。
4. 实例变量或方法属于类的一个实例。它的使用与各自的实例相关联。静态变量是被同一个类的所有实例所共享的。可以在不使用实例的情况下调用静态方法。
5. 类的每个实例都能访问这个类的静态变量和静态方法。然而，为清晰起见，最好使用“类名.变量”和“类名.方法”来调用静态变量和静态方法。
6. 可见性修饰符指定类、方法和数据是如何被访问的。公共的 (`public`) 类、方法或数据可以被任何客户访问，私有的 (`private`) 方法或数据只能在本类中被访问。
7. 可以提供 `get` (访问器) 方法或者 `set` (修改器) 方法使客户程序能够看到或修改数据。
8. `get` 方法具有方法签名 `public returnType getPropertyName()`。如果返回值类型 (`returnType`) 是 `boolean` 型，则 `get` 方法应该定义为 `public boolean isPropertyName()`。`set` 方法具有方法签名 `public void setPropertyName(dataType propertyValue)`。
9. 所有传递给方法的参数都是值传递的。对于基本类型的参数，传递的是实际值；而若参数是引用数据类型，则传递的是对象的引用。
10. Java 数组是一个可以包含基本类型值或对象类型值的对象。当创建一个对象数组时，它的元素被赋予默认值 `null`。
11. 一旦被创建，不可变对象 (`immutable object`) 就不能被改变了。为了防止用户修改一个对象，可以定义该对象为不可变类。
12. 实例变量和静态变量的作用域是整个类，无论该变量在什么位置定义。实例变量和静态变量可以在类中的任何位置定义。为一致性考虑，本书都在类的开始部分定义。
13. `this` 关键字可以用于引用进行调用的对象。它也可以用于在构造方法中来调用同一个类的另外一个构造方法。

## 测试题

在线回答本章节的测试题，位于 [www.cs.armstrong.edu/liang/intro10e/quiz.html](http://www.cs.armstrong.edu/liang/intro10e/quiz.html)。

## 编程练习题

🔧 教学提示：第 9 ~ 13 章的练习题要达到下面三个目标：

- 设计类并画出 UML 类图。
- 实现 UML 中的类。
- 使用类开发应用程序。

学生可以从配套网站上下载偶数题号练习题的答案，教师可以从同一个网站下载所有答案。

### 9.2 ~ 9.5 节

9.1 (矩形类 `Rectangle`) 遵照 9.2 节中 `Circle` 类的例子，设计一个名为 `Rectangle` 的类表示矩形。这个类包括：

- 两个名为 `width` 和 `height` 的 `double` 型数据域，它们分别表示矩形的宽和高。`width` 和 `height` 的默认值都为 1。
- 创建默认矩形的无参构造方法。
- 一个创建 `width` 和 `height` 为指定值的矩形的构造方法。
- 一个名为 `getArea()` 的方法返回这个矩形的面积。

- 一个名为 `getPerimeter()` 的方法返回周长。

画出该类的 UML 图并实现这个类。编写一个测试程序，创建两个 `Rectangle` 对象——一个矩形的宽为 4 而高为 40，另一个矩形的宽为 3.5 而高为 35.9。按照这个顺序显示每个矩形的宽、高、面积和周长。

9.2 (股票类 `Stock`) 遵照 9.2 节中 `Circle` 类的例子，设计一个名为 `Stock` 的类。这个类包括：

- 一个名为 `symbol` 的字符串数据域表示股票代码。
- 一个名为 `name` 的字符串数据域表示股票名字。
- 一个名为 `previousClosingPrice` 的 `double` 型数据域，它存储的是前一日的股票值。
- 一个名为 `currentPrice` 的 `double` 型数据域，它存储的是当时的股票值。
- 创建一支有特定代码和名字的股票的构造方法。
- 一个名为 `getChangePercent()` 的方法，返回从 `previousClosingPrice` 变化到 `currentPrice` 的百分比。

画出该类的 UML 图并实现这个类。编写一个测试程序，创建一个 `Stock` 对象，它的股票代码是 ORCL，股票名字为 Oracle Corporation，前一日收盘价是 34.5。设置新的当前值为 34.35，然后显示市值变化的百分比。

### 9.6 节

\*9.3 (使用日期类 `Date`) 编写程序创建一个 `Date` 对象，设置它的流逝时间分别为 10000、100000、1000000、10000000、100000000、1000000000、10000000000，然后使用 `toString()` 方法分别显示上述日期。

\*9.4 (使用随机类 `Random`) 编写一个程序，创建种子是 1000 的 `Random` 对象，然后使用 `nextInt(100)` 方法显示 0 到 100 之间前 50 个随机整数。

\*9.5 (使用公历类 `GregorianCalendar`) Java API 有一个在包 `java.util` 中的类 `GregorianCalendar`，可以使用它获得某个日期的年、月、日。它的无参构造方法构建一个当前日期的实例，`get(GregorianCalendar.YEAR)`、`get(GregorianCalendar.MONTH)` 和 `get(GregorianCalendar.DAY_OF_MONTH)` 方法返回年、月和日。编写一个程序完成两个任务：

- 显示当前的年、月和日。
- `GregorianCalendar` 类有方法 `setTimeInMillis(long)`，可以用它来设置从 1970 年 1 月 1 日算起的一个特定时间。将这个值设置为 1234567898765L，然后显示这个年、月和日。

### 9.7 ~ 9.9 节

\*9.6 (秒表) 设计一个名为 `StopWatch` 的类，该类包含：

- 具有访问器方法的私有数据域 `startTime` 和 `endTime`。
- 一个无参构造方法，使用当前时间来初始化 `startTime`。
- 一个名为 `start()` 的方法，将 `startTime` 重设为当前时间。
- 一个名为 `stop()` 的方法，将 `endTime` 设置为当前时间。
- 一个名为 `getElapsedTime()` 的方法，以毫秒为单位返回秒表记录的流逝时间。


画出该类的 UML 图并实现这个类。编写一个测试程序，用于测量使用选择排序对 100 000 个数字进行排序的执行时间。

9.7 (账户类 `Account`) 设计一个名为 `Account` 的类，它包括：

- 一个名为 `id` 的 `int` 类型私有数据域（默认值为 0）。
- 一个名为 `balance` 的 `double` 类型私有数据域（默认值为 0）。
- 一个名为 `annualInterestRate` 的 `double` 类型私有数据域存储当前利率（默认值为 0）。假设所有的账户都有相同的利率。
- 一个名为 `dateCreated` 的 `Date` 类型的私有数据域，存储账户的开户日期。
- 一个用于创建默认账户的无参构造方法。

- 一个用于创建带特定 id 和初始余额的账户的构造方法。
- id、balance 和 annualInterestRate 的访问器和修改器。
- dateCreated 的访问器。
- 一个名为 getMonthlyInterestRate() 的方法，返回月利率。
- 一个名为 withdraw 的方法，从账户提取特定数额。
- 一个名为 deposit 的方法向账户存储特定数额。

画出该类的 UML 图并实现这个类。

 提示：方法 getMonthlyInterest() 用于返回月利息，而不是利率。月利息是  $\text{balance} \times \text{monthlyInterestRate}$ 。monthlyInterestRate 是  $\text{annualInterestRate}/12$ 。注意，annualInterestRate 是一个百分数，比如 4.5%。你需要将其除以 100。

编写一个测试程序，创建一个账户 ID 为 1122、余额为 20 000 美元、年利率为 4.5% 的 Account 对象。使用 withdraw 方法取款 2500 美元，使用 deposit 方法存款 3000 美元，然后打印余额、月利息以及这个账户的开户日期。

9.8 (风扇类 Fan) 设计一个名为 Fan 的类来表示一个风扇。这个类包括：

- 三个名为 SLOW、MEDIUM 和 FAST 而值为 1、2 和 3 的常量，表示风扇的速度。
- 一个名为 speed 的 int 类型私有数据域，表示风扇的速度（默认值为 SLOW）。
- 一个名为 on 的 boolean 类型私有数据域，表示风扇是否打开（默认值为 false）。
- 一个名为 radius 的 double 类型私有数据域，表示风扇的半径（默认值为 5）。
- 一个名为 color 的 string 类型数据域，表示风扇的颜色（默认值为 blue）。
- 这四个数据域的访问器和修改器。
- 一个创建默认风扇的无参构造方法。
- 一个名为 toString() 的方法返回描述风扇的字符串。如果风扇是打开的，那么该方法在一个组合的字符串中返回风扇的速度、颜色和半径。如果风扇没有打开，该方法就会返回一个由 “fan is off” 和风扇颜色及半径组合成的字符串。

画出该类的 UML 图并实现这个类。编写一个测试程序，创建两个 Fan 对象。将第一个对象设置为最大速度、半径为 10、颜色为 yellow、状态为打开。将第二个对象设置为中等速度、半径为 5、颜色为 blue、状态为关闭。通过调用它们的 toString 方法显示这些对象。

**\*\*9.9** (几何：正  $n$  边形) 在一个正  $n$  边形中，所有边的长度都相同，且所有角的度数都相同（即这个多边形是等边等角的）。设计一个名为 RegularPolygon 的类，该类包括：

- 一个名为 n 的 int 型私有数据域定义多边形的边数，默认值为 3。
- 一个名为 side 的 double 型私有数据域存储边的长度，默认值为 1。
- 一个名为 x 的 double 型私有数据域定义多边形中点的  $x$  坐标，默认值为 0。
- 一个名为 y 的 double 型私有数据域定义多边形中点的  $y$  坐标，默认值为 0。
- 一个创建带默认值的正多边形的无参构造方法。
- 一个能创建带指定边数和边长度、中心在 (0,0) 的正多边形的构造方法。
- 一个能创建带指定边数和边长度、中心在 (x,y) 的正多边形的构造方法。
- 所有数据域的访问器和修改器。
- 一个返回多边形周长的方法 getPerimeter()。
- 一个返回多边形面积的方法 getArea()。计算正多边形面积的公式是：

$$\text{面积} = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}$$

画出该类的 UML 图并实现这个类。编写一个测试程序，分别使用无参构造方法、RegularPolygon(6,4) 和 RegularPolygon(10,4,5.6,7.8) 创建三个 RegularPolygon 对象。

显示每个对象的周长和面积。

\*9.10 (代数: 二次方程式) 为二次方程式  $ax^2+bx+c=0$  设计一个名为 `QuadraticEquation` 的类。这个类包括:

- 代表三个系数的私有数据域 `a`、`b` 和 `c`。
- 一个参数为 `a`、`b` 和 `c` 的构造方法。
- `a`、`b`、`c` 的三个 `get` 方法。
- 一个名为 `getDiscriminant()` 的方法返回判别式,  $b^2-4ac$ 。
- 名为 `getRoot1()` 和 `getRoot2()` 的方法返回等式的两个根:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{和} \quad r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

这些方法只有在判别式为非负数时才有用。如果判别式为负, 这些方法返回 0。

画出该类的 UML 图并实现这个类。编写一个测试程序, 提示用户输入 `a`、`b` 和 `c` 的值, 然后显示判别式的结果。如果判别式为正数, 显示两个根; 如果判别式为 0, 显示一个根; 否则, 显示 “The equation has no roots.” (这个方程无根)。参见编程练习题 3.1 的运行示例。

\*9.11 (代数:  $2 \times 2$  的线性方程) 为一个  $2 \times 2$  的线性方程设计一个名为 `LinearEquation` 的类:

$$\begin{array}{lcl} ax+by=e & x = \frac{ed-bf}{ad-bc} & y = \frac{af-ec}{ad-bc} \\ cx+dy=f \end{array}$$

这个类包括:

- 私有数据域 `a`、`b`、`c`、`d`、`e` 和 `f`。
- 一个参数为 `a`、`b`、`c`、`d`、`e`、`f` 的构造方法。
- `a`、`b`、`c`、`d`、`e`、`f` 的六个 `get` 方法。
- 一个名为 `isSolvable()` 的方法, 如果  $ad-bc$  不为 0 则返回 `true`。
- 方法 `getX()` 和 `getY()` 返回这个方程的解。

画出该类的 UML 图并实现这个类。编写一个测试程序, 提示用户输入 `a`、`b`、`c`、`d`、`e`、`f` 的值, 然后显示它的结果。如果  $ad-bc$  为 0, 就报告 “The equation has no solution.”。参见编程练习题 3.3 的运行示例。

\*\*9.12 (几何: 交点) 假设两条线段相交。第一条线段的两个端点是  $(x_1, y_1)$  和  $(x_2, y_2)$ , 第二条线段的两个端点是  $(x_3, y_3)$  和  $(x_4, y_4)$ 。编写一个程序, 提示用户输入这四个端点, 然后显示它们的交点。如编程练习题 3.25 所讨论的, 可以通过对一个线性方程求解来得到。使用编程练习题 9.11 中的 `LinearEquation` 类来求解该方程。参见编程练习题 3.25 的运行示例。

\*\*9.13 (位置类 `Location`) 设计一个名为 `Location` 的类, 定位二维数组中的最大值及其位置。这个类包括公共的数据域 `row`、`column` 和 `maxValue`, 二维数组中的最大值及其下标用 `int` 型的 `row` 和 `column` 以及 `double` 型的 `maxValue` 存储。

编写下面的方法, 返回一个二维数组中最大值的位置。

```
public static Location locateLargest(double[][] a)
```

返回值是一个 `Location` 的实例。编写一个测试程序, 提示用户输入一个二维数组, 然后显示这个数组中最大元素的位置。下面是一个运行示例:

```
Enter the number of rows and columns in the array: 3 4
Enter the array:
23.5 35 2 10
4.5 3 45 3.5
35 44 5.5 9.6
The location of the largest element is 45 at (1, 2)
```