Java笔记

#对Java项目进行编译和执行:在项目根目录JavaPro下,

**javac** main.java 🡺编译项目,生成main.class文件

**java** main 🡺执行项目,不需要写.class

#基本语句:

package com.sayhello;

public class JavaPro{

public static void main(String[]args){

System.out.println(“Hello”);

}

}

System.out.println(…)打印并换行 print(…)不换行

快速套模板快捷键:

psvm ⬄ public static void main(String[]args)

sout ⬄ System.out.println(…)

类的开放性与包管理:

每个.java文件只能定义一个public类,且类名必须和文件名相同;

private和无开放性定义的可以无限声明.

同一包package相当于写在同一个作用域,可以直接使用;

不同的包如果需要引入则需要import;

简而言之,public的类可以给其他包用,而非public的只能给自己的包或者文件用.

简单类型:int short long float double char String boolean

引用类型:字符串,数组,自定义类.

空值:null

Scanner:接收用户输入.

**import java.util.Scanner;**

**Scanner input=new Scanner(System.in);**

int num=input.**nextInt()**;

input.close();

\*\*Scanner的成员函数\*\*:

|  |  |  |
| --- | --- | --- |
| next() | 读取下一个​​字符串​​（空格分隔） | sc.next() |
| nextLine() | 读取一行（包括空格，以回车结束） | sc.nextLine() |
| nextInt() | 读取下一个​​int​​ | int num = sc.nextInt(); |
| nextDouble() | 读取下一个​​double​​ | double d = sc.nextDouble(); |
| hasNext() | 是否有下一个字符串（空格分隔） | |
| hasNextLine() | 是否有下一行 | |
| hasNextInt() | 是否有下一个int | |
| hasNextDouble() | 是否有下一个double | |
| （其他类型类似，如hasNextLong()） |  | |
| useDelimiter(String pattern) | 设置分隔符（默认是空格/换行） | |
| close() | 关闭Scanner释放资源 | |

输入处理:

1. 用分隔符切割字符串输入:

Scanner sc=new Scanner(“1,2,3”).useDelimiter(“,”);

while(sc.hasNextInt())System.out.println(sc.nextInt());

1. 处理混合分隔符:用useDelimiter(“**[**,;**]**”)之类.
2. 处理未知数量数据:

while(sc.hasNext())…

1. 直接换行:sc.nextLine();
2. 未知行数的输入:

while(true){

String line=sc.nextLine();

if(line**.isEmpty()**)break;

…

}

字符串转换为各种类型:如

int num=**Integer.parseInt**(str);

double d=**Double.parseDouble**(str);

…

字符串格式化: String result=**String.format**(“%.1f”,avg)限制小数位数

字符串常用方法:

String str=new String();

str**.length()**返回字符串长度

str**.equals**(otherStr)比较两个字符串内容是否一致

判定字符串相等:不能用==,要用”abc”.equals(“abs”)

原因:==运算符对于简单类型比较真实值,而对于引用类型则比较地址.

str**.toLowerCase()**转为小写 **toUpperCase()**转为大写

str**.concat**(otherStr)拼接字符串,返回拼接后的字符串

str**.indexOf**(s)寻找s第一次出现的位置,返回索引,没找到返回-1

str**.lastIndexOf**(s)反着找

str**.substring**(a) 提取[a,:]的子串 substring(a,b) 提取[a,b)子串

str**.trim()**返回左右去掉空格的子串

str**.replace**(oldChar,newChar)将所有的oldChar替换为newChar

str**.split**(regex)将字符串以某字符切割并返回String[],可以使用正则表达式如"[,;]"

StringBuffer:可变字符串,当对字符串进行大量拼接的时候可以大大提高程序执行效率:

StringBuffer sb=new StringBuffer();

sb**.append**("NewYork");

sb**.insert**(2,"ww");//插入子串

String result=sb**.toString()**;//返回最终字符串

枚举: public **enum** Days{MONDAY,TUESDAY,WEDNESDAY}

包装类:如Integer,Double,Character,Boolean;

包装类可以有构造,可以直接赋值给基本类型,也可以直接由基本类型赋值产生.

显式返回包装类: 可以使用.**valueOf**返回包装类,自动转换:如Integer.valueOf(2);

**final**:常量

数组:int[]arr=new int[10];类似于cpp,但是不能用指针

int[]arr2=new int[]{1,2,3};直接初始化,注意不能直接写{ }而忽略new int[]

arr**.length**返回数组长度

排序: **import java.util.Arrays;**

**Arrays.sort**(arr)//升序排序

#面向对象:

面向对象三大特性:封装,继承,多态

this.<menber>成员变量

继承:class Dog **extends** Pet{…}

继承时,this指向自身类,super指向父类对象.super不能访问父类的private成员.

Object类:

**package java.lang**

Object可以作为所有类的父类,两个成员函数:

**toString()**返回自身的所有信息,以字符串形式返回

**equals**(other)比较是否是同一对象

两个成员函数常用于被子类重写.

抽象类与抽象方法:

public **abstract** class Pet{//抽象类

public **abstract** void eat(int energy);//抽象方法

}

抽象类可以有构造函数,但不可创建对象;抽象方法不能有方法体,只能由子类重写此方法.

子类重写抽象方法:在上面加一个**@Override**,如:

public class Dog extends Pet{

@override

public void eat(int num){…}

}

多态: 用子类对象初始化父类对象.

1. 父类转为子类:

Pet pet=new Dog();//pet认定为Pet类,能使用父类的方法,但方法实现是子类

🡺编译时为Pet,运行时为Dog

Dog dog=(Dog)pet;//强制类型转换,dog认定为Dog类.

1. 子类转为父类:自动转换,可以直接使用
2. **instanceof**关键字: **动态(运行时)**类型检查, 对象 instanceof 类/接口,返回boolean.

应用:如管理系统中的多态函数login(Person person),传入一个Person person=new Admin(…),此时login函数需要添加对此类的验证,即:

if(person instanceof Admin)…

else if(person instanceof User)…

#接口:

public **interface** Artist{//接口类

public void dance();//接口方法

}

接口类不能实例化, 接口方法都是抽象方法,不能有实现.

一个类可以实现多个接口;接口中的变量都是静态变量static.

类与类之间的关系:继承

类与接口的关系:**实现**.

类实现接口不使用extends,而是使用implements:

public Dancer **implements** Artist{

**@Override**

public void dance(){…}

}

#捕获异常:

try{

…//可能产生异常

}catch(Exception e){

…//catch中代码在try块产生异常时执行

}finally{

…//finally块中代码必然执行

}

抛出异常:throw抛出异常,抛出的必须是Throwable或其子类如Exception,RuntimeException;

throws声明异常,声明可能抛出的异常类型,如:

public void readFiles(String path) **throws** IOExcetion{…}

try{…}catch(IOException e){…}

e**.printStackTrace()**输出调用堆栈信息

e**.getMessage()**返回异常信息

#集合框架:**improt java.util.ArrayList等**



Collection接口存储不唯一,无序的对象

🡪List接口存储不唯一,有序的对象

🡪Set接口存储唯一,无序的对象

Map接口存储键值对{key,value}.

ArrayList ⬄ Python的list(动态长度,类型不唯一)

ArrayList alist=new ArrayList();

ArrayList<int>arr=new ArrayList<>(10);//初始设定10个长度

ArrayList<int>arr=new ArrayList<>(**List.of**(1,2,3,4));直接由数值创建

alist**.add**(Object)尾部追加

alist**.insert**(index,Object)在alist[index]处插入Object

alist**.remove**(index)删除alist[index]

alist**.get**(index)获取alist[index]

alist**.set**(index,val)给alist[index]赋值val

alist**.size()**返回列表长度

alist**.contains**(1.2)判断是否存在指定元素

LinkedList ⬄ 链表

LinkedList list=new LinkedList();

list**.addFirst**(Object)头插

list**.addLast**(Object)尾插,=**add**

list**.getFirst**()获取头部元素

list**.getLast**()获取尾部元素

list**.removeFirst**(Object)头删

list**.removeLast**(Object)尾删

list**.get**(index)获取list[index]

HashSet ⬄ 哈希表(单列)

HashSet hs=new HashSet();

hs**.add**(Object)插入

hs.size()返回大小

HashMap ⬄ 哈希表(双列,key-value)

HashMap map=new HashMap();

map**.put**(key,value)插入键值对

map**.get**(key)由键获取值

map**.keySet()**返回所有键的HashSet

map**.values()**返回所有值的HashSet.

map**.entrySet()**返回所有Map.Entry的HashSet.

**Map.Entry<Key,Value>**:键值对{key,value}

Map.Entry entry;

entry**.getKey()**返回key,entry**.getValue()**返回value.

map**.size()**返回大小

map**.containsKey**(key)判断是否含有key

迭代器Iterator: Collection接口的iterator()方法获取Iterator对象;

ArrayList list=new ArrayList();

**Iterator it=list.iterator();**

it**.hasNext()**//判断是否存在下一个可访问元素

it**.next()**返回并迭代到下一个可访问元素

泛型集合:只存储单一类型的数据,可以使用for(Object obj:list)等进行直接迭代.

注意:泛型集合的模板类只能存储引用类型如String,Integer,不能存储int等基本类型.

1.List:

List**<**Book**>**list=new ArrayList**<**Book**>**();

for(Book book:list)…

2.Map: 有几种遍历方法:

(1)用key遍历

for(String key: map.keySet())…

(2)用Map.Entry遍历

for(Map.Entry<String,Book>entry: map.entrySet())…

(3)直接遍历values

for(Book value:map.values())…

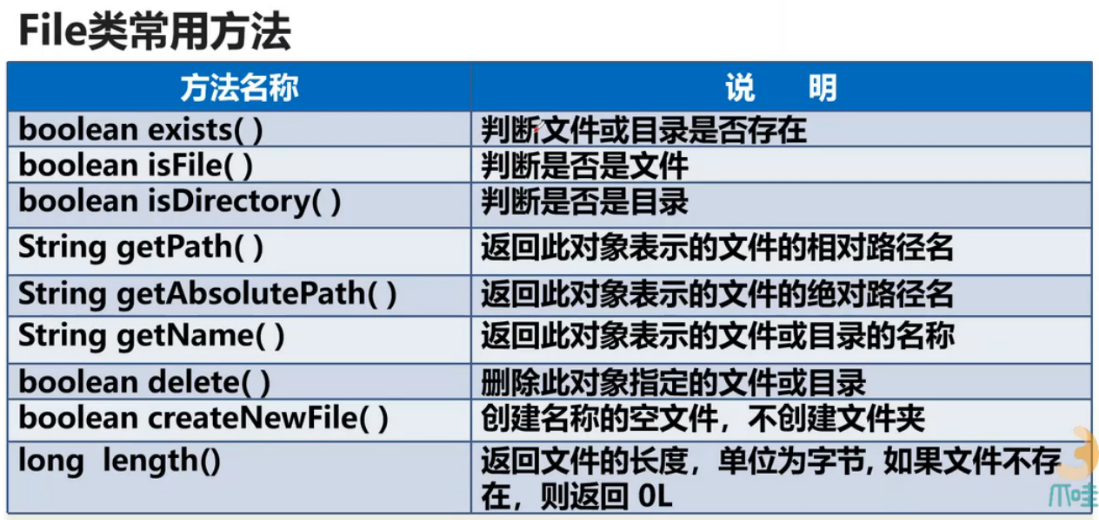
#实用类:

Math类:**import java.lang.Math**

abs,max(a,b),pow(a,b),sqrt等.

#文件IO:

访问文件:



**import java.io.File;**

File file=new File(path);//由路径创建文件对象

if(!file.exists()){

try{file.createNewFile();}

catch(IOException e){e.printStackTrace();}

}

读写文件:FileInputStream读取文件(extends InputStream)

文本, 信件

AI 生成的内容可能不正确。

**import java.io.FileInputStream;**

FileInputStream ifs=null;

try{

ifs=new FileInputStream(path);

//读取数据

while(**ifs.available()!=0**){//还没有到达流的末尾

}

}

catch(FileNotFoundException e){ e.printStackTrace();}

catch(IOException e){ e.printStackTrace();}

finally{

try{**ifs.close();**}

catch(IOException e){ e.printStackTrace();}

}