计算机科学技术学院
School of Computer Science and Technology

Java 程序设计

第 7 章　内部类与异常类

导读

- 主要内容
  - 内部类
  - 匿名类
  - 异常类
  - 断言
- 重点和难点
  - 重点：内部类和异常类的理解
  - 难点：异常类的使用

7.1 内部类

- Java 支持在一个类中声明另一个类，这样的类称作**内部类**，而包含内部类的类成为内部类的**外嵌类**
- 例如：某种类型的农场饲养了一种特殊种类的牛，但不希望其他农场饲养这种特殊种类的牛，那么这种类型的农场就可以将创建这种特殊种牛的类作为自己的内部类
- 有一个RedCowForm（红牛农场）类，该类中有一个名字为 RedCow （红牛）的内部类

7.1 内部类

```
class RedCowForm {
        static String formName;
        RedCow cow;
        RedCowForm() {}
        RedCowForm(String s) {
                cow = new RedCow(150, 112, 50000);  formName = s; }
        void showCowMess() {
                cow.speak(); }
        class RedCow {
                String cowName = " 红牛 "; int height, weight, price;
                RedCow(int h, int w, int p) {
                        height = h; weight = w; price = p; }
                void speak() {
                        System.out.println(" 身高：  " + height + " cm\n  体
重：                    " + weight + " kg\n  生活在：  " + formName); }
        }
}
```

7.1 内部类

- 内部类的使用规则：
  - 声明内部类如同在类中声明方法或变量一样，一个类把内部类看作是自己的成员
  - 外嵌类的类体中可以用内部类声明的对象，作为外嵌类的成员。
  - 外嵌类的成员变量在内部类中仍然有效，内部类中的方法也可以调用外嵌类中的方法。
  - 内部类的类体中不可以声明**类变量和类方法**。
  - 外嵌类和内部类在编译时，生成**两个 .class 文件**。

7.2 匿名类

① 和子类有关的匿名类

- Java 允许我们直接使用一个类的**子类的类体**创建一个子类对象
- 创建子类对象时，除了使用父类的构造方法外还有类体，此类体被认为是一个子类**去掉类声明后的类体**，称作匿名类。

假设 Bank 是类，那么下列代码就是用 Bank 的一个子类（匿名类）创建对象：

```
new Bank () {
        匿名类的类体
};
```

7.2 匿名类

```
class Bank {
   String getBankName() {return "";}  }
```

```
class ABC extends Bank {
    String getBankName() {
        return " 中国农业银行 ";   } }
```

```
class BankInfo {
    void showBankName(Bank bank) {
        System.out.println(bank.getBankName());  }}
```

```
BankInfo bi = new BankInfo();
bi.showBankName(new ABC());
bi.showBankName(new Bank() {
    String getBankName() {
        return " 宇宙小银行" ;  }  }); }
```

7.2 匿名类

② 和接口有关的匿名类

- 假设 Computable 是一个接口，那么， Java 允许直接用接口名和一个类体创建一个匿名对象，此类体被认为是实现了 Computable 接口的类去掉类声明后的类体，称作匿名类

下列代码就是用实现了 Computable 接口的类（匿名类）创建对象：

```
new Computable() {
        实现接口的匿名类的类体
};
```

7.2 匿名类

```
interface ISayHello {
    void say();  }
```

```
class People {
    void sayHello(ISayHello s) {
        s.say();
}  }
```

```
People c = new People();
c.sayHello(new ISayHello() {
    public void say() {
        System.out.println(" 你好 ");
}  });
```

7.3 异常类

- 所谓异常就是程序运行时可能出现一些错误，比如试图打开一个根本不存在的文件等，异常处理将会改变程序的控制流程，让程序有机会对错误作出处理
- 程序运行出现异常时， Java 运行环境就用异常类 Exception 的相应子类创建一个异常对象，并等待处理

7.3 异常类

- 异常对象可以调用如下方法得到或输出有关异常的信息：
  - public String getMessage();
  - public void printStackTrace();
  - public String toString();

7.3.1 try~catch 语句

- Java 使用 try~catch 语句来处理异常，将可能出现的异常操作放在 try~catch 语句的 try 部分，将发生异常后的处理放在 catch 部分。
- try~catch 语句的格式如下：

```
try {
        包含可能发生异常的语句
}
catch(ExceptionSubClass1  e) {
      …
}
catch(ExceptionSubClass2  e) {
      …
}
```

7.3.1 try~catch 语句

- 带 finally 子语句的 try~catch 语句，语法格式如下：

```
try{}
catch(ExceptionSubClass e){ }
finally{}
```

- 其执行机制是在执行 try~catch 语句后，执行 finally 子语句，也就是说，无论在 try 部分是否发生过异常， finally 子语句都会被执行。

例 子 7(DangerException.java , CargoBoat.java , Example7_7.java) 中模拟向货船上装载集装箱，如果货船超重，那么货船认为这是一个异常，将拒绝装载集装箱，但无论是否发生异常，货船都需要正点启航。运行效果如图 7.8 。

```
目前装载了600吨货物
目前装载了1000吨货物
超重
无法再装载重量是367吨的集装箱
货船将正点启航
```

图 7.8 货船装载集装箱

7.3.2 异常的声明

- 一个方法不处理它产生的异常，而是沿着调用层次向上传递，由调用它的方法来处理这些异常，叫声明异常 .
- **声明异常的方法**：在产生异常的方法名后面加上要抛出 (throws) 的异常的列表：

如： void compute(int x) throws ArithmeticException

{// 这里有异常发生，但是并没有处理… }

7.3.2 自定义异常类

- 我们也可以扩展 Exception 类定义自己的异常类，然后规定哪些方法产生这样的异常。
- 一个方法在声明时可以使用 throws 关键字声明要产生的若干个异常，并在该方法的方法体中具体给出产生异常的操作，即用相应的异常类创建对象，并使用 throw 关键字抛出该异常对象，导致该方法结束执行。

7.3.2 自定义异常类例题

- 通常情况下，计算两个整数之和的方法不应当有任何异常放出，但是，对某些特殊应程序，可能不允许同号的整数做求和运算，比如当一个整数代表收入，一个整数代表支出时，这两个整数就不能是同号。

例子5(Example7_5.java)中，Bank类中有一个income(int in,int out)方法，对象调用该方法时，必须向参数in传递正整数、向参数out传递负数，并且int+out必须大于等于0，否则该方法就抛出异常(BankException.java)。因此，Bank类在声明income(int in,int out)方法时，使用throws关键字声明要产生的异常。程序运行效果如图7.5

```
本次计算出的纯收入是:100元
本次计算出的纯收入是:200元
本次计算出的纯收入是:300元
银行目前有600元
计算收益的过程出现如下问题:
入账资金200是负数或支出100是正数，不符合系统要求.
银行目前有600元
```

图7.5 自定义异常

7.4 断言

- 断言语句用于调试代码阶段。在调试代码阶段让断言语句发挥作用，这样就可以发现一些致命的错误，当程序正式运行时就可以关闭断言语句，但仍把断言语句保留在源代码中，如果以后应用程又需要调试，可以重新启用断言语句。
- 使用关键字 assert 声明一条断言语句，断言语句有以下两种格式：
  - assert booleanExpression;
  - assert booleanExpression : messageException;

7.4 断言例题

例 6 中，使用一个数组放着某学生 5 门课程的成绩，程序准备计算学生的成绩的总和。在调试程序时使用了断言语句，如果发现成绩有负数，程序立刻结束执行。程序调试开启断言语句运行效果如图 7.6，关闭断言语句运行效果如图 7.7 。

```
C:\z>java -ea Example7_6
Exception in thread "main" java.lang.AssertionError: 负数不能是成绩
        at Example7_6.main(Example7_6.java:7)
```

图 7.6 开启断言语句

```
C:\z>java  Example7_6
总成绩:286
```

图 7.7 关闭断言语句

总结

- Java 支持在一个类中声明另一个类，这样的类称作内部类，而包含内部类的类成为内部类的外嵌类
- 和类有关的匿名类就是该类的一个子类，该子类没有明显的用类声明来定义，所以称做匿名类。
- 和接口有关的匿名类就是实现该接口的一个类，该子类没有明显的用类声明来定义，所以称做匿名类
- Java 使用 try~catch 语句来处理异常

作业

- 习题 7 ： 3,4

本章结束

Thank You !