第 7 章 内部类与异常类

7.1 内部类

1. 若在一个类中再声明一个类，则里面的类称为内部类，外面的类称为外嵌类（外围类）。

2. 内部类不可以声明类变量和定义类方法。

3. 内部类可以使用外嵌类的成员变量和调用外嵌类的方法。

4. 外嵌类可以声明内部类的对象名和引入内部类的对象。

5. 原则上内部类仅供外嵌类使用，若要供其他类使用，则必须是静态的（只有内部类才有可能是静态的）。静态内部类不可以使用外嵌类的实例变量和调用外嵌类的实例方法。

6. 内部类的字节码文件名为“外嵌类名$内部类名”。

pp. 162-163 代码 1

```
public class RedCowForm {
   static String formName;
   RedCow cow;  //内部类声明对象
   RedCowForm() {
   }
   RedCowForm(String s) {
      cow = new RedCow(150,112,5000);
      formName = s;
   }
```

```
   public void showCowMess() {
      cow.speak();
   }
   class RedCow {  //内部类的声明
      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("偶是"+cowName+",身高:"+height+"cm 体重:"+
                            weight+"kg,生活在"+formName);
      }
   }              //内部类结束
}                 //外嵌类结束

Example7_1.java

public class Example7_1 {
   public static void main(String args[]) {
      RedCowForm form = new RedCowForm("红牛农场");
      form.showCowMess();
      form.cow.speak();
   }
}
```

7.3 异常类

1. try-catch 语句

（1）形式

```
try {
    可能出现异常的操作
}
catch (异常类的子类名 1 e) { //形式参数无需非要取名“e”,用于传入异常类的
                              //子类 1 的对象的引用
    异常 1 的处理
}
catch (异常类的子类名 2 e) {
    异常 2 的处理
}
…
```

（2）若执行 try 部分时抛出了异常类的子类的对象，则转向相应的 catch 部分进行异常处理。

2. throw 语句

throw 异常类的子类的对象;

pp.167-168 代码 1

```
public class Example7_4 {
  public static void main(String args[ ]) {
     int n = 0,m = 0,t = 1000;
     try{  m = Integer.parseInt("8888");
           n = Integer.parseInt("ab89"); //发生异常,转向 catch
           t = 7777;  //t 没有机会被赋值
     }
     catch(NumberFormatException e) {
           System.out.println("发生异常:"+e.getMessage());
     }
     System.out.println("n="+n+",m="+m+",t="+t);
     try{  System.out.println("故意抛出 I/O 异常! ");
           throw new java.io.IOException("我是故意的"); //故意抛出异常
           //System.out.println("这个输出语句肯定没有机会执行,必须注释,否则编译
                                出错");
     }
     catch(java.io.IOException e) {
           System.out.println("发生异常:"+e.getMessage());
     }
  }
}
```

3. 自定义异常类的子类

（1）异常类的子类头的格式为“class 异常类的子类名 extends Exception”。

（2）可能抛出异常类的子类的对象的方法的头部格式为“类型 方法名(形式参数表) throws

异常类的子类名”。

p. 168 代码 2

```
public class BankException extends Exception {
   String message;
   public BankException(int m,int n) {
      message = "入账资金"+m+"是负数或支出"+n+"是正数,不符合系统要求.";
   }
   public String warnMess() {
      return message;
   }
}
```

pp. 168-169 代码 3

```
public class Bank {
   private int money;
   public void income(int in,int out) throws BankException {
      if(in<=0||out>=0||in+out<=0) {
         throw new BankException(in,out); //方法抛出异常,导致方法结束
      }
      int netIncome = in+out;
      System.out.printf("本次计算出的纯收入是:%d 元\n",netIncome);
      money = money+netIncome;
   }
   public int getMoney() {
      return money;
   }
}
```

p. 169 代码 2

```
public class Example7_5 {
   public static void main(String args[]) {
      Bank bank = new Bank();
      try{  bank.income(200,-100);
            bank.income(300,-100);
            bank.income(400,-100);
            System.out.printf("银行目前有%d 元\n",bank.getMoney());
            bank.income(200, 100);
            bank.income(99999,-100);
      }
      catch(BankException e) {
            System.out.println("计算收益的过程出现如下问题:");
            System.out.println(e.warnMess());
      }
      System.out.printf("银行目前有%d 元\n",bank.getMoney());
   }
}
```

7.4 断言

1. assert 语句

（1）形式

① assert 逻辑表达式;

② assert 逻辑表达式: 出错信息字符串;

（2）若逻辑表达式的值为 true，则继续执行，否则终止程序执行（并输出出错信息字符串）。

2. 启用与关闭 assert 语句

p.170 代码 1

```
import java.util.Scanner;
public class Example7_6 {
  public static void main (String args[ ]) {
       int [] score = {-120,98,89,120,99};
       int sum = 0;
       for(int number:score) {
          assert number>=0:"负数不能是成绩";
          sum = sum+number;
       }
       System.out.println("总成绩:"+sum);
   }
}
```

7.5 应用举例

对于带 finally 子语句的 try-catch 语句，执行 try 部分时不管是否抛出了异常类的子类的对象，都要执行 finally 子语句。

p.171 代码 1

```
public class DangerException extends Exception {
   final String message = "超重";
   public String warnMess() {
      return message;
   }
}
```

p.171 代码 2

```
public class CargoBoat {
    int realContent;  //装载的重量
    int maxContent;   //最大装载量
    public void setMaxContent(int c) {
       maxContent = c;
    }
    public void loading(int m) throws DangerException {
      realContent += m;
      if(realContent>maxContent) {
         realContent-=m;
         throw new DangerException();
      }
      System.out.println("目前装载了"+realContent+"吨货物");
    }
}
```

pp.171-172 代码 3

```
public class Example7_7 {
```

```
    public static void main(String args[]) {
       CargoBoat ship = new CargoBoat();
       ship.setMaxContent(1000);
       int m = 600;
       try{
           ship.loading(m);
           m = 400;
           ship.loading(m);
           m = 367;
           ship.loading(m);
           m = 555;
           ship.loading(m);
       }
       catch(DangerException e) {
           System.out.println(e.warnMess());
           System.out.println("无法再装载重量是"+m+"吨的集装箱");
       }
       finally {
          System.out.printf("货船将正点启航");
       }
    }
}
```