

```
package p1;

public class A {
    ? int i;
    ? void m() {
        ...
    }
}
```


a)

```
package p2;

public class B extends A {
    public void m1(String[] args) {
        System.out.println(i);
        m();
    }
}
```

b)

## 11.15 防止扩展和重写

 **要点提示：**一个被 `final` 修饰的类和方法都不能被扩展。被 `final` 修饰的数据域是一个常数。

有时候，可能希望防止类扩展。在这种情况下，使用 `final` 修饰符表明一个类是最终的，是不能作为父类的。`Math` 类就是一个最终类。`String`、`StringBuilder` 和 `StringBuffer` 类也可以是最终类。例如，下面的类 `A` 就是最终类，是不能被继承的：


```
public final class A {
    // Data fields, constructors, and methods omitted
}
```

也可以定义一个方法为最终的，最终方法不能被它的子类重写。

例如，下面的方法是最终的，是不能重写的：

```
public class Test {
    // Data fields, constructors, and methods omitted

    public final void m() {
        // Do something
    }
}
```

 **注意：**修饰符 `public`、`protected`、`private`、`static`、`abstract` 以及 `final` 可以用在类和类的成员（数据和方法）上，只有 `final` 修饰符还可以用在方法中的局部变量上。方法内的最终局部变量就是常量。

### 复习题

11.41 如何防止一个类被扩展？如何防止一个方法被重写？

11.42 指出下面语句是对还是错：

- 被保护的数据或方法可以被同一包中的任何类访问。
- 被保护的数据或方法可以被不同包中的任何类访问。
- 被保护的数据或方法可以被任意包中它的子类访问。
- 最终类可以有实例。
- 最终类可以被扩展。
- 最终方法可以被重写。

## 关键术语

actual type (实际类型)

casting object (转换对象)

constructor chaining (构造方法链)

declared type (声明类型)

dynamic binding (动态绑定)	protected (受保护修饰符)
inheritance (继承)	single inheritance (单一继承)
instanceof (运算符, 是……类型的实例)	subclass (子类)
is-a relationship (是关系)	subtype (子类型)
method overriding (方法重写)	superclass (父类)
multiple inheritance (多重继承)	supertype (父类型)
override (重写)	type inference (类型推导)
polymorphism (多态)	

## 本章小结

1. 可以从现有的类定义新的类, 这称为类的继承。新类称为次类、子类或继承类; 现有的类称为超类、父类或基类。
2. 构造方法用来构造类的实例。不同于属性和方法, 子类不继承父类的构造方法。它们只能用关键字 `super` 从子类的构造方法中调用。
3. 构造方法可以调用重载的构造方法或它的父类的构造方法。这种调用必须是构造方法的第一条语句。如果没有显式地调用它们中的任何一个, 编译器就会把 `super()` 作为构造方法的第一条语句, 它调用的是父类的无参构造方法。
4. 为了重写一个方法, 必须使用与它的父类中的方法相同的签名来定义子类中的方法。
5. 实例方法只有在可访问时才能重写。这样, 私有方法是不能重写的, 因为它不能在类本身之外访问的。如果子类中定义的方法在父类中是私有的, 那么这两个方法是完全没有关系的。
6. 静态方法与实例方法一样可以继承。但是, 静态方法不能重写, 如果父类中定义的静态方法在子类中重新定义, 那么父类中定义的方法被隐藏。
7. Java 中的每个类都继承自 `java.lang.Object` 类。如果一个类在定义时没有指定继承关系, 那么它的父类就是 `Object`。
8. 如果一个方法的参数类型是父类 (例如: `Object`), 可以向该方法的参数传递任何子类 (例如: `Circle` 类或 `String` 类) 的对象。这称为多态。
9. 因为子类的实例总是它的父类的实例, 所以, 总是可以将一个子类的实例转换成一个父类的变量。当把父类实例转换成它的子类变量时, 必须使用转换记号 (子类名) 进行显式转换, 向编译器表明你的意图。
10. 一个类定义一个类型。子类定义的类型称为子类型, 而父类定义的类型称为父类型。
11. 当从引用变量调用实例方法时, 该变量的实际类型在运行时决定使用该方法的哪个实现。这称为动态绑定。
12. 可以使用表达式 `obj instanceof AClass` (对象名 instanceof 类名) 测试一个对象是否是一个类的实例。
13. 可以使用 `ArrayList` 类来创建一个对象, 用于存储一个对象列表。
14. 可以使用 `protected` 修饰符来防止方法和数据被不同包的非子类访问。
15. 可以用 `final` 修饰符来表明一个类是最终类, 是不能被继承的; 也可以表明一个方法是最终的, 是不能重写的。

## 测试题

回答位于 [www.cs.armstrong.edu/liang/intro10e/quiz.html](http://www.cs.armstrong.edu/liang/intro10e/quiz.html) 中本章的测试题。

## 编程练习题

### 11.2 ~ 11.4 节

11.1 (三角形类 Triangle) 设计一个名为 Triangle 的类来扩展 GeometricObject 类。该类包括:

- 三个名为 side1、side2 和 side3 的 double 数据域表示这个三角形的三条边, 它们的默认值是 1.0。
- 一个无参构造方法创建默认的三角形。
- 一个能创建带指定 side1、side2 和 side3 的三角形的构造方法。
- 所有三个数据域的访问器方法。
- 一个名为 getArea() 的方法返回这个三角形的面积。
- 一个名为 getPerimeter() 的方法返回这个三角形的周长。
- 一个名为 toString() 的方法返回这个三角形的字符串描述。

计算三角形面积的公式参见编程练习题 2.19。toString() 方法的实现如下所示:

```
return "Triangle: side1 = " + side1 + " side2 = " + side2 +  
      " side3 = " + side3;
```

画出 Triangle 类和 GeometricObject 类的 UML 图, 并实现这些类。编写一个测试程序, 提示用户输入三角形的三条边、颜色以及一个 Boolean 值表明该三角形是否填充。程序应该使用输入创建一个具有这些边并设置 color 和 filled 属性的三角形。程序应该显示面积、边长、颜色以及表明是否填充的真或者假的值。

### 11.5 ~ 11.14 节

11.2 (Person、Student、Employee、Faculty 和 Staff 类) 设计一个名为 Person 的类和它的两个名为 Student 和 Employee 的子类。Employee 类又有子类: 教员类 Faculty 和职员类 Staff。每个人都有姓名、地址、电话号码和电子邮件地址。学生有班级状态(大一、大二、大三或大四)。将这些状态定义为常量。一个雇员涉及办公室、工资和受聘日期。使用编程练习题 10.14 中定义的 MyDate 类为受聘日期创建一个对象。教员有办公时间和级别。职员有职务称号。覆盖每个类中的 toString 方法, 显示相应的类别名字和人名。

画出这些类的 UML 图并实现这些类。编写一个测试程序, 创建 Person、Student、Employee、Faculty 和 Staff, 并且调用它们的 toString() 方法。

11.3 (账户类 Account 的子类) 在编程练习题 9.7 中定义了一个 Account 类来建模一个银行账户。一个账户有账号、余额、年利率、开户日期等属性, 以及存款和取款等方法。创建两个检测支票账户 (checking account) 和储蓄账户 (saving account) 的子类。支票账户有一个透支限定额, 但储蓄账户不能透支。

画出这些类的 UML 图并实现这些类。编写一个测试程序, 创建 Account、SavingsAccount 和 CheckingAccount 的对象, 然后调用它们的 toString() 方法。

11.4 (ArrayList 的最大元素) 编写以下方法, 返回一个整数 ArrayList 的最大值。如果列表为 null 或者列表的大小为 0, 则返回 null 值。

```
public static Integer max(ArrayList<Integer> list)
```

编写一个测试程序, 提示用户输入一个以 0 结尾的数值序列, 调用该方法返回输入的最大数值。

11.5 (课程类 Course) 重写程序清单 10-6 中的 Course 类, 使用 ArrayList 代替数组来存储学生。为该类绘制新的 UML 图。不应该改变 Course 类的原始合约 (即, 构造方法和方法的定义都不应该改变, 但私有的成员可以改变)。

11.6 (使用 ArrayList) 编写程序, 创建一个 ArrayList, 然后向这个列表中添加一个 Loan 对象、

一个 Date 对象、一个字符串和一个 Circle 对象，然后使用循环调用对象的 toString() 方法，来显示列表中所有的元素。

11.7 (打乱 ArrayList) 编写以下方法，打乱一个整数 ArrayList 中的元素。

```
public static void shuffle(ArrayList<Integer> list)
```

\*\*11.8 (新的 Account 类) 编程练习题 9.7 中给出了一个 Account 类，如下设计一个新的 Account 类：

- 添加一个 String 类型的新数据域 name 来存储客户的名字。
- 添加一个新的构造方法，该方法创建一个具有指定名字、id 和收支额的账户。
- 添加一个名为 transactions 的 ArrayList 类型的新数据域，用于为账户存储交易。每笔交易都是一个 Transaction 类的实例。Transaction 类的定义如图 11-6 所示。
- 修改 withdraw 和 deposit 方法，向 transactions 数组线性表添加一笔交易。
- 其他所有属性和方法都和编程练习题 9.7 中的一样。

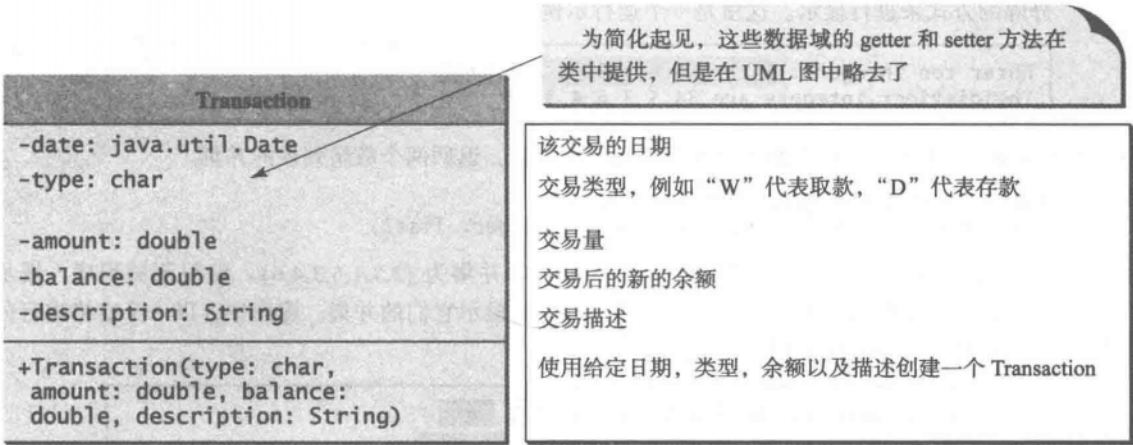



图 11-6 Transaction 类描述银行账户的一笔交易

编写一个测试程序，创建一个年利率为 1.5%、收支额为 1000、id 为 1122 而名字为 George 的 Account。向该账户存入 30 美元、40 美元和 50 美元并从该账户中取出 5 美元、4 美元和 2 美元。打印账户清单，显示账户所有者名字、利率、收支额和所有的交易。

\*11.9 (最大的行和列) 编写程序，随机将 0 和 1 填入一个  $n \times n$  的矩阵，打印该矩阵，并且找出具有最多 1 的行和列。

 提示：使用两个 ArrayList 来存储具有最多 1 的行和列的下标。  
这里是程序的一个运行示例：

```
Enter the array size n: 4
The random array is
0011
0011
1101
1010
The largest row index: 2
The largest column index: 2, 3
```

11.10 (利用继承实现 MyStack) 在程序清单 11-10 中，MyStack 是用组合实现的。扩展 ArrayList 创建一个新的栈类。

画出这些类的 UML 图并实现 MyStack 类。编写一个测试程序，提示用户输入五个字符串，然后以逆序显示这些字符串。

11.11 (对 ArrayList 排序) 编写以下方法, 对一个数值的 ArrayList 进行排序:

```
public static void sort(ArrayList<Integer> list)
```

编写测试程序, 提示用户输入 5 个数字, 将其存储在一个数组列表中, 并且以升序进行显示。

11.12 (对 ArrayList 求和) 编写以下方法, 返回 ArrayList 中所有数字的和:

```
public static double sum(ArrayList<Double> list)
```

编写测试程序, 提示用户输入 5 个数字, 将其存储在一个数组列表中, 并且显示它们的和。

\*11.13 (去掉重复元素) 使用下面的方法头编写方法, 从一个整数的数组列表中去掉重复元素:

```
public static void removeDuplicate(ArrayList<Integer> list)
```

编写测试程序, 提示用户输入 10 个整数到列表中, 显示其中不同的整数, 并以一个空格分隔的方式进行显示。这里是一个运行示例:

```
Enter ten integers: 34 5 3 5 6 4 33 2 2 4 Enter
The distinct integers are 34 5 3 6 4 33 2
```

11.14 (结合两个列表) 使用下面的方法头编写一个方法, 返回两个数组列表的并集。

```
public static ArrayList<Integer> union(
    ArrayList<Integer> list1, ArrayList<Integer> list2)
```

例如, 两个数组列表 {2,3,1,5} 和 {3,4,6} 的并集为 {2,3,1,5,3,4,6}。编写测试程序, 提示用户输入两个列表, 每个列表有 5 个整数, 然后显示它们的并集。输出中, 以一个空格进行分隔。这里是一个运行示例:

```
Enter five integers for list1: 3 5 45 4 3 Enter
Enter five integers for list2: 33 51 5 4 13 Enter
The combined list is 3 5 45 4 3 33 51 5 4 13
```

\*11.15 (凸多边形面积) 如果一个多边形中连接任意两个顶点的线段都包含在多边形中, 则称为凸多边形。编写一个程序, 提示用户输入一个凸多边形中的顶点数, 并顺时针输入点, 然后程序显示多边形的面积信息。这里是一个程序的运行示例:

```
Enter the number of the points: 7 Enter
Enter the coordinates of the points:
-12 0 -8.5 10 0 11.4 5.5 7.8 6 -5.5 0 -7 -3.5 -13.5 Enter
The total area is 292.575
```


\*\*11.16 (加法测试) 重写程序清单 5-1, 如果用户重复输入了相同的答案, 则给出用户警告。

 提示: 使用一个数组列表来存储答案。

这里是一个运行示例:


```
What is 5 + 9? 12 Enter
Wrong answer. Try again. What is 5 + 9? 34 Enter
Wrong answer. Try again. What is 5 + 9? 12 Enter
You already entered 12
Wrong answer. Try again. What is 5 + 9? 14 Enter
You got it!
```

\*\*11.17 (代数: 完全平方) 编写一个程序, 提示用户输入一个整数  $m$ , 然后找到最小的整数  $n$ , 使得  $m*n$  是一个完全平方。

 **提示：**存储所有  $m$  的最小因子到一个数组列表，则  $n$  是列表中出现奇数次的因子的乘积。例如，考虑  $m=90$  的情况，保存因子 2,3,3,5 到一个数组列表中。列表中 2 和 5 出现了奇数次数，因此， $n$  是 10。)

这里是一个运行示例：

```
Enter an integer m: 1500   
The smallest number n for m * n to be a perfect square is 15  
m * n is 22500
```

```
Enter an integer m: 63   
The smallest number n for m * n to be a perfect square is 7  
m * n is 441
```