Java.lang 包含了最基本的java库，基本数据类型,Class,String,Enum,character,clone都在此中

Java.io包含了数据的输入流和输出流

Java.lang.reflect包含了反射的函数

Java.text 包含的SinpleDateFormat 日期的设定

Java.util 工具包,一些常用的类库，日期操作都在此种包含了数组map的一些输入和输出操作

Java.text提供了一些国际化的显示处理类库

synchronized 同步的,用于线程问题

strictfp 绑？

transient 短暂的，不能被序列化

volatile 不稳定的

工厂设计:实际操作跟类的汇合点！

数据类型

Long 8\*8=64 Int 4\*8=32 Short 2\*8=16 Byte 1\*8=8

Char 1\*8=8 2位?

Float 4\*8=32 7.4f -6.3e64

Double 8\*8=64 Boolean

三目运算符 max=x>y?x:y;

Switch(oper) { oper为Int char

Case 1:System.out.println();bread;

}

数组的方法 int score[] = new int[3]; //score在栈内存，int[3]在堆内存

Int score[][] = new int[][]={{1},{2}}

System.arraycopy(i1,3,i2,1,3); i3从3开始的3位拷贝到数组i2从1开始后的3位

String常用方法

str = “hello” ;str2 = new String(“hello”);str3=str2;只有str2和str3是相等的

==和str.equals(str)的区别，==比较的是地址,equals比较的是值

Public String(char[] value); //将数组转换为字符串,构造方法

Public String(byte[] value); //将byte数组转换为字符串,构造方法

Public char[] toCharArray();//将字符串转换为数组

Public byte[] getBytes();//将字符串转换为字节数组

Public String toUpperCase();//转化为大写

Public String toLowerCase();//转化为小写

构造方法

一个好的类，必须覆写Object中的equals(),hashCode(),toString()三个方法

Public Boolean equals(Object obj){

If(this==obj){retrun true};

If(!(obj instanceOf Person)){return false;}

Person per = (Person)obj;

If(per.name.equals(this.name)&&per.age=this.age){return true;}

else{return false;}

}

如果确实要继承两个类，可以使用多层继承的方法，但不能多重继承

构造方法的实例化中，会首先编译①静态方法和变量②调用构造方法，如果有父类，则父类静态方法和变量，子类静态方法和变量，父类构造方法，子类构造方法的顺序进行

Override方法的重载:重载的方法一定只是参数上的类型或个数的不同，而不是返回值的不同

抽象类:public abstract void fun();不需要{}就可以定义方法

Java.util.Arrays.sort(int x[]);是最早用于数组排序的算法

Instanceof关键字判断一个对象到底是哪个类的实例，其结构是所有的父类都算是同实例,返回true，但是不包括子类,子类跟父类有关系，但是父类跟子类是没关系的

数据类型的装箱

Integer.MAX\_VALUE 可以获得装箱后数据的最大值

>>>无符号数右移位

异常:finally关键字：出现异常时，一定会执行的代码

Int i =Interget.parseInt(str);将字符串转换成int型

RuntimeException是不需要处理的异常

断言的应用Assert x = 0;与实际相反也会发生错误

Import java.util.\*跟import java.util.Set性能是一样的，没有用到的包不会被导入

Jar :jar –cvf my.jar org 因为c是create创建新的jar包，名字为my.jar

Jar –tvf my.jar查看jar包中的内容

Jar –xf my.jar解压jar包

访问权限：private 同一类default同一包 protected不同包的子类 public其他包

多线程

Thread方法：继承Thread类，覆写其中的run()方法；

MyThread mt1 = new MyThread();

mt1.start();线程的运行

资源不能共享：卖票问题！

Runnable方法：继承Runnable接口，覆写其中的run()方法；

资源可共享，本身自带同步！

线程的状态：创建，就绪，运行，阻塞，终止；

线程的方法：public static Thread currentThead();//返回现在正在运行的线程

Public final void setPriority(int x);//设置线程和获得线程的优先级

Public static void sleep(long millis)throws InterrupttedException;现成的 休眠。

同步与死锁(只有runnable接口共享资源才有的问题):加入延迟效果更明显，线程一开始大家都去取ticket，减一后才还回来，如果返回的速度不够快，就出现了问题！

同步代码块定义：一个支持多线程的类中，被同步的代码块只能允许一个线程进程。

Synchronized(this){if(ticket>0){}};this指的是要同步的对象，一般是this;

同步方法：public synchronized void sale(){if(ticket>0){}};

死锁：在同步的资源中，你需要我的，我需要你的，双方都在等对方的线程完成。

解决方法：一开始就抢占自己所需要的全部资源。

泛型：可以定义多个泛型class Person<T,V>{};<?>通配符

<? extends 类>可以是父类跟子类，<？super 类>只能是父类或Object。

Java常用类库

StringBuffer:

构造方法：StringBuffer buf = new StringBuffer();

主要方法:buf.append(str);//可以是char,string和buf

Buf.length();buf.insert(index,str);buf.delete(start,end);buf.toString();

Runtime:表示运行时的操作lang包中的类

构造方法：Runtime run = Runtime.getRuntime();

Run.freeMemory();run.maxMemory();run.gc();垃圾回收;

run.exec(“notepad.exe”);执行本机命令

ResourceBundle:获取资源文件的方法

Locale:import java.util.Locale;

构造方法：Locale zhLoc = new Locale(“zh”,”CN”);

构造方法：import java.util.ResourceBundle;

Message.properties文件内容：info = hello

ResourceBundle rb = ResourceBundle.getBundle(“message”，locale);

Rb.getString(“info”); //其实是map中的方法

System:

基本方法:System.exit(1);System.gc();垃圾回收;

public static long currentTimeMillis();取得当前时间

以下基本都属于java.util.\*包中

Date:构造方法:Date date = new Date();

Calendar:构造方法 Calendar calendar = new GregorianCalendar();

基本方法:calendar.get(Calendar.YEAR);

DateFormat:

构造方法:DateFormat df = DateFormat.getInstance();df.format(new Date());

Java.text.SimpleDateFormat;

SimpleDateFormat:

构造方法:String pattern = “yyyy-MM-dd HH:mm:ss.SSS”;

SimpleDateFormat sdf = new SimpleDateFormat(pattern);

Sdf.format(new Date());

Math:属于lang包中的static类，可直接调用

基本方法:Math.sqrt(9.0);Math.max(10,30);Math.min(10,30);Math.pow(2,3);

Math.round(33.6);四舍五入的方法

Random:java.util.Random包中，生成各种随机数

构造方法:Random r = new Random();

基本方法:r.nextBoolean();r.nextInt(100);生成小于100的int类型数据;r.nextLong();

NumberFormat:java.text.NumberFormat;设计数据的显示

构造方法:NumberFormat nf = NumberFormat.getInstance();

Nf.format(1000000);得到10, 000,000;

BigInteger:java.math.BigInteger;用于大数的操作

构造方法:BigInteger bi1 = new BigInteger(“123456789”);

Bi.add(bi2);bi.substract(bi2);bi1.multiply(bi2);bi1.divide(bi2);

BigDecimal:java.math.BigDecimal;用于精确计算float,double

构造方法:BigDecimal bd= new BigDecimal(double val);可以Int,float,string都行

基本方法:public BigDecimal add(BigDecimail value);

B1.multiply(b2).doubleValue();

Clone:继承Cloneable接口即可完成克隆

class Person implements Cloneable{};e

Person per2 = (Person)per1.clone();

Arrays:java.util.Arrays;数组,static方法，不需要构造

常用方法Arrays.sort(temp);Arrays.toString(tem);

Comparable<T>:java.util.接口，类要实现排序就实现comparable<T>接口

需要覆写其中的compareTo()方法，就是比较大小，返回值为Int1,0.-1三种

实现后可以用Arrays.sort(per[]);排序

也可以把一个数据定义为Comparable类型

比如：private Comparable data。则在比较同类数据的时候可以data.compareTo(data);

Comparator<T>:java.util接口，运用于类开始没定义，后面补救的方法

public PersonComparator implements Comparator<T>{};

Arrays.sort(per[],new PersonComparator());

Pattren类和Matcher类:java.util.regex.Matcher;java.util.regex.Pattren;设计正则表达

构造方法:String str = “1993-07-27”;String pattern = \\d{4}-\\d{2}-\\d{2};

Pattern p = Pattern.compile(pattren);

Matcher m = p.matcher(str);

boolean matches();

String对正则也是支持的,str1.mathes(pattern);

TimerTask:java.util.TimerTask;定时调度功能,调用的类需要继承TimeTask类

构造方法：MyTask extends TimeTask{};

MyTask mt = new Mytask();

Timer t = new Timer();

t.schedule(mt.1000,2000); //1s后开始，每2s执行一次

JAVA IO

File:操作文件的的类

构造方法:File file = new File(filepath);

基本方法:public static final String separator;

boolean exists();Boolean delete();long length();

RandomAccessFile:自由读取文件类

构造方法:RandomAccessFile raf = new RandomAccessFile(file,”rw”); //以写方式

RandomAccessFile raf = new RandomAccessFile(file,”r”); //以读方式

基本方法:writeBytes(str);writeInt(int);readByte(byte[] b);int readInt();

InputStream和OutputStream:输入流和输出流FileInputStream和FileOutputSteam

构造方法:InputStream input = new FileInputStream(file);

基本方法:input.read(byte[]);input.read();

构造方法:OutputSteam out = new FileOutputSteam(file);

基本方法:out.write(byte[]);out.write(int b);

Writer和Reader：字符输入和输出流FileWriter和FileReader

构造方法:Writer out = new FileWriter(file);

基本方法:write(str);write(char[])；

如果需要转行符，则可以添加”\r\n”

构造方法:Reader input = new FileReader(file);

基本方法:int read();read(char[])；

字节流字符流之间是可以相互转换的

构造方法:Write out = new OutputStreamWriter(new FileOutputSteam(file));

Reader input = new inputStreamWriter(new FileinputSteam(file));

ByteArrayInputStream和ByteArrayOutputStream内存操作流

PipedInputStream和PipeOutputStream管道流

PrintStream打印流,跟c的Printf差不多

构造方法:PrintSteam ps = new PrintStream(new FileOutputSteam(file));

基本方法:ps.print(any);可以输入任何字符

ps.printf();跟C一样的方法%c,%s,%d,%f;

System.out其实就是PrintStream的对象

OutputStream out = System.out;

System.in其实就是InputSteam的实例对象

InputSteam input = System.in;

输入输出重定向System类提供方法

重定向输入方法:System.setOut(new PrintSteam(new FileOutputStream(file)));

重定向输出方法:System.setIn(new FileInputStream(file));

BufferReader:能读入键盘中输入的字符

构造方法:BufferReader buf = new BufferReader(new InputSteamReader(System.in));

基本方法:String readLine(); //执行读取操作

Scanner:java.util.Scanner输入数据的一个类

构造方法:Scanner scan = new Scanner(System.in);

基本方法:Boolean hasNextInt();//下个数字是否为int型

String next(); //接受下个字符串

Float nextFloat(); //接受下个小数

DataOutputSteam:数据的操作,相当于Map

构造方法:DataOutputSteam dos = new DataOutputSteam(FileOutputStream(file));

基本方法:writeInt(int);writeChars(str);

DataInputSteam:数据的操作,相当于map

构造方法:DataInputSteam dos = new DataInputSteam(FileInputStream(file));

基本方法:readInt();readChar();

SequenceInputStream:合并流

构造方法:SequenceInputStream sis = new SequenceInputStream(out1,out2);

Serializable:对象系列化，需要继承此接口

ObjectOutputStream:对象输出接口

ObjectOutputSteam oos = new ObjectOutputSteam(FileOutputSteam(file));

基本方法:write.Object(new Person);

ObjectInputSteam ois = new ObjectInputSteam(FileInputSteam(file));

基本方法:Object obj = ois.readObjct();

java类集:解决对象数组的问题

接口:Collection,Iterator,ListIterator,List,Set,Map,SortedSet,SortMap,Queue,Map.Entry,

Enumeration

List: Collection进行大量扩充，里面的内容是可以重复的；

ArrayList:

构造方法:List<String> list = new ArrayList<String>();

基本方法:list.add(str);add(index,str);

remove(index),remove(str);size();get(index);

Vector:挽救的子类,跟ArrayList几乎一样

Queue:队列接口,实现队列的操作

LinkedList:链表的操作,同时继承了Queue接口

构造方法:LinkedList<String> link = new LinkedList<String>();

基本方法:add(str);addFirst(str);addLast(str);

peek();找到表头poll();找到并移除表头

Set: Collection的子接口，没有扩充，不允许放重复元素,没有顺序

HashSet<String>:

构造方法:Set<String> set = new HashSet<Sring>();

基本方法:add(str);

TreeSet<String>:

构造方法:Set<String> set = new TreeSet<String>();

基本方法:add(str);

SortedSet<String>:

构造方法:SortedSet<String> ss = new TreeSet<String>();

基本方法:add(obj);

Map<K,V>:存放对值的最大接口,跟Collection没关系key---value

HashMap<K,V>:无序存放,key不允许重复，异步方法

构造方法:Map<String,String> map = new HashMap<String,String>();

基本方法:put(K k,V v);增加新的;get(K k);

SortMap:存放一对值，按照key排序,使用比较器排序

TreeMap<K,V>:有序存放,按key排序

构造方法:Map<String,String> map = new TreeMap<String,String>();

基本方法:put(K,V);

Hashtable<K,T>:旧的操作类,key不允许重复,同步方法，效率较慢

IdentityHashMap<K,T>:唯一可以重复的Map子类

Iterator<T>:只能从前向后输出

构造方法:Iterator<String> iter = new ArrayList<String>().iterator();

基本方法:hasNext();next();remove();

ListIterator<T>:可以从后向前输出

构造方法:ListIterator<String> iter = new ArrayList<String>().listIterator();

基本方法:iter.hasNext();iter.hasPrevious();iter.next();

foreach:

方法for(String str:list){};

Enumeration:最早的输出接口,基本已经不用,之前跟vector搭配使用

Map.Entry:保存Map的key-value的实例，每个map都有多个Map.Entry接口实例

Collections:是List，Set的操作集合,可以完成对Set,List各种操作

public int hashCode(){return this.name.hashCode()\*this.age;}

Stack:栈的应用

构造方法:Stack<String> s = new Stack<String>();

基本方法:push(obj);入栈;pop();出栈;

Properties:是Hashtable类的子类,跟Map非常类似的操作,用于文件的操作

构造方法:Properties pro = new Properties();

基本方法:setProperties(String,String);load(new FileInputSteam(file));

枚举:java.lang

构造方法:enum Color{RED,GREEN,BLUE};

基本方法:int ordinal();取得下标；String name();名字

EnumMap:

构造方法:Map<Color,String> desc = new EnumMap<Color,String>(Color.class);

基本方法:Color.values();取的全部内容;desc.keySet();取得所有key;desc.values();

EnumSet:

构造方法:EnumSet<Color> es = EnumSet.allOf(Color.class);

基本方法:add(Color.RED);for(Color c:es);

反射Class:java.lang.reflect;反射就是将所有的类反射到Object中，Object为所有类的父类，可以统一操作，完成操作后可以向下转型完成输出

构造方法:Class.forname(classpath); //通过路径反射,其实就是第三种反射

new Person().getclass(); //Object类中的方法,通过实例反射

Person.class; //直接的类反射

基本方法:T newInstance();需要转型,需要无参构造方法的支持

Constructor<?> cons[] = c.getConstructors();//取得构造方法

Class<?> c[] = getInterfaces();//取得全部接口;

Class<?> c[] = getSupperclass();//取得父类

Method m[] = getMethods();//取得所有方法

Field f[] = getDeclaredFields();//取得本类的属性

Method m = getMethod(name,parameter.class…);取得单个方法

Annotation：java.lang.annotation.Annotation

构造方法:public @interface Annotation的名称{数据类型 变量名称()};

@Override:覆写的Annotation

@Deprecated:不赞成使用的Annotation

@SuppressWarnings:压制安全警告的Annotation

@Retention 设置作用方位 @Retention(value=RUNTIME)

SOURCE:不会被保存到class中,只保存在\*.java中

CLASS:会保存到class中，但不在JVM中运行

RUNTIME:会加载到JVM

@Target 定义Annotation能使用的范围

@Documented 能够生成方法的doc文档，需要在编译的时候 javadoc –d doc \*.java

@Inherited希望被继承

java数据库操作

public static final String DBDRIVER = "org.gjt.mm.mysql.Driver";

public static final String DBURL = "jdbc:mysql://localhost:3306/mldn";

public static final String DBUSER = "root";

public static final String DBPASS = "mysqladmin";

Connection conn = null; <！-- 数据库连接函数 -->

PerparedStatement pstmt = null; <！-- sql语句导入函数 -->

ResultSet rs = null; <!-- 实际写入和读取函数-->

Class.forName(DBDRIVER);

conn = DriverManager.getConnection(DBURL,DBUSER,DBPASS);

String sql = "select empno,ename,job,sal,hiredate from emp";

pstmt = conn.prepareStament(sql);

rs = pstmt.executeQuery();