B站杜老师：

# 一：Java零基础小白入行全攻略视频教程(editplus工具使用)

# 若是要有更高的发展，数据结构需要学习

# 2.学习Java的核心库（JAVASE），集合，IO流，反射机制，网络编程

# 3.搞定数据库的课程，推荐MySQL，同时公司访问量大，会搭建数据库集群，应付高并发

# 4.WEB 前端，从事JAVA开发，主要是BS结构，B端就是WEB前端，包括JS，JQuery框架

# 5.WEB后端，后端可以是多种语言，PHP，C，C++，若选择JAVA，则需要学习Servlet，Filter，EL，MVC，后端学习完，同时需要学习AJAX，AJAX其实不属于后端，属于前端浏览器的技术。到此JAVA基础也是学完了，但是这些技术在实际开发中不会用到，为了开发效率，会使用封装好的框架

# 6.在此做个项目，将所有的知识点串联起来，在项目的开发过程，学习自动构建工具，Maven是要精通的，为了协作开发，Git/SVN是要学会使用，目前git使用次数多

# 7.学习高级框架，新项目大部分使用的都是SpringBoot；对于SSM框架，因为互联网项目需要进行SQL优化，Mybatis的SQL优化很方便，因此SSM使用次数多；对于政府企业使用SSH框架多

# 8.在此做个项目来完成，SSM做个项目

# 9.若薪资需要达到15K，需要学习分布式相关的技术，比如分布式框架Dubbo，SpringCloud，MQ，Redis，Nginx

# 10.学习完后就忘了，是学习方法的问题（不要摘抄代码，要有自己的思想去编写代码，根据老师的思路去思考写代码。先不要注重代码质量，只要功能完成即可）

# 11.学习的框架在公司中很少用，多是公司封装的框架，动力节点中有structs108录课程中详细讲解源码，学习完后学习公司框架容易

# 12.如何学习新技术，每个月抽出时间进行学习

# 三．动力节点史上最全Java基础

# 1）J2SE:平台标准版，可以做桌面程序开发

# J2EE:平台企业版，可以做网站

# J2ME:平台微型版，做移动端程序开发（感兴趣的自己去学习）

# 备注：在JDK5.0之后就J2SE就叫做JavaSE，J2EE是JavaEE，J2ME是JavaME

# 2）JAVA语言的特性

# 简单性：省略了C++的多继承，指针

# 安全性：数据类型检查；运行时堆栈溢出

# 健壮性：运行的垃圾通过gc机制的垃圾自动回收

# 移植性：编译一次，各个平台到处运行

# 3）JVM是java虚拟机，用于运行字节码文件，JVM不可单独安装，解决方案是：安装JDK同时安装了JVM(JDK是开发工具环境)；

# 安装JRE同时安装了JVM（测试环境不需要开发，只需要运行环境则只安装JRE即可）；JRE，JDK，JVM之间的关系：JDK是java开发工具包；JRE是java运行环境，JVM是java虚拟机，JDK安装的同时也就把JRE和JVM安装了，JRE安装的同时也就把JVM安装了备注：JVM不可单独安装。比如：在安装JDK软件中途的时候，提示“是否安装JRE”，此时可以退出安装，因为已经安装了jre，路径在C:\Program Files\Java\jdk1.7.0\_80下面。而路径是C:\Program Files\Java\jre7的jre7可不必安装

# 代码初入：

# public static void main (String [] args){}该方法是程序运行的唯一入口，若一个类中没有此main方法，该类可以编译成功，但无法运行成功

1. public class在源文件中只能存在一个该类
2. classpath的配置可以在任意路径下运行字节码文件(class文件)，默认运行字节码文件只能在当前路径下进行，classpath配置的路径是字节码的存放目录，classpath有两种配置方式，有临时配置和永久配置。永久配置的在系统配置里面配置即可（备注：classpath中加分号和不加分号的区别）
3. 关键字，比如public class 等，关键字都是小写字母，在editplus软件中是以蓝色表示。
4. 标识符，指的是类名称，方法名称，参数名称，变量名称等名称，标识符命名规则：以字母、下划线、$开头，其后是字母、下划线、$、数字，记住不要用关键字命名。建议使用规范的标识符规则。
5. 字面值，指的是整型字面值，浮点型字面值等
6. 字符编码，按某种格式某种规定将数据存在计算机中。比如：ASCII只能存放英文编码；iso8859-1是西欧语言编码，兼容ASCII；GBK/GB2312只能是存中文编码，Unicode统一全球所有国家的文字，常用的有utf-8，utf-16。备注：java源代码采用unicode编码。通常出现乱码的情况是因为编码和解码分别有一套对照表，如果对照表不一致，则导致乱码，编码是将字符编写成对应的码值，解码是将码值转成对应的字符。（1.乱码的原因没懂，需后期补充）
7. 若需要输出变量，则必须声明变量并同时赋予值
8. 转义符，\n，[\\，\t，\u](file:///\\，\t，\u)的用法
9. 变量分类：局部变量-当前方法中声明的和成员变量-类中声明的；基本数据类型和引用数据类型的区别在于内存中存放的内容不同导致的。Java每次运行时，在内存中JVM会自动分配空间，栈，堆，方法区。

栈：方法和局部变量 方法区：代码片段，常量池，静态属性

堆：存方法对象，对象里面存着成员变量

例子：class Day03{

public static void main(String [] args){

int a=3,b=4;

double c=3.4;

String str="abc";

}

}

main方法

5.0x1111=abc

4.str=0x1111

3.c=3.4

3.b=4

2.a=3

1.Day03.class

栈

方法区

堆

1. 数据类型之间的转换和运算符的认识
2. equals和==的使用方式
3. if判断方法，若是等值操作判断的时候，可以用switch case语法，

但switch中的表达式可以是byte，short,int,char类型的，但在jdk7.0以后还可以是String 类型，枚举类型，记住case中加入break，不然出现穿透的现象

1. for，while，do while循环的各个用法
2. break和continue区别，break退出当前的循环体；continue结束本此循环，后面的语句不执行，接着执行下一次循环
3. 方法的基本语法格式

[修饰符] 返回值类型 方法名称([参数列表]){

方法体

}

方法的内存解析，执行到当前方法的时候，则方法进栈，方法执行完，则出栈；当方法中声明的变量是局部变量，其中形参也是局部变量，明白方法调用的时候的内存解析；方法的重载的理解，名字相同，参数列表不同，与返回类型无关。

1. 面向对象的开始

class Student{

//成员变量

String name;

int age;

char sex;

String address;

public void study(){

System.out.println("正在学习中");

}

}

class Test{

public static void main(String []args){

Student su=new Student();

}

}

4.su对象，存放的是堆里面的地址0x1111

args参数

2.main方法

3.new Student()，这里存放成员变量,并赋予了初始化的值

0x1111

1.Test.class

class Student{

//成员变量

String name;

int age;

char sex;

String address;

public void study(){

System.out.println("正在学习中");

}

}

class Test{

public static void main(String []args){

Student su=new Student();

System.out.print(su.name);

su.name="lzm";

su.age="3";

su.sex='男';

System.out.println("姓名"+su.name+"年龄"+su.age);

}

}

上面的成员变量赋值在内存中的位置需要清楚

1. 成员变量和局部变量的区别

成员变量是有初始值（默认值），局部变量是无初始值的；在同一个类中，成员变量和局部变量是可以同名的，优先是局部变量，若用成员变量则前面加上this

1. 面向对象的3大特征

封装、继承、多态

封装，对某些类的属性无法直接调取，使用方法调取属性值，属性设置成Private，本类可以直接调用，外类无法调用，外类若直接调用，使用get方法进行获取。为了安全期间需要使用封装

1. 基本数据类型的赋值

class Method{

public static void print(int a,int b){

a+=5;

b+=10;

System.out.println(a);

System.out.println(b);

}

public static void main(String []args){

int a=0,b=0;

System.out.println(a);

System.out.println(b);

//在调用本类的方法时，可忽略this

print(a,b);

System.out.println(a);

System.out.println(b);

}

}

分析在内存中的解析

10

5

4.print

5.a b

6.print方法调用完后，弹出栈

0

2.a 3.b

0

1. main

引用

引用数据类型赋值在内存中的解析

1. static修饰属性的用法

static 修饰的属性是属于类的，static修饰的属性会存在静态方法区中，若类存在的同时static属性也存在；若对象值为null，则仍可以输出null.static属性的值，因为static存在方法区中，不会出现空指针异常。Static修饰的属性也叫做类的变量，而非static的变量也叫做实例变量

1. static修饰方法的用法

使用的性质和修饰属性一致，其他的性质是static方法只能访问static方法和属性

1. 若初始化类的实例的变量（非静态的属性），可以使用构造方法，若初始化类的类的变量（静态的属性），可以使用static代码块，静态代码块在类第一次被载入时则执行
2. 为何会出现static这种方式？

因为有多个对象时，会产生重复的属性值，从而使用static修饰的属性，不需要每次初始化，从而多个对象都可以使用它。

1. 继承

继承的两大特性：具有传递性（B继承A，C继承B，此时C具有了B和A）和单继承

子类继承父类，无法重写父类的构造方法（规定）；

子类无法重写父类的属性，因为重写是定义于方法的；

子类无法重写父类的静态方法；（规定）

子类无法重写父类的私有方法；

1. 继承的出现，导致出现了重写方法，子类重写了父类，优先调用子类的重写方法。重写的规则和作用范围需要明白
2. Super访问的是父类中，this访问的是子类中的，若调用的属性和方法在子类中不存在，再去父类中去寻找。子类实例化的时候，父类做了何操作？

在实例化子类的时候，首先会找到相匹配的构造函数，在子类当前的构造函数中会默认调用父类的无参构造函数。若调用父类的有参构造函数，则使用super(参数…)来调用。额外的添加的知识点：若类中没有任何构造函数，创建对象的时候，则默认调用空参构造函数；若类种有有参构造函数，创建对象的时候没带参数，则会报错。因此手动编写构造函数的时候，首先编写无参构造，再编写有参构造

继承中构造函数这里一定要注意！！！

1. 多态：
2. 引用名称只能调取父类的属性和方法，但若子类中重写了父类的方法，优先调用子类的方法。将多个对象调用同一个方法，因子类不同，输出不同的结果。
3. 多态的好处

减少代码的冗余性

1. 多态有两种转换形式，向上和向下转型。父类类型 引用名称= new 子类类名()；向下转型， 若访问子类的属性和方法，建议使用instanceof判断对象是否为当前的类，若是，则进行向下转型。
2. 面向对象
3. 明白包之间类的使用方式，使用import导入外包的类
4. Eclipse安装的环境配置：

在jdk安装路径的外面有jre安装包，这是公共的JRE，建议重新安装JDK，不需要安装外面的JRE；

不需要配置classpath，使用set classpath查看配置项，在JDK5版本之后不用配置；

确认JDK的版本；

1. 下载Eclipse，不同的版本对应不同的JDK版本而设计

Eclipse neon JDK8； Eclipse Oxygen JKD9;

Eclipse Photon JDK10

1. Import导入是类，而不是源文件（切记），例子看视频151即可，而不是源码。

34.在我们导入java源文件的时候，往往本地的源文件的编码格式和eclipse工作区编码格式不一致，会出现乱码。源文件是以ANSI编码保存，ANSI不是文件编码，是一种根据操作系统而定的GBK，若是中文的WINDOWS，则ANSI是中文编码的GBK

# 二：Java零基础小白入行全攻略视频教程(eclipse工具使用)

# 1.类的权限：公共类和非公共类

# 公共类的命名要和源文件名一致；

# 只有公共类才能被其他包使用；

# 实例变量的权限：自行脑补权限表格

# Public > Protect > 无修饰符 > Private

# 有权限的目的是遵循权限最小化原则

# 若只需要当前类访问配置Private；

# 若当前类和继承类可以访问则配置Protect；

# final的用法

# 原因：为何会出现final，因为部分属性和类有特殊限制，因此出现final；

# Final修饰的类无法被继承；

# Final修饰的成员变量包括静态变量和实例变量；

static的变量可以通过static块来赋值;final变量只可以显示赋值或者构造方法赋值（语法规定）,final修饰的变量无法二次赋值;通常final和static同时使用；

final修饰的方法，子类可以继承，但无法被重写；

final修饰的局部变量，不可二次赋值；

final修饰的形参变量，方法体中不可再次赋值；

1. 抽象类

抽象类无法实例化对象；

抽象类的引用需要赋值子类对象；

抽象类可以赋值匿名内部类对象；

抽象类中即可有抽象方法，也可以有普通方法；

抽象类被子类继承，子类要重写父类的所有的方法，否则子类要被定义成抽象类

抽象类的出现是为了抽象体现事物才出现的方式

1. 接口

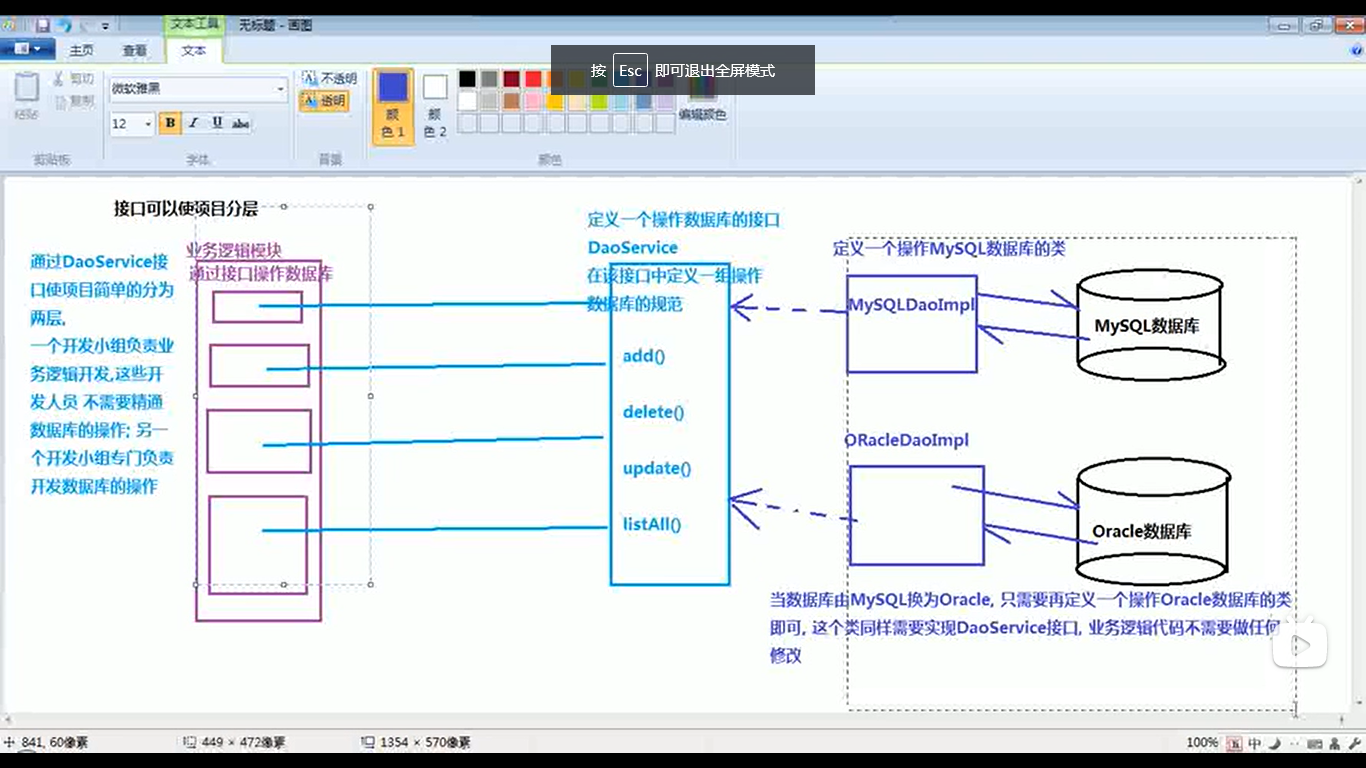
Jdk1.8和jdk1.7在接口中定义的方法有区别，jdk1.8增加了public static和public default方法

接口可以继承接口；类可以多实现接口；

接口的出现是为了对功能的封装和定义一组操作规范

接口在实际编程中使用灵活，接口容易扩展功能

接口使项目分层：



1. 类和类之间的关系
2. 继承（泛化）

一个类继承了另一个类，一个接口继承了另一个接口

1. 实现

一个类实现了接口

1. 依赖

当前类的方法中形参类型，返回值类型，局部变量类型是另一个类的，成当前类依赖另一个类

Person{

Void get(Student student);

}

1. 关联

Person{

Student student;

}

当前类的成员变量使用了另一个类，当前类关联了另一个类

1. 聚合

聚合是关联的一种；A类由很多B类组合而成，但无法决定B类的生命周期，称A类为聚合类。

老师类和学生类

6）组合

组合是关联的一种；A类由很多B类组合而成，但可以决定B类的生命周期，称A类为组合类

人类和细胞

1. 异常

运行时异常和受检异常（编译异常），受检异常定义是在定义方法的时候就抛出了异常。如何对异常进行处理？

try{

}catch(){

通常异常捕捉到日志文件或者打印出来

}catch(){

若捕获的异常有继承关系，则先执行子类，再执行父类；若没有继承关系，catch子句没有顺序

}

finally{

无论是否发生异常，肯定都会执行finally，通常finally是用来释放资源的

}

例如：下面的语句如何执行？

**try** {

**return** a;

}**finally**{

a++;

**return** a;

}

学会分析eclipse中异常栈中的信息, e.printStackTrace();

在异常的情况下对重写的方法有了新的概念，若父类抛出异常，则子类可抛出一样的异常或者是父类异常的子异常或者不抛出异常，重写中子类的返回值可以是父类返回值的子类对象或者相同的，子类修饰符的权限大于父类的

自定义异常，自定义异常的操作如何进行？

自定义异常类的主要作用是区分异常发生的位置

# 三：Java零基础小白入行全攻略视频教程(基本的java计算方式)

# 1.数组的理解,明白数组的a含义？数组如何根据索引找到数值？堆中是连续的存储空间

**int** [] a = **new** **int**[4];

# a[1]=10;

10

0(0x1111) 1 2 3

a 0x1111

2.数组的扩容

数组的扩容可以自定义，可以使用定义好的方法Arrays. copyOf(**int**[] original, **int** newLength)来扩容;

3.数组的初始化

数组的初始化有动态和静态初始化两种；一维数组和二维数组静态初始化的格式

4.二维数组

**int** [][]a = **new** **int**[3][];

定义了一个二维数组，有5个元素，每个元素是int[]类型的一维数组

5.多次练习数组的使用方式

6.二分法查找，快速排序，冒泡排序

7．字符类型的数据实际存放的是码值

**四**：Java零基础小白入行全攻略视频教程(**常用的类**)

1. 通过学习String 类的学习，接触到字符编码含义：字符按照哪种字节存储下来，若按照UTF-8规则，则一个字符占到3个字节；按照GBK规则，则一个字符占到2个字节。编码就是按照规则对字符进行翻译成对应的二进制数（1byte=8bit）。String的字符串和字符数组之间的转化，字符串和字节数组之间的转化；
2. 通过学习包装类的学习，清楚了//int / Interger / String 之间的相互转化，其中字符串转化为基本数据类型，针对取数据库的数据常用。

**int** number3 = Integer.*parseInt*("123");

**double** number4 = Double.*parseDouble*("23.0");

1. 通过学习正则表达式，需要注意的是*String.matcher("\\")括号中的是正则表达式，正则表达式本身就需要一个\,第二个\就会和后面的"构成转义字符*
2. 通过学习类之间的转化，“Ctrl+o”表示打开显示类的成员；
3. Date日期类的转化。
4. 数学类的Math，BigDecimal类

在双引号中的字符串遇到了\，表示转义字符的开始，因此要多加上一个\ 例如： String str = "lizhiming01@126.com ";

String pattern = "\\w{6,}@\\w{2,}\\.(com|net|cn)\t";

1. String 类的再次定义

matches方法里面匹配正则表达式，注意预定义转义字符和转义字符在使用过程中的区别，预定义转义字符需前面多加上\（且注意\w和\t的区别），反斜杠\代表转义字符,如果想表示\w的话,需要写成\\w,来消除影响。

1. String str3 = "123";

String str4 = "123";

System.***out***.println(str3==str4);

//双引号中的字符串是保存到了常量池中，首先将其引用给了变量str3,当执行第二行的时候，会直接把常量池中的 //赋值给了str4;最终两值相等。在jdk1.7之后字符串常量池在堆中存放，之前在方法区中存放。字符串的比较还有几个特殊的地方，常量字符串的优化解读。 P231。因为拼接过程中会产生多个对象，最终得知String 类型不适合拼接字符串，要明白String 拼接的原理

1. 可变字符串，StringBuffer和StringBuider，StringBuider不是线程安全的，StringBuffer是线程安全的。（StringBuffer和StringBuider需好好练习）
2. 包装类

Object的子类是Number，Boolean，Character

其中Number的子类是 [AtomicInteger](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/util/concurrent/atomic/AtomicInteger.html) ， [BigDecimal](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/math/BigDecimal.html) ， [Byte](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/lang/Byte.html) ， [Double](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/lang/Double.html) ， [Float](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/lang/Float.html) ， [Integer](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/lang/Integer.html) ， [Long](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/lang/Long.html) ， [Short](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/lang/Short.html)

//为何基本数据类型可以赋值给引用类型

//基本数据类型自动转化为包装类型，包装类型再赋值给了引用类型，同时该引用类型是包装类的父类，因此可赋值

**public** **class** ComString6 {

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

*m*(1.23);

*m*(**true**);

*m*(3);

*m*("fe");

}

**public** **static** **void** m(Object o) {

}

}

包装类的创建，包装类的使用，包装类如何转化为基本数据类型

总结：记住数组遍历常用的方法

**五**：Java零基础小白入行全攻略视频教程（集合）

1. 需要清楚Collection集合接口的意义和Colletion的扩展图
2. Set集合：无序，不可重复
3. HashSet中添加元素，底层是添加到了HashMap的key中，集合的是HashMap的key所有元素

hashSet.add("add");

hashSet.add("delete");

1. TreeSet底层是TreeMap，TreeSet的所有元素保存到了TreeMap的key中。TreeSet可以指定comparator比较器对元素进行排序。在创建TreeSet的时候创建Comparator的比较器，若没有指定Comparator比较器，则需要TreeSet里面的元素实现Comparable接口，否则TreeSet无法添加元素。TreeSet集合会查看是否有Comparator比较器，若没有，再去找元素实现的Comparable接口。在实际开发中，一般使用Comparable接口去表述默认的比较规则，但若有多个比较规则的时候，则使用Comparator比较器
2. List集合

有序，可重复；

相比于Colletion接口，增加的方法是针对集合元素索引的操作；

若使用remove的方法删除元素，集合的元素是自定义对象，则需要在对象的类中重写equals方法；

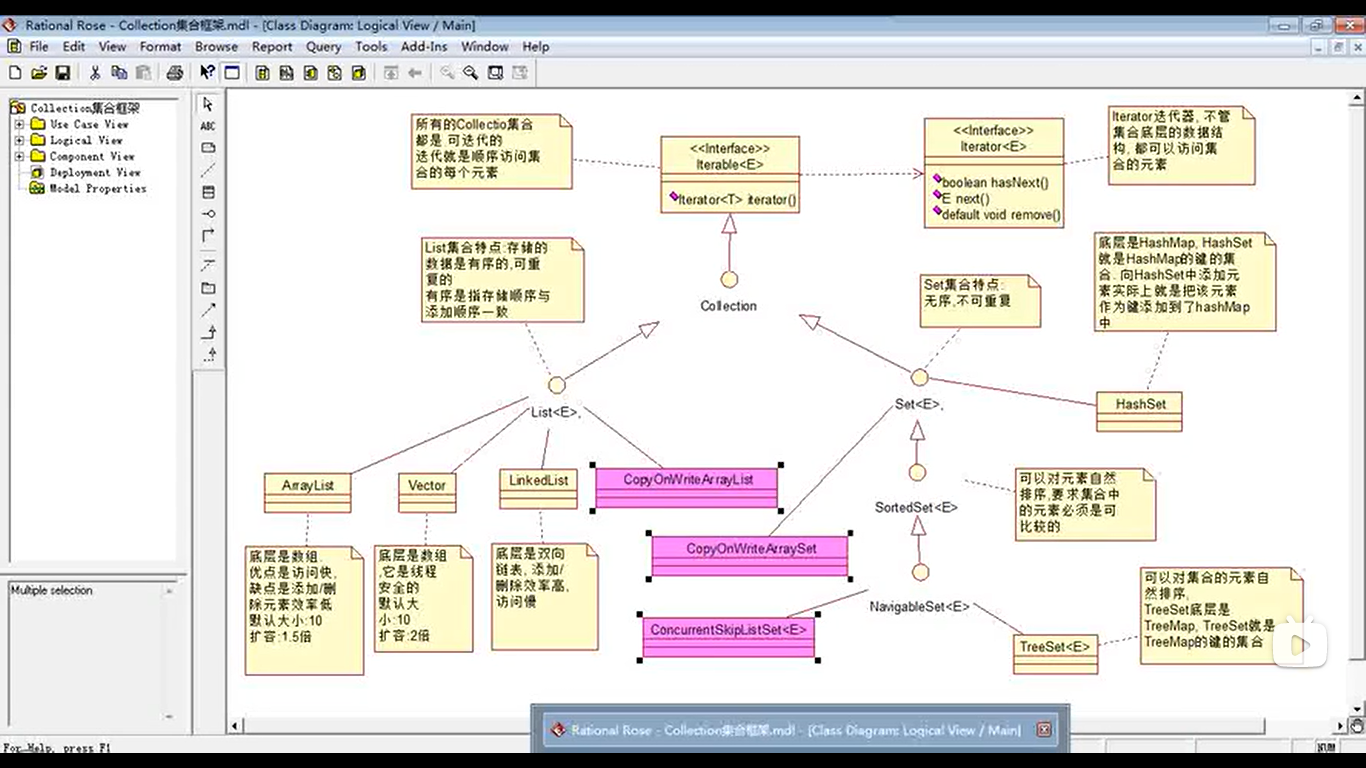
即可实现Comparable接口排序，也可以使用Collection类中的方法进行排序；

ArryList和Vector的区别

1. 总结Collection集合（自己总结）
2. 在集合泛型中，在定义集合的时候泛型才有用，泛型就是引用数据类型

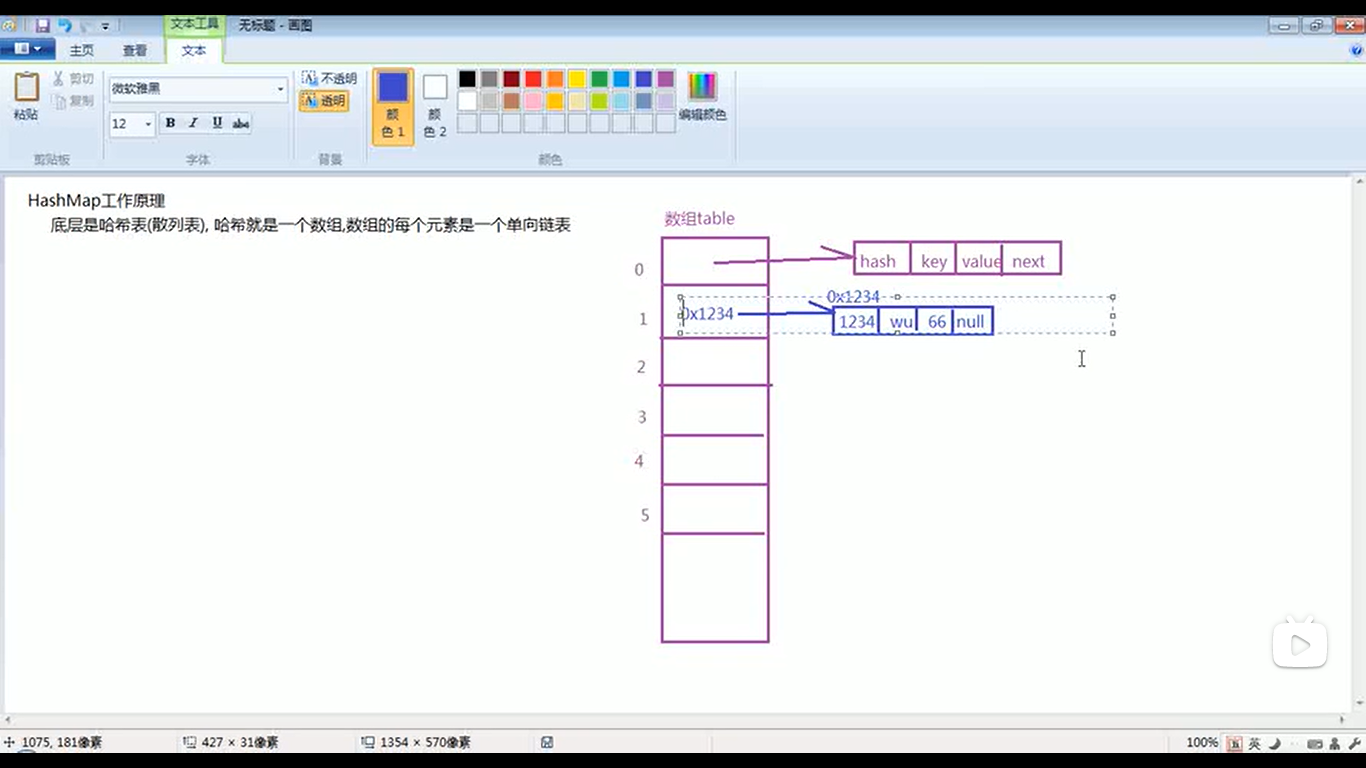
举例：HashSet<String> hashSet = new HashSet<>();

在前面的括号中定义数据类型才能约束数据类型；在后面的括号中不起效果。

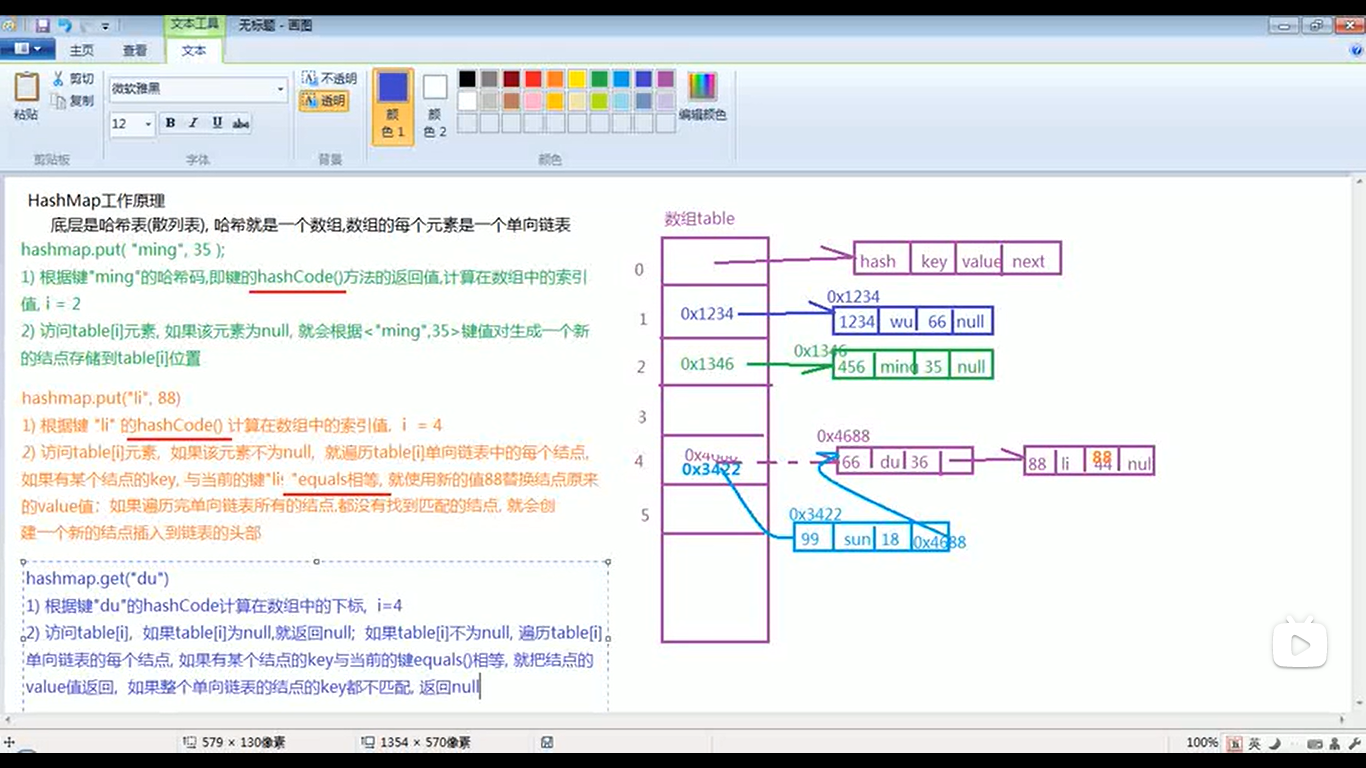


1. 在对于线程安全中有专属的集合工具包[java.util.concurrent](mk:@MSITStore:D:\动力节点\工具包\jdk%20api%201.8_google.CHM::/java/util/concurrent/package-frame.html)
2. Iterator iterator = list.iterator();该方法用于集合的遍历
3. Map的使用，通过灵活使用Map的特点可以解决许多算法的问题
4. HashMap的注意事项：添加的顺序和存储顺序不一致；HashMap的使用方法

何为哈希表：哈希表就是一个数组，数组中的元素存放的是单向链表



1. HashMap的工作原理



1. HashTable和HashMap的区别需要记住，面试用，HashTable在实际工作中不会使用
2. 工作中实际常用的是ProPerties类，该类继承的是HashTable，用于读取properties中的文件信息
3. TreeMap实现了SortedMap，可以根据键自然排序，排序原理是二叉树原理，TreeMap在添加的时候也需要实现Comparator比较器
4. 清楚单向链表和双向链表的区别

**六**：Java零基础小白入行全攻略视频教程（IO流）

IO流

1. 流的概念：流是有起点和终点的有序字节序列
2. 流的分类：

输入流/输出流：以当前程序为起点，程序从外面读取数据的是输入流，程序输出到外面的数据是输出流；

字符流/字节流：以字符为单位处理流的数据是字符流，以字节为单位处理流的数据是字节流；

节点流/处理流：从数据源读取数据即是节点流，处理流是对节点流的包装；

在API的java.io包中，以Stream为结尾的类是字节流，以Reader结尾的类是字符输入流，以Writer结尾的类是字符输出流

1. 输入流和输出流代码写法不同，可以手动关闭流，stream.close；也可以自动关闭流，将创建流写入到try()catch{}语句中的try()的括号里面即可自动关闭流

4. 文本文件的复制，使用FileReader类和FileWriter类进行文件中字符之间的复制，即可以单个字符的复制，也可以字符数组之间的复制。这两种类只能读写与编码环境兼容的文本文件

**1）int** read = fileReader.read();

**2）char**[] c = **new** **char**[1024];

**int** read = fileReader.read(c);

5．如若不理解解码和编码，可以看一下代码

//使用InputStreamReader类可以避免文本文件的编码和当前环境(Eclispse环境)的编码不冲突，因为可以指定编码

/\*从字节流解码到字符流，指定编码集GBK，文本文件是GBK的，这样不乱码

\* InputStreamReader 是字节流通向字符流的桥梁：它使用指定的 charset 读取字节并将其解码为字符\* \*/

InputStream inputStream1 = **new** FileInputStream("E:/source/1.txt");

InputStreamReader inputStream = **new** InputStreamReader(inputStream1,"GBK");

**int** read = inputStream.read();//这里的read是字符的码值

**while**(read!=-1) {

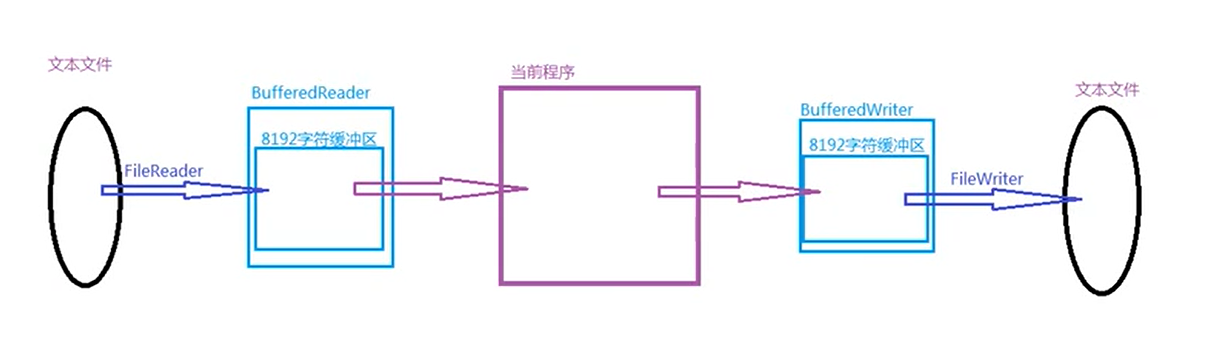
System.***out***.print((**char**)read);

read = inputStream.read();

}

1-5总结：流本身就是字节序列，只针对读写文本文件的类有FileReader和FileWriter，以及InputStreamReader和OutputStreamReader这4中，两种都是读写字符的，FileReader和FileWriter只适用于文本文件和当前程序编码环境一致的情况适用；InputStreamReader和OutputStreamReader可以解决文本文件和当前程序编码环境不兼容的情况，因为可以指定字符集，而这两种转换流采用的是适配器设计模式

6.字符缓冲流



字符缓冲流也称为包装流（处理流），输出字符缓冲流中缓存的数据不是直接写入到文件中，符合以下条件才可写入到文本中，1)bw.flush(); 2)bw.close(); 3)缓冲区数据满了

 把m3方法看懂了，就理解编码和解码了。其中读取的文件是GBK编码

1. 对象序列化

对象的序列化和反序列化，序列化是将对象写入到文件中，反序列化是将文件中的对象读取出来。类实现了Serializable接口，系统会自动为该类添加序列号版本号。若对该类添加或删除某个字段，编译后序列号版本号会重新生成，所以在反序列化的时候出现序列号冲突的异常。解决方案：序列号和反序列号时候序列号版本要一致，因此类里面可以手动添加序列号解决该问题。序列化的作用序列化时为了保持版本的兼容性，即在版本升级时反序列化仍保持对象的唯一性。

1. 字节输入流

1.首先形成输入流通道，读取文件，以字节数组形式进行读取

2.将输入流里面的内容读取到字节数组里面。read（bytes【】）。该方法的返回值是字节数量

3.有一种方法是将字节直接转为字符串， 是new String（bytes），这种方式是将字节数组解码，以字节数组方式进行读取。

进行一个一个字节的读取，最终关闭流，如何对异常进行处理

1.在jdk1.7之前需要手动关闭流通道

2.将流通道的行程，文件内容的读取，最后写入到文件中的过程进行整体的try，catch，最后对流通道进行关闭。

3.fis.close方法会提示流通道未初始化，因为若读取文件产生的流通道失败，则导致fis没有值，因此提示未初始化。若初始化完成了，则会有空指针异常的出现，因此判断若不为空，则进行关闭流。但该方法仍然有ioe exception异常，进行try，catch处理

4.以上就是手动关闭流的处理方式

1.在jdk1.7以后，包括jdk1.7，可以对流进行自动关闭

2.需要将流通道建立在try的括号中，则可以实现自动关闭流

9.FileOutStream字节输出流

10.字节输入流和输出流之间的文件复制

11.字符输入流和输出流之间的文件复制

**七**：Java零基础小白入行全攻略视频教程（线程）

线程的介绍

1.进程和线程

2.进程是属于系统里面的一个程序运行

3.进程里面有多个线程，一个线程就是一个执行单元

4.每个线程都有属于自己独立的栈空间，线程栈都是相互独立的，线程可以共享堆和方法

5.jvm启动的主线程就是main方法，用户线程就是新线程的建立及运行，守护线程是为其他线程做辅助的，不能单独运行，比如jvm里面的垃圾回收就是守护线程

6.因为线程共享堆和方法区的内容，堆里面是对象，实例变量，方法区里面是常量值，导致数据发生不一致的现象，称为线程安全。

7.解决线程安全的方式是，每个线程访问各自的局部变量；若同时访问实例变量，静态变量时，则采用线程同步技术。线程同步技术：

Synchronized(锁对象){同步代码块}，其中锁对象一般是常量代替。

工作原理：

1. 任意对象都可作为锁对象，每个对象中就有一个内置锁，该对象可以是常量，对象，字节码
2. 某一时刻，锁对象最多只能被一个线程持有
3. 若线程持有了锁对象，直到执行完同步代码块才释放
4. 线程要执行同步代码块，必须先获取锁对象

**private** **final** **static** Object ***object*** =**new** Object();

**public** **void** getCount() {

**synchronized** (***object***) {

System.***out***.println("我是"+Thread.*currentThread*().getName()+"当前账户是"+count);

count -= 1000;

System.***out***.println("我是"+Thread.*currentThread*().getName()+"现在账户是"+count);

}

}

1. 要搞清Synchronized(锁对象){同步代码块}锁对象的使用方法，即可灵活使用同步方法，通常锁对象使用的是this。
2. 学会使用synchronized修饰的方法。Synchronized默认的锁对象是当前类对象
3. 以上分别是同步代码块和同步方法使用方法

8.线程中经典的生产消费者模式（）

多个线程扮演生产者，多个线程扮演销售者，生产的产物放到仓库中

若仓库满了，则停止生产，等仓库有位置了，则继续生产；

若仓库空了，则停止销售，等仓库有产品了，则继续销售；

在生产过程中，若没有生产完，则不能销售；

在销售过程中，若没有装满，则不能销售

当仓库满了，其中之一的生产线程在等待，需要notitfy唤醒其他线程，会使用wait和notify。同时注意若唤醒了已经在等待的线程一定需要在判断一下仓库是否满了，因此代码中使用了while语句。

1. 创建线程的方式有3种：

第一种：继承Thread

第二种：实现Runnable接口

第三种：实现Callable接口

实现Callalbe接口和Runnable接口的差异是Callable有返回值

1. 线程的生命周期

在线程的生命周期中，它要经过新建(New)、就绪（Runnable）、运行（Running）、阻塞(Blocked)和死亡(Dead)5种状态