# JAVA学习总结

## java命令执行点class文件

1. 对于JAVAC命令编译点java文件，仅需要在当前目录下存在该名称对应的点java文件就可以了，但是对于点class文件的执行却要严格的多。当使用java命令执行某个点class文件时例如“C:\Users\17485\Desktop\java TestPackage”，如果在CLASSPATH环境变量中配有“.;”那么编译器会从当前目录寻找点class文件，将执行java命令的目录“C:\Users\17485\Desktop\”加上java命令后跟的点class文件前的包名作为具体路径，若文件前未加上包名，则默认为在当前目录下寻找该点class文件“TestPackage.class”,即寻找“C:\Users\17485\Desktop\TestPackage.class”文件。如果找到了，则会读取该文件中有没有package关键字指定的包信息，如果没有则可以正常执行，如果有则会比对package指定的路径“qiang/zhong/gao/”是否是包含在执行java命令的路径下“C:\Users\17485\Desktop\”,且“qiang\zhong\gao\”目录下是否存有TestPackage.class文件，如果以上都满足，则java命令可以正常执行，反之则会报错找不到主类。
2. 在WINDOWS环境中，CMD中执行包路径需要用“/”隔开，即“C:\Users\17485\Desktop\java qiang/zhong/gao/TestPackage”。
3. 使用“javac -d . 点java文件”可以在当前目录下创建点java文件中package关键字指定的目录结构。

## 面向对象于面向过程变成的区别

1. 面向对象编程思想是将一切行为都归属于一个实体对象，这个对象可以有众多的属性，各异的行为。
2. 面向过程编程思想是按照具体行为的不同部分发生的顺序描述行为。
3. 二者的区别在于面向对象会将一系列具有相同属性的行为包装成这些行为的主体的一个可以支持不同种特异性类别的行为，比如打篮球，踢足球，拍羽毛球都是对象人进行一项运动行为，运动的器具是某个物品，这就可以包装成一个对象“人”，具有运动的行为，运动行为支持不同的器具。而面向过程仅仅描述某一行为的发生过程，打篮球，踢足球，拍羽毛球这三项过程并没有多大的关联性，属于三个不同的过程。
4. 面向对象的优势在于基于是对于一系列具有相同属性行为的总结作为对象的一项行为，那么该对象的该行为可以同时支持多种特异性的行为，且这些行为具有相同的对象类型，便于统一管理。面向过程的又是在于不用考虑各行为之间的相互之间的关系，每个过程都是整体的一个独立的环节。

## java核心机制

1. java虚拟机（JVM）使得java程序仅需编写一次，在不同系统都可以运行，JVM支撑了java程序的跨平台属性。因为运行在不同操作系统上的java程序操作的是对应不同系统的JVM，向不同系统的JVM传达操作指令，JVM再把java程序的不同操作指令转换为对各自操作系统的操作。就如同java程序作为一个外国人，来到不同的国度，java程序仅需要向翻译人员表达请求，再由翻译人员想本地人员解释该如何执行。
2. 垃圾回收机制（GC，garbage collection）。相对于C，C++语言需要手动回收不使用的内存，java通过垃圾回收机制进行内存的回收，手动回收内存的优点在于可以在指定的地方立即回收内存，不需要等待，内存回收更加快速高效，缺点就在于程序员是否有意识地进行内存回收，如果忘记了就会存在内存被一直占用的风险，导致内存溢出。对比于C，C++的手动内存回收机制，java的自动内存回收机制因为是自动的，由系统进程自动进行内存回收，就不存在会忘记回收内存而导致内存溢出的情况，但是也有缺点，那就是内存的回收是不及时的，什么时候回收均由系统进程决定。

## 支持多版本开发

1. 可以在一台电脑上下载多个版本的JDK，解压缩在指定路径下，当需要不同版本的JDK时，可以通过修改环境变量中配置的JAVA\_HOME变量的地址，来达到多个版本相互切换，来满足多个不同项目开发的需求。

## System.out.print方法打印信息

1. 使用System.out.print()方法打印信息，可以在打印信息中防止转义字符来达到相应的效果，比如“\n”即为换行。

## 关键字

1. 定义：被java语言赋予了特殊的含义，用作专门用途的字符串。
2. 特点：关键字中所有字母均为小写。
3. 定义数据类型的关键字：class，interface，enum，byte，short，int，long，float，double，char，boolean，void
4. 定义数据类型值的关键字：false，true，null
5. 定义流程控制的关键字：if，else，switch，case，default，while，do，for，break，continue，return
6. 保留字：现有java版本未使用的后续版本可能使用的关键字，自己命名标识符时应尽量便面使用这些保留字，包括：byValue，cast，future，generic，inner，operator，outer，rest，var，goto，const

## 标识符命名规范

1. 包名：多单词组成时所有字母都小写。
2. 类名，接口名：多单词组成时，所有单词的首字符大写。
3. 变量名，方法名：多单词组成时， 第一个单词首字母小写，其余单词首字母大写。
4. 常量名：所有字母大写，多单词用下划线连接每一个单词。

## 数据类型

1. 在java语言中包含两大类数据类型：基本数据类型，引用数据类型。
2. 基本数据类型包括八种类型：整数类型（int，byte，short，long），浮点数型（float，double），字符型（char），布尔型（boolean）。
3. 引用数据类型包括三种：类，接口，数组。
4. 特别说明，String类型是作为引用数据类型的类。
5. 对于引用数据类型均可以赋值未null。
6. byte，char，short类型的变量在相互计算时会先转换成int，在进行计算，计算结果为int。
7. 基本数据类型容量大小为：double>float>long>int>char,byte,short。有多种基本数据类型的数据在混合计算时，会将所有数据的类型转换为其中容量最大的基本数据类型，再进行计算。
8. 任何基本类型的数据与String类型的数据相+，String左边进行正常的基本数据类型的加法运算，String右边全转换为String一次与String连接，最终结果为String类型。
9. 强制类型转换：（基本类型）数据，但是无法使用这种方式强制转换String数据为基本类型数据。

## 存储问题

1. java中内存分配涉及到的内存区域包括：寄存器（最快，无法在程序中控制），栈（速度仅次于寄存器，存放基本数据类型的变量及其数据以及对象的引用），堆（速度次于栈，存放由new创建的对象），静态域（存放对象中定义的静态成员），常量池（存放常量），非RAM存储（硬盘等永久存储空间）。
2. 栈中的数据的生命周期是确定的，存放于栈中的基本类型变量及其数量或引用当程序运行到他们作用域之外，或者引用不再指向对象，这些内存将会被java自动释放掉，可以被分配给其他变量使用。
3. 堆用来为由new创建的对象，数组分配内存，由java虚拟机的自动垃圾回收器来管理。可以为在堆中由new创建的对象，数组在栈中创建一个特殊的变量，让这个变量的值等于堆中对象，数组在堆内存中的首地址，栈中的这个变量就是堆中对象，数组的引用变量。可以通过引用变量来访问堆中的对象，数组。不同于存在于栈中的引用变量那样内存在程序运行到作用域之外时就会被系统回收，堆中的对象，数组即使没有引用指向，或者程序运行也超出了其作用域，他们在堆中的内存也不会被立即回收，但确定的时在某个时间会被回收掉。
4. 由于栈中的变量指向堆中的对象，数组，可以被看作为java中的指针。
5. 堆中的内存是在运行时动态分配的，，因此堆中的读取速度会比较慢，并且堆内存的回收是由垃圾回收器负责的。而栈中的数据，生存期都是确定的，确实灵活性，主要存放基本类型变量的数据（如int a = 10；即意味这会在栈中分配存放10的内存）于对象的应用。
6. 栈中数据的共享性：即在栈中不会创建两个或多个相同的数据，假设当前栈中不存在一个值为3的整型数据，那么此时通过int i = 3就会在栈中创建一个基本类型int型的变量存储数据为3，当第二次创建另一个变量且值也为3的时候即int j = 3，编译器会优先查找栈中是否已经存在整型3的数据，由于已经存在，则编译器会直接将变量j指向该整型3的数据，不会另外创建基本类型变量的数据。这和两个引用同时指向一个对象是有区别的，两个引用指向同一个对象，那么通过任一引用修改对象，必会影响另以引用的数值，但是栈中两个变量指向相同数据，改变一个变量的数据时他会重复刚创建时的行为，先寻找是否已存在，再创建，并不会影响之前指向的数据，因此并不会影响栈中另一个指向该数据的变量的值。
7. 成员变量：定义再方法外部，类的内部的变量。存储在堆内存中，由java的垃圾回收器负责回收。
8. 局部变量：定义在方法或语句块内部的变量，必须初始化。形式参数是局部变量。局部变量存储于栈中（如果局部变量是基本数据类型，那么局部变量的数据也存储于栈中，其他的类型则存储于堆中），随着方法的消失而有java系统立即回收。
9. 常量池：指的是在编译期被确定，并被保存在已编译的点class文件中的一些数据。包含代码中定义的基本类型和对象类型（String和数组）的常量值（final），以及一些以文本形式出现的符号引用如类和接口的全限定名、字段的名称和描述符、方法的名称和描述符。
10. JAVA中的String字符串：1）如果是编译期就已经创建好即直接使用双引号定义于初始化（如String str = “abc”；创建的时候编译器会先去常量池中寻找是否已存在相同内容“abc”的字符串，如果存在则会直接让str指向该字符串，如果不存在则会在常量池中创建一个内容为“abc”的字符串，由str指向），那么该String就存储于常量池中，如果是运行期才能确定String的具体内容即是用new关键字动态创建（如 String str = new String(“ABC”)；那么该String对象创建后存储于堆中，即使堆中已经存在内容为“ABC”的String对象，依然会重新创建另一个内容为“ABC”的String对象，String对象由str指向，str引用存储于栈中），那么该String则存储于堆中。2）如果两个字符串使用equals方法判等结果为true，那么如果这两个字符串存储于常量池中，则其实这两个字符串是同一个字符串，两个String变量同时指向而已；而对于这两个字符串都是由new创建的那么他们是存储于堆中不同的字符串，只不过String内容相同。3）使用双引号定义一个String对象，相比较于由new创建，既可以节省内存，由可以在一定程度上提交程序的运行速度，因为JVM会自动根据栈中数据的实际情况决定是否有必要创建新的对象，但是对于new关键字，则JVM一概会在堆中创建一个全新的对象。4）java会确保常量池中的String数据是唯一的即String字符串常量只有一个拷贝。由+号连接的多个字符串常量结果为常量存储于常量池中；由+号连接的字符串中有由new创建的字符串或者指向字符串常量的引用亦或者指向由new创建的字符串对象的引用，那么不管+号连接的字符串中是否存在字符串常量，由于在编译期无法确定字符串确定的值，所以结果是为一个存储于堆中的字符串对象。5）任意引用，不管是指向常量，亦或是指向new创建的对象，在编译期都无法确认其的值。6）对于连接多个字符串，如果使用+号运算符，那么其底层实现方式是每一次+操作，都创建一个StringBuilder对象，append +号操作符后的字符串，之后销毁这个StringBuilder对象，如果有多个+号，则会创建多个StringBuilder对象，每次使用完都会销毁掉；如果直接使用StringBuilder进行append，则可以仅创建一个StringBuilder对象来达到多个字符串连接的效果即StringBuilder sb = new StringBuilder(strA);

Sb.append(strB).append(strC).......，可以节省创建和销毁多个StringBuilder对象的时间，当需要循环拼接多个字符串是推荐使用创建一个StringBuilder对象进行，不使用+号此操作符；并且由于字符串的不变性，由+号操作符进行字符串拼接会在内存产生多个临时变量即为strA，strA+strB，strA+strB+strC.....都会产生一个字符串临时标量存储于常量池中，浪费存储空间。

1. 比较类中的数值是否相同使用点equals方法；比较两个对象引用是否同时指向同一个对象则使用“==”。对于String比较特殊，由于使用双引号定义的相同内容的String存储于常量池中，且两个变量同时指向该常量池String对象，所以str1.equals（str2）结果为true，str1 == str2结果也为true；但是当由new创建两个相同内容的对象，在堆中是存储于堆中不同位置的两个对象，那么分别指向这两个对象的变量并不是指向同一个对象，使用str1.equals(str2)结果为true，使用str1 == str2结果为false。
2. 对于final关键字，它仅是限定引用只能指向初始指向的那个对象（底层就是限制引用的值即为对象的地址只能是初始指向的对象的地址），改变指向的对象的值是可行的，但不能改变引用指向其他地址。

## （）