河南中医药大学信息技术学院（智能医疗行业学院）智能医学工程专业《计算机程序设计》课程

第01章：初始Java

黄子杰

河南中医药大学信息技术学院（智能医疗行业学院）与河南方和信息科技有限公司 联合建设

河南中医药大学信息技术学院智能医疗教研室

https://aitcm.hactcm.edu.cn

2025/2/16

本章概要

- 初识Java

Java是由 Sun Microsystems 公司于 1995 年 5 月推出的高级程序设计语言，历经几十年的发展已成长为从互联网到企业平台应用最广泛的编程语言，可运行于多个平台，易于编写，内置垃圾回收机制，不必考虑内存和指针问题。

本章概要

- Java市场有多大?

超过80%的高端企业级应用程序都在使用JAVA平台（电信，银行等），JAVA是具有数10年历史的成熟产品。亚马逊，谷歌，eBay，淘宝，京东，阿里巴巴和其他大型电子商务品牌都使用Java进行后台处理。如此众多的公司青睐Java，主要原因是Java具有良好的可伸缩性并可以处理更多的客户数据。

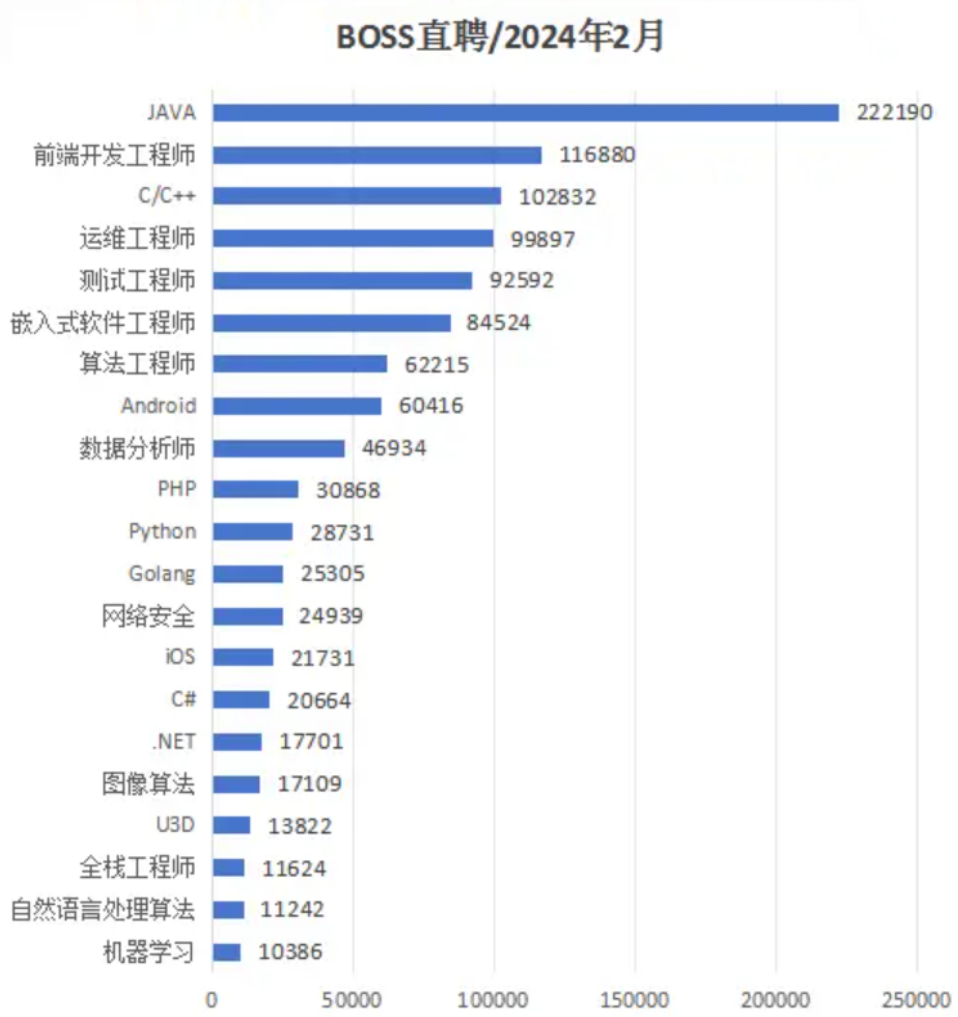
本章概要

- Java市场占有率

做为世界上最流行的编程语言之一，Java广泛应用于企业级应用开发、Android开发以及各种服务器端应用。根据Oracle的官方数据，截至2023年，Java的市场占有率大约为30%左右。

本章概要

- Java岗位需求



本章概要

- Java岗位薪资

1.1 Java语言发展简

- Java 是由 Sun Microsystems 公司于 1995 年 5 月推出的 Java 面向对象程序设计语言和 Java 平台的总称。它是由 James Gosling和同事们共同研发，并在 1995 年正式推出。2009年4月 Sun 公司被 Oracle （甲骨文）公司收购，Java 也随之成为 Oracle 公司的产品。

- Java分为三个体系：

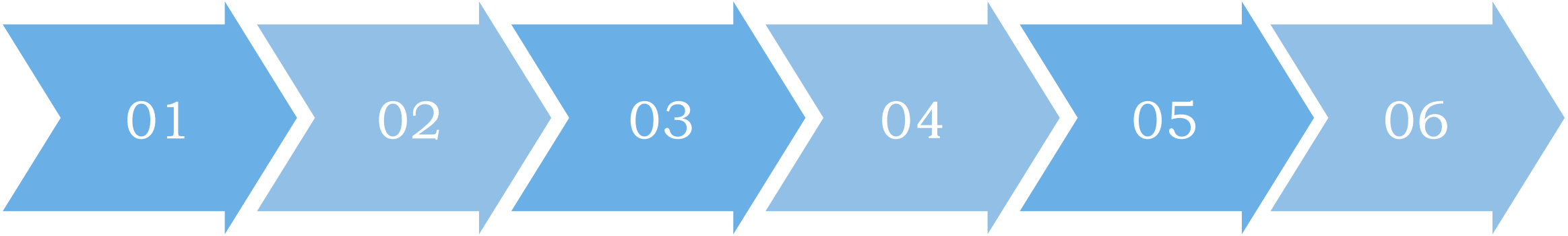
  JavaSE(J2SE) (Java2 Platform Standard Edition，java平台标准版)

  JavaEE(J2EE) (Java 2 Platform,Enterprise Edition，java平台企业版)

  JavaME(J2ME) (Java 2 Platform Micro Edition，java平台微型版)

- 2005 年 6 月，JavaOne 大会召开，SUN 公司公开 Java SE 6。此时，Java 的各种版本已经更名，以取消其中的数字 "2"：J2EE 更名为 Java EE，J2SE 更名为Java SE，J2ME 更名为 Java ME。

1.1 Java语言发展简



1996年1月

SUN公司推出了JDK1.0。在JDK1.0发布后，有很多程序员学习并运用Java来制作网页，JDK包括两大部分：开发工具和运行环境。

1998年12月8日

JDK1.2——第二代Java平台的企业版J2EE发布。

1999年6月

Sun公司把Java体系分为三个方向：

J2ME(微型版)

J2SE(标准版)

J2EE(企业版)

2000年5月

JDK1.3、JDK1.4和J2SE1.3相继发布，J2SE1.3是对J2SE1.2的补充和扩展，从应用领域的角度分析，JavaSE1.3已经涵盖了数据库、WEB、网络、图形、多媒体、电话、影像等大部分的信息技术领域。

2005年6月

在Java One大会上，Sun公司发布了Java SE 6。此时，Java的各种版本已经更名，已取消其中的数字2，如J2EE更名为JavaEE，J2SE更名为JavaSE，J2ME更名为JavaME。

2009年4月20日

甲骨文74亿美元收购Sun，取得Java的版权； 收购后，Java进入了新的发展阶段，其持续的影响力和应用范围得到了进一步的扩展。

1.2 认识Java语言

- Java是一种面向对象的编程语言，它设计的基础是“Write Once, Run Anywhere”原则，也就是一次编写，到处运行。Java有一个广泛的标准库，这使得开发者能够编写一次，然后在任何支持Java的平台上重复使用。

- 一个最简单的Java程序

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

要运行这个程序，你需要有一个Java编译器（JDK），然后使用以下命令来编译和运行程序：

```
javac HelloWorld.java
java HelloWorld
```

1.2 认识Java语言

- **简单**

Java 语言的语法与 C 语言和 C++ 语言很接近，使得大多数程序员很容易学习和使用。另一方面，Java 丢弃了 C++ 中很少使用的、很难理解的、令人迷惑的那些特性，如操作符重载、多继承、自动的强制类型转换。特别地，Java 语言不使用指针，而是引用。并提供了自动分配和回收内存空间，使得程序员不必为内存管理而担忧。

- **面向对象**

Java 语言提供类、接口和继承等面向对象的特性，为了简单起见，只支持类之间的单继承，但支持接口之间的多继承，并支持类与接口之间的实现机制（关键字为 implements）。Java 语言全面支持动态绑定，而 C++语言只对虚函数使用动态绑定。总之，Java语言是一个纯的面向对象程序设计语言。

- **分布式**

Java 语言支持 Internet 应用的开发，在基本的 Java 应用编程接口中有一个网络应用编程接口（java net），它提供了用于网络应用编程的类库，包括 URL、URLConnection、Socket、ServerSocket 等。Java 的 RMI（远程方法激活）机制也是开发分布式应用的重要手段。

- **健壮**

Java 的强类型机制、异常处理、垃圾的自动收集等是 Java 程序健壮性的重要保证。对指针的丢弃是 Java 的明智选择。Java 的安全检查机制使得 Java 更具健壮性。

- **安全**

Java通常被用在网络环境中，为此，Java 提供了一个安全机制以防恶意代码的攻击。除了Java语言具有的许多安全特性以外，Java 对通过网络下载的类具有一个安全防范机制（类 ClassLoader），如分配不同的名字空间以防替代本地的同名类、字节代码检查，并提供安全管理机制（类 SecurityManager）让 Java 应用设置安全哨兵。

- **体系结构中立**

Java 程序（后缀为 java 的文件）在 Java 平台上被编译为体系结构中立的字节码格式（后缀为 class 的文件），然后可以在实现这个 Java 平台的任何系统中运行。这种途径适合于异构的网络环境和软件的分发。

- **方便移植**

这种可移植性来源于体系结构中立性，另外，Java 还严格规定了各个基本数据类型的长度。Java 系统本身也具有很强的可移植性，Java 编译器是用 Java 实现的，Java 的运行环境是用 ANSI C 实现的。

- **解释型语言**

Java 程序在 Java 平台上被编译为字节码格式，然后可以在实现这个 Java 平台的任何系统中运行。在运行时，Java 平台中的 Java 解释器对这些字节码进行解释执行，执行过程中需要的类在联接阶段被载入到运行环境中。

- **高性能**

与那些解释型的高级脚本语言相比，Java 的确是高性能的。Java 的运行速度随着 JIT(Just-In-Time）编译器技术的发展越来越接近于 C++。

- **多线程**

在 Java 语言中，线程是一种特殊的对象，它必须由 Thread 类或其子（孙）类来创建。通常有两种方法来创建线程：其一，使用型构为 Thread(Runnable) 的构造子类将一个实现了 Runnable 接口的对象包装成一个线程，其二，从 Thread 类派生出子类并重写 run 方法，使用该子类创建的对象即为线程。值得注意的是 Thread 类已经实现了 Runnable 接口，因此，任何一个线程均有它的 run 方法，而 run 方法中包含了线程所要运行的代码。线程的活动由一组方法来控制。Java 语言支持多个线程的同时执行，并提供多线程之间的同步机制（关键字为 synchronized）。

- **动态语言**

Java 语言的设计目标之一是适应于动态变化的环境。Java 程序需要的类能够动态地被载入到运行环境，也可以通过网络来载入所需要的类。这也有利于软件的升级。另外，Java 中的类有一个运行时刻的表示，能进行运行时刻的类型检查。

1.2 认识Java语言

- Java类库是Java的应用程序接口(API)，通常以包的形式组织类库。

1、java.lang:默认导入的包，提供程序设计的常用基础类和接口；

2、java.util:工具类库，提供包含集合框架、集合类、日期时间等工具类；

3、java.io:Java的标准输入输出流；

4、java.applet:实现Java Applet 小程序的类库；

5、java.net:提供实现网络应用与开发的类库；

6、java.sql:提供访问并处理存储在数据源(通常是关系型数据)中的数据；

7、java.awt和java.swing:提供用户构建图形用户界面的类库；

8、java.awt.event:图形界面中用户交互控制和事件相应类库；

- Java.lang类库

String类

1.String对象是用来保存字符串的，也就是一组字符序列

2.字符串常量是用双引号括起来的字符序列例如"你好","12.97"等

3.字符串使用Unicode编码，一个字符（不管是汉字还是字母）占两个字节

4.String有多个构造器

5.String类实现了接口Serializable（可以串行化：在网络上传输）和接口Comparable（可以进行大小比较）

6.String是final类，不能被其他的类继承

7. value是final类型，不能更改（是他指向不可修改，而不是指向的里面的内容不可修改）

- Java.lang类库

StringBuffer类

1. java.lang.Stringbuffer代表可变的字符序列，可以对字符串内容进行增删。

2.很多方法与String相同，但StringBuffer是可变长度的。

3. StringBuffer是一个容器。

- Java.lang类库

String 与 StringBuffer对比

1. String保存的是字符串常量，里面的值不能更改，每次String类的更新实际上就是更改地址，效率较低。

2. StringBuffer保存的是字符串变量，里面的值可以更改,每次StringBuffer的更新实际上可以更新内容，不用每次更新地址，效率较高。

- Java.lang类库

StringBuilder类

1.一般用在单线程，和StringBuffer不同StringBuffer用在多线程

2.一个可变的字符序列，此类提供一个与StringBuffer兼容的API，但不保证同步(StringBuilder不是线程安全的)。用在字符串缓冲区被单个线程使用的时候，该类可以用作StringBuffer的一个简易替换，大多数实现中StringBuilder比StringBuffer要快。

3.在StringBuilder上的主要操作是append和insert方法，可重载这些方法以接受任意类型的数据。

1.2 认识Java语言

- Java.lang类库

String、 StringBuffer、 StringBuilder比较

1. StringBuffer和StringBuilder非常类似，均代表可变的字符序列，而且方法也一样
2. String：不可变字符序列，效率较低，但复用率高。
3. StringBuffer:可变字符序列，增加删除时效率较高，且线程安全的。
4. StringBuilder:可变字符序列，效率最高，线程不安全。

String使用注意说明:

String str = "a";//创建了一个字符串

str += "b";//实际上原来的"a"字符串对象已经丢弃了，现在又产生了一个字符串"ab"。如果多次执行这些改变字符串内容的操作会导致大量副全字符串对象留在内存中，降低了效率。如果这样的操作放在循环中会极大影响程序的性能。

替换方案:

1.如果字符串存在大量的修改操作，一般使用StringBuffer或StringBuilder

2.如果字符串存在大量的修改操作并在单线程的情况，使用StringBuilder

3.如果字符串存在大量的操作操作并在多线程的情况，使用StringBuffer

4.如果字符串很少修改且被多个对象引用，使用String，如配置信息、中间信息

- Java.util类库

java.util是Java标准类库中的一个非常重要的组成部分，它提供了一系列对程序开发非常有用的类和接口。这个包主要包含集合框架、日期时间类、事件模型、随机数生成器以及其他实用工具类。

- 常用类及接口

1. 集合
2. 日期时间类
3. 事件模型
4. 随机数生成器
5. 其他实用工具类

- 集合

集合是java.util包中最重要的部分之一，它提供了一系列的数据结构和算法，帮助开发者存储和操作数据集合。它包含:

Collection - 集合框架的根接口，代表一组对象，这些对象称为元素。

List - 有序集合，每个元素都有其在集合中的特定位置。

Set - 不允许重复元素的集合。

Map - 键值对的集合，也称为字典或哈希表。

ArrayList - 实现了List接口的动态数组。

LinkedList - 实现了List接口的双向链表。

HashSet - 实现了Set接口的哈希表集合。

LinkedHashSet - 继承自HashSet，维护元素插入顺序的集合。

TreeSet - 实现了Set接口的基于红黑树的集合，元素按自然顺序或自定义顺序排序。

HashMap - 实现了Map接口的哈希表。

LinkedHashMap - 继承自HashMap，维护元素插入顺序或访问顺序的映射。

TreeMap - 实现了Map接口的基于红黑树的映射，键按自然顺序或自定义顺序排序。

- 日期时间类

提供了操作日期和时间的类。它包含:

Date - 表示特定的瞬间，精确到毫秒。

Calendar - 可以用于日期的计算，比如添加或减少天数、获取星期等。

TimeZone - 表示时区。

Locale - 表示特定的语言、国家和区域的设置。

- **事件模型**

用于处理事件和监听器模式。它包含:

EventObject - 所有事件的超类。

EventListener - 所有事件监听器接口的标记接口。

EventSource - 能够生成事件的对象。

- 随机数生成器

提供随机数生成的功能。它包含:

Random - 基于线性同余生成器的随机数生成器。

java.security.SecureRandom - 提供更安全的随机数生成。

- **其他实用工具类**

提供了一系列静态方法，用于执行各种任务。它包含:

Arrays - 提供静态方法来操作数组，如排序、搜索等。
Collections - 提供静态方法来操作集合，如排序、搜索等。
Objects - 提供静态方法来操作对象，如空安全的比较、计算哈希码等。
Timer - 可以调度任务在将来的某个时间点执行。
UUID - 用于生成UUID（通用唯一标识符）。

- Java作为近几十年最受欢迎的编程语言之一，因其具有跨平台、面向对象、安全性高等优点，广泛应用于各种领域。

1. Web应用

Java是Web应用开发的主流选择。Java 的一个最重要的应用场景就是基于Java的Web应用程序。Java Servlet、JavaServer Pages（JSP）和JavaServer Faces（JSF）等技术，已经成为普遍可用的Web应用服务器解决方案。

2. 移动应用

随着移动设备越来越普及，Java技术也开始广泛应用于移动端应用程序的开发。Android系统基于Java技术栈构建，使得Java成为了移动应用开发的重要工具。许多大型企业都在使用Java技术来开发自己的移动应用程序。

3. 大数据处理

Java广泛用于大数据处理，如Hadoop框架等。Hadoop是处理大数据的一个开源软件项目，它底层使用的是Java语言。Hadoop将数据切分为多个数据块，存储在不同的服务器上，程序可以并行执行，从而实现快速、高效的数据处理。

4. 金融业

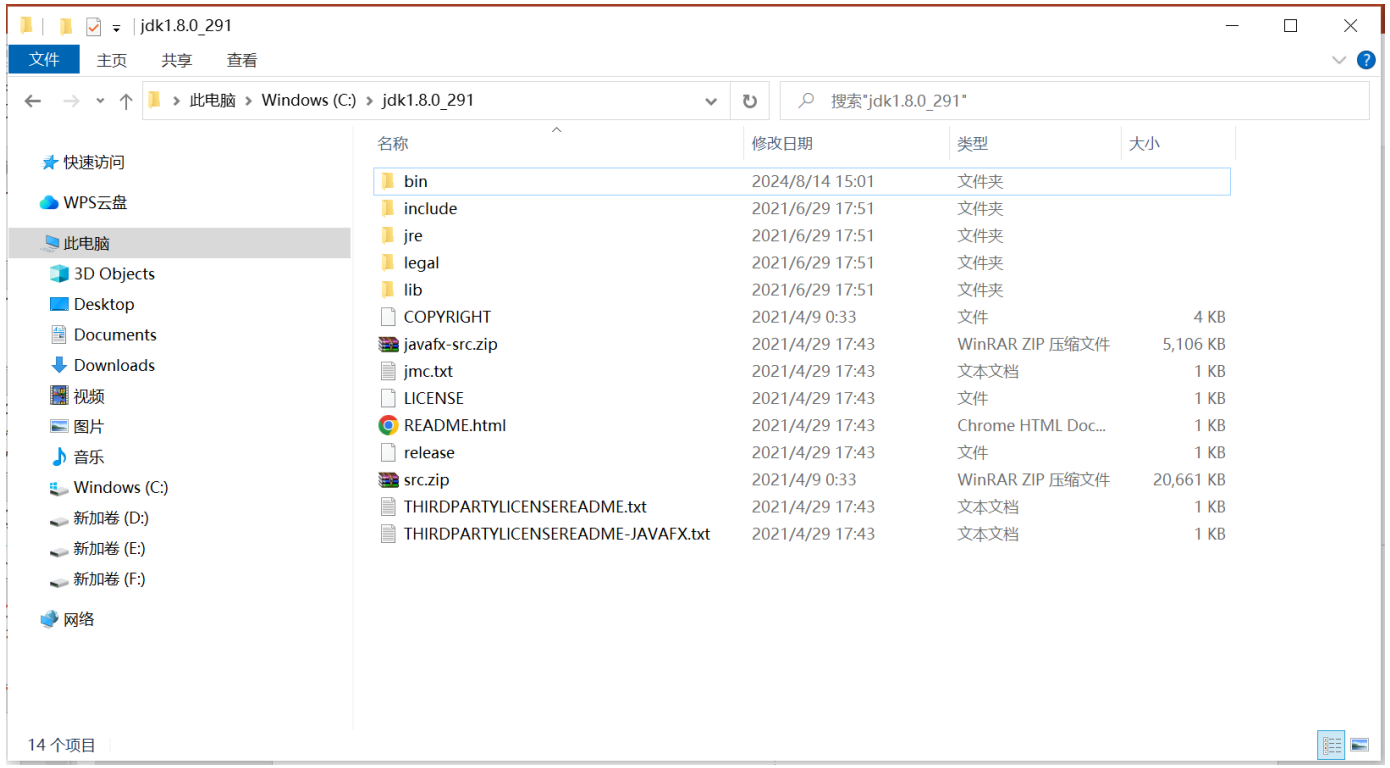
Java在金融领域有着广泛的应用，如证券市场交易系统、银行核心处理系统等。Java的高安全性和可靠性，使得它成为了这些应用的首选编程语言。

5. 游戏开发

Java技术也开始应用于游戏开发领域。Java能够处理游戏中的图形和音频，还可以方便地使用外部库和API进行游戏开发。例如，我的世界(Minecraft)游戏就是基于Java平台开发的。

JDK

JDK是SUN提供的一套Java开发环境，全称JavaDevelopmentKit，简称JDK，它是整个Java的核心，其中包括Java编译器、Java运行工具、Java文档生成工具、Java打包工具等。
JDK分解压版和安装版，在JDK安装完毕后，会在硬盘上生成一个目录，该目录就是JDK的安装目录。



JDK目录和文件

- bin目录

该目录用于存放一些可执行程序，如javac.exe（Java编译器）、java.exe（Java运行工具）、jar.exe（打包工具）和javadoc.exe（文档生成工具）等。

- db目录

db目录是一个小型的数据库。从JDK 6开始，Java中引入了一个新的成员Java DB，这是一个纯Java实现、开源的数据库管理系统。这个数据库不仅很轻便，而且支持JDBC 4.0所有的规范，在学习JDBC时，不再需要额外地安装一个数据库软件，选择直接使用Java DB即可。

- include目录

由于JDK是通过C和C++实现的，因此在启动时需要引入一些C语言的头文件，该目录就是用于存放这些头文件的。

JDK目录和文件

- jre目录

此目录是Java运行时环境的根目录，它包含Java虚拟机，运行时的类包、Java应用启动器以及一个bin目录，但不包含开发环境中的开发工具。

- lib目录

lib是library的缩写，意为Java类库或库文件，是开发工具使用的归档包文件。

- javafx-src.zip文件

该压缩文件内存放的是Java FX（Java图形用户界面工具）所有核心类库的源代码。

JDK目录和文件

- src.zip文件

src.zip为src文件夹的压缩文件，src中放置的是JDK核心类的源代码，通过该文件可以查看Java基础类的源代码。

- README

说明性文档。

配置环境变量

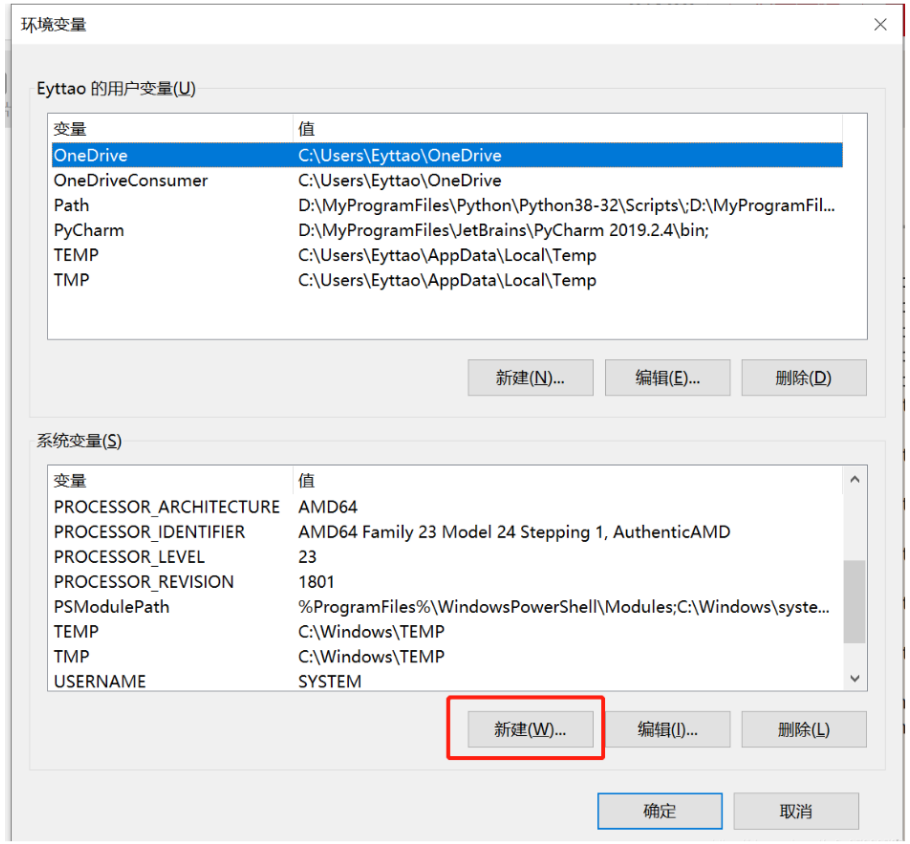
1.右击“我的电脑”→“属性”→“高级系统设置”→“高级”→“环境变量”；

配置环境变量

2.选择"高级"选项卡，点击"环境变量";

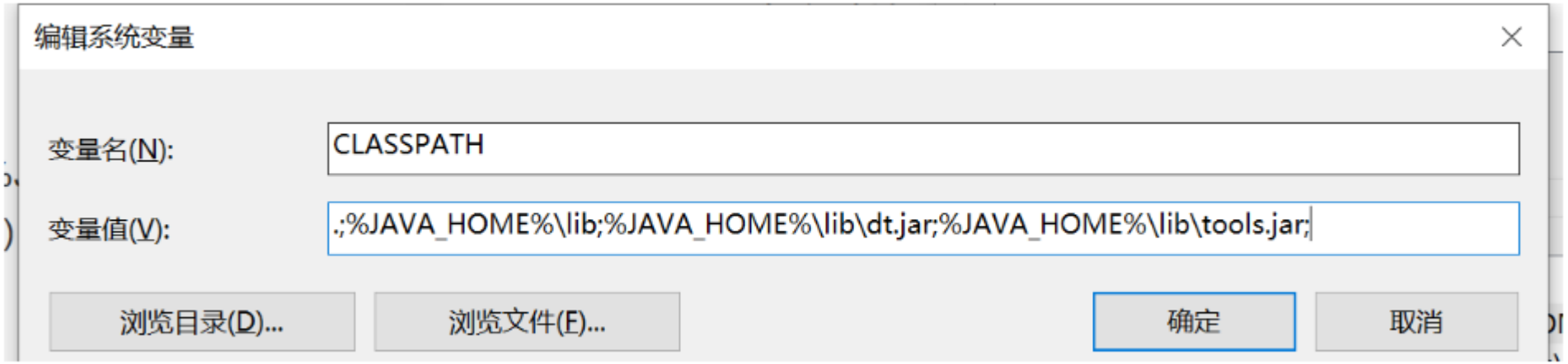
配置环境变量

3. 新建“JAVA_HOME”系统变量（点击“系统变量”下方的“新建”按钮，填写变量名与变量值，点击“确定”）



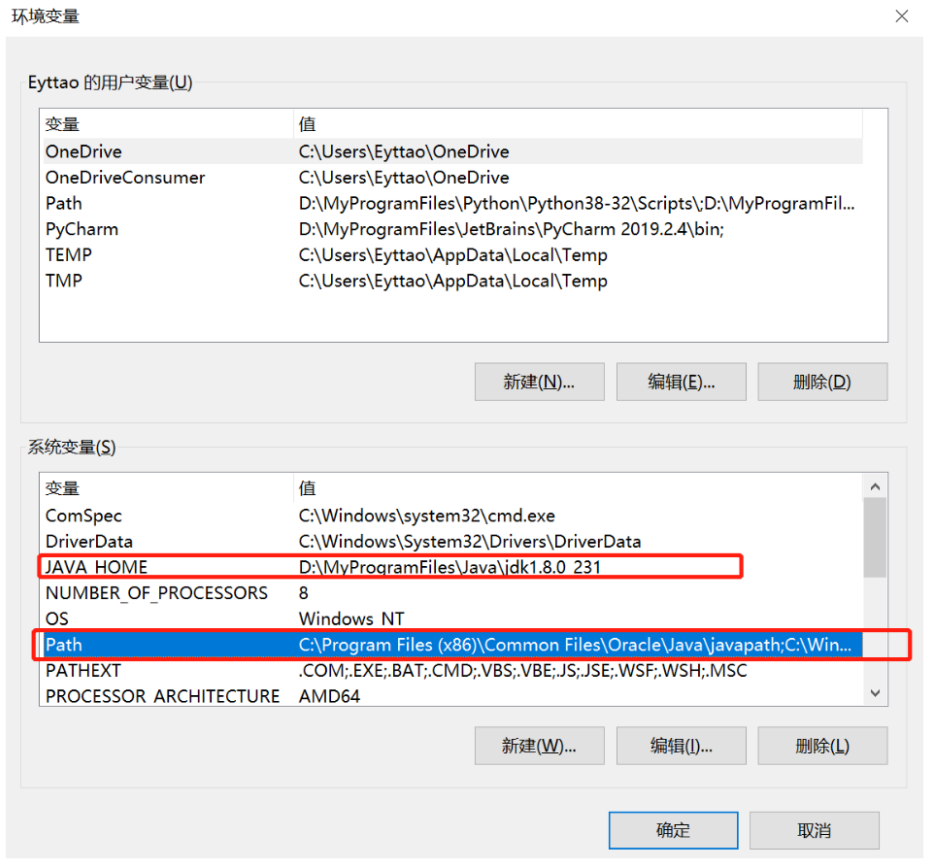
配置环境变量

4. 同上，新建“CLASSPATH”系统变量，变量值为“.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;”。（引号内的全部内容，注意最前方是点不是逗号）



配置环境变量

5. 双击“系统变量”下的“Path”变量进行编辑。（此时可以看到JAVA_HOME已经存在于系统变量中），（有的电脑"Path"也写作“PATH”）



配置环境变量

在"系统变量"中设置3项属性，JAVA_HOME,PATH,CLASSPATH(大小写无所谓)，若已存在则点击"编辑"，不存在则点击"新建"。

总结一下:

环境变量设置参数

变量名：JAVA_HOME

变量值：C:\Program Files (x86)\Java\jdk1.8.0_91　　// 要根据自己的实际路径配置

变量名：CLASSPATH

变量值：.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;　//记得前面有个"."

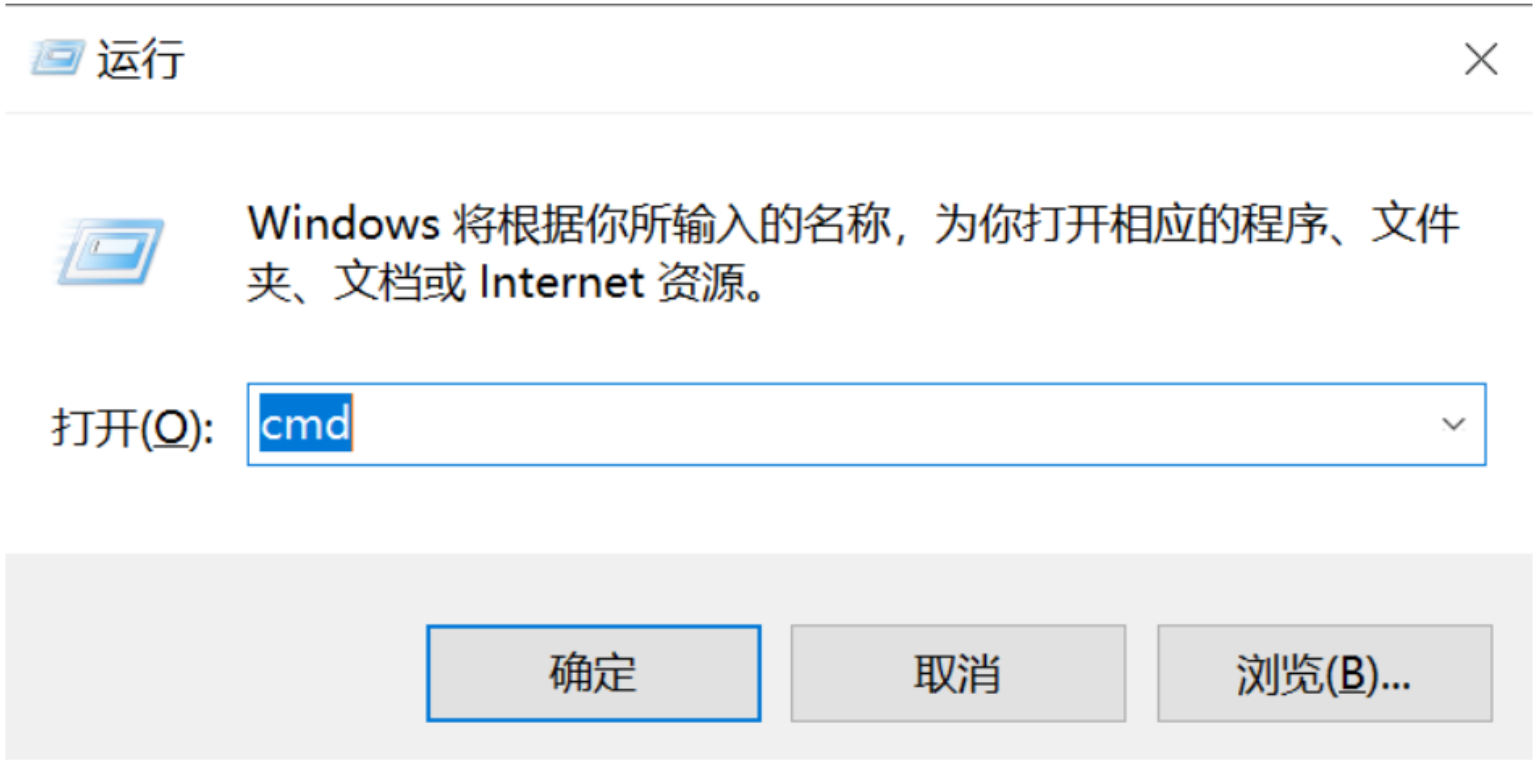
变量名：Path

变量值：%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;

1.2 认识Java语言

通过控制台测试JDK是否安装成功

1、同时按键盘上“win”、“R”两个键打开运行，输入“cmd”确定打开控制台。



通过控制台测试JDK是否安装成功

2、键入命令: java -version、java、javac 几个命令，出现以下信息，说明环境变量配置成功;

通过控制台测试JDK是否安装成功

2、键入命令: java -version、java、javac 几个命令，出现以下信息，说明环境变量配置成功;

```
C:\Users\Lenovo>javac
用法: javac <options> <source files>
其中, 可能的选项包括:
  -g                         生成所有调试信息
  -g:none                    不生成任何调试信息
  -g:{lines,vars,source}     只生成某些调试信息
  -nowarn                    不生成任何警告
  -verbose                   输出有关编译器正在执行的操作的消息
  -deprecation               输出使用已过时的 API 的源位置
  -classpath <路径>            指定查找用户类文件和注释处理程序的位置
  -cp <路径>                   指定查找用户类文件和注释处理程序的位置
  -sourcepath <路径>           指定查找输入源文件的位置
  -bootclasspath <路径>        覆盖引导类文件的位置
  -extdirs <目录>              覆盖所安装扩展的位置
  -endorseddirs <目录>         覆盖签名的标准路径的位置
  -proc:{none,only}          控制是否执行注释处理和/或编译。
  -processor <class1>[,<class2>,<class3>...] 要运行的注释处理程序的名称; 绕过默认的搜索进程
  -processorpath <路径>        指定查找注释处理程序的位置
  -parameters                生成元数据以用于方法参数的反射
  -d <目录>                    指定放置生成的类文件的位置
  -s <目录>                    指定放置生成的源文件的位置
  -h <目录>                    指定放置生成的本机标头文件的位置
  -implicit:{none,class}     指定是否为隐式引用文件生成类文件
  -encoding <编码>             指定源文件使用的字符编码
  -source <发行版>              提供与指定发行版的源兼容性
```

1.2 认识Java语言

通过控制台测试JDK是否安装成功

2、键入命令: java -version、java、javac 几个命令，出现以下信息，说明环境变量配置成功;

```
C:\Users\Lenovo>java
用法: java [-options] class [args...]
           (执行类)
   或  java [-options] -jar jarfile [args...]
           (执行 jar 文件)
其中选项包括:
    -d32          使用 32 位数据模型 (如果可用)
    -d64          使用 64 位数据模型 (如果可用)
    -server       选择 "server" VM
                  默认 VM 是 server.

    -cp <目录和 zip/jar 文件的类搜索路径>
    -classpath <目录和 zip/jar 文件的类搜索路径>
                  用 ; 分隔的目录, JAR 档案
                  和 ZIP 档案列表, 用于搜索类文件。
    -D<名称>=<值>
                  设置系统属性
    -verbose:[class|gc|jni]
                  启用详细输出
    -version      输出产品版本并退出
    -version:<值>
                  警告: 此功能已过时, 将在
                  未来发行版中删除。
                  需要指定的版本才能运行
    -showversion  输出产品版本并继续
    -jre-restrict-search | -no-jre-restrict-search
                  警告: 此功能已过时, 将在
                  未来发行版中删除。
                  在版本搜索中包括/排除用户专用 JRE
```

- Debug

Debug，意为“调试”，是程序员必备技能之一。

- **Debug的基本方法**

1. 断点（breakpoint）

打上断点以后，程序运行到断点处就会暂停，可以一步一步观察运行情况。

2. 跟踪（trace）

跟着流程一步一步走，看看程序代码的执行流程。

跟着流程一步一步走，看看变量动态的变化情况。

3. 监视（watch）

即时监视：鼠标指向变量

快速监视：点右键，Inspector

添加监视：点右键，Watch

1.2 认识Java语言

- IntelliJ_IDEA的Debug

最简单的程序

先随着一个最简单的程序看IDEA的Debug流程。

我们选择一个1+2+…+10的程序，我们通过这个最基本的程序了解如何去Debug。

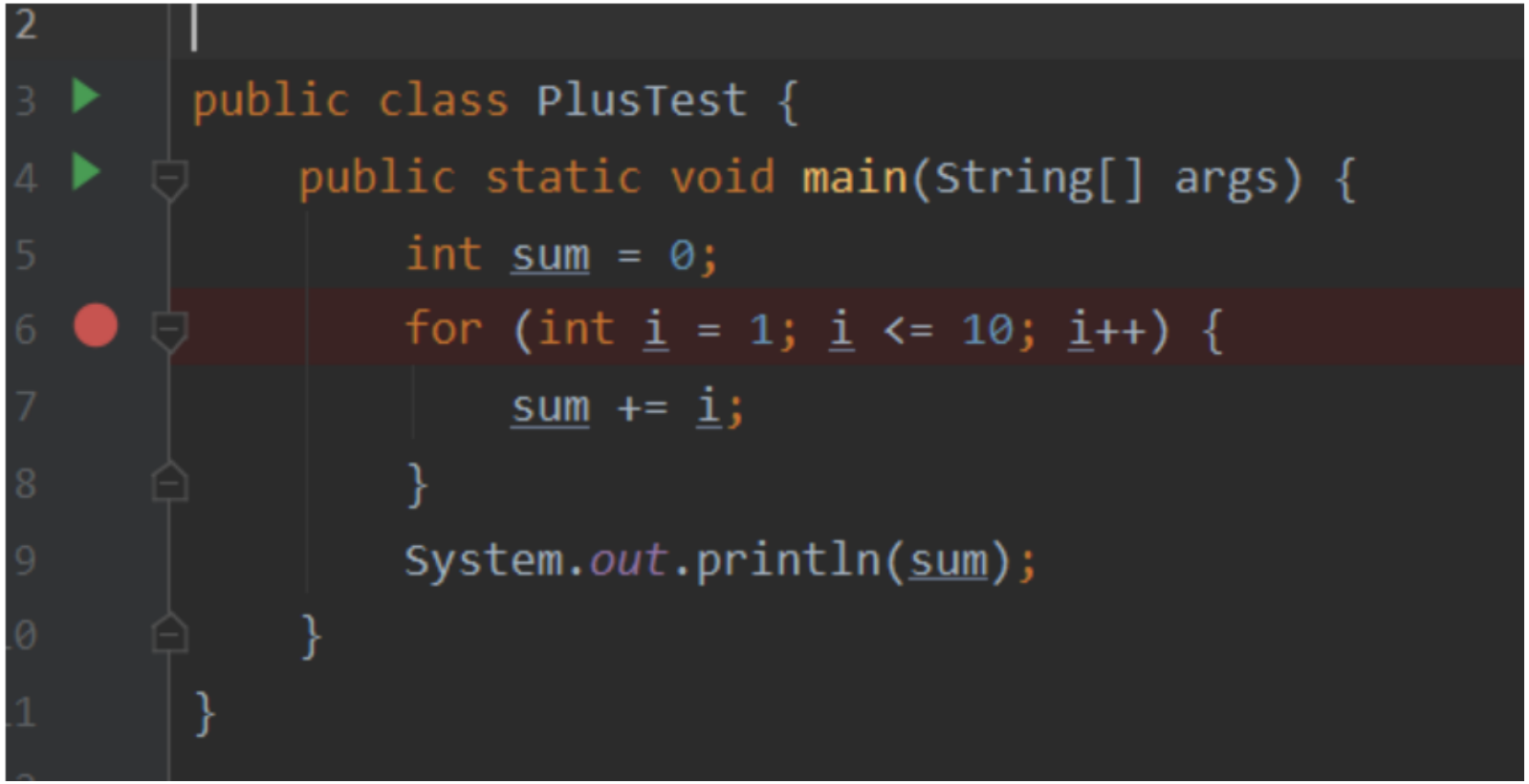
推荐用F8来逐行运行观察，发现问题再追踪。

```
public class PlusTest {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 1; i <= 10; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

1.2 认识Java语言

- IntelliJ_IDEA的Debug

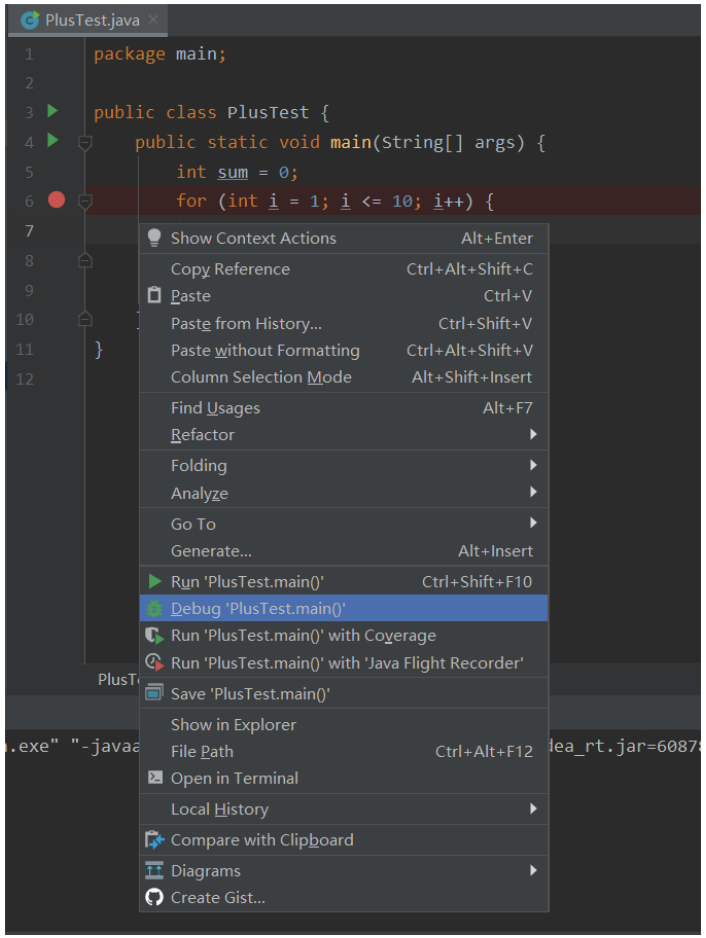
在标注行数的左边栏的空白处，点一下，会出现一个红点，带出一条红线，这叫断点。打断点的目的是使程序Debug运行到这里的时候会停住，我们能逐步地观察变量的变化、程序语句执行的流程。



1.2 认识Java语言

- IntelliJ_IDEA的Debug

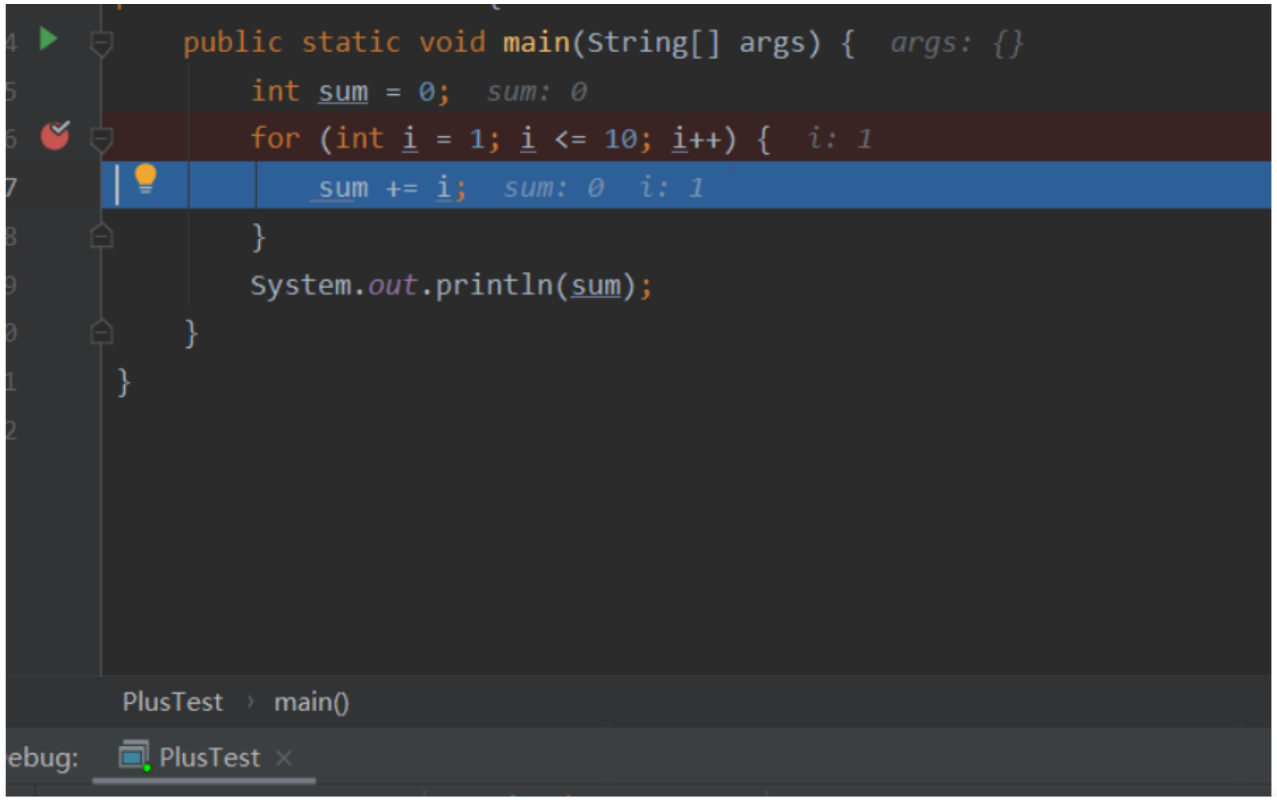
右键Debug运行，注意不要Run，Run的话断点就无效了。



1.2 认识Java语言

- IntelliJ_IDEA的Debug

开始运行后，可以看出程序停在了打断点的那一行处，并且有很多灰色的k-v对，下方的Variable栏就会出现各种变量的值，可以追踪各个变量的当前值和值的变化。

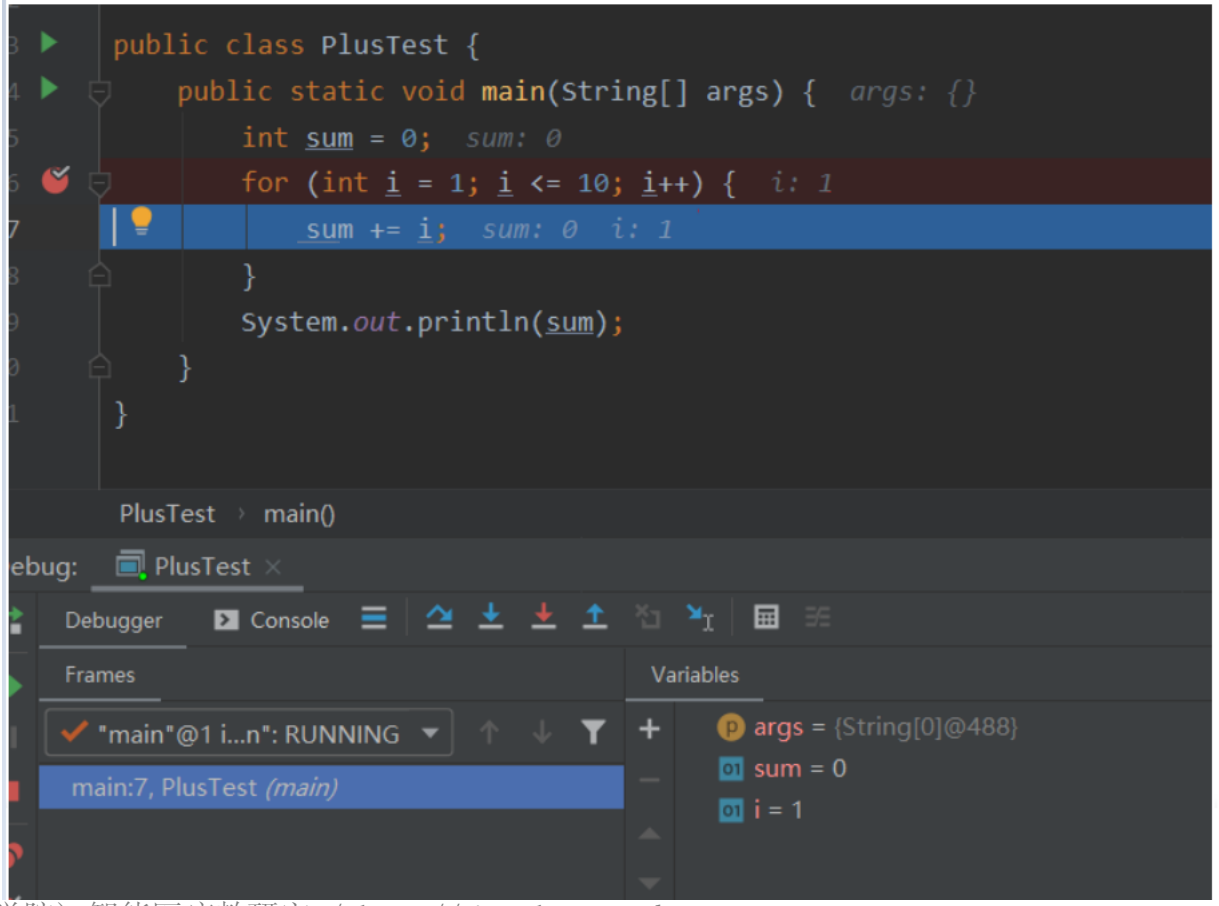


1.2 认识Java语言

- IntelliJ_IDEA的Debug

开始运行后，可以看出程序停在了打断点的那一行处，并且有很多灰色的k-v对，下方的Variable栏就会出现各种变量的值，可以追踪各个变量的当前值和值的变化。

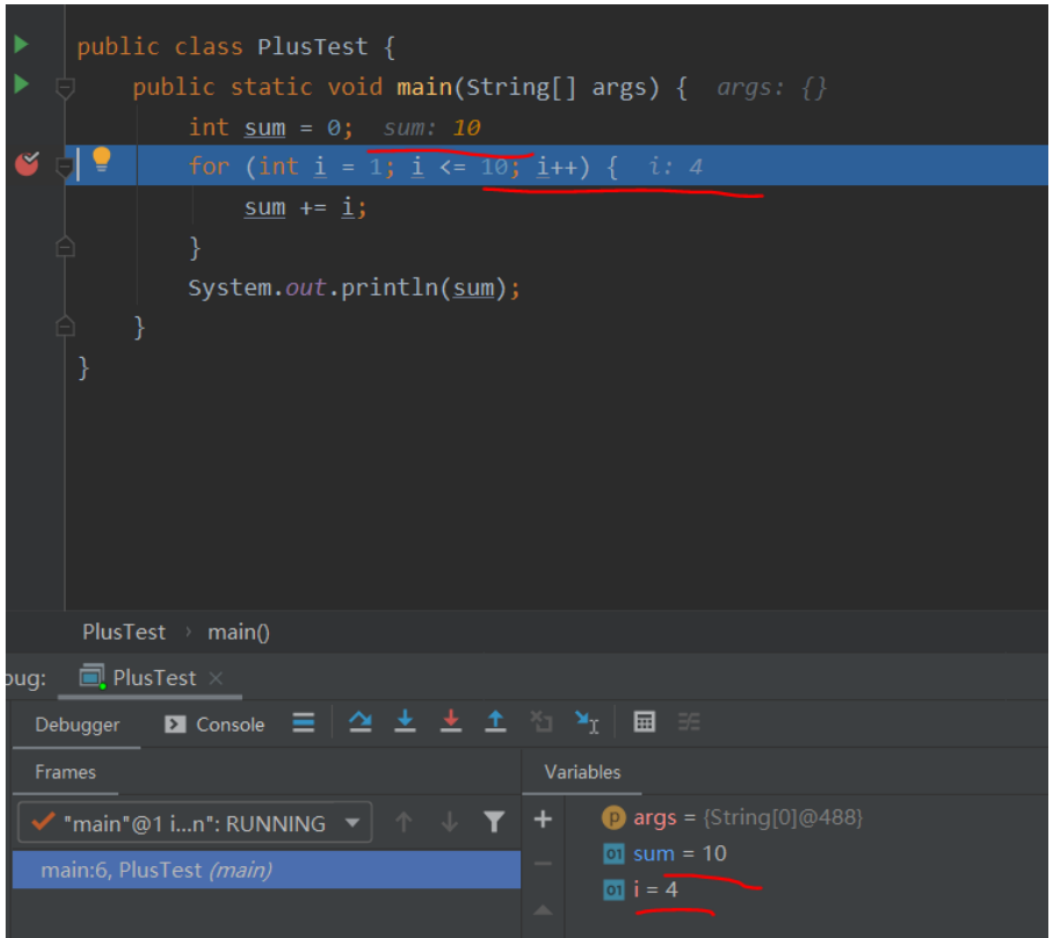
按F8就可以继续执行。

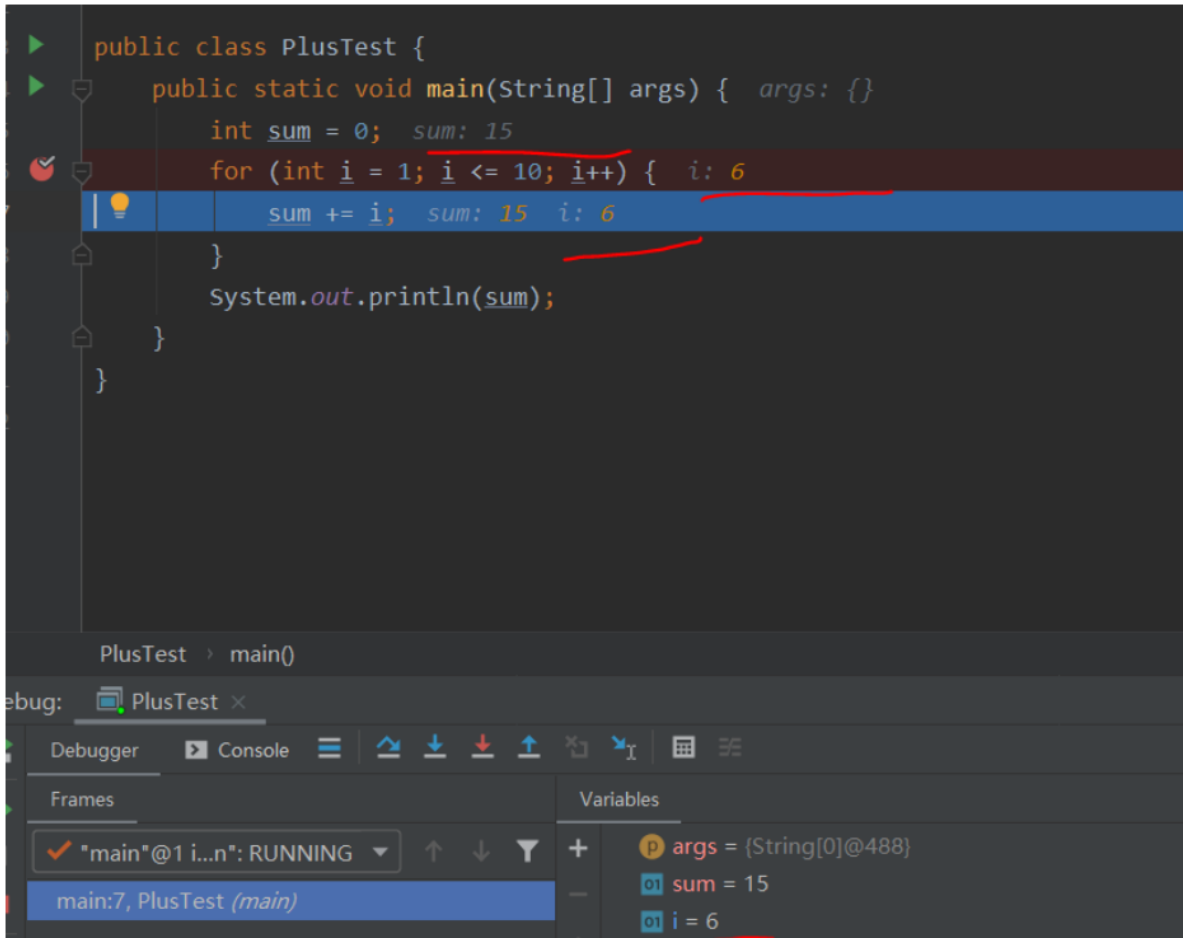


1.2 认识Java语言

- IntelliJ_IDEA的Debug

Variable栏中蓝色的变量表示刚刚被修改过：

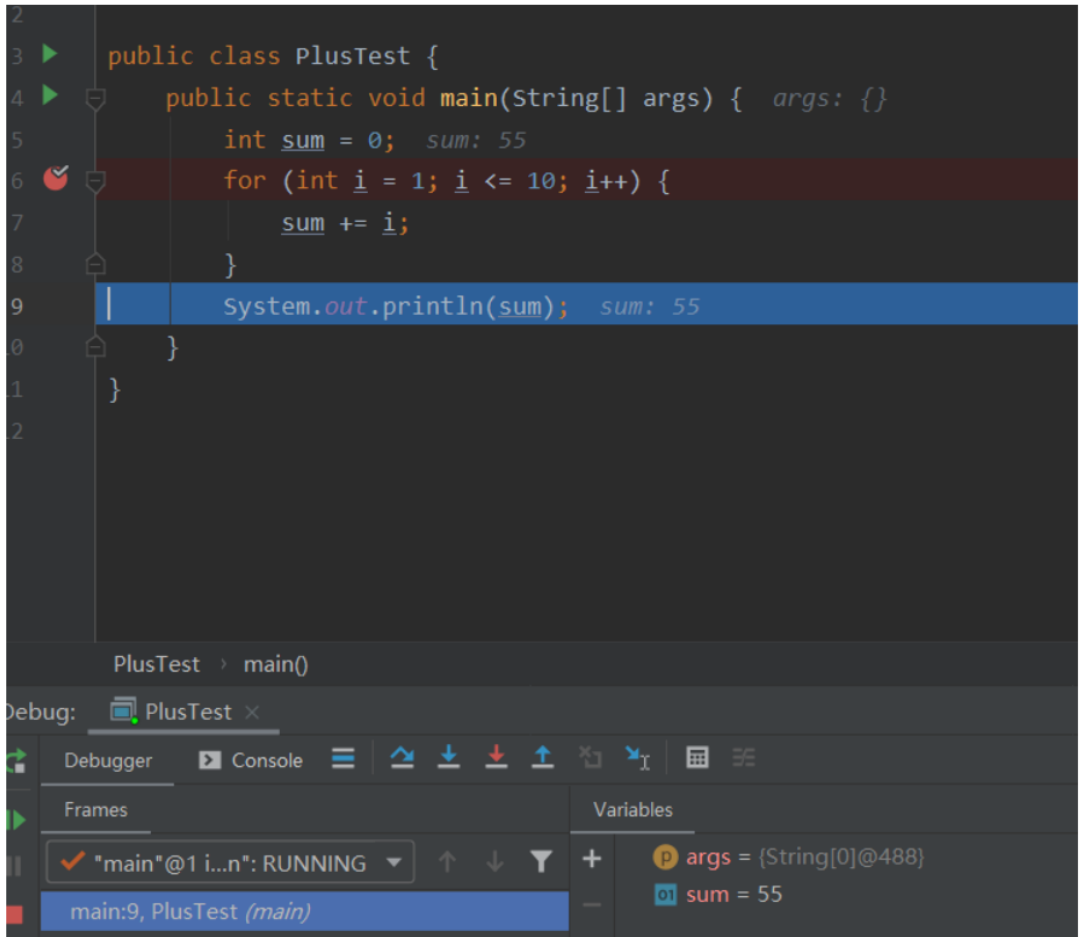




1.2 认识Java语言

- IntelliJ_IDEA的Debug

运行结束，得到结果：（事实上如果真有Bug，能发现的话在过程中就发现了）



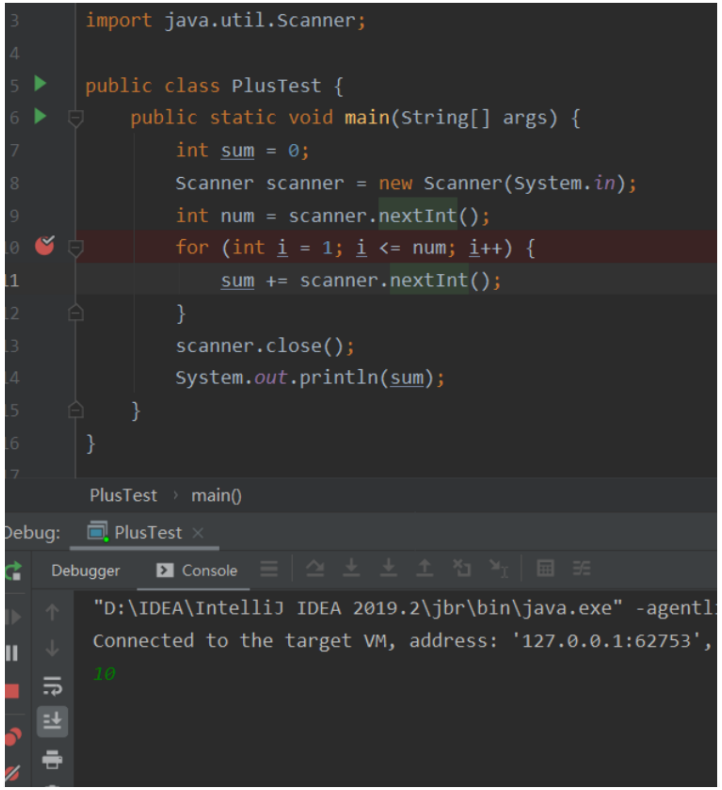
1.2 认识Java语言

- IntelliJ_IDEA的Debug

不断的输入

我们可能需要命令行输入很多数据，有时候我们选择复制后一股脑输入Debug的Console中，但很多时候我们输入几个值并按回车就自动进入了Debug。而Debug运行到需要IO的地方就会阻塞。

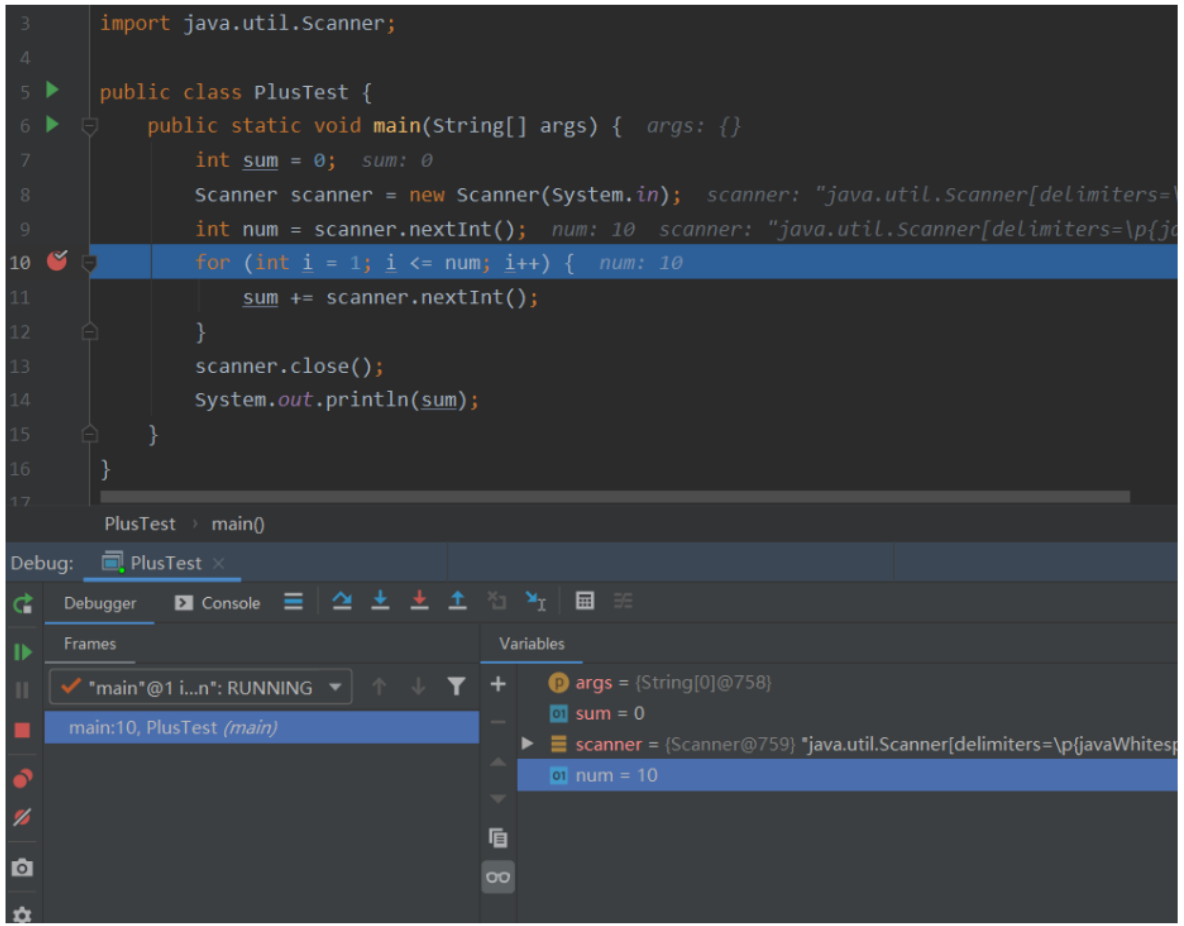
先输入10，然后回车：



1.2 认识Java语言

- IntelliJ_IDEA的Debug

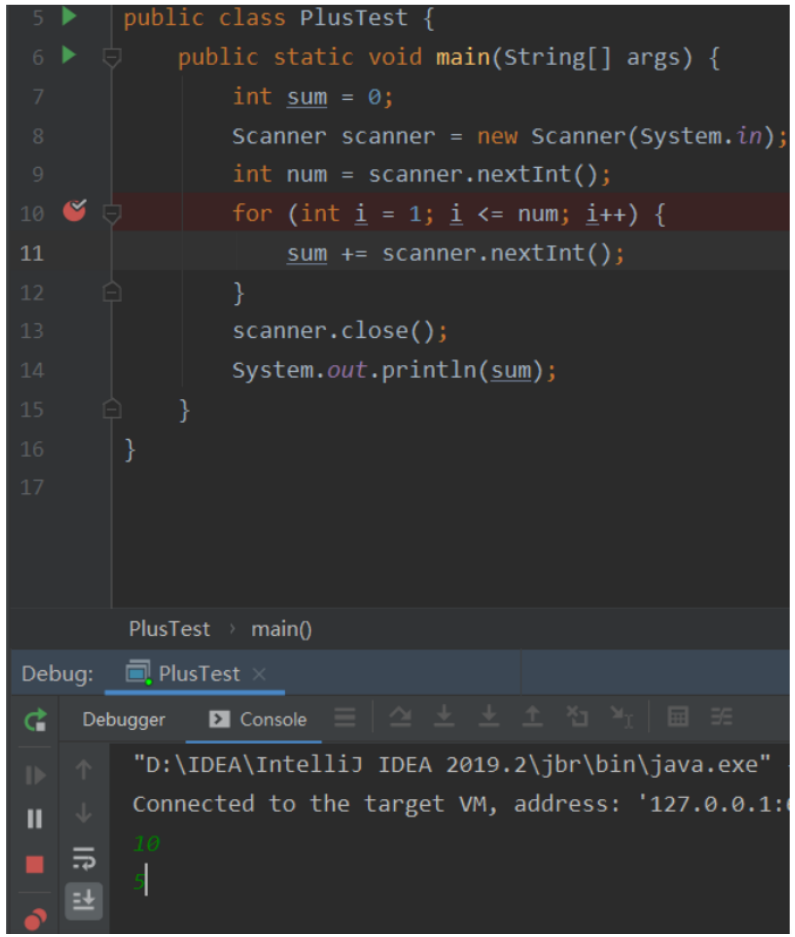
程序跳到上面讲过的Debug界面：



1.2 认识Java语言

- IntelliJ_IDEA的Debug

顺着执行，会被IO卡住，那怎么办呢？点击Console，切到Console界面，输入新的值，回车：

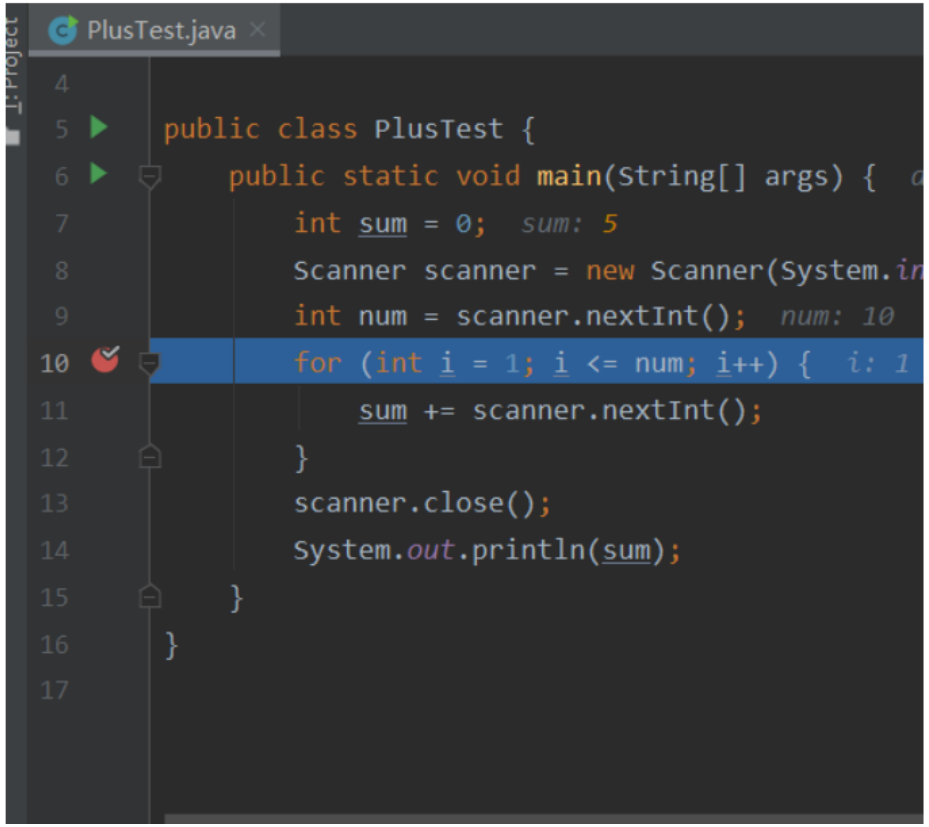


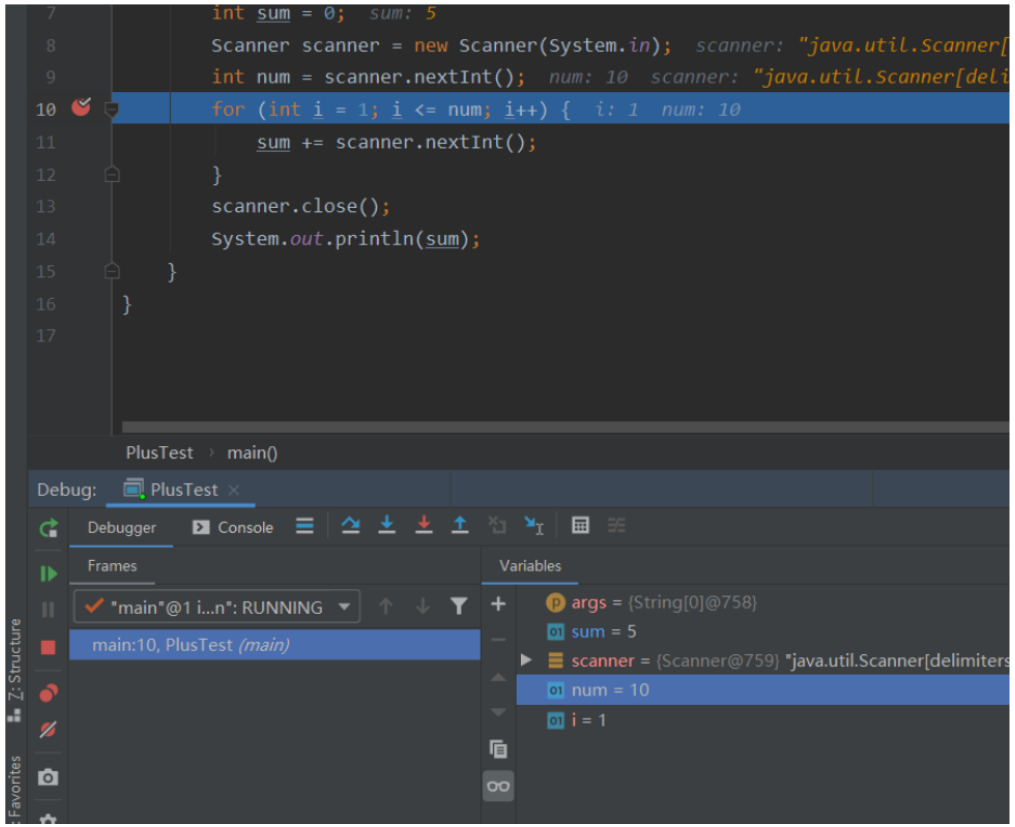
1.2 认识Java语言

- IntelliJ_IDEA的Debug

这次就不是自动切回去了，需要点击下方的Debugger

Console识别在一次IO后就切Debugger，所以比较省事的做法是，第一次就把所有的数据输进去。





1.2 认识Java语言

- IntelliJ_IDEA的Debug

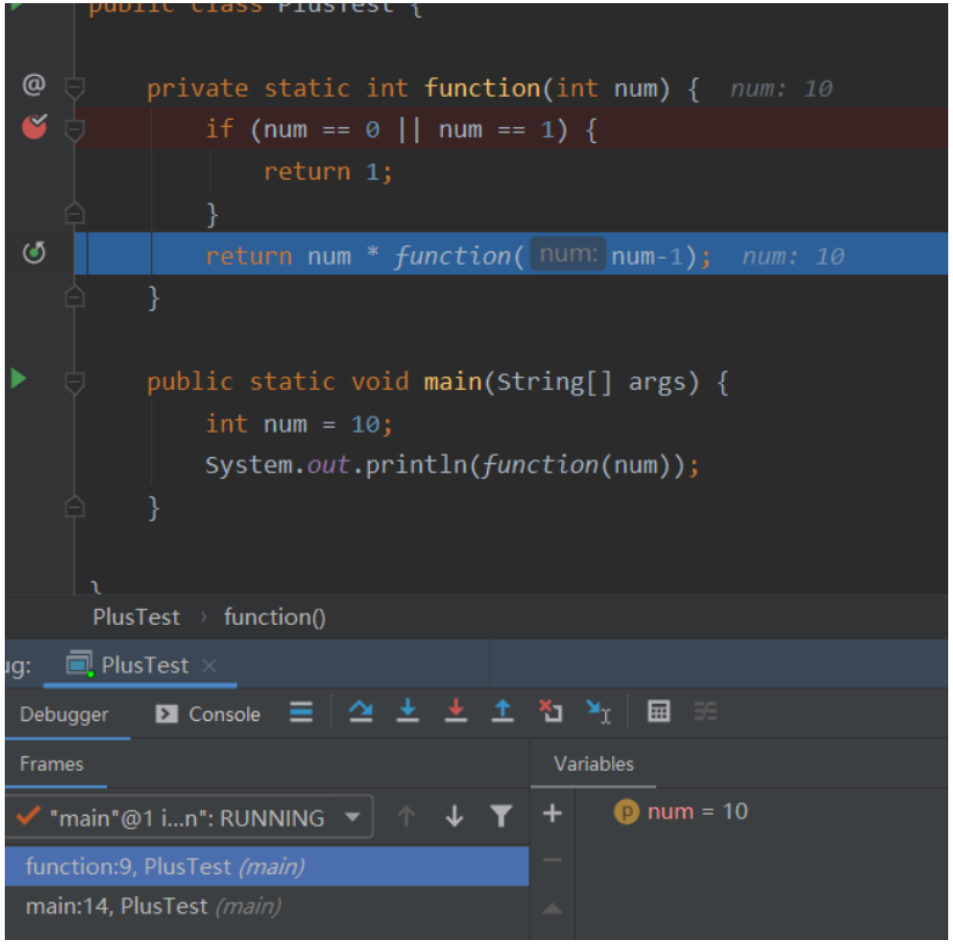
递归函数的Debug

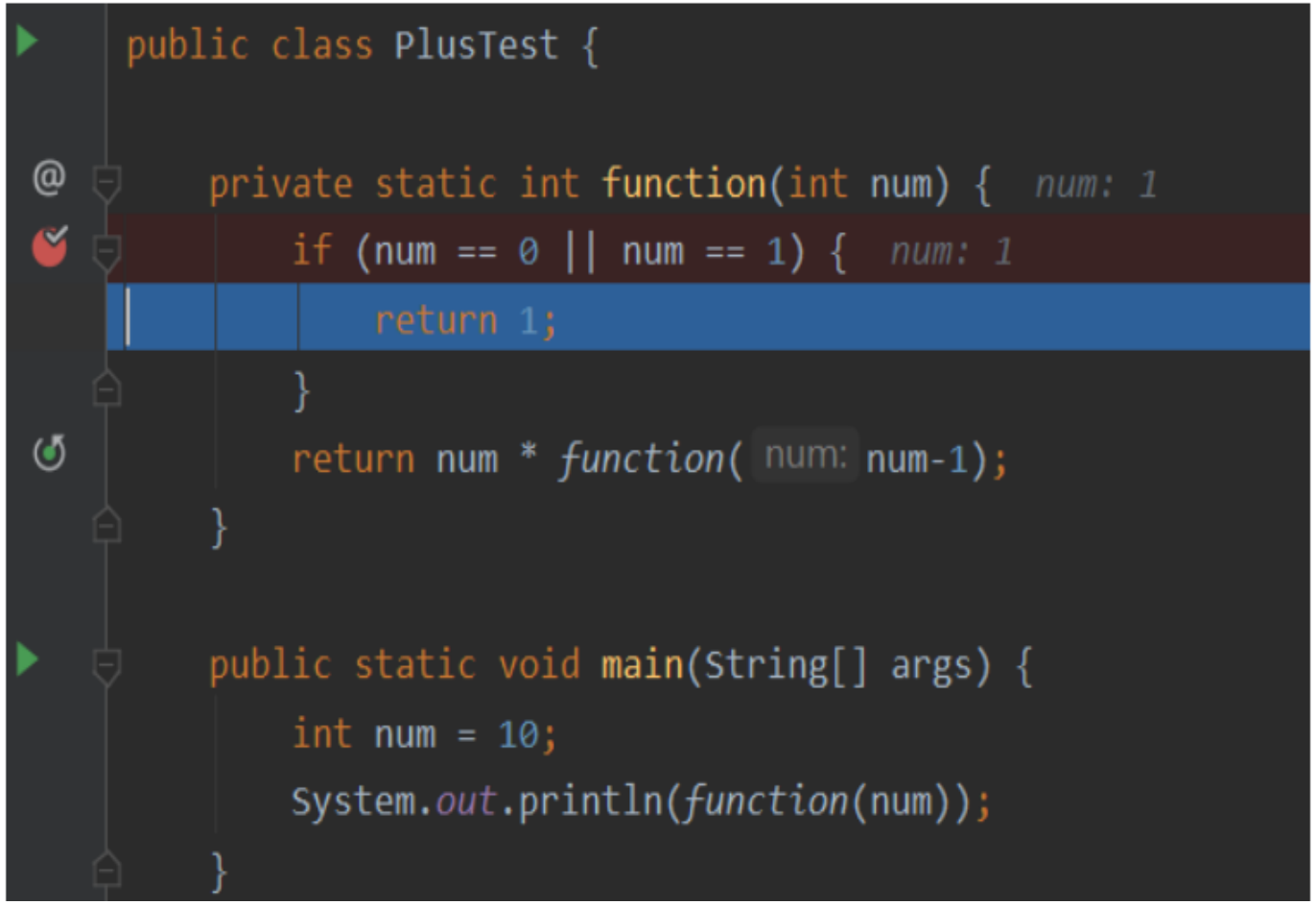
```java
public class PlusTest {

    private static int function(int num) {
        if (num == 0 || num == 1) {
            return 1;
        }
        return num * function(num-1);
    }

    public static void main(String[] args) {
        int num = 10;
        System.out.println(function(num));
    }

}
```

1.2 认识Java语言

- IntelliJ_IDEA的Debug

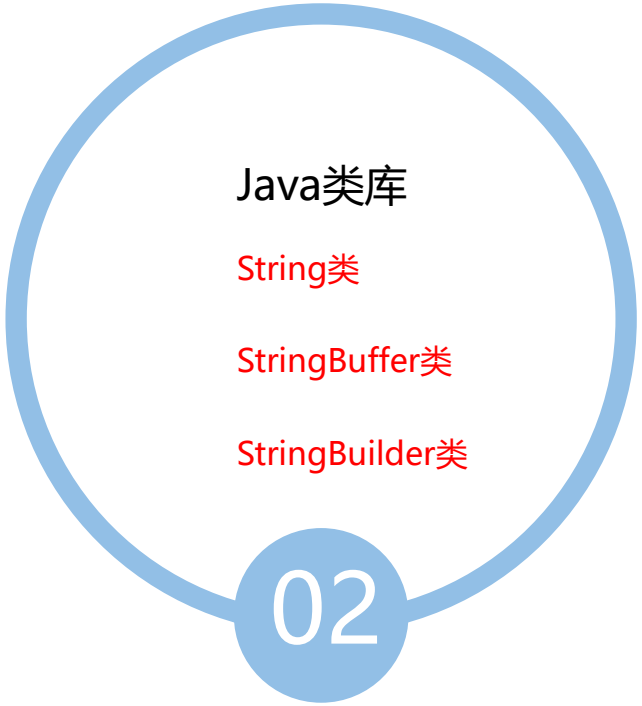
递归函数的Debug





本章重点


Java特性
面向对象
解释型语言
多线程
01


Java类库
String类
StringBuffer类
StringBuilder类
02


调试程序
使用Debug打断点调试程序
03

本章作业

1 配置JDK环境

要求:能够查看jdk当前版本;

输入java和javac命令显示帮助信息;

2 完成一个HolloWord实例

要求:运行Java代码后在控制台显示HolloWord

3 完成一个简单的运算实例

要求:该程序会请求用户输入两个整数，然后计算并打印出这两个整数的和、差、乘积以及它们的商;

信创智能医疗系统研发课程体系

河南中医药大学信息技术学院（智能医疗行业学院）

河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

河南中医药大学医疗健康信息工程技术研究所