**每日作业卷**

就业班JavaSE--day05【异常、Object类】

# 关卡1

## 训练案例1

### 训练考核知识点

Object类

### 训练描述

请谈谈你对Object类的理解

### 操作步骤描述

1. 请说出Object类在继承层次结构中的位置

处在类层次最高点，包含了所有Java类的公共属性

1. 请说出 Object 类的特点

如果一个类没有指定父类,那么默认继承Object类(所有类的爹)

## 训练案例2

### 训练考核知识点

equals方法

### 训练描述

请阐述你对equals方法的理解

### 操作步骤描述

1. 请说出equals方法在Object类中的默认实现什么?.

Object默认的quals方法的比较规则同==，比较的是两个引用变量中存储的值（地址信息），判断两个变量是否指向相同的对象；请说出我们自定义类什么时候需要对equals方法进行重写.

1. 请说出重写equals方法注意事项

（1）自反性：对于任何非空引用x，x.equals(x)应该返回true。  
（2）对称性：对于任何引用x和y，如果x.equals(y)返回true，那么y.equals(x)也应该返回true。  
（3）传递性：对于任何引用x、y和z，如果x.equals(y)返回true，y.equals(z)返回true，那么x.equals(z)也应该返回true。  
（4）一致性：如果x和y引用的对象没有发生变化，那么反复调用x.equals(y)应该返回同样的结果。  
（5）非空性：对于任意非空引用x，x.equals(null)应该返回false。

## 训练案例3

### 训练考核知识点

toString方法

### 训练描述

请阐述你对toString()的理解

### 操作步骤描述

1. 请说出toString()方法在Object类默认返回的是什么?

Object类的toString()方法默认返回该对象实现类的“类名+@+hashcode”值

1. 请说出什么时候需要重写toString()方法?

toString()方法是一个“自我描述”的方法，当输出某实例对象时，可以通过重写自定义等方式为对象实现输出自我描述的信息

1. 在什么时候,会自动调用这个对象的toString()方法

如果你在类里编写了toString，相当于覆盖了类中原有的toString

在System.out.print中会自动调用。

## 训练案例4

### 训练考核知识点

异常

### 训练描述

请阐述异常的分类

### 操作步骤描述

1. 请说出异常的继承体系

Throwable是java中异常和错误的顶层父类，只有继承Throwable类的子类才能够通过throws语句或者java虚拟机抛出去

1. 请说出你对错误(Error)的理解

Error (错误:一出现就是致命的)

服务器宕机,数据库崩溃等

1. 请说出你对异常(Expection的理解)

异常分为两大类：

* 运行时异常(RuntimeException或者是其子类)
* 编译时异常(除了运行时异常就是编译时异常)

1. 请说出你对运行时异常(RuntimeException)的理解

RuntimeException(运行时异常,一般都是程序员犯的错误,需要修改源码的.)

编译时异常:在编译时必须进行处理,不处理无法通过编译.

## 训练案例5

### 训练考核知识点

错误(Error)

异常(Exception)

### 训练描述

请阐述 错误(Error)与异常(Exception) 的区别

### 操作步骤描述

1. 请说出什么是错误(Error),错误可以捕获和抛出吗?

Error (错误:一出现就是致命的)，服务器宕机,数据库崩溃等

可以，但不是必须

1. 请说出什么是异常(Exception),异常可以捕获和抛出吗?

异常指的是在程序运行过程中发生的异常事件，通常是由外部问题（如硬件错误、输入错误）所导致的；可以

## 训练案例6

### 训练考核知识点

异常

### 训练描述

请说出虚拟机处理异常的方式

如果在运行时抛出了一个异常，那么java虚拟机会**按顺序搜索整个异常表找到匹配的条目，并且仅会匹配到在其指定范围内的异常**。当找到第一个匹配的条目后，虚拟机便**将程序计数器设置为新的偏移位置，然后继续执行指令**。如果没有条目被匹配到，java虚拟机会弹出当前的栈帧（停止执行当前方法），并继续向上（调用remainder方法的方法）抛出同样的异常。

### 操作步骤描述

略

## 训练案例7

### 训练考核知识点

运行时异常

### 训练描述

请说出四种运行时异常

NullPointerException - 空指针引用异常  
ClassCastException - 类型强制转换异常。  
IllegalArgumentException - 传递非法参数异常。  
ArithmeticException - 算术运算异常  
ArrayStoreException - 向数组中存放与声明类型不兼容对象异常  
IndexOutOfBoundsException - 下标越界异常  
NegativeArraySizeException - 创建一个大小为负数的数组错误异常  
NumberFormatException - 数字格式异常  
SecurityException - 安全异常  
UnsupportedOperationException - 不支持的操作异常

### 操作步骤描述

略.

## 训练案例8

### 训练考核知识点

异常的处理

### 训练描述

请说出throw与throws的区别

### 操作步骤描述

1. 请说出throw的使用位置,作用是什么?

throw出现在函数体,throw是语句抛出一个异常，一般是在代码块的内部，当程序出现某种逻辑错误时由程序员主动抛出某种特定类型的异常t

1. 请说出 throws的使用位置,作用是什么?

throws出现在方法函数头, 当某个方法可能会抛出某种异常时用于throws 声明可能抛出的异常，然后交给上层调用它的方法程序处理

## 训练案例9

### 训练考核知识点

异常的处理

### 训练描述

请简述异常的处理方式

### 操作步骤描述

* + - 1. 异常处理方式有几种,分别是什么?

(1).捕获异常,自己处理.  
(2).throw出去,让别人处理.

2. 详细阐述每种方式对异常是如何处理的

(1).捕获异常,自己处理:

使用try{

//如果里面有异常就执行catch里的语句

}catch（Exception e）{

//exption

}语句块

(2).throw出去,让别人处理:

Class A{

Void a()throws Exception//往上抛

}

## 训练案例10

### 训练考核知识点

抛出异常

异常处理

### 训练描述

请用代码实现如下需求:

1. 写一个方法实现获取字符串某个索引上的字符的功能
2. 写一个测试测试方法调用上面写的方法,使用thows方式进行处理异常,并在main方法中调用这个测试方法
3. 再写一个测试测试方法调用上面写的方法,使用try...catch方式进行处理异常,并在main方法中调用这个测试方法

### 操作步骤描述

1. 创建包com.itheima.level1\_10
2. 定义 Test10类
3. 提供一个静态方法:public static char charAt(String str, int index) 用于获取字符串str,index索引处的字符
4. 在方法中charAt(String,int index)
   1. 当字符串为null抛出 Exception("字符串不能为null")异常,
   2. 当字符串为"" 的时候抛出 Excetion("字符串不能为空") 异常;
   3. 当index不在 [0,str.length()-1] 之间的时,抛出Exception("索引越界") 异常.
   4. 否则返回str在index索引的字符

测试类：

**public** **class** Test10 {

**public** **static** **void** main(String[] args) **throws** Exception {

String str = **null**;

*charAt*(str, 8);

// String str = "";

// charAt(str, 0);

// String str = "aaaasdf";

// charAt(str, 8);

}

**public** **static** **char** charAt(String str, **int** index) **throws** Exception {

**if** (**null** == str) {

**throw** **new** Exception("字符串不能为null");

}

**if** (str.equals("")) {

**throw** **new** Exception("字符串不能为空");

}

**if** (index < 0 || index >= str.length()) {

**throw** **new** Exception("索引越界" + index);

}

**return** str.charAt(index);

}

}







1. 提供一个静态的方法testThrows() 用于测试charAt(String,int index)方法
   1. 在方法中调用charAt(String,int index)方法;
      1. 注意需要测试每一种情况
      2. 体会: 程序一旦遇到异常后面的代码就不执行了
      3. 注释运行过测试的代码,继续测试下一种情况
   2. 使用throws方式对异常进行处理
   3. 在main方法中,调用本方法

**public** **static** **void** testThrows() **throws** Exception{

**char** c;

c = *charAt*(**null**, 0);

c = *charAt*("", 0);

c = *charAt*("abc",4);

c= *charAt*("abc",1);

System.***out***.println(c);

}

1. 提供 一个静态方法testTryCatch(),用于测试charAt(String,int index)方法
   1. 在方法中调用charAt(String,int index)方法;
      1. 注意需要测试每一种情况
   2. 使用try...catch方式对异常进行处理
   3. 在main方法中,调用本方法.

public static void testTryCatch() {

try {

char c;

c = charAt(null, 0);

// c = charAt("", 0);

// c = charAt("asdasf", 1);

System.out.println(c);

} catch (Exception e) {

System.out.println("捕获异常");

}

}

}

## 训练案例11

### 训练考核知识点

向文件中写入数据

try...catch的方式处理异常

### 训练描述

使用代码实现

写一个实现把字符串中数据,写入项目根目录下的content.txt文件中

### 操作步骤描述

1. 创建包com.itheima.level1\_11
2. 定义类(Test11)
3. 写一个静态方法 void write(String content),在方法中
   1. 定义字符缓冲输出流变量BufferedWriter bw;初始值为null
   2. 写一个try{ }catch(IOException e){ }finally{ }代码块
   3. 在try{ }在代码块中
      1. 创建BufferedWriter对象,绑定content.txt文件,赋值给bw
      2. 调用bw的write()方法,传入content
   4. 在catch代码块中,打印异常信息
   5. 在finally代码块关闭流
      1. 写try{}catch(IOException ex){} 代码块
      2. 在try 代码块中,如果bw!=null,调用bw.close()方法
      3. 在catch代码块,打印异常信息
4. 在main方法中调用write(String conent)方法

**public** **class** Test {

**public** **static** **void** main(String[] args) {

*write*("11");

}

**public** **static** **void** write(String content){

BufferedWriter bw = **null**;

**try** {

bw = **new** BufferedWriter(**new** FileWriter("content.txt"));

bw.write(content);

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

**try** {

**if**(bw!=**null**){

bw.close();

}

} **catch** (Exception e2) {

e2.printStackTrace();

}

}

}

}

## 训练案例12

### 训练考核知识点

自定义异常

### 训练描述

请使用代码实现

每一个学生(Student)都有学号,姓名和分数,分数永远不能为为负数

如果师徒给学生赋值一个负数,抛出一个自定异常

### 操作步骤描述

1. 创建包com.itheima.level1\_12
2. 定义异常类NoScoreException,继承RuntimeException
   1. 提供空参和有参构造方法
3. 定义学生类(Student)
   1. 属性:name,score
   2. 提供空参构造
   3. 提供有参构造;
      1. 使用setXxx方法给名称和score赋值
   4. 提供setter和getter方法
      1. 在setScore(int score)方法中
         1. 首先判断,如果score为负数,就抛出NoScoreException,异常信息为:分数不能为负数:xxx.
         2. 然后在给成员score赋值.
4. 定义测试类Test12
   1. 提供main方法,在main方法中
      1. 使用满参构造方法创建Student对象,分数传入一个负数,运行程序
      2. 由于一旦遇到异常,后面的代码的将不在执行,所以需要注释掉上面的代码
      3. 使用空参构造创建Student对象
      4. 调用setScore(int score)方法,传入一个正数,运行程序
      5. 调用setScore(int score)方法,传入一个负数,运行程序

异常类：

**public** **class** NoScoreException **extends** RuntimeException{

**public** NoScoreException() {

**super**();

}

**public** NoScoreException(String message) {

**super**(message);

}

}

Student类：

**public** **void** setScore(**int** score) {

**if** (score<0) {

**throw** **new** NoScoreException("分数不能为负数");

}**else**{

**this**.score = score;

}

}

}

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

// Student s1= new Student("aa",-5);

Student s2=**new** Student();

s2.setScore(100);

s2.setScore(-5);

}

}



# 关卡2

## 训练案例1

### 训练考核知识点

复制文件

异常处理

### 训练描述

请使用代码实现

写一个方法用于复制文本文件;

要求:使用try...catch...finally对异常进行处理

### 操作步骤描述

1. 创建包com.itheima.level2\_01
2. 定义测试类Test13
   1. 写一个复制文件的静态方法:copyFile(String src,String dest)
      1. 这里的src为源文件路径,dest为目标文件路径
      2. 定义BufferedReader br变量,初始值为null
      3. 定义BufferedWriter bw 变量,初始值为null
      4. 写try...catch...finally代码块
      5. 在try代码块中
         1. 创建BufferedReader的对象,绑定源文件,并把对象赋值给br
         2. 创建BufferedWriter的对象,绑定目标文件,并把对象赋值给bw
         3. 定义一个字符数组 chs,长度为1024,用于临时存储读取到的数据
         4. 定义整型变量len,用于记录读取到的有效字节个数
         5. 使用while循环,读一个字符数组,写一个字符数组,并刷新流
      6. 在catch代码块中,打印异常信息
      7. 在finally代码块
         1. 写一个try...catch代码块
         2. 在try代码块中: 如果bw不为null就关闭bw
         3. 在catch代码块中打印异常信息
         4. 在写一个try...catch代码块
         5. 在try代码块中,如果br不为null就关闭br
         6. 在catch代码中打印异常信息
   2. 提供main方法,在main方法中
      1. 在main方法,调用刚刚写copyFile方法,传入源文件路径和目标文件路径

**public** **class** Test13 {

**public** **static** **void** main(String[] args) {

*copyFile*("content.txt","aa.txt");

}

**public** **static** **void** copyFile(String src,String dest){

BufferedReader br = **null**;

BufferedWriter bw = **null**;

**try** {

br = **new** BufferedReader(**new** FileReader(src));

bw = **new** BufferedWriter(**new** FileWriter(dest));

**char** chs[] = **new** **char**[1024];

**int** len;

**while**((len=br.read(chs))!=-1){

bw.write(chs);

bw.flush();

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

**try** {

**if**(br!=**null**){

br.close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

**try** {

**if**(bw!=**null**){

bw.close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

}

## 训练案例2

### 训练考核知识点

自定义异常

使用代码验证知识点

继承

方法重写

使用throws的使用

### 训练描述

通过代码验证

1. 方法重写,子类只能抛比父类小的异常(父类所抛异常或父类所抛异常的子类)

a. 重写方法不能抛出与父类平级的其他异常

b. 重写方法不能抛出比父类高级的异常

c. 重写方法: 可以抛出和父类同样的异常

d. 重写方法: 可以抛出父类抛出异常的子类异常.

e. 重写方法: 可以抛出多个 "父类抛出异常的子类异常".

2. 方法重写, 父类没有抛异常,子类重写父类方法也不能抛异常?

a. 父类不抛出异常,子类也不能抛出异常

b. 如果子类内部抛出了异常(或调用抛出异常的方法)只能自己处理,不允许向外抛.

注: 这里所说的异常都是编译异常

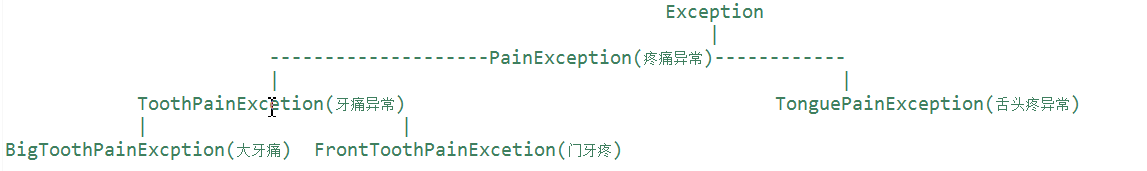
### 操作步骤描述

分析:

1. 要完成上面的验证必须要一个异常体系,可以使用现有的,也可以自定义异常体系;我们这里自定义异常体系,注意异常体系中的所有异常都是编译异常;
2. 定义用于测试的父类和子类,然后逐条验证上面的说法是否正确
3. 把报编译异常的代码可以使用注释,注释起来,上面写明验证的知识点.

步骤:

1. 创建包com.itheima.level2\_02
2. 构建异常体系



* 1. 定义PainExecption(疼异常)类继承Exception
     1. 提供空参和有参构造,注意有参构造必须调用父类的有参构造
  2. 定义ToothPainException(牙疼异常)类和TonguePainException(舌头疼异常)类继承PainException
     1. 提供空参和有参构造,注意有参构造必须调用父类的有参构造
  3. 定义BigToothPainException(大牙疼),FrontToothPainExcption(门牙疼)继承ToothPainException(牙痛)
     1. 提供空参和有参构造,注意有参构造必须调用父类的有参构造

1. 定义用于测试的父类Father
   1. 提供一个吃(eat)的成员方法,抛出牙疼异常
      1. 输出吃到一个石子
      2. 抛出一个牙疼的异常
   2. 提供一个喝(drink)的成员方法不抛出异常
      1. 输出: 喝什么都没有问题
2. 定义用于测试的子类Son,继承Father
   1. 验证: 重写方法不能抛出与父类平级的其他异常
      1. 重写父类的eat方法,抛出与牙疼平级舌头疼异常,编译报错,说明这句话是对的
      2. 注释掉这个重写的方法,验证下一个
   2. 验证: 重写方法不能抛出比父类高级的异常
      1. 重写父类的eat方法,抛出比牙疼高级疼异常,编译报错,说明这句话是对的
   3. 验证:重写方法: 可以抛出和父类同样的异常
      1. 重写父类的eat方法,也抛出牙疼异常,编译通过,说明这句话是对的
   4. 验证:重写方法: 可以抛出父类抛出异常的子类异常.
      1. 重写父类的eat方法,抛出牙疼异常的子类异常大牙疼异常,编译通过,说明这句话是对的
   5. 验证: 重写方法: 可以抛出多个 "父类抛出异常的子类异常".
      1. 重写父类的eat方法,抛出牙疼异常的两个子类异常,大牙疼和门牙疼,编译通过,说明这句是对的
   6. 验证:父类不抛出异常,子类也不能抛出异常(编译时异常)
      1. 重写父类的drink方法,抛出(牙疼,舌头疼,门牙疼)任意一个编译异常,编译都报错,说明这句话是对的
   7. 验证: 如果子类内部抛出了异常(或调用抛出异常的方法)只能自己处理,不允许向外抛.
      1. 重写父类的drink方法
      2. 输出喝到了100度的水
      3. 抛出舌头疼异常
      4. 这个时候编译报错
      5. 把上面两句代码放到try代码块中
      6. 在catch代码块中打印异常信息
      7. 编译通过.

## 训练案例3

### 训练考核知识点

自定义异常类

### 训练描述

请用代码描述:

在一款角色扮演游戏中,每一个人都会有名字和生命值;角色的生命值不能为负数

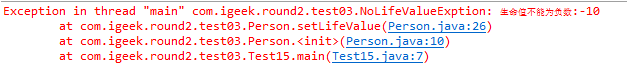
要求:当一个人物的生命值为负数的时候需要抛出自定的异常

### 操作步骤描述

1. 创建包com.itheima.level2\_03
2. 自定义异常类NoLifeValueExption继承RuntimeException
   1. 提供空参和有参构造
   2. 在有参构造中,需要调用父类的有参构造,把异常信息传入
3. 定义Person类
   1. 属性:名称(name)和生命值(lifeValue)
   2. 提供空参构造
   3. 提供有参构造;
      1. 使用setXxx方法给name和lifeValue赋值
   4. 提供setter和getter方法
      1. 在setLifeValue(int lifeValue)方法中
         1. 首先判断,如果 lifeValue为负数,就抛出NoLifeValueException,异常信息为:生命值不能为负数:xxx.
         2. 然后在给成员lifeValue赋值.

3. 定义测试类Test15

* 1. 提供main方法,在main方法中
     1. 使用满参构造方法创建Person对象,分数传入一个负数,运行程序



* + 1. 由于一旦遇到异常,后面的代码的将不在执行,所以需要注释掉上面的代码
    2. 使用空参构造创建Person对象
    3. 调用setLifeValue(int lifeValue)方法,传入一个正数,运行程序
    4. 调用setLifeValue(int lifeValue)方法,传入一个负数,运行程序

NoLifeValueException：

**public** **class** NoLifeValueExption **extends** RuntimeException{

**public** NoLifeValueExption() {

**super**();

}

**public** NoLifeValueExption(String message) {

**super**(message);

}

}

Person：

**public** **void** setLifeValue(**int** lifeValue) {

**if**(lifeValue<0){

**throw** **new** NoLifeValueExption("生命值不能为负数:"+lifeValue);

}**else**{

**this**.lifeValue = lifeValue;

}

}

测试类：

**public** **class** Test15 {

**public** **static** **void** main(String[] args) {

// Person p = new Person("aa", -10);

Person p = **new** Person();

p.setLifeValue(100);

p.setLifeValue(-10);

}

}

## 训练案例4

### 训练考核知识点

自定义异常

### 训练描述

请使用代码描述

写一个方法实现用户登陆,插,传入用户名和密码

如果用户名错误,就抛出自定义登陆异常(LoginException),异常信息为用户名不存在

如果密码错了就也抛出登陆异常,异常信息为密码错误

如果用户名和密码都对了,输出: 欢迎xxx

说明:正确用户名和密码都是admin

### 操作步骤描述

1. 创建包com.itheima.level2\_04
2. 定义异常类LoginException继承Exception
   1. 提供空参和有参构造
3. 创建测试类Test
   1. 提供一个用于登陆的方法login(String name,String pwd),在放方法中
      1. 如果用户名错误,就抛出自定义登陆异常(LoginException),异常信息为用户名不存在
      2. 如果密码错了就也抛出登陆异常,异常信息为密码错误.
      3. 如果能来到下面,就说明用户和密码都是对的,输出: 欢迎xxx
   2. 提供main方法,在main方法
      1. 使用try...catch代码块处理异常
      2. 调用login方法,传入错误用户名,运行程序,报运行时异常,然后注释这行代码
      3. 调用login方法,传入正确用户名,错误的命名,运行程序,报运行时异常,然后注释这行代码
      4. 调用login方法,传入正确的用户名和密码

LoginException类：

**public** **class** LoginException **extends** Exception{

**public** LoginException() {

**super**();

}

**public** LoginException(String message) {

**super**(message);

}

}

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**try** {

// login("bb","123");

// login("aa", "12");

*login*("aa", "123");

} **catch** (LoginException e) {

e.printStackTrace();

}

}

**public** **static** **void** login(String name,String pwd) **throws** LoginException {

**if**(name.equals("aa")&&pwd.equals("123")){

System.***out***.println("欢迎aa");

}**else** **if**(!name.equals("aa")){

**throw** **new** LoginException("用户名不存在");

}**else** {

**throw** **new** LoginException("密码错误");

}

}

}







# 关卡3

## 训练案例1

### 训练考核知识点

集合

IO流

异常处理

### 训练描述

已知集合ArrayList中有, ”张三丰”,”景甜”,”林思意”,”鹿晗”;

把集合中人名写入当前项目的根目录下的names.txt中,每一个人名占一行;

要求: 使用try...catch...finally对异常进行处理

**public** **class** Test {

**public** **static** **void** main(String[] args) **throws** IOException {

ArrayList<String> arrayList = **new** ArrayList<>();

arrayList.add("张三丰");

arrayList.add("景甜");

arrayList.add("林思意");

arrayList.add("鹿晗");

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("names.txt"));

**try** {

**for**(String s:arrayList){

bw.write(s);

bw.newLine();

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

**try** {

**if**(bw!=**null**){

bw.close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

}

## 训练案例2

### 训练考核知识点

集合

IO流

异常处理

### 训练描述

请使用代码描述

把names.txt中的人名,读取到程序中,存储到ArrayList集合中;每一个人名为ArrayList集合中一个元素;

要求:使用try...catch...finally对异常进行处理

**public** **class** Test {

**public** **static** **void** main(String[] args) **throws** IOException {

ArrayList<String> strArr = **new** ArrayList<>();

BufferedReader br = **new** BufferedReader(**new** FileReader("names.txt"));

**try** {

String len=**null**;

**while** ((len=br.readLine())!=**null**) {

strArr.add(len);

}

**for**(String s:strArr){

System.***out***.println(s);

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

**try** {

**if**(br!=**null**){

br.close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

}

## 训练案例3

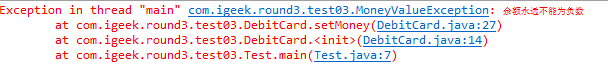
### 训练考核知识点

自定义异常

### 训练描述

请使用代码描述:

每一张借记卡(DebitCard)都有账号(accountNumber):和余额(money); 余额永远不能为负数

要求:当借记卡的余额为负数的时候需要抛出自定的异常

MoneyValueException类：

**public** **class** MoneyValueException **extends** Exception{

**public** MoneyValueException() {

**super**();

}

**public** MoneyValueException(String message) {

**super**(message);

}

}

DebitCard类：

**public** **void** setMoney(**int** money) **throws** MoneyValueException {

**if**(money<0){

**throw** **new** MoneyValueException("余额永远不能为负数");

}

**this**.money = money;

}

}

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) **throws** MoneyValueException {

DebitCard dCard = **new** DebitCard("1001",-1);

}

}