1. 入门知识

1. 环境变量Path：相当于在电脑中存储一些路径，因此在cmd命令界面中，不需要进入到相对应的路径中即可。
2. bin-存放各种工具命令，如javac和java；conf：存放相关配置文件；include：存放特定的头文件；jmods：存放各种模块；legl:存放各种授权文件；lib-存放工具的一些补充JAR包
3. 程序运行分为两步：**编译**（javac xxx.java）；**执行**（java xxx），执行的时候不需要后缀
4. Java SE：java语言的标准版，用于桌面应用的开发，是其他两个版本的**基础**。（但是并不合适，常见的是C和C++）

   Java ME：java语言的小型版，用于嵌入式电子设备和小型移动设备（已凉）

   Java EE：java语言的企业版，用于Web方向的网站开发。网站开发分为浏览器+服务器，java主要用于后者（主力）
5. java优点：面向对象；安全性（代码漏洞少）；多线程；简单易用；开源；跨平台（指操作系统，Windows, Mac, Linux）
6. 编译型语言(c, c++)：整体翻译；解释型语言（python）：逐行翻译。

   混合型语言(java)：首先整体编译为.class二进制字节码文件，再按行交给设备运行（运行在**虚拟机**中）。java不直接运行在操作系统中，而是在虚拟机中，这就是java能够多平台运行的原因
7. JVM: java virtual machine, java虚拟机； 核心类库：java预先定义的内容

   javac：编译工具；java：运行工具；jdb 调试工具；jhat 内存分析工具

   JDK：java开发工具，包含JVM, 核心类库，核心类库，开发工具；

   JRE: java运行环境，包含JVM， 核心类库，运行工具

2. 小概念

1. 注释 - 单行注释//， 多行注释/* */，文档注释 /** */，主要用于说明文档；
2. 关键字：被java赋予**特定含义**的英文单词。特点：关键字的字母全部**小写**；常用的代码编译器，会将关键字用**特殊颜色**标出。举例：

   class: 用于创建或定义一个类，是java最基本的组成单元
3. 字面量：主要用于告诉程序员，数据在程序中的书写格式。如：整数和小数类型；字符串类型和字符类型（前者需要“”， 后者需要''）；布尔类型(true, false)和空类型（null）

   常见特殊字符：\t, 制表符，在打印的时候吧前面字符串的长度补齐到8或8的倍数，最少补1个空格，最多补8个空格
4. 变量：在程序中可能会改变的量。定义格式：数据类型 变量名 = 数据值；

   变量注意事项：i. 只能存一个值；ii. 变量名不允许重复； iii. 一条语句可以定义多个变量； iv: 变量**使用前一定要赋值**； v: 变量要注意**作用范围**
5. **类名一定要与文件名统一！！！**
6. 计算机的存储规律：主要分为文本、图片、声音三类，但是都是以二进制的形式来存储。

   计算机中各种进制的表型形式：二进制 - 0b；十进制 - 无前缀；八进制 - 0； 十六进制 - 0x；
7. 数据类型：基本数据类型+引用数据类型。基本数据类型：整数（byte, short, int, long）；浮点数（float, double）；字符（char）;布尔类型（boolean）

8. 标识符：给类、方法、变量起的名字。

   硬性要求：

   i. 有数字、字母、下划线_、美元$构成

   ii. 不能以数字开头

   iii. 不能是关键字

   iv. 区分大小写

   软性建议：

   i. 小驼峰命名法：一个单词时全部小写（name），多个单词时第一个单词首字母小写，后续的单词首字母大写(nameFirst)。**主要用于方法、变量**

   ii. 大驼峰命名法：一个或多个单词组成时，全部单词首字母大写。**主要用于类名**

   iii. **见名知意**

9. 键盘录入**Scanner**类，可以接受键盘输入的**数字。**

   使用步骤：

   i. 导包(import java.util.Scanner);

   ii. 创建对象(Scanner sc = new Scanner(System.in));

   iii. 接受数据(int i = sc.nextInt())

   Scanner的两套体系：

   第一套体系（空格，制表符或回车 停止接收）：

   i. nextInt() : 接收整数

   ii. nextDouble() : 接收小数

   iii. next() : 接收字符串

   第二套体系（仅停车 停止接收，其余皆可）：

   i. nextLine() : 接收字符串

10. IDEA项目结构：project(项目)；module（模块）；package（包）；class（类）

    对于包package，使用时常常创建多级包，用来表示所属关系。如com.itheima.demo1

3. 运算符

1. 隐式转换（自动类型提升）：将取值范围小的数值，转换为取值范围大的数值。char, short, byte类型都会先转换为int类型，之后再参与计算
2. 强制装换：将一个取值范围大的数值，赋值给取值给取值范围小的数值。（强制转换的优先级比较低，最好在后面添加小括号）
3. 字符串相加：+操作中出现字符串时，此时相当于字符串的拼接，产生一个新的字符串。如："123"+123 = "123123"。连续+号，从左到右逐个进行
4. 字符相加：先提升为int，再进行计算
5. java中包含自增运算符++和自减运算符--
6. 逻辑运算符：& | ^ ! ; 短路运算符：&& || 用于提高程序运行效率，只要能够判断结果，就不再继续向后运算。
7. 三元运算符：A?B:C ;

4. 流程控制语句

1. 三种控制语句：顺序结构；分支结构；循环结构

2. if语句：

   第一种格式：if(关系表达式){语句体;}

   第二种格式：if(关系表达式){语句体;} else {语句体;}

   第三种格式：if(关系表达式){语句体;} else if(关系表达式){语句体} ... else {语句体}

   switch语句：switch(表达式){case 值1: 语句体; break; case 值2: 语句体; break; default: 语句体; break;}

   如果每个case语句后面只有一个执行语句，也可以这样：

   switch(表达式){case 值1 -> 语句体; case 值2 -> 语句体; ... default -> 语句体}

   switch(表达式){case 值1,2,3 -> 语句体; case 值4, 5 -> 语句体}

3. switch其他知识

   i. default的位置和省略：default可以省略，此时如果不匹配则无执行内容； default需要设置在最后

   ii. case的穿透：case如果匹配且后面无break，就会向下继续执行（可以用于case语句重复时的简化代码，如周1-4是工作日，周5-7是休息日）

4. for循环：

   for(int i=1; i<= 10; i++) { ... }

   while循环：

   while(i<10) { ... }

   do ... while循环：

   do { ... } while { ... }

5. break, continue语句：不解释

6. 生成随机数：import java.util.Random; Random r = new Random(); int num = r.nextInt(100) (小括号内写的，是随机数的范围，以0开始，给定数-1为阶数);;

5. 数组

1. 数组：用于存储同种数据类型的多个值

2. 定义方式：

   i. 数组类型[] 数组名， 如：int[] array;（更常用）

   ii. 数据类型 数组名[]，如：int array[];

3. 数组的初始化：

   静态初始化：int[] array = new int[] {1, 2, 3, 4, 5}。**注意：没有float类型的数组，一般小数默认为double类型！**

   简写格式：int[] array = {1, 2, 3, 4, 5};

   动态初始化：int[] array = new int[50]; **注意：动态初始化时，new后面的数组长度可以是变量**

   对于动态初始化，整数默认为0，小数默认为0.0，字符默认为'/u0000'，即空格，布尔默认为false，引用默认为null

4. 数组打印：直接打印arr，得到的是数组的地址值。实际打印采用：arr[10]类型格式

5. 数组遍历：arr.length为数组的长度，随后for循环即可遍历。

   简写方法：for(int num : arr) { ... } 这时的缺点在于，不知道此时遍历元素的索引。

6. **java内存分配：**主要分为五部分：栈，堆，方法区，本地方法栈，寄存器。

栈：**方法**运行时使用的内存，如main方法运行，进入方法栈中执行

堆：存储**对象或者数组**，new来创建的东西在该块内存中开辟空间并产生地址 。**比如a=10这样的语句，没有使用new，因此不适用堆空间，而是在栈内存！！！**

方法区：存储可以运行的**class文件**

本地方法栈：JVM在使用操作系统功能时使用，与开发过程无关

寄存器：与CPU相关

6. 方法

1. 方法(method)是程序中**最小的执行单元**，要么全部执行，要么不执行。对于**重复的代码、具有独立功能的代码**，可以抽取到方法中。优点：提高代码的复用性、可维护性。
2. 方法定义：

   public static void 方法名(参数类型 参数, ...) {方法体; return 返回值}
3. 形参：**方法定义**中的参数；

   实参：实际参数，**方法调用**中的参数
4. 方法注意事项：

   i. 方法与方法之间是平级关系，不能互相嵌套

   ii. 方法的编写顺序与执行顺序无关
5. 方法重载

   在同一个类中，定义了多个**同名的方法**，这样方法有**同种的功能**，但是具有**不同的参数类型**或者**参数个数。**

   参数不同分为三种：个数不同，类型不同，顺序不同
6. 基本数据类型：整数、浮点数、布尔、字符（变量中存储的是**真实数据**）

   引用数据类型：例如array（使用new创建的，都是引用数据类型，在**堆**中开辟空间。变量在栈中记录的是地址，在堆中记录的是数据）
7. **在java中，如果传递的是基本数据类型，形参的改变不影响实参的值。**

   如果传递的是引用数据类型（如数据），此时传递的是地址值，因此会改变实参的值。

7. 面向对象

1. 面向对象编程：通过获取**特定对象**来完成任务
2. 面向对象学习什么：

   i. 获取已有对象并使用

   ii. 自己设计对象并使用
3. 类：对象共同特征的描述。（相当于设计图）

   对象：真实存在的具体物体。

   类的组成：成员变量；成员方法；构造器；代码块；内部类。

   对象的定义：类名 对象名 = new 类名();

   对象的调用：**对象.成员变量**； 对象.成员方法
4. 定义类的补充注意事项：

   Javabean类：用于描述一类事物的类，不写main函数

测试类：可以创建javabean类的对象并进行赋值使用

i. 类名首字母大写，使用**大驼峰命名法**

ii. 一个java文件中可以定义多个class类，**但只能一个类是public修饰，且该类名必须称为代码文件名。**在实际开发中，**一个文件尽量定义一个class类**

iii. 成员变量的完整定义格式：**修饰符 数据类型 变量名称 = 初始化值**；一般无需指定初始化值，存在默认值。

5. **开发中类的设计**：一般来说，名词设计为类，动词设计为类对应的方法。
6. **面向对象的三大特征：封装，继承，多态**

   封装：如何正确设计对象的属性和方法。对象代表什么，就得封装对应的数据，并提供数据对应的行为。**简单来说，以人关门为例，门自身具有一个状态（开关），对该状态改变的方法就属于人**
7. **private**：是一个权限修饰符，可以修饰成员（成员变量和成员方法），**只能在本类中访问。**

   作用：使代码更健壮、安全（如age需要代表的范围）。

   常用方法：设置属性为private，但是设置public void setAge, public int getAge.

   对于每个私有化的成员变量，都要提供get和set方法
8. 就近原则：在类的方法内和方法外的变量同名时，使用该变量会就行
9. **构造方法**：也叫构造器，构造函数。

   作用：在创建变量时，给成员变量进行赋值。

   格式：修饰符 类名（参数）{方法体；}

   特点：

   i. 方法名与类名相同，大小写一致

   ii. 没有返回值类型，连void都没有

   iii. 没有返回值，因此不能有return

   执行时机：

   创建对象时有虚拟机调用， 不能手动调用；每创建一个对象，就调用一次。

   一般来说，我们在创建类时，还是会写空参构造。因为在写了有参构造后，虚拟机就不再自动生成无参构造
10. 注意事项

    i. 未定义构造方法时，系统会构造默认的午餐构造

    ii.构造函数的重构。方法名相同，参数不同

    iii. **一般来说，我们至少要写两个构造方法：无参构造方法和带全部参数的构造方法**

    iv. 调用构造方法知识创造对象的一部分，而不是全部。构造方法的作用：给成员变量进行初始化
11. 标准JavaBean注意事项：

    i. 类名见名知意

    ii. **成员变量使用private修饰**

    iii. 至少有两个构造方法：午餐构造和带全部参数的构造

    iv. 要构造setXxx和getXxx
12. 原空间：字节码文件时，进入的内存

    栈内存：方法运行时进入的内存，变量也是在这里

    堆：new创建的对象

一个对象的内存图：（以 Student s = new Student();为例）加载class文件；申明局部变量；在堆内存中开辟一个空间；默认初始化；显示初始化；构造方法初始化；将堆内存中的地址值赋 值给左边的局部变量。

13. this的内存原理：菊粉局部变量和成员变量。this的本质：所在方法调用者的地址
14. 成员变量：类中方法外的变量

    局部变量：方法中的变量

    区别：

    类中位置不同（前者在类中方法外，后者在方法内）；

    初始化值不同（前者有默认初始化值，后者没有，需要赋值）

    内存位置不同（前者在堆内存中，后者在栈内存中）

    生命周期不同（前者随着对象的创建而存在，随着对象的消失而消失；后者随着方法的调用而存在，随着方法的运行结束而消失）

    作用域不同（前者在整个类中有效，后者在当前方法中有效）

8. API & 字符串

1. API: application programming interface: 应用程序编程接口。在java中，即jkd中提供的各种功能的java类。
2. 如何使用帮助文档：i. 打开API帮助文档 ii. 点击显示，并找到索引下面的输入 iii. 在输入框中输入类名并点击显示 iv. 查看类所在的包
3. 字符串的一般开发使用类：String，StringBuilder，StringJoiner, StringBuffer, Pattern, Matcher
4. **字符串的内容是不会发生改变的，他的对象在创建后不能被更改。但是变量赋值可以改变。**
5. 创建字符串的两种方式：

    i. 直接赋值。如String name = "张三"，**实际使用最多**

    ii. new，调用构造函数。如new String(String original), new String(), new String(char[] chs), new String(byte[] chs) (此时会翻译为字母). **后两者在实际开发中有相应应用。主要针对字符串内容无法修改，因此需要先将字符串转化为字符数组，修改后再重新转换位字符串。**
6. 字符串的内存占用：最早为**串池**，只有直接创建的字符串才会在串池中。从jdk7之后，**串池被移动到堆内存中。**

    直接赋值：首先在串池中查询是否存在该字符串，不存在则新建，**存在则复用。**

    new：以字符数组为例，首先在堆内存中产生字符数组，随后在堆中开辟新创建字符串，随后赋值。**不复用。**
7. Java的常见方法（比较）

    ==的原理：如果==两边是基本数据类型，则比较数据值；如果是引用数据类型，则比较地址值。

    字符串比较内容的方法：

    i. A.equals(B)　ii. A.equalsIgnoreCase(B) 后者忽略大小写
8. java键盘录入的字符串属于new的对象，因此与直接赋值得到的地址不同
9. A.charAt(i) 将字符串视为字符数组进行处理，就可以堆字符串进行遍历

    想要将1直接转换为字符1，最好直接使用字符串的拼接
10. A.substring(0, 3)，包左不包右，截取字符串
11. A.replace(旧值, 新值)，字符串的替换。

12. **StringBuilder()** 一个容器，创建之后里面的**内容是可变**的。**作用：不会产生中间字符串，提高字符串的操作效率。**

    原因：s1+s2+s3+s4+s5 每次相加都会产生一个中间字符串，影响程序的运行效率。

    创建方法：

    i. public StringBuilder() 创建一个空白可变字符串对象

    ii. public StringBuilder(s0) 根据字符串创建对象

    常用方法：

    public StringBuilder append()

    public StringBuilder reverse()

    public int length();

    public String toString()

    不常用方法：

    capacity() - 获取StringBuilder的容量

    因为StringBuilder是Java已经写好的类，所以java在底层对它进行了特殊处理，打印对象不是地址值而是属性值

    StringBuilder的常用场景：翻转字符串；拼接字符串

13. **链式编程**：当我们在调用一个方法的时候，不需要用变量接收它的结果，可以继续调用其他方法

14. **StringJoiner**: StringJoiner sj = new StringJoiner(", ", "[", "]")。也是一个容器，创建后内容可变。

    作用：提高字符串的操作效率，带你吗编写简介，但是在jdk8之后出现，因此用的人较少。

    创建： public StringJoiner(间隔符号), 或 public StringJoiner(间隔符号, 开始符号, 结束符号)

    常用方法：add, length, toString

15. jdk7及以前字符串拼接的底层原理：

    i. 拼接时没有变量参与：如String s = "a" + "b" + "c", 触发字符串的优化机制，在编译的时候已经是最终结果了。即运行时可视为 String s = "abc"

    ii. 拼接时有变量参与：如String a1 = "a", String s = s1 + "b"；内存中现在串池中创建"a", 随后添加"b"，然后再内存中创建StringBuilder, 最后再通过toString转换为一个新的字符串"ab"，在堆内存中，但不在串池中。

    jadk8字符串拼接的底层原理：

    预估字符串长度，创建一个数组，分别存储各个变量，再将整体变为字符串。

16. StringBuilder源码分析：

    i. 创建一个字符数组，默认容量为16。在StringBuilder中实际存储的是字符的ASCII表。

    ii. 如果容量不够则扩容，新容量 = 老容量*2+2.

    iii. 如果超出扩容容量，则以实际容量为准

9. 集合

1. 集合产生原因：需要一个长度可变的容器
2. 集合 - 只能存引用数据类型，不能存基本数据类型

   数组 - 可以存基本数据类型，引用数据类型
3. 泛型：用来限制数组中数据的类型，(api帮助文档中用表示)

集合创建方式：

ArrayList list = new ArrayList<>(); (**后面的泛型可省略**)

4. **集合的成员方法（增删改查）**

   add, remove, set, get, size

10. static

1. static: **所有类共享的对象**，被static修饰的变量，甚至可以通过类名直接调用。可以修饰成员方法，也可以修饰成员变量

   调用方式：i. 类名调用（推荐）　ii. 对象名调用

   static内存图：在堆内存中单独开辟静态存储位置（**静态区**），存储该类所有的静态变量。

   静态变量随着类的加载而加载，优先于对象

2. 静态方法：被static修饰的成员方法。

   特点：i. 多用在测试类和工具类中　ii. Javabean中很少会用

   工具类：可以帮助我们做一些事情，但是不描述任何事物。

   工具类需要私有化构造方法！！！ 原因：防止在外界创造该对象，因为创建该类毫无意义。如Math

3. static的注意事项：

   i. 静态方法**只能访问静态变量和静态方法**

   ii. 非静态方法**可以访问静态变量或者静态方法，也可以访问非静态的成员和方法**

   iii. 静态方法中**没有this**

   内存角度解释：

   i. jdk7之后，静态变量在静态区中，**静态方法找变量的时候，只会在静态区中寻找，因此不能调用非静态变量（实例变量）**

   ii. 对于非静态方法，由于静态方法无法找到调用者（this），因此不能调用非静态方法

11. 继承

1. **继承**：Java提供extends关键字，我们可以让一个类与另一个类建立起继承关系。

   public class Student extends Person {}

   Student称为子类，Person称为父类

   优点：

   i. 提高代码的复用性

   ii. 子类可以在父类的基础上，增加新的功能，使得子类的功能更加强大

2. **什么时候用继承**：类与类之间存在**相同的内容**，并满足**子类是父类的一种**，就可以考虑继承

   继承的特点：Java只支持单继承，不支持多继承，但支持多层继承。

   每一个类都直接或者间接的继承于Object

3. **子类继承父类中的内容：**

   构造方法：均私有，**都不能继承**

   成员变量：都能被子类继承下来，**但是私有部分不能直接调用，可以通过get和set进行操作**

   成员方法：非私有（虚方法，非private, 非static, 非final）被继承下来，而私有被继承下来

内存图：

i. 子类的相应内存中，含有父类的私有变量，但是不能直接访问

ii. 从顶级父类开始，java设置虚方法表，储存可能被经常调用的方法。（非private, 非static, 非final）。虚方法表会继承给子类，子类也会添加自己新的虚方法

4. 继承中的就近原则：

成员变量在使用时，首先在当前方法中寻找；当前方法中找不到，就在当前类中寻找；当前类中找不到，就在父类中寻找。

更精准的用法：当前方法内直接使用，当前类中使用时添加this.前缀，父类变量使用时添加super.前缀

最多只能调用到父类的变量，不能调用到更高

就近原则针对this, super都是适用的

5. **方法重写**：当父类的方法不能满足子类现在的需求时，需要进行方法重写。

代码标志：子类中出现了和父类中一模一样的方法声明，**在重写的方法上放@Override**.

方法重写的本质：在子类的虚方法表中发生覆盖

方法重写的注意事项：

i. 重写方法的名称、形参列表必须与父类中一致

ii. **子类重写父类方法时，访问权限子类必须大于等于父类。（访问权限：public > protected > private**）

iii. **子类重写父类方法时，返回值类型子类必须小于等于父类**。如存在继承关系Animinal->Dog, 只能父类返回Animinal，子类返回Dog，不能相反

iv. 重写方法尽量和父类保持一致

v. 只有被添加到虚方法表中的方法才能被重写

6. **继承中构造方法的访问特点**：

i. 父类的构造方法不会被子类继承（否则与类名不一致），因此子类默认只有无参构造

ii. 子类中所有的构造方法默认线访问父类中的无参构造，再执行自己（因为需要完成父类数据空间的初始化）

iii. **子类构造方法的第一行语句默认都是：super(), 不写也存在，且必须在第一行。如果想要调用父类的有参构造，必须手写super进行调用**

7. this, super使用总结

i. this: 理解为一个变量，表示当前方法调用者的地址值。**在虚拟机眼中，this只是一个局部变量**

使用空参创造对象时，如果需要**设置默认值**，方法为：

this(null, 0, "电子科技大学")

此时this相当于调用本类其他的构造方法，且虚拟机不再默认调用super()，因为其他类已经调用过了

ii. super: 代表父类存储空间

12. 多态

1. 多态：同类型的对象，表现出的不同形态。例：Person p = new Student(); **将子类赋值为父类的形态**

表现形式：父类类型 对象名称 = 子类对象；

多态的前提：i. 有继承关系 ii. 有父类引用指向子类对象 iii. 有方法重写

2. 作用：

   例：register(Person p), 里面传入的参数可以是Student, Teacher或Administor, 同时里面调用的函数（如show）也会是对应类型的。

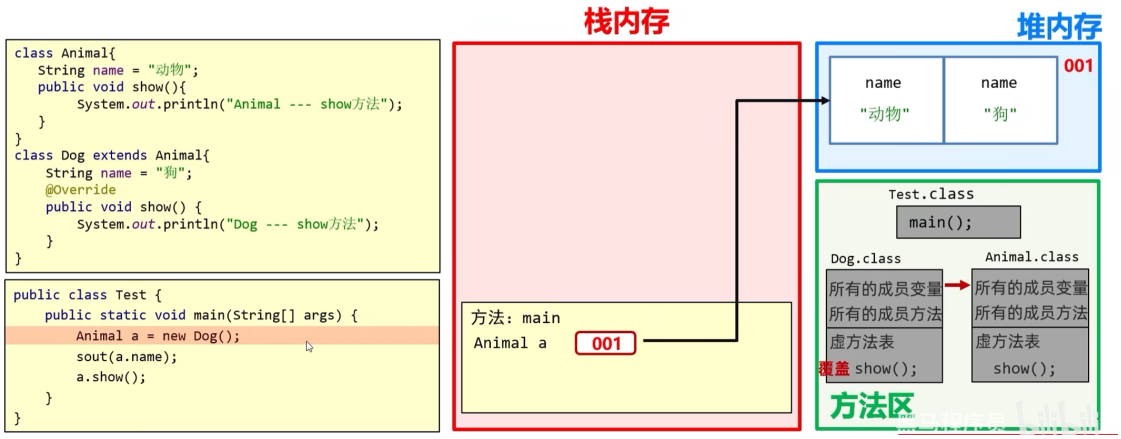
3. 多态调用成员的特点：

   主要针对：Animal a = new Dog();

   i. 变量调用：编译看左边，运行也看左边。（即Javac编译代码的时候，会看左边的父类中有没有这个变量。如果有，则编译成功；反之，则编译失败。）

   ii. 方法调用：编译看左边，运行看右边。（即Javac编译代码的时候，会看左边的父类中有没有这个方法。但是在运行的时候，如果子类进行了方法重写，则运行的是子类的方法。）

   内存图解：



4. 多态的优势：

   i. 在多态形式下，右边对象可以实现解耦合，便于扩展和维护。

   ii. 定义方法的时候，使用父类型作为参数，可以接收所有子类对象，体现多态的扩展性与便利。例：对于所有对象，都可以用Object指代。

   多态的弊端：

   不能调用子类的特有功能（解决方法：变回子类类型。如Dog d = (Dog) a; **这里不能随意转类型，比如将Dog转换为Cat。**这里可以使用instanceof进行判断，对象是否属于相应的类。）

   （jdk14之后提出新特性：a instanceof Cat d. 作用：如果a是Cat类型，则强转为Cat并赋值给d，否则不强转）

*—. IDEA快捷键

1. psvm: public static void main(String[] args);
2. sout: System.out.println();
3. fori：构建for(int i=0, i< ; i++)循环体
4. 构造类的时候的快捷键：alt+insert (或alt+fn+insert)。可以选择构造函数，set和get
5. 插件PTG 1s生成Javabean: 右键生成Javabean
6. 对于java自带的类，只需要在idea中输入一部分并按回车，就会自动补全import部分
7. 使用鼠标滚轮，可以竖着对idea中的代码进行选择