



# 内部类、匿名类和异常类



邹国兵

上海大学

计算机学院



# 第7章 内部类、匿名类和异常类

1. 内部类
  2. 匿名类
  3. 异常类



# 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，而包含内部类的类称为外嵌类。

- ① 内部类的外嵌类的成员变量在内部类中仍然有效，内部类中方法也可以调用外嵌类中的方法。
- ② 外嵌类的类体中可以用内部类声明对象，作为外嵌类的成员。
- ③ 内部类仅供它的外嵌类使用，其他类不可以用某个类的内部类声明对象。

# 1、内部类

```
1 package shu.ces.java.chap7;
2
3 public class RedCowForm {
4     static String formName;
5     RedCow cow; //内部类声明对象
6
7     RedCowForm() {
8
9
10    RedCowForm(String s) {
11        cow = new RedCow(150,112,5000);
12        formName = s;
13    }
14
15    public void showCowMess() {
16        cow.speak();
17    }
18
19    class RedCow {} //内部类的声明
20    String cowName = "红牛";
21    int height,weight,price;
22
23    RedCow(int h,int w,int p){
24        height = h;
25        weight = w;
26        price = p;
27    }
28    void speak() {
29        System.out.println("偶是"+cowName+",身高:"+height+"cm 体重:"+weight+"kg,生活在"+formName);
30    }
31 } //内部类结束
32 }
```

Eclipse演示



## 第7章 内部类与异常类

- 1. 内部类
  - 2. 匿名类
  - 3. 异常类
- 



## ■■■ 2、匿名类

### 2.1 和子类有关的匿名类

- 假如没有显式地声明一个类的子类，如何用子类创建一个对象？
- 使用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子类类体，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**。



## 2.1 和子类有关的匿名类

```
1 package shu.ces.java.chap7;  
2  
3 public class ShowBoard {  
4     void showMess(OutputAlphabet show) {  
5         show.output();  
6     }  
7 }  
8  
9  
10
```

```
1 package shu.ces.java.chap7;  
2  
3 abstract class OutputAlphabet {  
4     public abstract void output();  
5 }  
6  
7  
8 |
```

```
1 package shu.ces.java.chap7;  
2  
3 public class OutputEnglish extends OutputAlphabet {  
4     public void output() {  
5         for(char c='a';c<='z';c++) {  
6             System.out.printf("%3c",c);  
7         }  
8     }  
9 }  
10
```

## 2.1 和子类有关的匿名类

```
1 package shu.ces.java.chap7;  
2  
3 public class Example7_2 {  
4     public static void main(String args[]) {  
5         ShowBoard board=new ShowBoard();  
6         board.showMess(new OutputEnglish()); //向参数传递OutputAlphabet的子类对象  
7         board.showMess(new OutputAlphabet() { //向参数传递OutputAlphabet的匿名子类对象  
8             public void output() {  
9                 System.out.println();  
10                for(char c='α';c<='ω';c++) //输出希腊字母  
11                    System.out.printf("%3c",c);  
12            }  
13        }  
14    );  
15 }  
16 }
```

Eclipse演示

---

a b c d e f g h i j k l m n o p q r s t u v w x y z  
α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ο σ τ υ φ χ ψ ω

## 2.1 和子类有关的匿名类

- 匿名类就是一个子类，由于无名可用，所以不可能用匿名类声明对象，但可直接用匿名类创建一个对象。
- 使用匿名类时，必然是在某个类中直接用匿名类创建对象，因此匿名类一定是**内部类**。
- 匿名对象的引用可以传递给一个匹配的参数，匿名类的常用的方式是向方法的参数传值。

## 2.2 与接口有关的匿名类

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该接口的类去掉类声明后的类体，也称作匿名类。

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

如果某个方法的参数是接口类型，则可使用接口名和类体组合创建一个匿名对象传递给方法的参数。

## 2.2 与接口有关的匿名类

```
1 package shu.ces.java.chap7;
2
3 interface SpeakHello {
4     void speak();
5 }
6 class HelloMachine {
7     public void turnOn(SpeakHello hello) {
8         hello.speak();
9     }
10 }
11 public class Example7_3 {
12     public static void main(String args[]) {
13         HelloMachine machine = new HelloMachine();
14         machine.turnOn( new SpeakHello() {
15             public void speak() {
16                 System.out.println("hello, you are welcome!");
17             }
18         });
19     };
20     machine.turnOn( new SpeakHello() {
21         public void speak() {
22             System.out.println("你好，欢迎光临！");
23         }
24     });
25 };
26 }
27 }
28 }
```

Eclipse演示

## ■■■ 3、异常类

- Java异常出现在方法调用过程中，即在方法调用过程中抛出异常对象，终止当前方法的继续执行，导致程序运行出现异常，进行异常处理。



## ■■■ 3.1 try～catch语句

- Java使用try～catch语句处理异常。
- 将可能出现的异常操作放在try部分，当try部分中的某个方法调用发生异常后，try部分将立刻结束执行，而转向执行相应的catch部分，程序将发生异常后处理放在catch部分。
- try～catch语句可以由几个catch组成，分别处理发生的相应异常。
- finally{}在try～catch语句后，执行finally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。



## 3.1 try~catch语句

- try~catch语句的格式如下：

```
try {  
    包含可能发生异常的语句  
}  
catch (ExceptionSubClass1 e1) {  
    ...  
}  
catch (ExceptionSubClass2 e2) {  
    ...  
}  
Finally {  
    ...  
}
```

- catch参数中的异常类都是Exception的某个子类，表明try部分可能发生的异常。这些子类之间不能有父子关系，否则保留一个含有父类参数的catch即可。

## 3.2 自定义异常类与异常处理

- 编写程序时可以扩展Exception类定义自己的异常类，然后根据程序的需要来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；
- 在方法体中具体给出产生异常的操作，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。

## 3.2 自定义异常类与异常处理

```
public void someMethod()
```

```
    throws SomeException {
```

```
        if (someCondition()) {
```

```
            throw new SomeException("错误原因");
```

```
}
```

调用该方法时试  
图捕获异常

... ... ...

声明该方  
法可能抛  
出的异常

构造并抛出  
异常对象

```
try {
```

```
    someMethod();
```

```
} catch (SomeException e) {
```

```
    //异常处理代码;
```

```
}
```

... ... ...

方法是可能抛出异常的

定义处理异  
常的代码

### 3.3 异常类举例

```
1 package shu.ces.java.chap7;
2
3 public class BankException extends Exception {
4     String message;
5
6     public BankException(int m,int n) {
7         message="入账资金"+m+"是负数或支出"+n+"是正数，或者银行亏本，不符合系统要求。";
8     }
9
10    public String warnMess() {
11        return message;
12    }
13}
14
```

```
1 package shu.ces.java.chap7;
2
3 public class Bank {
4     private int money;
5     public void income(int in,int out) throws BankException {
6         if(in<=0||out>=0||in+out<=0) {
7             throw new BankException(in,out); //方法抛出异常，导致方法结束
8         }
9         int netIncome=in+out;
10        System.out.printf("本次计算出的纯收入是：%d元\n",netIncome);
11        money=money+netIncome;
12    }
13    public int getMoney() {
14        return money;
15    }
16}
```

## 3.3 异常类举例

```
1 package shu.ces.java.chap7;
2
3 public class Example7_5 {
4     public static void main(String args[]) {
5         Bank bank=new Bank();
6         try{ bank.income(200,-100);
7             bank.income(300,-100);
8             bank.income(400,-100);
9             System.out.printf("银行目前有%d元\n",bank.getMoney());
10            bank.income(200, 100);
11            bank.income(99999,-100);
12        }
13        catch(BankException e) {
14            System.out.println("计算收益的过程出现如下问题：");
15            System.out.println(e.warnMess());
16        }
17        System.out.printf("银行目前有%d元\n",bank.getMoney());
18    }
19 }
```

Eclipse演示