# Java相关知识点回顾与扩展

**第一阶段Java基础课程**

1. 1

Java基础课程内容涉及：Java开发介绍、Java数组、Java面向对象、常用基础类、集合、IO流、多线程、异常、反射。

### 2第一部分：Java开发介绍

1. DOS常用命令

一、DOS的基本介绍

Dos： Disk Operating System 是磁盘操作系统的缩写，是个人计算机上的一类操作系统。

完整的DOS由五部分组成：

1.引导程序（ BOOT ）：由格式化程序直接写入磁盘初始扇区。

2.基本输入/输出管理程序(PC-DOS为IBMBIO. COM、MS-DOS为IO.SYS)。

3.文件管理和系统功能调用程序。

4.命令处理程序(COMMAND. COM)。

5.各种外部命令：完成各种辅助功能的可执行文件。

1.1、通配符

\*号表示0-N个任意字符

？号表示1个字符

主文件名.扩展名 best.jpg ,best是主文件名，jpg是扩展名，扩展名决定了文件的类型

示例：

. 所有文件

请搜索 主文件名中包含字母o的jpg图片 o.jpg o.jpg

请找出d:下所有的文本文件（txt）,且文件名以字符a打头 a\*.txt

主文件名为3个字符的所有类型的文件 ???.\*

找出第个字符为a的所有png图片, ?a\*.png

找出最后一个字符为a的gif图片 \*a.gif,???a.gif

1.2、文件夹与目录

DOS中的目录 == Windows中的文件夹

Windows目录结构 Windows的目录结构采用的是树形结构。

在树形结构中,树根结点没有前驱结点,其余每个结点有且只有一个前驱结点。

叶子结点没有后续结点,其余每个结点的后续节点数可以是一个也可以是多个。

1. JVM、JRE、JDK之间的关系

JVM：java Virtual Machine java运行的虚拟机

可以将编译好的.class文件放在虚拟机上运行

JRE :java runtime Environment Java 运行环境

JDK ：java development kit java开发工具

JDK：

是java开发工具包，是Sun公司针对java开发人员的产品，jdk中包含jre，在jdk的安装目录下有一个jre的目录，里面有两个文件夹bin和lib，在这里可以认为bin里面就是jvm，lib中是java工作需要的类库，而jvm和lib合起来就称为jre

JRE:

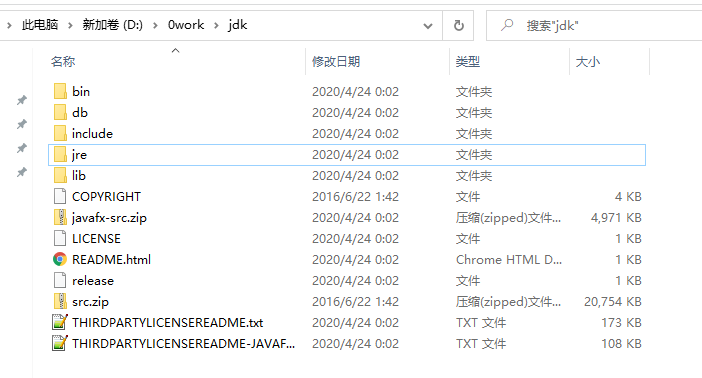
是运行基于java语言编写的程序所不可缺少的运行环境，也是通过它，java的开发者才可以将自己开发的程序发布到用户手中，让用户使用，与大家熟悉的jdk不同，jre是运行环境，并不是一个开发环境，所以没有包含任何开发工具（如编译器，调试器）等，知识针对使用java程序的用户

JVM:

就是我们常说的java虚拟机，它是整个java实现跨平台的最核心的部分，所有的java程序首先被编译为.class文件，这种类文件可以在虚拟机上运行，也就是说class并不直接与机器的操作系统相对应。而是经过虚拟机间接的与操作系统交互，由虚拟机将程序解释给本地系统执行，只有jvm还不能将class执行，因为在解释class的时候jvm需要调用解释所需要的类库lib，而jre包含lib类库，jvm屏蔽了具体操作系统平台的相关信息，使得java程序只需要生成在java虚拟机上运行的目标代码。可以在多种平台（操作系统）上不加修改的运行

关系：JDK包含JRE ，JRE包含JVM

Bin(jvm)+lib=jre+java=jdk



1. Java开发环境的搭建：安装JDK，配置环境变量

**下载，安装，配置windows命令**

**%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin**

在cmd中分别输入java，javac，java -version 命令验证Java环境

1. Java入门程序（Java的开发流程）

略：待补充helloworld

1. Java的注释，标识符、标识符的命名规范

Java的注释：1.单行注释2.多行注释3.文档注释

标识符：

标识符的命名规范

三，java命名规范

1、项目名全部小写

2、包名全部小写

3、类名首字母大写，若类名由多个单词构成，每个单词首字母大写，即驼峰命名法

4、变量名、方法名首字母小写，若其由多个单词构成，每个单词首字母大写，即小驼峰命名法

5、常量名全部大写

6、以上所有命名遵循规则：

名称只能由数字、字母、下划线、$符组成

不能以数字开头

不能是有Java的关键字和保留字

严禁使用汉字和拼音

1. Java基本数据类型

有 byte short int long Boolean float double char

byte类型：

占用字节数：1字节

占用位数：8位

最大值：27-1 (127)

最小值：-27 (-128)

short类型：

占用字节数：2字节

占用位数：16位

最大值：215-1（32 767）

最小值： -215（-32 768）

int类型：

占用字节数：4字节

占用位数：32位

最大值： 231－1

最小值： －231

long 类型：

占用字节数：8字节

占用位数：64位

最大值：263-1

最小值：-263

1. 变量和常量的定义及初始化

(1)变量的定义

变量是内存中的一个存储区域，该区域有自己的名称（变量名）和类型（数据类型），Java中每个变量必须先声明，后使用 该区域的数据可以在同一类型范围内不断变化

定义变量的格式：数据类型 变量名 = 初始化值

1. Java的运算符

1.算术运算符 作用是数字的计算，包括：正号+,负号-,乘\*,除/,余%,加+,减-，其算法与数学中的运算相同。 算术运算符实例（假设变量A=10，变量B=20）

2.位运算将数字转成int型后，把二进制的0当作false，1当作true，每一位进行逻辑运算，运算结果为int型。运算符包括：位非~,位与&,位或|,位异或^,位左移<<,位右移>>,位补零右移>>>位运算实例（假设变量A=60（二进制为00111100），变量B=13（二进制伟00001101））

1）位左移：将int型整数的二进制每位向左移动若干位，前面若干位左移出int范围的bit丢弃，数字位移动到符号位，后面若干位空出的bit补0。如果符号位没有发生改变，则等同于乘2的n次方 2）位右移：将int型整数的二进制每位向右移动若干位，后面若干位右移出int范围的bit丢弃，前面若干位空出的bit全都补符号位数字。等同于除2的n次方。 3）位补零右移：将int型整数的二进制每位向右移动若干位，后面若干位右移出int范围的bit丢弃，前面若干位空出的bit全都补0。正数等同于除2的n次方，负数补零右移后变正数。

3.字符串处理 只有一个操作符"+" ， 作用是将多个字符串连接成1个大的字符串。 字符串运算实例（变量A=“abc”,变量B=123）

4.关系运算 判断2者的关系，返回布尔值。包括：判断相等==,判断不等!=，判断大于>，判断大于等于>=,判断小于<,判断小于等于<=。 关系运算符实例（假设变量A=10，变量B=20）

5.逻辑运算 关于关系的复合运算，返回布尔值。包括：逻辑反!,逻辑与&,逻辑或|,逻辑异或^。 逻辑运算实例（假设布尔变量A为真，B为假）：

1.逻辑短路与(&&)， 与逻辑与(&)不同的是，一旦判断前一关系结果为假，则后续关系不做判断，直接返回结果为假。 2.逻辑短路或(||)，与逻辑或(|)不同的是，一旦判断前一关系结果为真，则后续关系不做判断，直接返回结果为真。

6.赋值运算 自增一++,自减一–,赋值=，自乘\*=,自除/=,自余%=,自加+=,自减-=,自与&=,… 赋值运算实例

7.条件运算符 逻辑判断"?:"， 用法：变量 x = (表达式) ? value if true : value if false ， 解释：逻辑判断?值1：值2，作用是判断逻辑是真还是假，如果是真则返回值1，否则返回值2。 例如：(1>2)?3:4，因1>2为假，所以返回4，即(1>2)?3:4的结果为4。 使用三元运算符时，如果出现类型不一致，java会使用自动拆装箱的机制

8.Java运算符优先级 当多个运算符出现在一个表达式中，会涉及到运算的优先级别的问题，在一个多运算符的表达式中，运算级别的不同会导致结果差别很大。 顺序：算术→位运算→字符串运算→关系运算→逻辑运算→赋值运算

10. Java分支语句之if...else

If();

If(){

}else{

}

If(){

}else if(){

}else if(){

}

If(){

If(){}

}

11. 循环的嵌套

for

while(){}

do{}while()

12. 方法的定义

Public/protected/default/private void/class name(class c){}

13. 方法的形参和实参

形参：只出现在函数定义中，在整体函数内部都可以使用，离开函数则不可以使用

实参：只出现在主调函数中，进入被调函数后，实参变量也不能使用

进阶：内存？

14. 方法的递归调用

方法自身调用自身,方法的发展方向是趋于结束的方向

### 3第二部分：Java数组

1. Java 数组的定义

2. Java 数组的声明

3. 数组的优势与局限

4. 数组的遍历访问（普通循环，增强for循环）

5. 数组元素的顺序查找

6. 数组元素的冒泡法排序

7. Arrays工具类的使用

8. 二维数组

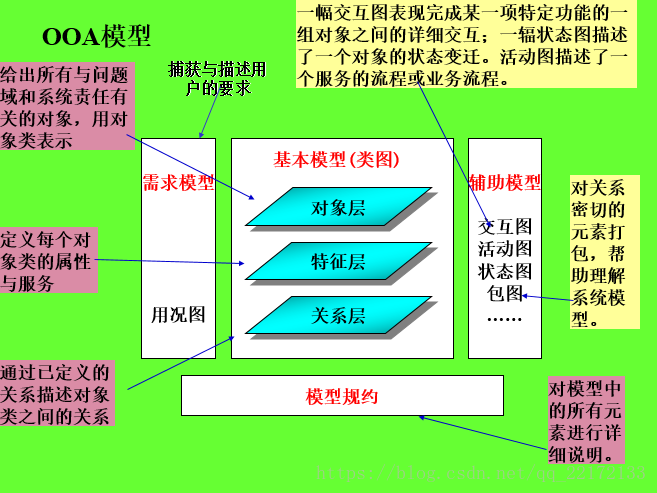
### 4第三部分：Java面向对象

1. 面向对象设计思想

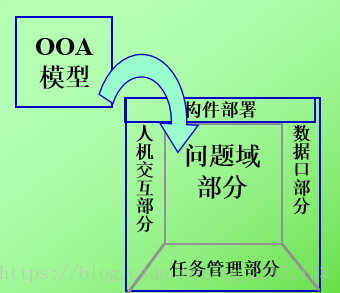
继承 封装 多态

1. 面向对象的分析与设计
2. 面向对象的分析（Object Oriented Analysis, OOA），就是运用面向对象方法进行系统分析。其基本任务即运用面向对象方法，对问题域和系统责任进行分析和理解，找出描述问题域及系统责任所需的对象，定义对象的属性、操作以及它们之间的关系。其目标是建立一个符合问题域、满足用户需求的OOA模型。
3. OOA是分析，是软件生命周期的一个阶段，具有一般分析方法共同具有的内容、目标及策略；强调运用面向对象方法进行分析，用面向对象的概念和表示法表达分析结果。
5. 问题域（problem domain）：被开发系统的应用领域，即在现实世界中由这个系统进行处理的业务范围。
6. 系统责任（system responsibilities）：所开发的系统应该具备的职能。

2、OOA模型

1. 

四、面向对象设计

1. 从OOA到OOD不是转换，是调整和增补。使OOA作为OOD模型的问题域部分；增补其它四个部分，成为完整的OOD模型。
2. 
3. 有不同的侧重点和不同的策略
4. OOA主要针对问题域，识别有关的对象以及它们之间的关系，产生一个映射问题域，满足用户需求，独立于实现的OOA模型。
5. OOD主要解决与实现有关的问题，基于OOA模型，针对具体的软、硬件条件（如机器、网络、OS、GUI、DBMS等）产生一个可实现的OOD模型。
6. Java与面向对象

面向对象中的基本原则

1.抽象

过程抽象：任何一个完成确定功能的操作序列，其使用者都可把它看作一个单一的实体，尽管实际上它可能是由一系列更低级的操作完成的。

数据抽象:根据施加于数据之上的操作来定义数据类型，并限定数据的值只能由这些操作来修改和观察。

客观事物－>对象－>类－>一般类

不同开发阶段需要进行不同程度的抽象，便于实现模块的可替换性

2.封装

把对象的属性和操作结合成一个独立的系统单位，并尽可能隐蔽对象的内部细节。只是向外部提供接口，降低了对象间的耦合度。

封装的意义：

使对象能够集中而完整地描述并对应一个具体事物。

体现了事物的相对独立性，使对象外部不能随意存取对象的内部数据，避免了外部错误对它的“交插感染”。

对象的内部的修改对外部的影响很小，减少了修改引起的“波动效应”。

公开静态的、不变的操作，而把动态的、易变的操作隐藏起来。

封装机制保证：数据不能被对象的使用者直接访问。只允许通过由对象提供的方法或代码访问数据。

3.委托

借助消息传递，工作可从一个对象（客户）传递到另一个对象（代理），因为从客户的观点，代理具有客户所需要的操作。工作连续地传递，直到到达了既有数据又有方法（代码）能完成这项工作的对象。

注意：委托是执行任务的权利，而不是责任。

4.分类

把具有共同性质的事物划分为一类，得出一个抽象的概念。分类帮助我们组织我们所生活的复杂世界。我们可以对在一个特殊分类中的对象做一些假设。如果一个对象是分类（类）的一个实例，它将符合该分类的总体模式。所有的对象都是类的实例。实例能够在运行时被产生（初始化）或销毁（删除）。对象怎样提供操作,由该对象为其实例的类所决定。这样，同一个类的所有对象在响应特定的操作请求（功能调用）时使用相同的方法

5.泛化

无多态性的泛化：类可以由层次继承结构所组织。在该结构中，子类将从位于层次结构高层的父类中继承属性、操作和关系。抽象的父类是指仅用来定义子类的超类。这样，抽象类就没有直接的实例。

有多态的泛化：可以使用层次继承结构组织类，子类可以继承位于层次结构的高层的父类的属性、操作和关系。然而，子类可以定义它自己的操作来代替其任何超类的同名操作。

6.消息通信

即要求对象之间只能通过消息进行通讯。消息传递机制与函数调用机制的区别 ：

在消息传递机制中，每一个消息被发送给指定的接收者（对象）。在命令式编程范型中，函数调用机制没有指定的接收者。

消息的解释（用来完成操作请求的方法或操作/代码集）依赖接收者，并且因接收者的不同而异。

在面向对象的范型中，通常在运行时才能知道给定消息的特定的接收者。

7.关系机制

关系机制为我们提供了用同等（关联、依赖）和层次（一般化/特殊化和聚合）结构组织类/对象的方法。很多面向对象的专家把模型的这部分结构称作静态模型。我们使用James Martin和James Odell的术语，将其称为结构分析。然而，一个应用/系统有了结构分析并不充分，还需要进行行为分析。行为分析是我们用来考察一个对象（类）是怎样提供它的操作的过程。 从分析的视点，有两种类型的行为：

静态的：在静态行为中，实现操作的代码不被任何外部或内部的事件（动作）所影响。

动态的：在行为中发生这些变化的原因可能是由于对象存在很多不同的状态。随后，对象根据它的状态做出反映。使用命令式编程技术不能很好地处理这种类型的行为。使用另外的一种称为有限状态机的机制会更好地捕获这样的方法。

8.复杂性控制

引入包(package) 的概念，使模型具有大小不同的粒度层次，以利于控制复杂性。 如把分析和设计阶段的模型分别用包进行组织。

说明：书上=》抽象、分类、封装、消息通信、多态性、行为分析、复杂性控制

4、关键字说明

对象

对象是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位。一个对象由一组属性和对这组属性进行操作的一组操作构成。

对象标识

对象标识就是对象的名字，有“外部标识”和“内部标识”之分类。

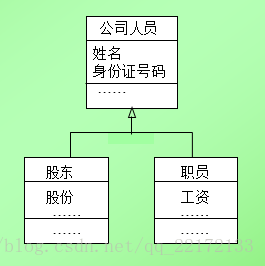
类 是具有相同属性和操作的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述，其内部包括属性和操作务两个主要部分。类的作用是用来创建对象，对象是类的一个实例抽象

抽象（化)忽略事物的非本质特征，只注意那些与当前目标有关的本质特征，从而找出事物的共性。

分类

把具有共同性质的事物划分为一类，得出一个抽象的概念，叫做分类。

继承



特殊类拥有其一般类的全部属性与操作，称作特殊类对一般类的继承。继承意味着自动地拥有，或曰隐含地复制子类从父类中继承属性和操作，根据需要添加自己的属性和方法。继承简化了人们对事物的认识和描述，非常有益于软件复用，是OO技术提高软件开发效率的重要原因之一。一般类与特殊类之间的关系叫泛化关系（继承关系），简称泛化。例：

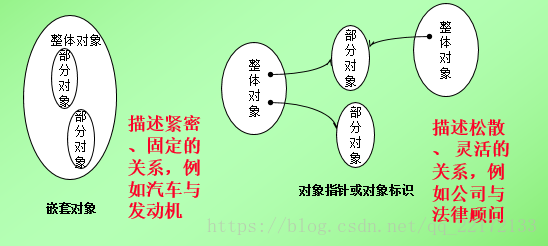
多态

多态是指同一个命名可具有不同的语义。OO方法中，常指在一般类中定义的属性或操作被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。

消息

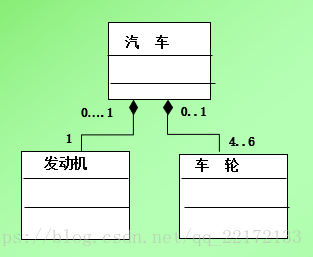
对象通过它对外提供的操作在系统中发挥作用。当系统中的其他对象或其他系统成分（在不要求完全对象化的语言中，允许有不属于任何对象的成分，例如C++程序中的main函数），请求这个对象执行某个操作时，该对象就响应这个请求，完成该操作。在OO方法中，把向对象发出的操作请求称为消息。目前在大部分面向对象的编程语言中，消息其实就是函数（或过程）调用。但是，函数调用只是实现消息的方式之一，上述理解只适合于顺序系统。

聚合



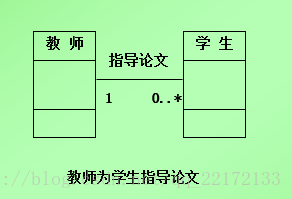
一个（较复杂的）对象由其他若干（较简单的）对象作为其构成部分，称较复杂的对象为聚集，称较简单的对象为成分，称这种关系为聚合。如：

UML表示举例：



关联 类之间的静态联系称作关联。在实例化后，由类产生对象，由关联产生连接对象的链。

链是关联的实例。 关联的表示符号也称作实例连接：



5、面向对象优点

与传统方法相比，面向对象方法的主要优点：

把易变的数据结构和部分功能封装在对象内并加以隐藏，

一是保证了对象行为的可靠性；

二是对它们的修改并不会影响其他的对象，有利于维护，对需求变化有较强的适应性。

封装性和继承性有利于复用对象。

把对象的属性和操作捆绑在一起，提高了对象（作为模块）的内聚性，减少了与其他对象的耦合，这为复用对象提供了可能性和方便性。在继承结构中，特殊类对一般类的继承，本身就是对一般类的属性和操作的复用。

1. 类中成员变量的定义与意义

成员变量声明

构造器声明

成员方法声明

1. 构造方法的定义和调用

Class(){super}

1. 面向对象的封装特性

 封装（Encapsulation）是面向对象方法的重要原则，就是把对象的属性和操作（或服务）结合为一个独立的整体，并尽可能隐藏对象的内部实现细节。

**封装的优点**

1. 良好的封装能够减少耦合。
2. 类内部的结构可以自由修改。
3. 可以对成员变量进行更精确的控制。
4. 隐藏信息，实现细节。
5. 局部变量和成员变量的作用域问题

成员变量：

（1）定义在类里，方法之外的变量。

（2）成员变量可以不显式初始化，它们可以由系统设定默认值；

（3）成员变量在所在类被实例化后，存在堆内存中

局部变量：

（1）定义在方法体内部的变量。

（2）局部变量没有默认值，所以必须设定初始赋值。

（3）局部变量在所在方法调用时，存在栈内存空间中。

8. 静态属性、静态方法、静态代码块

用static修饰符修饰的属性和方法叫作静态属性和静态方法

静态代码块

静态代码块只能写在类中方法外，不能写在方法中，它会随着类的加载而优先于各种代码块和构造方法的加载，并且只会加载一次，如果出现多个静态代码块，会按照书写顺序加载

9. 面向对象的继承特性

继承是使用已存在的类的定义作为基础建立新类的技术，新类的定义可以增加新的数据或新的功能，也可以用父类的功能，但不能选择性地继承父类。

1、子类拥有父类非private的属性和方法。

2、子类可以拥有自己属性和方法，即子类可以对父类进行扩展。

3、子类可以用自己的方式实现父类的方法。（以后介绍）。

10. 继承中构造方法的细节

先执行父类的构造，再执行子类的构造

11. 面向对象的多态特性

Father f = new Son();

12. 抽象方法与抽象类

由abstract修饰的方法为抽象方法，抽象方法只有方法的定义，没有方法体实现，用一个分号结尾；

一个类中如果包含抽象方法，该类应该用abstract关键字声明为抽象类；

如果一个类继承了抽象类，必须重写其抽象方法（除非该类也声明为抽象类）。

13. 接口

 就是一种特殊的类。抽象层次比抽象类更高。

接口中定义的都是抽象方法。

接口的本质就是定义规范。

称为标记型接口

接口可以多实现，抽象类只能单继承。

### 5第四部分：异常

1. 异常的概念

异常就是有异于常态，和正常情况不一样，有错误出现。在java中，阻止当前方法或作用域的情况，称之为异常

1. 异常的分类

Throwable

1.Error是程序中无法处理的错误，表示运行应用程序中出现了严重的错误。此类错误一般表示代码运行时JVM出现问题。通常有Virtual MachineError（虚拟机运行错误）、NoClassDefFoundError（类定义错误）等。比如说当jvm耗完可用内存时，将出现OutOfMemoryError。此类错误发生时，JVM将终止线程。非代码性错误。因此，当此类错误发生时，应用不应该去处理此类错误。

2.Exception

程序本身可以捕获并且可以处理的异常。

运行时异常(不受检异常)：RuntimeException类极其子类表示JVM在运行期间可能出现的错误。编译器不会检查此类异常，并且不要求处理异常，比如用空值对象的引用（NullPointerException）、数组下标越界（ArrayIndexOutBoundException）。此类异常属于不可查异常，一般是由程序逻辑错误引起的，在程序中可以选择捕获处理，也可以不处理。

非运行时异常(受检异常)：Exception中除RuntimeException极其子类之外的异常。编译器会检查此类异常，如果程序中出现此类异常，比如说IOException，必须对该异常进行处理，要么使用try-catch捕获，要么使用throws语句抛出，否则编译不通过

1. 异常的理解

运行中发生一些预料中和没有预料中的会打断程序运行的事件

1. 常见异常介绍

runtimeException IOException EOFException NullPointerException

outofBoundsException

1. 运行时异常

unchecked exceptions（运行时异常）都是RuntimeException类及其子类异常，就是我们在开发中测试功能时程序终止，控制台出现的异常，比如：   
NullPointerException(空指针异常)、   
IndexOutOfBoundsException(下标越界异常)   
ClassCastException(类转换异常)   
ArrayStoreException(数据存储异常，操作数组时类型不一致)   
IO操作的BufferOverflowException异常

6. 编译时异常

 checked exceptions，非运行时异常 （编译异常）：是RuntimeException以外的异常，类型上都属于Exception类及其子类。从程序语法角度讲是必须进行处理的异常，如果不处理，程序就不能编译通过。如IOException、SQLException等以及用户自定义的Exception异常，一般情况下不自定义检查异常。   
通俗的话说就是在写代码时出现红线，需要try catch或者throws时出现的异常。

1. 运行和编译异常区别
2. 异常的处理方式之捕获异常

Try{

}catch(Exception e){

Throw new Excption();

}

1. 异常的处理方式之抛出异常

10. 异常的处理方式之断点调试

11. 断点调试之引导运行步骤

12. 自定义异常

对于具体的业务场景，定义对应的异常类。

13. 异常关键字处理

对于捕获的相关异常信息进行对应的处理。

14. 关键字执行流程

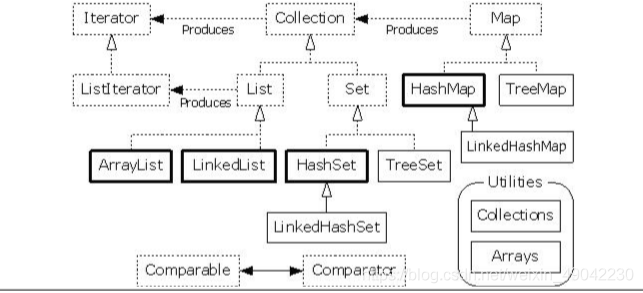
15. 异常总结

### 6第五部分：集合

1. 集合的概念和作用

长度可变数组

#### 集合框架结构图

  
Collection:单值存储的集合  
Map：双子存储的集合（键值对的形式）  
Iterator：迭代器（对所有的集合进行迭代的实现，用迭代器获取数据结构的数据对这个数据结构来说是最优的）

1. 集合和数组的区别



1. 集合框架体系介绍
2. 集合框架之Collection接口

1.集合层次结构中的根界面 。 集合表示一组被称为其元素的对象。 一些集合允许重复元素，而其他集合不允许。 有些被命令和其他无序。 JDK不提供此接口的任何直接实现：它提供了更具体的子接口的实现，如Set和List 。 该界面通常用于传递集合，并在需要最大的通用性的情况下对其进行操作。

包或多重集 （可能包含重复元素的无序集合）应直接实现此接口。

2.所有通用的Collection实现类（通常通过其子接口间接实现88446144404803）应提供两个“标准”构造函数：一个void（无参数）构造函数，它创建一个空集合，以及一个构造函数， Collection ，它创建一个与其参数相同的元素的新集合。 实际上，后一个构造函数允许用户复制任何集合，生成所需实现类型的等效集合。 没有办法强制执行此约定（因为接口不能包含构造函数），而是所有Java平台库中的通用Collection实现。

3.包含在该界面中的“破坏性”的方法，即，修改其经营的收集方法，被指定抛出UnsupportedOperationException如果此collection不支持该操作。 如果是这样的话，可能会，但不要求这些方法，抛出一个UnsupportedOperationException如果调用会对收集没有影响。 例如，如果要添加的集合为空，则可以在不可修改的集合上调用addAll(Collection)方法，但不是必须抛出该异常。

4.Some collection implementations have restrictions on the elements that they may contain.例如，一些实现禁止空元素，有些对它们的元素的类型有限制。 尝试添加不合格元素会引发未经检查的异常，通常为NullPointerException或ClassCastException 。 尝试查询不合格元素的存在可能会引发异常，或者可能只是返回false; 一些实现将展现出前者的行为，一些实现将展现出后者。 更一般来说，尝试对不符合条件的元素进行操作，其完成不会导致将不合格元素插入到集合中可能会导致异常，或者可能会成功执行该选项。 此异常在此接口的规范中标记为“可选”。

5.每个集合决定自己的同步策略。 在没有实现的更强保证的情况下，未定义的行为可能是由于对由另一个线程进行突变的集合的任何方法的调用而导致的; 这包括直接调用，将集合传递给可能执行调用的方法，并使用现有的迭代器来检查集合。

6.在集合框架接口的许多方法在来定义equals方法。 例如，对于在本说明书contains(Object o)方法表示：“返回true当且仅当这个集合包含至少一个元素e使得(o==null ? e==null : o.equals(e))”。 该规范不应该被解释为意味着具有非空参数调用o Collection.contains会导致o.equals(e)被调用任何元素e。 实现可以自由地实现优化，从而避免equals调用，例如，首先比较两个元素的哈希码。 （ Object.hashCode()规范保证具有不等的哈希码的两个对象不能相等。）更一般地，各种Collections Framework接口的实现可以随意使用底层Object方法的指定行为，无论执行者认为合适。

7.执行递归遍历集合的一些集合操作可能会失败，而自引用实例的异常会导致集合直接或间接包含其自身。 这包括clone() ， equals() ， hashCode()和toString()方法。 实现可以可选地处理自引用场景，然而大多数当前实现不这样做。

1. ArrayList和LinkedList的方法使用

ArrayList 基于index的存储，适合作查询，需要从端点迭代查询 不适合删减，增加删除需要移动当前元素和之后的所有元素，时间复杂度O(n)

LinkedList 双向链表的结构适合两端做读取的操作

和插入删除操作 时间复杂度O(1)

1. ArrayList和LinkedList各自的工作原理分析原理分析

LinkedList需要更多的内存，因为ArrayList的每个索引的位置是实际的数据，而LinkedList中的每个节点中存储的是实际的数据和前后节点的位置。

优缺点及使用场景：

（1）如果应用程序对数据有较多的随机访问，ArrayList对象要优于LinkedList对象；

( 2 ) 如果应用程序有更多的插入或者删除操作，较少的随机访问，LinkedList对象要优于ArrayList对象；

（3）不过ArrayList的插入，删除操作也不一定比LinkedList慢，如果在List靠近末尾的地方插入，那么ArrayList只需要移动较少的数据，而LinkedList则需要一直查找到列表尾部，反而耗费较多时间，这时ArrayList就比LinkedList要快。

8. 使用多种方式遍历集合

Iterator it = *list*.iterator();  
 while(it.hasNext()) {  
 Object obj = it.next();  
 System.*out*.println(obj);  
 }

Iterator<Map.Entry<String,String>> ii = *map*.entrySet().iterator();  
while(ii.hasNext()){  
 Map.Entry<String,String> entry = ii.next();  
  
}

9. HashSet和LinkedHashSet各自的工作原理分析

10. 集合框架之Map接口

11. 泛型的使用

12. Collections工具类的使用

### 7第六部分：IO流

1. File类的作用

2. File类中常用方法的使用介绍

3. 使用File类操作文件和文件夹

4. 相对路径和绝对路径的介绍

5. IO流的概念和工作原理

6. IO流的分类

7. 文件流的使用

8. 转换流的使用

9. 缓冲流的使用

10. 对象流的使用

11. 内存流的使用

12. 使用不同的流实现文件内容的拷贝

### 8第七部分：多线程

1. 进程和线程的介绍

2. 进程和线程之间的区别与联系

3. 线程实现方式之继承自Thread类

4. 线程实现方式之实现Runnable接口

5. 线程实现方式之线程池

6. 线程的生命周期

7. 线程中常用方法的使用

END

**第二阶段：JavaWeb**

1. 1

JavaWeb课程内容涉及：HTML5课程、CSS3、JavaScript、MySQL使用、JDBC连接池、Servlet、JSP、AJAX、jQuery、Bootstrap。

### 2第一部分：HTML5

1. html概述

2. html基本标签

3. 图片标签

4. 超链接标签

5. 表格标签

6. 无序列表标签

7. 有序列表标签

8. 定义列表标签

9. div标签

10. 语义化标签

11. 表单标签

### 3第二部分：CSS3

1. CSS3简介和用途

2. CSS3书写规范

3. CSS3基本语法

4. CSS3三种使用方式

5. 标签选择器

6. Id选择器

7. Class选择器

8. 属性选择器

9. 层次选择器

10. 文字、文本属性

11. 背景、列表属性

12. 尺寸、显示、轮廓属性

13. 浮动、定位属性

14. 盒子模型

15. CSS3高级属性

### 4第三部分：JavaScript

1. JavaScript概述

2. JavaScript发展史

3. JavaScript组成

4. 数据类型

5. 运算符

6. 分支语句

7. 循环语句

8. 系统函数、自定义函数、匿名函数

9. 常见事件

10. 数组、字符串、日期对象

11. 正则表达式

12. BOM对象：window、history、location

13. DOM概念

14. DOM操作：创建、修改内容、删除

15. Css样式操作

### 5第四部分：MySQL使用  或oracle数据库

1. 数据库简介

2. MySQL的安装与使用

3. MySQL客户端和服务器配置

4. 用户权限管理

5. DDL创建数据库和表

6. 约束与外键

7. DML实现添加、修改、删除数据

8. 聚合函数、日期函数等

9. DQL简单查询

10. 排序、分组、筛选

11. 多表查询

12. 子查询

13. in和exists关键字

14. 分页查询

15. Mysql SQL语句优化介绍

### 6第五部分：JDBC连接池

1. JDBC概述

2. JDBC使用步骤

3. Connection、Statement、ResultSet接口

4. 使用JDBC 完成数据库DML操作

5. PreparedStatement对象

6. 大数据的操作

7. 批量处理与元数据

8. 事务处理

9. 隔离级别

10. 自定义连接池

11. DBCP、C3p0连接池

12. DBUtis工具类

13. QueryRunner

14. 自定义结果集处理

15. 连接池知识总结

### 7第六部分：Servlet

1. Tomcat与Eclipse的集成开发

2. 什么是Servlet

3. Servlet的进化史

4. Servlet的优化过程

5. Servlet中应用的多线程概念

6. Servlet接收客户端请求信息

7. Servlet响应客户端信息

8. Servlet的生命周期

9. 重定向写法

10. 转发写法

11. 转发和重定向区别

12. HttpServletRequest获取中文乱码问题

13. HttpServletResponse响应中文乱码问题

14. Servlet路径映射问题

15. Servlet多路径映射

### 8第七部分：JSP

1. 什么是JSP

2. JSP的内置对象

3. 透析JSP的本质

4. JSTL+EL

5. MVC介绍

6. 分层的重要性

7. Web工程中事务实现

8. Session的使用

9. Cookie的使用

10. web工程中连接池的使用

11. 实现用户的自动登录

12. 文件的上传、下载

13. 过滤器、 监听器

14. 反射、注解、泛型

15. 综合案例练习

### 9第八部分：AJAX

1. Ajax技术背景

2. Ajax技术的理解

3. XMLHttpRequest使用步骤

4. XMLHttpRequest GET/POST的区别

5. 同步请求&异步请求

6. 同步请求&异步请求的区别

7. Ajax获取不同的返回值类型

8. Ajax使用原生JavaScript和Jquery的区别

9. Ajax响应状态处理

10. JSON语法

11. Java JSON 解析

12. Java JSON 生成

13. Java JSON 第三方工具快速解析

14. JavaScript JSON转换

15. Jquery JSON转换

END

**Java框架课程**

### 1Java框架课程内容涉及：

1. Spring、Spring MVC、MyBatis、SVN、Git、Maven、SpringBoot

### 2第一部分：Spring

1. 框架原理介绍

2. 框架环境搭建

3. 快速入门

4. 创建Bean的方式及实现原理

5. Bean种类

6. Bean生命周期

7. Bean的作用域

8. Bean的注值方式

9. 整合Junit测试

10. 注解装配Bean

11. 注解装配Bean

12. AOP思想、原理解剖

13. AOP底层JDK动态代理

14.AOP编程

15. AspectJ介绍及实现AOP开发

16.事务

17. Web项目集成Spring

### 3第二部分：Spring MVC

1. Spring MVC 概述

2. 使用 @RequestMapping映射请求

3. 请求处理方法签名概述

4. 处理方法签名详细说明

5. 使用 HttpMessageConverter

6. 处理模型数据

7. 数据绑定流程剖析

8. 数据格式化

9. 数据校验

10. 视图和视图解析器

11. 处理方法的数据绑定

12. 视图和视图解析器

13. 文件上传下载

14. JSON处理

15. 拦截器

### 4 第三部分：MyBatis

1. MyBatis数据框架介绍

2. MyBatis对比Hibernate框架

3. MyBatis配置、查询数据

4. MyBatis全局配置文件

5. MyBatis全局配置文件元素详细介绍

6. Mapper映射文件

7. MyBatis两种开发方式

8. MyBatis中别名配置

9. MyBatis # 和 $ 拼接参数区别

10. 关联查询，动态sql

11. MyBatis缓存机制、整合ehcache

12. MyBatis二级缓存处理

13. MyBatis、Spring、SpringMVC整合

14. 分页插件

15. MyBatis逆向工程

### 5第四部分：SVN   和git

1. SVN简介

2. SVN和Git对比

3. SVN服务器搭建

4. SVN创建仓库

5. SVN创建用户以及分配权限

6. SVN签出、提交、更新

7. Eclipse整合svn插件

8. Idea整合svn插件

9. 使用SVN更新提交

10. SVN仓库分析

11. SVN协同修改&冲突解决

12. SVN代码整合注意事项

13. SVN权限管理

14. SVN数据回滚

15. TortoiseSVN

### 6第五部分：Maven

1. Maven环境搭建

2. Maven构建项目

3. Maven本地仓库配置

4. Maven中央仓库配置

5. Maven基本概念

6. Maven基本命令介绍

7. Maven于Eclipse整合

8. Maven于Idea整合

9. Maven创建web工程

10. pom.xml、依赖管理

11. 坐标、依赖、生命周期等

12. Eclipse下的Maven使用

13. Maven中 继承、聚合

14. Maven构建SSH/SSM应用

15. 自动部署、持续集成、持续部署

1. 第六部分

项目开发