java基础知识点：

1.关键字

含义：被java赋予特殊含义的单词

特点：全部小写

注意事项：

goto和const不是关键字，但是保留字，在以后的jdk版本升级中可能会被提升为关键字

一些高级记事本中会对关键字进行变色

2.标识符

含义：就是为类，接口，方法，变量等起名字的字符序列

组成规则：

英文大小写字母

阿拉伯数字

$和\_

注意事项：

不能以数字开头

不能是java中的关键字

区分大小写

起名字要见名之意

包：就是硬盘上面的文件夹，为了区分同名的文件的

一般是以公司的域名反转之后的前两位作为包名的

[www.edu360.cn->cn.edu360.www->cn.edu360](www.edu360.cn-%3Ecn.edu360.www-%3Ecn.edu360)

一个单词：全部小写        cn

多个单词：全部小写，每级包之间用"."隔开

类和接口：

一个单词：首字母大写

多个单词：每个单词首字母大写

方法名和变量：

一个单词：全部小写

多个单词：从第二个单词开始首字母大写

常量：

一个单词：全部大写

多个单词：全部大写，每个单词之间用"\_"隔开

3.常量

含义：其值在程序执行的过程中不可以发生改变的量

字面值常量

字符串常量        就是用双引号括起来的内容        "哈哈"

整数常量        所有整数，包括正负数        12 -12

小数常量        所有小数，包括正负小数        12.12 -12.12

字符常量        就是用单引号括起来的内容        '男'

布尔常量        只有两个值要么true要么false

空常量                数组章节会说

常量的常用四种变现形式：

计算机电子元件中一般以0和1表示高电压和低电压，也就是关和开，但是两种信号远远不够表示现实生活的数据的，所以就用8个这样的电信号表示一个单位

这个单位就是字节

进制数越大，表现形式越短

二进制：以0b开头，由0和1组成

八进制：以0开头，由0-7组成

十进制：默认就是十进制，由0-9组成

十六进制：以0x开头，由0-f组成

x进制如何转换成y进制？

x进制先转换成十进制：

位值：就是每一位上面对应的数值

进制数：x进制，进制数就是x

次方：从右向左，从0开头编号，对应位上面的编号就是该位的次方

将每一位的位值\*进制数的次方相加就得到了十进制

十进制如何转换成y进制：

除以y直至商为零，余数翻转

一个整数如何在计算机中存储的？

所有的整数在计算机中都是以补码的方式进行存储和运算的

想要知道补码，必须先要知道原码和反码

原码：就是讲一个整数转换成二进制，然后最高位用0表示整数，用1表示负数，其他位表示数值，也叫数值位

反码：正数的反码就是原码，负数的反码所有的位都取反，除了最高位

补码：正数的补码就是原码，负数的补码在反码的基础上末尾+1

自定义常量(面向对象的时候会说)

4.变量和数据类型

变量含义：其值在程序执行的过程中在一定范围之内可以发生改变的量

变量定义格式：

数据类型 变量名 = 初始化值；

变量注意事项：

变量在哪一级大括号定义的，那么这个大括号就是它的作用域，一个作用域中不能出现同名的变量

变量不赋值，不能直接使用

一行上面可以定义多个相同类型的变量，用","隔开

数据类型：

基本数据类型：

整数类型：byte,short,int,long

小数类型：float,double

字符类型：char

布尔类型：boolean

字节        位数        默认值

byte         1                8                 0

short         2                16                0

int         4                32                0

long         8                64                0

float         4                32                0.0

double         8                64                0.0

char         2                16                ''

boolean        1                8                false

默认的小数都是double类型，如果想声明成float类型，要在数值后面加f或者F

默认的整数都是int类型，如果想声明成long类型，要在数值后面加l或者L

数据类型之间的运算规则：

booelan类型不能喝其他数据类型之间运算

byte，char，short之间不直接相互运算，而是转换成int类型之后再相互运算

低精度向高精度转换顺序：

byte,short,char->int->long->float->double

如果一个变量可以接收指定的值，但是因为数据类型不匹配可以使用数据类型强制转换

目标数据类型 变量名 = (目标数据类型)(要被转换的值或者表达式)；

任何数据类型和字符串做运算结果都是字符串，如果遇到括号或者乘除法先运算

表达式的运算顺序：从左向右

引用数据类型：接口，类，数组(暂时不说)

运算符都有一个返回结果

1.算术运算符

单独使用：前++和后++，前--和后--的结果都是一样的

参与运算：

前++：先使自身的值+1，然后再参与运算

后++：先参与运算，然后再使自身值+1

前--：先使自身的值-1，然后再参与运算

后--：先参与运算，然后再使自身值-1

/：如果两个整数相除，那么结果是整数，如果想要得到小数，可以在除数或者被除数后面乘以1.0f

2.赋值运算符

=是最基本的一个赋值运算符

+=，-=，\*=，/=，%=

//包含了隐式强制转换

a += b;->a = (a的数据类型)(a+b);

3.关系运算符

它们的结果都是boolean类型的值

==是否相等的意思

=是赋值运算符

4.逻辑运算符

&和&&的结果都是一样的，&&具有短路效果

与：无false则true，有false则false

|和||的结果都是一样的，||具有短路效果

或：有true则true，无true则false

^：相同false，不同true

!：非真即假，非假即真

在数学中1<x<3是这样写的，但是在java中需要借助逻辑运算符与才能实现

5.位运算符

它是对整数的补码直接进行位运算的

左移：向左移动了几位就相当于乘以2的几次方，最高位会被丢弃，右边空缺的补0

右移：向右移动了几位就相当于除以2的几次方，最高位是0就补0，最高位是1就补1

6.三目运算符

(关系表达式)?表达式1:表达式2;

执行流程：

首先计算出关系表达式的值，看其结果是true还是false

如果是true就执行表达式1

如果是false就执行表达式2

7.选择结构if语句

if语句格式2和三目运算符的关系：

三目运算符能实现的，if语句格式2都可以实现，但是反之不成立

什么时候不成立？

当控制的语句是一条输出语句时，三目运算符就实现不了了，因为三目运算符是一个运算符，必须有一个结果返回

if语句格式3：

if(关系表达式1){

语句体1；

}else if(关系表达式2){

语句体2；

}else if(关系表达式3){

语句体3；

}。。。

else{

语句体n+1;

}

执行流程；

计算出关系表达式1的值，看其结果是ture还是false

如果是true就执行语句体1

如果是false计算关系表达式2的值，看其结果是true还是false

如果是true就执行语句体2

如果是false计算关系表达式3的值，看其结果是true还是false

如果是true就执行语句体3

如果是false计算关系表达式n的值，看其结果是true还是false

如果是true执行语句体n

如果是false，代表上述条件都不满足，执行else里面的语句体n+1

1.选择结构

if语句：

if(表达式1){

语句体1；

}else if(表达式2){

语句体2；

}else if(表达式3){

语句体3；

}...

else{

语句体n+1；

}

执行流程：

执行表达式1的值，看其结果是true还是false

如果是true，则执行语句体1

如果是false执行表达式2的值，看其结果是true还是false

如果是true，则执行语句体2

如果是false执行表达式3的值，看其结果是true还是false

如果true，则执行语句体3

。。。

如果上述条件都不满足，则执行else里面的语句体n+1

注意事项：

if后面的表达式的值都是boolean类型的

if默认只能控制一条语句体，建议不要将大括号省略

switch语句：

switch(表达式){

case 常量值1：

语句体1；

break;

case 常量值2：

语句体2；

break;

case 常量值3：

语句体3；

break;

default:

默认语句体

break;

}

格式解析：

表达式可以取的值：byte,short,int,long,char,String

case：表示匹配的意思，后面的常量值和表达式的值进行匹配

语句体：就是我们要做的事情，可以是一条或者多条语句

break：是结束switch语句的

default：当所有的case都不匹配时，就执行default里面的语句

执行流程：

首先计算出表达式的值，然后和case后面的常量值进行匹配，一旦匹配成功就执行里面的语句体，直到遇到break才会结束，如果没有匹配成功且有default就执行

default里面的默认语句体

注意事项：

case后面只能跟常量值

case和default的顺序可以随意，没有前后之分

break是可以省略的，如果一但省略，case地下的所有语句都会执行，除非遇到break才会结束

default也是可以省略的，如果你不需要默认值，是可以省略的，它是相当于if语句中的else

switch什么时候会结束？

遇到break

执行完毕了switch语句

2.循环结构

for语句：

for(初始化语句；判断条件语句；控制条件语句){

循环体语句；

}

执行流程：

A:执行初始化语句

B:执行判断条件语句，看其结果是true还是false

如果是true，继续C步骤

如果是false，结束for循环

C:执行循环体语句

D:执行控制条件语句，然后回到B步骤

while语句：

初始化语句；

while(判断条件语句){

循环体语句；

控制条件语句；

}

执行流程：

A:执行初始化语句；

B:执行判断条件语句，看其结果是true还是false

如果是true，则继续C步骤

如果是false，结束while循环

C:执行循环体语句

D:执行控制条件语句，然后回到B步骤

do...While语句：

初始化语句；

do{

循环体语句；

控制条件语句；

}while(判断条件语句);

执行流程：

A：执行初始化语句

B：执行循环体语句

C:执行控制条件语句

D:执行判断条件语句，看其结果是true还是false

如果是true则继续步骤B

如果是false，则结束循环

while：更适合求未知数的循环，因为在while循环结束之后，初始化的那个变量还能被访问

for：更适合做固定范围之内的循环

do...while：更适合做循环体语句要被执行一次的循环

while和for它们的循环体语句只有在判断条件返回结果为true的时候才能执行

do...while它的循环体语句至少会被执行一次

3.跳转控制语句

break语句：中断当前循环的意思

continue语句：结束本次循环，直接开始下次循环

return语句：结束当前的方法，将执行权交给上层调用者，return后面的代码都会执行

方法 数组

1.方法：就是完成特定功能的代码块，为了解决代码重复编写问题的

方法格式：

修饰符 返回值类型 方法名(参数列表){

方法体；

return 返回值；

}

格式解析：

修饰符：先阶段就使用public static

返回值类型：由返回值决定的，返回的这个值具体是类型，如果一旦有明确返回值类型了，那么必须有一个对应值返回

方法名：为了方便调用功能，为了方便调用特点代码块

参数列表：接收编写方法体需要的参数，它可以有多个，用","隔开

方法体：就是我们要编写的代码，功能的主体

return：结束方法，并将返回值返回给上层调用者

返回值：是方法体产生的产物，也是调用者关心的结果，由return进行返回

如何定义一个方法？

我们发现只有返回值类型和参数列表当时是不好确定的，但是在编写方法体的过程中发现少什么就通过参数列表传递什么

方法体编写完成之后，结果就产生了，这样一来返回值类型也就确定了

方法的分类：

有明确返回值类型的方法：

单独调用，没有人任何意义，因为它是有返回值的，它是需要你接收然后自己处理这个结果的

输出调用，意义并不大，因为如果仅仅是想要输出，那么就不必返回了，直接在方法内部输出就完事了

赋值调用，这个是有明确返回值类型方法的初衷，一般是将结果进行再加工的

没有明确返回值方法：就是返回值类型写成void，这个时候就不需要返回值了，不能通过return返回任何的值，只能通过return结束方法

方法的注意事项：

方法需要调用才能执行

方法与方法之间是平级关系，它们不能嵌套定义

方法如果有明确的返回值，那么必须通过return返回一个具体的值

方法的参数如果有多个，需要定义每个参数的类型，并用“,”隔开

方法重载：

方法名相同+方法的参数个数或者参数类型不同，和返回值类型无关

2.数组

数组即可以存储基本数据类型，还可以存储引用数据类型

一维数组

就是一个存储相同数据类型元素的集合，其实就是一个容器

数组的定义格式：

数据类型[] 数组名；

但是数组仅仅是声明了，并没有向内存申请空间，所以使用会报错，需要将数组初始化

一维数组的初始化：

动态初始化：由我们指定数组的长度，由系统为每个数组元素赋值

格式1：

数据类型[] 数组名 = new 数据类型[数组长度];

int[] arr = new int[3];//表示定义了一个一维数组，这个数组可以存储3个int类型的值

如何取值：

索引：从0开始编号，最大的索引=数组的长度-1

数组名[索引]

如何赋值：

数组名[索引] = 要赋的值;

java内存分配：

变量都需要先赋值，再使用

栈：

存放的变量都不会赋值，需要我们手动的赋值

放在栈里面的变量叫做局部变量，局部变量就是定义在方法里面或者参数列表上

方法的执行其实是在栈中执行的，又因为方法里面有变量，所以这些变量不会被赋值

栈中的变量会在方法执行完毕之后被释放掉

堆：

存放在堆中的变量，系统会自动为它们赋值

引用类型：null

byte,short,int,long        0

float,double        0.0

char        ''

boolean         false

存放在堆中的东西都是new出来

它们会在栈中没有引用指向的时候被当做垃圾，等待着gc的回收

静态初始化：由我们指定数组元素的值，由系统根据数组元素的个数决定数组的长度

格式：

数据类型[] 数组名 = new 数据类型[]{元素值1，元素值2，元素值3...};

简化格式：

数据类型[] 数组名 ={元素值1，元素值2，元素值3...};

静态初始化和动态初始化不能同时存在

二维数组

小牛学堂每个班有很多学生，每个班的学生可以用一个数组来存储

小牛学堂又有很多个班级，这很多个班级也可以用一个数组来存储，那么这个数组的元素就是一个班，一个班又是一个一维数组

所以这样一来，这就是一个二维数组

二维数组：就是一个数组元素为一维数组的一个数组

格式1：

数据类型[][] 数组名 = new 数据类型[m][n];

m：表示这个二维数组有多少个一维数组

n：表示每一个一维数组有多少个元素

int[][] arr = new int[3][20];//表示小牛学堂有3个班级，每个班级有20个学生

取值：

数组名[m][n]：表示获取数组的第m+1个一维数组中的第n+1个元素

格式2：

数据类型[][] 数组名 = new 数据类型[m][];

m：表示这个二维数组有多少个一维数组，但是每个一维数组的长度没有指定，需要我们手动指定

int[][] arr = new int[3][];表示我申请了一个二维数组中有三个一维数组，每个一位数组的长度没有指定

arr[0] = new int[8];//第一个一维数组的长度为8

arr[1] = new int[4];//第二个一维数组的长度为4

arr[2] = new int[3];//第三个一维数组的长度为3

格式3：

数据类型[][] 数组名 = new 数据类型[][]{{元素值1，元素值2...},{元素值1，元素值2...},{元素值1，元素值2...}};

大括号里面有多少个大括号就表明这个二维数组有多少个一维数组

最里面的大括号有多少个元素，就表明当前的一维数组的长度为多少

简化格式：

数据类型[][] 数组名 = {{元素值1，元素值2...},{元素值1，元素值2...},{元素值1，元素值2...}};

代码块 继承 关键字

1.代码块

静态代码块：随着类的加载而加载，优先于对象而存在，而且只执行一次，一般是用于初始化整个应用程序都必须初始化的东西，比如引入C语言打包好的库

构造代码块：随着对象的创建而调用，优先于构造方法执行的，用于初始化所有构造方法都必须初始化的内容

普通代码块：放在普通代码块里面的变量会在执行完毕之后立马当做是垃圾，等待gc的回收

构造方法，构造代码块，静态方法的执行顺序？

静态代码块-构造代码块-构造方法

2.继承

概述：将多个类中存在的相同属性和行为封装到一个新类中，然后其余的类继承这个新类，然后其余的类都具有了这个新类中的属性和行为

继承格式：

class 子类名 extends 父类名

被继承的类叫做父类，超类或者基类

继承的这些类叫做子类，派生类

子类可以在继承父类的基础上，定义自己特有的属性和行为

继承的好处：

提高了代码的复用性，因为父类中有的非私有成员，子类中都有

提高了代码的维护性，如果父类中一个地方改动了，所有的子类也随着改变

让类与类之间产生了关系，是我们后面学习多态的前提

继承的特点：

java中只支持单继承，不支持多继承，但是支持多层继承

继承的注意事项：

父类中的私有成员，子类不能够被继承

不要为了部分功能去继承

父类中的构造方法，子类不能继承，但是可以通过super进行访问

继承中的子父类成员变量的关系

通过子类的对象中的一个成员方法访问一个变量的查找顺序：

首先在子类的成员方法局部范围找

再在子类的成员范围找

再在父类的成员范围找

如果没有找到，就会报错

super关键字：代表父类的引用

this关键字：代表本类的引用

super和this访问成员变量：

super.成员变量名;                this.成员变量名;

super和this访问构造方法：

super(参数);                        this(参数);

super和this访问成员方法：

super.成员方法名(参数);        this.成员方法名(参数);

继承中的子父类构造函数关系：

子类在进行初始化的时候，首先会初始化父类的成员，因为子类可能会使用到父类中的成员

在子类的构造方法的可执行代码第一行隐式的调用了super();从而访问父类的无参构造方法

访问父类的构造函数方法的代码必须放在可执行代码的第一行

如果父类中没有无参的构造函数怎么办呢？

子类可以通过super去手动调用父类中有参的构造函数

在本类中可以this调用本类的构造函数，通过this调用本类构造函数的代码也必须放在可执行代码的第一行

继承中的成员方法的关系：

通过子类对象访问一个成员方法：

首先在子类成员范围找

再到父类成员范围找

最后如果找不到，就会报错

方法的重写：子类中定义了一个好父类中一模一样的方法，但是子类的方法体又是自己定义的，这样的情况就叫方法的重写；

这样既沿袭了父类的功能，但是功能主体又是子类自己定义的

方法的注意事项：

父类中私有的成员方法，子类不能重写

子类重写的方法权限不能比父类的更低

子类重写父类中的静态方法时，子类的方法也必须用static修饰(不能算作方法的重写，多态的时候证明)

final关键字：

修饰类：类不能被继承

修饰变量：变量就变成了常量，只能被赋值一次，如果是成员常量必须在对象创建成功之前被赋值，如果局部常量必须在声明的时候赋值

修饰方法：方法就不能被重写

3.static关键字

static关键字是一个修饰符，可以修饰类，成员变量，成员方法

被static修饰的成员放在方法区中的静态区中

static修饰成员的特点：

随着类的加载而加载

优先于对象而存在的

可以直接使用类名调用，也可以使用对象名调用

static修饰成员的注意事项：

静态方法中没有this关键字存在

静态的成员方法只能访问静态成员(成员变量和成员方法)，静态的只能访问静态的

单例 多态 抽象 接口

1.单列设计模式

含义：让一个类永远只产生一个对象

恶汉式：

1.私有化构造函数

2.创建一个私有的静态的最终的本类对象

3.提供一个公共的静态的返回值类型为本类对象的一个方法

懒汉式：

1.私有化构造函数

2.声明一个私有的静态的本类对象引用变量

3.提供一个公共的静态的返回值类型为本类对象的一个方法，在方法体内部会判断当前的变量引用是否为null，如果为null则创建本类的对象，如果不为null则直接返回

开发中使用恶汉式较多，因为恶汉式是线程安全的

2.多态

某一个事物在不同环境下，所表现的不同状态

多态的前提和体现：

要有继承关系

父类引用指向子类对象

子类重写父类的方法

通过父类引用调用子类的重写方法

多态的弊端：

通过父类引用不能访问子类特有的方法，可以使用转型来解决

向上转型：

父类引用指向子类对象

从子到父

向下转型：要强转的类型必须和父类指向的对象类型一致

将父类引用强转成子类对象

从父到子

3.抽象类

当父类中无法给出具体的实现时，父类中仅仅给出一个声明，然后这个方法只有声明，没有方法体，这样的方法在java中被称为抽象方法，有抽象方法的类一定是一个抽象类

抽象类定义格式：

abstract class 类名

修饰符 abstract 返回值类型 方法名(参数列表);

抽象类成员特点：

成员变量：可以是常量，可以是变量

构造方法：是有的，因为抽象类中有具体的实现，子类想使用父类的具体成员，就必须初始化父类

成员方法：可以是抽象的方法，也可以是具体的方法

如何使用抽象类里面的方法呢？

抽象类中不一定有抽象方法，有抽象方法的类一定是一个抽象类

可以使用一个子类继承抽象类，然后由子类实现抽象方法，这是多态的一种思想，抽象类多态

抽象类的子类特点：要么是抽象类，要么实现抽象类里面的所有抽象方法

4.接口

接口一般是用于做扩展功能的，有些额外的功能不适合定义在父类中，有一部分子类想拥有这部分功能可以实现对应的接口

接口的定义格式：

interface 接口名

接口的成员特点：

成员变量：只能是成员常量，而且默认修饰符：public static final

构造方法：没有构造方法，因为接口里面没有具体的方法，只有常量，而且常量是静态的，可以直接通过接口名调用，所以要构造方法没有意义

成员方法：只能是抽象的，而且默认修饰符：public abstract

如何使用接口里面的抽象方法呢？

可以让一个子类实现该接口，子类实现所有的抽象方法，这也是多态的思想，接口多态

子类如何实现接口？

class 子类名 implements 接口名

接口的子类特点：要么是抽象类，要么实现接口中所有的抽象方法

5.类和接口以及彼此之间的关系

类与类：继承关系，单继承，多层继承，不支持多继承

类与接口：实现关系，可以单实现，多实现，多层实现，还可以在继承一个类的同时实现多个接口

接口与接口：继承关系，单继承，多继承，多层继承

6.抽象类和接口的区别

成员区别：

成员变量：

抽象类：常量，变量

接口：只能是常量，而且默认修饰符：public static final

构造方法：

抽象类：有

接口：没有

成员方法：

抽象类：可以有具体的，可以有抽象的

接口：只能有抽象的

类和接口关系区别：

类：继承关系，单继承，多层继承，不能多继承

接口：实现关系，单实现，多实现，多层实现

设计理念区别：

抽象类：体现共性功能的，将共性封装在类中，不给出具体的实现

接口：扩展功能的，将额外的功能封装到接口

7.包

概述：在电脑上就是文件夹，在java中称为包

作用：分类管理文件的，不能功能的类放在不同的包中

包定义：

包名：一般是以公司域名反转之后的前两位

package 包名；

包定义注意事项：

必须放在可执行代码第一行

一个源文件中只能出现一个package关键字

如果没有package关键字，就表明该类没有包名

包的编译和运行：

手动式：

1.javac 源文件名.java

2.手动建立对应的包名文件夹

3.将编译好的字节码文件放在对应的文件夹中

4.java 路径名.包含main方法的类

自动式：

1.javac -d . 源文件名.java

2.java 路径名.包含main方法的类

导包：为了让不同包的类方便访问，不需要再加全路径名

import 全路径名；

eclipse笔记：

注意：新建工程的第一件事情就是将工程的编码改成UTF-8，Window->preference->WorkSpaces->改成UTF-8

1.eclipse是一个高级的开发工具，这个开发工具提供了很多快捷键，还可以自动的检查语法错误

2.eclipse从www.eclipse.org这个网站进行下载的，直接下载绿色版本，直接解压执行eclipse.exe就可以运行了，最好将eclipse.exe发送到桌面快捷方式，方便下次使用

3.新建一个项目

右击Project Explorer会出现很多菜单栏->new Project->Java Project->给工程起个名字，然后下一步下一步就完成了

4.在src存放源代码的文件夹中创建类

选中src文件夹然后右击选择新建的类和包

5.更改类中的字体

Window->preference->搜索Colors and Fonts，然后找到Basic->Text Font->双击选择你想要的字体

6.运行java程序

选中main方法右击选择Run As——>Java Application

7.更改快捷键

Window->preference->搜索Keys->搜索Java Application——>改成你自己喜欢的快捷键

注意：这个快捷键在第一次使用的时候可能不会生效

8.快捷键

Ctrl+1：万能键

Alt+/：代码提示补全

Ctrl+Alt+上下：赋值当前行

Alt+上下：移动当前行

Ctrl+D：删除所选中的行

Alt+Shift+R：联动更改名字

Ctrl+M：最大化或者最小化代码窗口

Ctrl+W：关闭当前代码窗口

Ctrl+Shift+W：关闭所有的代码窗口

Shift+Enter：快速新建一行

按住Ctrl+鼠标左击或者直接按F3：查看源码

Ctrl+Shift+o:导入需要的包，删除无用的包

Ctrl+o：快速搜索方法

Ctrl+T：查看类的继承体系

Ctrl+向左的斜线：注释或者取消单行注释

Ctrl+Shift+向左的斜线：多行注释

Ctrl+Shift+向右的斜线：取消多行注释

Alt+Shift+S:自动生成一些方法的代码快捷键

Ctrl+Shift+F：快速格式化代码

Alt+Shift+M：将想要封装成方法的代码选中，按Alt+Shift+M

9.波浪线提醒

黄色波浪线不影响程序的运行

红色波浪线如果不处理，程序就无法运行

集合章节总结

1.Collection是单列集合的根类

List集合：元素都可以重复，存入的顺序和取出的顺序一致

ArrayList：底层结构是数组，查询比较快，增删比较慢

LinkedList：底层结构是链表，查询比较慢，增删比较快

Set集合：元素都是唯一的，底层是依靠元素对象的hashCode方法和equals方法进行元素判断是否重复的依据

HashSet：保证元素的唯一，存入的顺序和取出的顺序不保证一致

LinkedHashSet：保证元素的唯一，它保证元素存入的顺序和取出的顺序一致

2.遍历

普通的for循环

迭代器：在迭代的过程中不能直接通过集合进行元素的更改

增强for循环

将集合转换成数组，然后对数组进行遍历

3.增强for格式：

for(要被遍历的元素数据类型 遍历名：要遍历的数组或者集合){

对遍历的元素进行操作

}

4.可变参数

当定义参数列表时，不知道要定义几个参数合适时，可以用可变参数

可变参数本质上就是一个数组

修饰符 返回值类型 方法名(数据类型... 变量名){

}

注意事项：当参数列表中有多个参数时，如果参数列表中有可变参数，那么可变参数一定要放在最后的位置

5.泛型

左边的泛型和右边的泛型需要保持一致，在JDK1.7之后右边的泛型可以不用指定了

泛型好处：

提高了安全性

将运行时期的错误提前到了编译时期

省去了类型强转

6.Map集合

双列的

是键值对：key=value

保证键的唯一，不保证值的唯一

通过哈希表结构保证键的唯一，也是通过键元素对象的hashCode方法和equals方法

HashMap：存入的顺序和取出的顺序不保证一致

LinkedHashMap：存入的顺序和取出的顺序一致

map遍历的方式：

1.获取所有的键集，然后遍历键集，获取键对象，通过get方法获取值

2.获取键值对对象集合，然后遍历键值对对象集合，通过键值对对象获取每一个键和值

7.比较接口

Comparator：是比较器，一般是用来排序java定义好的类的元素

Comparable：是用来排序我们自定义类的元素对象的

当Comparable和Comparator同时存在时，以Comparator为主

第一个对象(o1,this)和第二个对比较：升序

第二个对象(o2,o)和第一个对象：降序

IO章节总结

1.异常

编译时期异常：是必须要显示处理的，不然程序不给运行

运行时期异常：可以显示处理，也可以不处理，一般情况下都是因为代码导致的，可以更改代码s

处理异常的两种方式：

捕获：当方法体的内部可以自己处理这个异常的时；当后续代码需要继续运行时，因为如果抛给JVM处理，后续程序就会终止

抛：当方法体内部不想或者不能处理时，可以抛给调用者

异常的注意事项：当父类的方法没有抛出异常时，那么子类重写的这个方法也不能抛出异常，只能内部进行处理

2.递归

方法体内部调用方法自身的一种现象

注意事项：

方法的递归次数不能太多，否则就会内存溢出

方法的递归需要有出口，否则就是死递归

构造方法不能递归

如何定义以递归？

找到规律

找到出口

3.IO流

通过流向分：

输出流

输入流

通过数据类型分：

字节流：

字节输出流：

OutputStream

FileOutputStream

ByteArrayOutputStream

BufferedOutputStream：它内部有一个缓冲区，如果在中途的时候有一部分数据着急刷新 出去，可以使用flush刷新；如果不着急可以在关闭流的时候或者缓冲区满的时候刷新出去

字节输入流：

InputStream

FileInputStream

ByteArrayInputStream

BufferedInputSteram

字符流：

字符输出流：它内部有一个缓冲区，如果在中途的时候有一部分数据着急刷新 出去，可以使用flush刷新；如果不着急可以在关闭流的时候或者缓冲区满的时候刷新出去

Writer

OutputStreamWriter：字节输出流+编码表

FileWriter：它是OutputStreamWriter的简写形式，它只能以平台默认的编码写出字符

BufferedWriter：它有一个特殊的方法，newLine

字符输入流：

Reader

InputStreamReader：字节输入流+编码表

FileReader：它是InputStreamReader的简写形式，它只能以平台默认的编码读字符

BufferedReader：它有一个特殊的方法，readLine

多线程总结

多线程有几种实现方案，分别是哪几种?

继承Thread类，重写run方法，然后创建该子类的实例，然后调用start方法

实现Runnable接口，实现run方法，然后将该子类的实例作为参数传递给Thread，然后调用start方法

同步有几种方式，分别是什么?

锁对象：必须保证唯一或者说多个线程需要使用同一把锁

同步代码块：的锁对象可以是任意对象，只要保证多个线程使用的锁对象一致就可以了

synchronized (锁对象) {}

同步方法：锁对象是this

将synchronized放在方法的修饰位置上

静态同步方法的锁对象是当前类的字节码文件对象

使用Lock子类对象：创建ReentrantLock子类对象然后使用lock和unLock分别加锁和解锁

启动一个线程是run()还是start()?它们的区别?

是用start，start方法会启动一个线程，然后jvm在通过线程调用run方法

直接调用run方法就相当于一个对象调用方法，最终还是运行在主线程中

sleep()和wait()方法的区别

sleep：是睡眠指定的时间，等到指定时间到了就苏醒

wait：只能在同步代码中通过锁对象调用，必须通过锁对象唤醒