Java学习笔记

在Java中，所有的int类型的a,与非零int类型的b满足以下等式：

（a/b）\*b+(a%b) =a

String的常用方法：

CompareTo:

如果指定的数小于参数，返回-1；

如果指定的数大于参数，返回1；

如果指定的数等于参数，返回0；

Date 的常用方法：

Date():生成一个当前的时间对象

Date(long date):根据指定的long类型整数来生成一个Date对象

Boolean after（Date when）:测试日期是否在指定的日期之前

Boolean before(Date when)：测试日期是否在指定的日期之前

getTime()返回该时间对应的long类型整数

Calendar的常用方法：

Void add(int field ,int amount):根据日历规则，为给定的日历添加或者减去制定的时间量，

Int get(int fied):返回给定的时间量

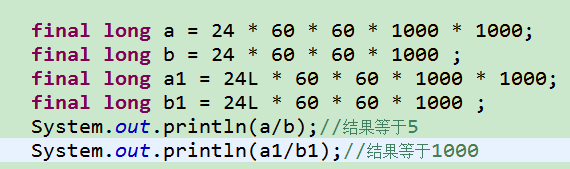
calendar.getActualMaximum(field)：返回指定日历可能拥有的最大值

calendar.getActualMinimum（field）:返回指定日历可能拥有的最小值

add(int field ,amount):改变Calendar特定字段的值，增加amount为正，减少amount为负

roll(int field,int amount):与add()方法类似，区别在于加上amount超过该字段所能表达的最大范围时，也不会想上级进位。

长整数相除或者是相乘，必须先转换为long类型的数值再进行相关操作



\* do while 和 while的区别是

\* while先判断循环条件，如果条件为真的话在执行循环体

\* do while则是先执行循环体，再判断条件，如果条件为真的话在执行下一次循环体

\*switch 语句的数据类型只能是 byte、int、char、short、还有枚举类型和String类型

\* 使用时注意不能省略每个case后面的break

# 正则表达式

String提供的可以匹配正则表达式的方法：

Boolean matches(String regex):判断该字符串是否匹配它指定的值

String replaceAll(String regex,String replacement):将该字符串中所有匹配regex的子串替换成replacement

String replaceFirst(String regex ,String repalcement):将该字符串中匹配regex的第一个子串替换成replacement;

String[] split(String regex):以regex为分隔符，把该字符串分割为多个子串

Pattern是不可变类：

* find()返回目标字符串中是否包含与pattern匹配的的字符串
* group()：返回上一次与pattern匹配的子串
* start()：返回上一次与pattern匹配的子串在目标字符串中的开始位置
* end()：返回上一次与pattern匹配的子串在目标字符串中的结束位置加1
* lookingAt():返回目标字符串前面部分与pattern是否匹配
* matches():返回整个目标字符串与pattern是否匹配
* reset（）将现有的Matcher对象应用于一个新的字符序列。

# Java集合

1. HashSet
2. HashSet是Set的典型实现类，具有良好的存取和查找功能，其中的元素是无序的，不是同步的，集合的值可以是null,HashSet判断两个元素相等是通过比较两个元素的hashCode()和equals（）
3. LinkedHashSet ：LinkedHashSet是HashSet的子类，其中的元素的顺序是有序的，插入的性能低于HashSet;
4. TreeSet：TreeSet是SortSet接口实现类，TreeSet可以使集合中的元素处于排序状态，

二、List

ArrayList和ProrityQueue

三、Map

操作集合的工具类：conllections

void reverser(list):反转list集合中元素的顺序

void shuffle(list):对集合中的元素进行随机排序

void sort(list):对指定集合中的元素按升序排列

void swap(list，int i ;int j):将指定List中的i处的元素跟j处的元素进行交换。

void rotate(list,int distance):当distance是正数时，将list集合后distance个元素移到前面

……………………………………………

同步控制:Conllections提供了多个synchronizedXxx保证线程安全。

# Java IO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 分类 | 字节输入流 | 字节输出流 | 字符输入流 | 字符输出流 |
| 抽象基类 | **InputStream** | **OuputStream** | **Reader** | **Writer** |
| 访问文件 | FileInputStream | FileOutputStream | FileReader | FileWriter |
| 访问数组 | ByteArrayInputStream | ByteArrayOutputStream | CharArrayRreader | CharArrayWriter |
| 访问管道 | PipedInputStream | PipedOutputStream | PipedReader | PipedWriter |
| 访问字符串 |  |  | StringReader | StringWriter |
| 缓冲流 | BufferedInputStream | BufferedOutputStream | BufferedReader | BufferedWriter |
| 转换流 |  |  | InputStreamReader | OutputStreamReader |
| 对象流 | ObjectInputStream | ObjectOutputStream |  |  |
| 抽象基类 | **FilterInputStream** | **FilterOutputStream** | **FilterReader** | **FilterWriter** |
| 打印流 |  | printStream |  | PrintSWriter |
| 推回输入流 | PushbackInputStream |  | PushbackReader |  |
| 特殊流 | DataInputStream | DataOutputStream |  |  |

# Java异常

先捕获小异常，再捕获大异常

一个catch捕获多个异常，捕获多个异常时，异常变量用使用隐性的final修饰

所有的异常包含的方法：

* getMessage():返回该异常的详细描述字符串
* printStackTrace():将该异常跟踪栈信息输出到标准错误输出
* printStackTrace(PrintStram s)将该异常的跟踪栈输出到指定的输出流
* getStackTrace()返回该异常的跟踪栈信息

finally回收资源：程序在try块里面打开了一些物理资源（数据库连接、网络连接、磁盘文件等），这些物理资源必须显示回收。

**异常链：程序先捕获原始异常，然后抛出一个新的业务异常。**

**throws: 当前方法不知道怎么处理这种异常，该异常应该由上一级调用者处理。子类方法声明抛出的异常类型应该与父类的相同或者是父类异常的子类。子类抛出的异常不允许比父类方法抛出的异常多。**

**使用throw抛出异常：可以与catch连用**

**异常处理规则：**

1. **使程序代码混乱最小化**
2. **捕获并保留诊断信息**
3. **通知合适的人员**
4. **采用合适的方式结束异常活动**

# 类加载机制与反射

**JVM进程结束后，该进程在内存中的状态将会丢失。**

**类的加载：**是将类的.class文件加载到内存中，并为之创建一个java.lang.Class对象

**类的连接：**当类被加载之后，会生成一个class对象，接着会将进入连接阶段，连接阶段负责把类的二进制数据合并到jre中，分为三个阶段：验证、准备、解析

**类的初始化：**虚拟机主要负责对类进行初始化，主要就是对类变量进行初始化。在java类中对类变量指定初始值有两种方式：1、声明类变量的时候指定初始值2、使用静态初始化块为类变量指定初始值。JVM初始化一个类遵循如下几个步骤：

* 假如这个类还没有被加载和连接，则程序先加载并连接该类
* 假如该类的直接父类没有被初始化，则先初始化该类的直接父类
* 假如类中还有初始化语句，则系统依次执行这些初始化语句。

当程序首次通过下面的六种方式使用某个类或者是某个接口时，系统就会初始化该类或者接口：

1. 创建类的实例 ：通过new 、反射、反序列化
2. 调用某个类的类方法（静态方法）
3. 访问某个类或者接口的类变量，或者为该类变量赋值
4. 使用反射的方式来强制创建某个类或接口对象的java.lang.Class对象。
5. 初始化某个类的子类
6. 直接使用java.exe命令来运行某个主类

除此之外，下面的几种情形需要特别的指出：

对于final型的了类变量，如果该类变量的值在编译时就可以确定下来，那么这个变量相当于是“宏变量”。Java编译器会在编译时直接把这个类变量出现的地方直接替换成它的值。因此即使程序使用该静态变量，也不会导致

## 类加载器

当JVM启动时，会形成三个类加载器组成的初始类加载器层次结构：

Bootstrap ClassLoader :根类加载器，加载java的核心类库

Extension ClassLoader :拓展类加载器 加载jre的拓展目录

System。ClassLoader：系统类加载器

## 类加载机制

JVM加载机制主要有三种：

1. 全盘负责
2. 父类委托
3. 缓存机制

## 反射

**反射机制和动态代理**

MapReduce

## 输入格式：

Hadoop常用类

Path

YARN：资源管理器、应用管理器

**1、输入分片**

InputFormat负责产生输入分片并将他们分割成记录

一个输入分片就是由一个map来处理输入块，每个map只处理一个输入分片。输入分片在java中被表示为InputSplit接口，InputSplit包含一个以字节为单位的长度和一组存储位置。分片不包含本身数据，而是指向数据引用。（优先处理大文件）。

2.FileInputFormat类

提供两个功能：一个指出作业的输入文件位置；一个是输**入文件生成分片的实现代码段**

一个被**指定为输入路径的目录**，其内容不会被递归处理，如果包含子目录的话，也会被解释为文件，解决：使用一个glob或者一个过滤器根据命名模式限定选择 目录中的文件，或者将mapred.input.dir.recursive设置为true

3.FileInputFormat输入分片

FileInputFormat只会分割比HDFS块大的文件，减少大量小文件可以使用SequenceFile将这些小文件合并成为大文件：将文件内容作为键，文件内容作为值。

4.避免切分

不希望文件被分割，使用FileInputFormat的子类，并重载isSplitable方法返回值设置为false；

5.mapper中的文件信息

处理文件输入分片的mapper可以从作业上配置对象的某些特定属性中读取输入分片的信息，Mapper的Context对象的getInputSplit()方法来实现。当输入的格式源自FileInputFormat时，该返回的的InputSplit可以强制转换为一个FileSplit

6．把整个文件作为一条记录处理

## 文本输入：

TextInputFormat

TextInputFormat是默认的InputFormat。每条记录是一行，key是字节偏移量

KeyValueTextInputFormat

主要是处理key/value格式的输入文件，Key是分隔符前面的字符串

NLineInputForamt

希望mapper收到固定的行数的输入，需要将NLineInputFormat作为InputFormat

多个输入：MultipleInputFormat

数据库输入输出;DBInputFormat(最好用于加载的小量数据集)

## 输出格式:

文本输出:TextOutputFormat:把没条记录写为文本行

二进制输出：SquenceFileOutputFormat

多个输出：MutilpleFileOutputFormat、MutipleOutputs，MultiOutputs输出的文本类型必须要与定义的一致。

## 连接：

Map端连接：

在两个大规模输入数据集之间的map端连接会在数据到达map函数之前就执行连接操作，为达到该目的，各map的输入数据必须先分区并且以特定方式排序。各数据集按被划分成相同数量的分区，并且均按相同的键排序。同一件的所有记录会放在同一分区。

Reduce端连接：

Map为各个记录标记源，然后以连接键为输出键，使得相同的值放在同一reduce中

构建Hadoop 集群

1. 安装java
2. 创建Hadoop用户
3. 安装hadoop
4. 测试安装
5. Ssh配置
6. Hadoop配置

* 配置管理
* 控制脚本
* Master节点场景
* Hadoop守护进程的关键属性

查看hadoop是否处于安全模式

Hadoop dfsadmin –safemode get

在执行某条命令之前，先退出安全模式

Hadoop dfsadmin –safemode wait

进入安全模式

Hadoop dfsadmin –safemode enter

离开安全模式

Hadoop dfsadmin –safemode leave

Hive:

Hive是被设计为mapreduce操作的HDSFS数据，对于大量的数据，使得数据的汇总，查询，分析更加简单。

Hive 一般在工作站上面运行，他把SQL查询转换为一系列在hadoop集群上面的Mapreduce作业，hive把数据组织为表，通过这种方式为存储在HDFS上面的数据赋予结构。元数据（如表的模式）存储在metastore数据库中。

Hive的外壳环境：

HiveQL是和SQL类似，一般是大小写不敏感的，执行hiveQL必须以分号结束，告诉hive立即执行。

Hive对数据的验证并不是在加载数据是进行，而是在查询时进行，称为读时模式，

读时模式可以使数据加载非常迅速，写时模式可以使数据查询性能提高；

Hive 特性：

* Hive不支持更新或者删除，但支持insert into
* Hive的索引有两种：紧凑和位图索引，紧凑型索引存储每个值的HDFS块号，不会占用太多的磁盘空间，位图索引使用压缩的位集合来高效存储具有某个特殊值的行，这种索引一般适合具有少取值的列
* 引入了表级和分区级的锁

## HiveQL

Hive的数据类型：

基本类型：TINYINT、SMALLINT、INT、BIGINT、BOOLEAN、STRING、BINARY、TIMESTAMP

复杂的数据类型：ARRAY：数组；MAP:映射；STRUCT：结构体、

操作与函数：

基本操作：关系操作、空值判断、模式匹配、算术操作以及逻辑操作

内置函数：可以在hive外壳输入SHOW FUNCTIONS查看，或者输入DESRCIBE FUNCTION xxx 查看函数的用法

类型转换：原子数据类型形成了一个hive函数和操作表达式进行隐式转换的层次，显式转换需要CAST操作

## 表

托管表和外部表：

托管表：加载数据到托管表时，Hive把数据移到仓库，加载操作其实就是文件系统中的文件的移动或者重命名，hive同样不检查表目录中的文件是否符合表声明的模式，如果有数据和模式不匹配，只有在查询时才会知道。

丢弃表：DROP TABLE table\_name;（元数据和数据会被同时删除）

外部表：外部表由你来控制数据的创建和删除，丢弃外部表时，hive不会碰数据，而只会删除元数据

## 分区和桶

Hive把表组织为分区（partion）。这是一种根据分区列的值对表进行粗略划分的机制。使用分区可以加快数据分片的查询速度。表或者分区可以进一步分为桶，他会为数据提供额外的结构以获得更高效的查询处理。

分区：创建表的时候用PARTIONED BY子句定义的

CREATE TABLE logs (ts BIGINT,line STRING)

PATITIONED BY (dt STRING ,country STRING);

桶：CLUATERED BY 更高的查询处理效率，取样更高效

CREATE TABLE bucketed(id INT ,name STRING)

CLUSTERED BY (id) INTO 4 BUCKETS;

## 存储格式

Hive从两个维度对表的存储进行管理：“行格式”和“文件格式”。行格式指行和一行中的字段是如何存储的，行格式的定义是SerDe(序列化和反序列化工具)

## 数据导入

INSERT语句

INSERT OVERWRITE TABLE target

SELECT col1，col2

FROM lgs ;

多表插入：

CREATE TABLE … AS SELECT语句

将查询输出的结果存放到一个新的表中

表的修改： ALTER TABLE xxx RENAME to target

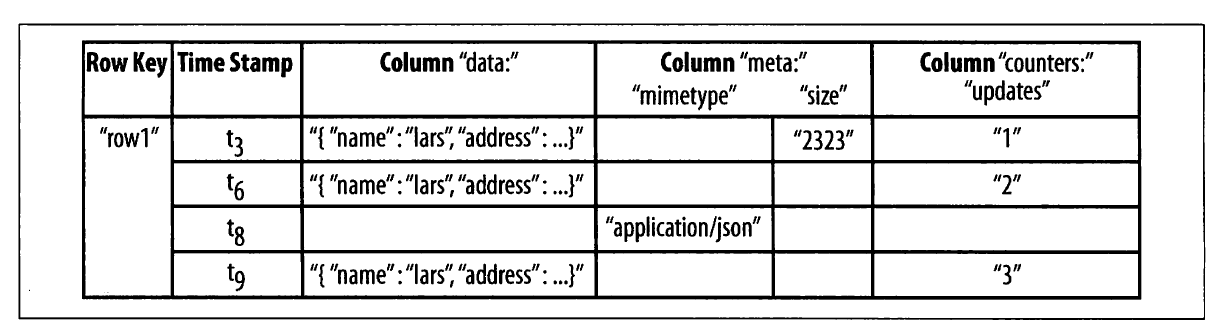
添加一个新列：ALTER TABLE target ADD COLUMNS (clo3 STRING)

HBASE

HBASE是一个在HDFS上面开发的面向列的分布式数据库，如果需要实时地访问大规模数据库，就需要使用HBASE。

**表、行、列、单元格**

最基本的单位是列、一列或者多列形成一行，并有唯一的行键来确定存储，一行由若干列组成，若干列又组成一个列族（column family ）。一个列族的所有列存储在同一个底层的存储文件里面，这个存储文件叫做HFile。列族需要在创建表时就定义好，并且不能修改的太频繁。



HBase中扩展和负载均衡的基本单位称为region ，

HBASE主要有三个组件：客户端库，一体主服务器、多个region服务器