Java技术体系平台

1. java se（标准版）

java ee （企业版）

java me（小型版）

2、java web（js+css+html）

3、编译指令java，javac，javadoc…..

4、jdk（包含jre和一些开发工具指令）//开发者需求

Jre（包括jvm和一些核心类库）

初学java易犯错误

1. 找不到文件

解决方法：源文件名不存在或者写错，或者当前路径错误。

1. 主类名和文件名不一致

解决方法：声明为public的主类应与文件名一致，否则编译错误。

1. 缺少分号

解决方法：编译错误，注意错误出现的行数。

文档注释

1. 注释内容可以被jdk提供的工具javadoc所解析，生成一套以网页文件体现的该程序的说明文档，一般写在类。
2. 格式：

/\*\*

\* @author 热心市民

\* @version 666.0

\*/

3、Javadoc标签

|  |  |  |
| --- | --- | --- |
| **标签** | **描述** | **示例** |
| **@author** | **标识一个类的作者** | **@author description** |
| **@deprecated** | **指名一个过期的类或成员** | **@deprecated description** |
| **{@docRoot}** | **指明当前文档根目录的路径** | **Directory Path** |
| **@exception** | **标志一个类抛出的异常** | **@exception exception-name explanation** |
| **{@inheritDoc}** | **从直接父类继承的注释** | **Inherits a comment from the immediate surperclass.** |
| **{@link}** | **插入一个到另一个主题的链接** | **{@link name text}** |
| **{@linkplain}** | **插入一个到另一个主题的链接，但是该链接显示纯文本字体** | **Inserts an in-line link to another topic.** |
| **@param** | **说明一个方法的参数** | **@param parameter-name explanation** |
| **@return** | **说明返回值类型** | **@return explanation** |
| **@see** | **指定一个到另一个主题的链接** | **@see anchor** |
| **@serial** | **说明一个序列化属性** | **@serial description** |
| **@serialData** | **说明通过writeObject( ) 和 writeExternal( )方法写的数据** | **@serialData description** |
| **@serialField** | **说明一个ObjectStreamField组件** | **@serialField name type description** |
| **@since** | **标记当引入一个特定的变化时** | **@since release** |
| **@throws** | **和 @exception标签一样.** | **The @throws tag has the same meaning as the @exception tag.** |
| **{@value}** | **显示常量的值，该常量必须是static属性。** | **Displays the value of a constant, which must be a static field.** |
| **@version** | **指定类的版本** | **@version info** |

Dos命令

1、md + 路径 + 文件名 //创建文件

rd + 路径 + 文件名 //删除文件

2、 相对路径：从当前目录开始定位，形成的路径。

绝对路径：从顶级目录：D开始定位，形成的路径。

3、 dir 查看当前目录文件。

cd 切换到其它盘符。

cd ..回到上一级目录

cd \回到根目录

4、tree 生成当前目录下的目录树

5、Cls 清屏

6、exit 退出dos

数据类型

1. 浮点型 = 符号位 + 指数位 + 尾数位
2. 尾数部分可能丢失，造成精度损失（小数都是近似值）。
3. Floa类型赋值时要加F在尾部声明为float常量；（默认小数是一个double类型）
4. Double类型特有的陷阱！！！double num=8.1/3.00;//8.1这时会被识别成一个8.000001这样的数，导致结果和预期不同。
5. 数据类型的自动转换，从低精度向高精度转换。（byte和short类型不能和char类型进行转换）这三种类型进行运算时会自动转换成int类型。

例如：char ->double

1. 当多种数据类型进行混合运算时，会将所有的数据转换成容量最大的那种数据类型

不像c++的类型转换，可以直接将int类型的变量赋给char类型。当要把一个int类型的变量赋值给一个char类型的变

基本数据类型和string类型的转换

1. 基本数据类型转成字符串类型。

a）

public class day2{

public static void main(String[] args) {

int a = 100000;

String n= a + "";//定义一个字符串类型的变量在要转的变量后加上“”；

System.out.println(n + 20000);

}

}

1. 将字符型转成基本数据类型
2. 通过基本类型的包装类调用parse方法。

public class day2{

public static void main(String[] args) {

String a = "123456",b = "99";

int num1 = Integer.parseInt(a);//通过parse方法将字符串类型转换成寻常数据类型。注意大写的地方！！！

int num2 = Integer.parseInt(b);

int c1 = num1 + num2;

System.out.println(c1);

}

}

1. 如果将字符串类型中的字符指定输出。

public class day2{

public static void main(String[] args) {

String a = "123456",b = "99";

System.out.println(a.charAt(0));//a.charAt(0);

}

}

键盘输入

1. 导入Scanner类import java.util.Scanner;
2. 创建Scanner对象，new创建一个对象。

import java.util.Scanner;

public class input{

public static void main(String[]args){

Scanner myscanner = new Scanner(System.in);

System.out.println("请输入名字");

String name = myscanner.next();//接收输入

System.out.println("请输入年龄");

int age = myscanner.nextInt();

System.out.println("请输入薪水");

double sal = myscanner.nextDouble();

System.out.println("name = " + name + "age = " + age + "sal = " + sal);

}

}

位运算符

1. 算术右移>>：低位溢出，符号位不变，并用符号位补溢出的高位。

例：int a = 1 >> 2;//=> 00000001 => 00000000;//两位被推出去并补了一个符号位。

可以做到某个数除上2的几次方的效果。

1. 算术左移<<：符号位不变，低位补0。//可以做到某个数乘上2的几次方的效果。

例：1 << 7；//结果为128

1. >>>逻辑右移（无符号右移）:低位溢出，高位补0。
2. 数组入门
3. 1、.length 计算数组的长度。
4. 2、动态创建int a[] = new int[5];//创建长度为5的一维数组等待赋值，初始值为0
5. int a[]//声明一个没有数组变量，但是不开辟空间。
6. new int[5]//此时为其开辟内存空间
7. 3、静态创建int b[] = new int[] {1,2,3};//声明并创建内存空间且赋值
8. 4、jvm的内存被分成了 堆、栈、方法区。数组存放地址，地址指向放在堆区中的元素。
9. 5、二维数组中存储的是一维数组，将一维数组存储在堆区中，而一维数组会再指向存放在堆区中的一维数组地址。

类与对象

1. 创建一个类以及对象：

public class a{

public static void main(String[]args){

Scanner myscanner = new Scanner(System.in);

Cat cat1 = new Cat();//cat1指向由new新建的对象cat

cat1.name ="甘甘"；

cat1.age = 114514；

cat1.color = "yellow";

Cat cat2 = new Cat();

cat1.name ="坤坤"；

cat1.age = 1145141919；

cat1.color = "blue";

}

}

class cat{//定义一个猫类

String name;

int age;

String color;

}

1. 对象的内存布局（引用类型，有点像数组的存放方式），当新建一个对象会存放一个地址在栈中，地址指向堆区，在堆区中存放对象中的属性。 属性中的字符串类型会指向方法区的常量池中
2. 内存分配补充：类信息在方法区中加载，新建对象在堆区中，栈中一个变量指向对象的地址，对象的string类型会存储在方法区的常量池中（string是引用类型），在常量区中的string类型会把他的地址再给堆区中的对象。
3. 成员变量 == 属性 == field
4. 访问修饰符：控制方法的访问范围。
5. public：
6. protected：
7. 默认：
8. Private：
9. 方法，方法中可以执行很多操作，类似循环，创建成员方法时需要public void xxx()；

可以给方法传参（感觉就是在类里面写一个函数）

class cat{

public void sum(){

int sum=0;

for(int i=0;i<1000;i++)sum+=i;

System.out.print(sum);

}

}

1. 方法调用机制
2. 首先，主方法Main会在栈中开辟一块空间。
3. 加载类信息，new之后会在堆区新建一个对象由新建的变量指向这个对象。
4. 当程序在执行对象的方法时，会在栈区中新建一个独立的空间，被执行方法的语句所指向。
5. 被调用的方法执行到return语句后，会返回到调用方法的位置并销毁他在栈区中的空间。

递归调用

1. 老鼠走迷宫
2. 详情略，具体请见代码文件夹mi gong.java文件。
3. 汉诺塔问题

a) 详情略，具体请见代码文件夹hanoi.java文件。

方法重载

1. 同一个类中，允许多个同名方法的存在，但要求形参列表不一致。

可变参数

1. java允许将同一个类中多个同名同功能但参数个数不同的方法封装成同一个方法，可以看做传的是数组。

class tool{

public int sum(int... nums){

System.out.println("接收的个数" + nums.length);

return 0;

}

}

可变参数的本质就是数组，可变参数的实参可以是数组。可变参数要放在形参列表最后。

一个形参列表只能有一个可变参数

作用域

1. 关于局部变量和全局变量的作用域。
2. 如果属性变量和局部变量重名，就近原则。
3. 全局变量可以加修饰符
4. 全局变量可以被本类使用或被其他类使用（通过对象调用）
5. 局部变量只能在本类中对应的方法中使用

构造器

1. 构造器的主要方法是完成对新对象的初始化。
2. 构造器的修饰符可以默认，也可以是public protected private
3. 构造器没有返回值，连void都不是
4. 方法名和类名字必须一样
5. 参数列表和成员方法一样的规则
6. 在创建对象时，系统会自动调用该类的构造器完成对对象的的初始化

代码示例：

public class a{

public static void main(String[]args){

Scanner in = new Scanner(System.in);

person a1 = new person("zhangsan",15);

System.out.print(a1.name + " " + a1.age);//输出测试

}

}

class person{

String name;

int age;

public person(String pname ,int page){//构造器方法名必须与类名相同，且不能 有返回值

name = pname;//直接赋值

age = page;

}

}

1. 构造器细节
2. 一个类可以定义多个不同的构造器，即构造器重载。
3. 如果没有定义构造器，系统会自动给类生成一个默认无参构造器。
4. 可以用javap返编译查看
5. 在新建一个无参构造器后，默认的无参构造器会被覆盖掉，除非再次显示的定义一次。

代码示例：

class person{

String name;

int age;

public person(int fage){

}//此时默认无参构造器被覆盖了

person(){}//再次定义，默认构造器复活

}

对象的赋值：默认初始化->显示初始化->构造器初始化。

This

1. 当对对象的属性使用this.时，会自己识别成是相对应对象的属性
2. 在堆中创建好的对象会有一个隐藏的this属性指向自身。
3. 使用细节
4. 只能在构造器中使用
5. 可以用来访问本类的属性、方法、构造器（不能使用父类）
6. 在构造器中用this访问另一个构造器时，this语句要在第一行（与继承相关）