上机实践 7　内部类与异常类

实验 1　内部购物券

1. 相关知识点

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内部类的外嵌类。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。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。

2. 实验目的

本实验的目的是让学生掌握内部类的用法。

3. 实验要求

手机专卖店为了促销自己的产品，决定发行内部购物券，但其他商场不能发行该购物券。编写一个 MobileShop 类（模拟手机专卖店），该类中有一个名字为 InnerPurchaseMoney 的内部类（模拟内部购物券）。

4. 运行效果示例

程序运行效果如图 7.1 所示。

```
手机专卖店目前有30部手机
用价值20000的内部购物券买了6部手机
用价值10000的内部购物券买了3部手机
手机专卖店目前有21部手机
```

图 7.1　内部购物券

5. 程序模板

请按模板要求，将【代码】替换为 Java 程序代码。

NewYear.java

```
class MobileShop {
   【代码1】//用内部类InnerPurchaseMoney声明对象purchaseMoney1
   【代码2】 //用内部类InnerPurchaseMoney声明对象purchaseMoney1
   private int mobileAmount;  //手机的数量
   MobileShop(){
    【代码3】 //创建价值为20000的purchaseMoney1
    【代码4】 //创建价值为10000的purchaseMoney2
   }
   void setMobileAmount(int m) {
     mobileAmount = m;
   }
   int getMobileAmount() {
      return mobileAmount;
   }
```

```
    class InnerPurchaseMoney {
        int moneyValue;
        InnerPurchaseMoney(int m) {
           moneyValue = m;
        }
        void buyMobile() {
           if(moneyValue>=20000) {
              mobileAmount = mobileAmount-6;
              System.out.println("用价值"+moneyValue+"的内部购物券买了6部手机");
           }
           else if(moneyValue<20000&&moneyValue>=10000) {
              mobileAmount = mobileAmount-3;
              System.out.println("用价值"+moneyValue+"的内部购物券买了3部手机");
           }
        }
    }
}
public class NewYear
{
   public static void main(String args[]) {
      MobileShop shop = new MobileShop();
      shop.setMobileAmount(30);
      System.out.println("手机专卖店目前有"+shop.getMobileAmount()+"部手机");
      shop.purchaseMoney1.buyMobile();
      shop.purchaseMoney2.buyMobile();
      System.out.println("手机专卖店目前有"+shop.getMobileAmount()+"部手机");
   }
}
```

6. 实验指导

✧ 静态（static）内部类不可以操作外嵌类中的实例成员。

✧ 内部类可以限制其他类用这个内部类实例化对象。

7. 实验后的练习

参照本实验，用内部类模拟一个实际问题。

8. 填写实验报告

实验编号：701 学生姓名：　　实验时间：　　教师签字：

实验效果评价	A	B	C	D	E
模板完成情况					
实验后的练习效果评价	A	B	C	D	E
练习完成情况					
总评					

实验 2　检查危险品

1．相关知识点

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，一旦 try 部分抛出异常对象，比如调用某个抛出异常的方法抛出了异常对象，那么，try 部分将立刻结束执行，而转向执行相应的 catch 部分。

2．实验目的

本实验的目的是让学生掌握使用 try-catch 语句。

3．实验要求

车站检查危险品的设备，如果发现危险品会发出警告。编程模拟设备发现危险品。

编写一个 Exception 的子类 DangerException，该子类可以创建异常对象，该异常对象调用 toShow()方法输出“属于危险品”。

编写一个 Machine 类，该类的方法 checkBag(Goods goods)当发现参数 goods 是危险品时（goods 的 isDanger 属性是 true）将抛出 DangerException 异常。

程序在主类的 main()方法中的 try-catch 语句的 try 部分让 Machine 类的实例调用 checkBag (Goods goods)方法，如果发现危险品就在 try-catch 语句的 catch 部分处理危险品。

4．运行效果示例

程序运行效果如图 7.2 所示。

```
苹果不是危险品！苹果检查通过
危险品！炸药被禁止！
西服不是危险品！西服检查通过
危险品！硫酸被禁止！
手表不是危险品！手表检查通过
危险品！硫磺被禁止！
```

图 7.2　检查危险品

5．程序模板

请按模板要求，将【代码】替换为 Java 程序代码。

Goods.java

```
public class Goods {
   boolean isDanger;
   String name;
   public void setIsDanger(boolean boo) {
      isDanger = boo;
   }
   public boolean isDanger() {
      return isDanger;
   }
   public void setName(String s) {
      name = s;
   }
   public String getName() {
      return name;
   }
}
```

DangerException.java

```
public class DangerException extends Exception {
```

```
    String message;
    public DangerException() {
        message = "危险品!";
    }
    public void toShow() {
        System.out.print(message+" ");
    }
}
```

Machine.java

```
public class Machine {
    public void checkBag (Goods goods) throws DangerException {
        if(goods.isDanger()) {
            DangerException danger=new DangerException();
            【代码1】   //抛出danger
        }
        else {
            System.out.print(goods.getName()+"不是危险品!");
        }
    }
}
```

Check.java

```
public class Check {
    public static void main(String args[]) {
        Machine machine = new Machine();
        String name[] ={"苹果","炸药","西服","硫酸","手表","硫黄"};
        Goods [] goods = new Goods[name.length]; //检查6件物品
        for(int i= 0;i<name.length;i++) {
            goods[i] = new Goods();
            if(i%2==0) {
                goods[i].setIsDanger(false);
                goods[i].setName(name[i]);
            }
            else {
                goods[i].setIsDanger(true);
                goods[i].setName(name[i]);
            }
        }
        for(int i= 0;i<goods.length;i++) {
            try { machine.checkBag(goods[i]);
                System.out.println(goods[i].getName()+"检查通过");
            }
            catch(DangerException e) {
                【代码2】 //e调用toShow()方法
                System.out.println(goods[i].getName()+"被禁止!");
            }
        }
    }
}
```

6. 实验指导

✧ try-catch 语句可以由几个 catch 组成，分别处理发生的相应异常。

✧ 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。

7. 实验后的练习

（1）是否可以将实验中 try-catch 语句中，catch 捕获的异常更改为 Exception?

（2）是否可以将实验中 try-catch 语句中，catch 捕获的异常更改为 java.io.IOException?

8. 填写实验报告

实验编号：702 学生姓名：　　实验时间：　　教师签字：

实验效果评价	A	B	C	D	E
模板完成情况					
实验后的练习效果评价	A	B	C	D	E
练习（1）完成情况					
练习（2）完成情况					
总评					

实 验 答 案

实验 1：

【代码 1】InnerPurchaseMoney purchaseMoney1;

【代码 2】InnerPurchaseMoney purchaseMoney2;

【代码 3】purchaseMoney1 = new InnerPurchaseMoney(20000);

【代码 4】purchaseMoney2 = new InnerPurchaseMoney(10000);

实验 2：

【代码 1】throw danger;

【代码 2】e.toShow();

自 测 题

1. 请说出下列程序的输出结果。

```
public class E {
   public static void main(String args[]) {
      int m=5,n=-3;
      try{
          for(int i=1;i<=100;i++) {
            if(m+n>=0)
               System.out.print(m+n);
            else
              throw new java.io.IOException();
```

```
            m--;
          }
        }
        catch(Exception e) {
            System.out.print("不能循环100次");
        }
    }
}
```

2. 请说出下列程序的输出结果。

```
class MyException extends Exception {
   String message;
   MyException(String str) {
     message=str;
   }
   public String getMessage() {
     return message;
   }
}
abstract class A {
   abstract int f(int x,int y) throws MyException;
}
class B extends A {
   int f(int x,int y) throws MyException {
     if(x>99||y>99)
        throw new MyException("乘数超过99");
     return x*y;
   }
}
public class E {
   public static void main(String args[]) {
     A a;
     a = new B();
     try{
        System.out.print(a.f(12,8)+" ");
        System.out.print(a.f(120,3)+" ");
     }
     catch(MyException e) {
        System.out.print(e.getMessage());
     }
  }
}
```

答案：

1. 210 不能循环 100 次
2. 96 乘数超过 99