**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.程序模板**

请按模板要求，将【代码】替换为Java程序代码。

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

InnerPurchaseMoney purchaseMoney1 ,purchaseMoney2;

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

purchaseMoney1 = **new** InnerPurchaseMoney(10000);

purchaseMoney2 = **new** InnerPurchaseMoney(20000);

}

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+"的内部购物卷买了6部手机");

}

else if(moneyValue<20000&&moneyValue>=10000) {

mobileAmount = mobileAmount-3;

System.out.println("用价值"+moneyValue+"的内部购物卷买了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");

}

}

**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，—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.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

danger.toShow();

}

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","硫磺"};

Goods [] goods = new Goods[name.length]; //检查6件货物

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

e.toShow();

System.out.println(goods[i].getName()+"被禁止!");

}

}

}

}