简介与安装

调整目标

1. 在讲 hello world 的时候体现 "顶级", "类级", "方法级" 这样的代码区域.
2. 针对所有前期课件, 在编译代码出错的时候把一些编译错误和运行错误的具体错误信息罗列出来. 强调编译错误和运行错误的概念
3. 讲讲编程是要干啥.
4. 把 Java 的背景省略省略, 不必这么详细.
5. 重点讲 javac, java, jvm, jre, jdk, 编译型, 解释型, 编译期, 运行期, 这些概念.
6. 在 "3" 中加一些 jdk 版本差异的一些重要特性.
7. 学了之后能找哪些岗位. 可以投一些公司的 jd, 给大家看看.
8. 环境搭建部分最好讲两遍. 先演示一次, 然后把环境都卸载掉, 再和同学们一起搭建一遍.
9. 后面就统一都上来就 IDEA, 命令行编译运行后面. (和 Linux 指令放到一起, 大概 SE 末尾). 讲这个时候最好拿 windows 的命令行作为过渡. 在这个环节要讲 args
10. 关于环境和学校的环境冲突问题, 可以考虑让同学们先创建个用户(拿一个班试试)
11. 作业中加上对于 类名 和 文件名 一致, 以及类名首字母大写的考点.
12. 环境: 这部分要单独拎出一个文档, 并且需要和测试课程的环境对一下.
    1. JDK 只安装 1.8. 安装路径尽量用默认路径.
    2. IDEA 2020.1 专业版(老郑给个破解方法), 安装路径尽量用默认路径.
    3. 不要配置任何环境变量
    4. Maven 使用 idea 内置的 Maven
    5. IDEA 不连 MySQL
    6. web 项目的配置, 使用 maven, 但是不使用模板, 手动创建目录结构.
    7. Tomcat 插件, 先使用一段时间, 然后讲解手动拷贝 war 包的
    8. 像 postman, fiddler 这些工具嵌入到课程内部.
    9. MySQL使用 8, jdbc 也要一起调整. 安装包使用二进制安装包, 不要解压缩的.
    10. Linux 环境统一使用云服务器. centos7
    11. Linux 的 jdk 就用 openjdk
    12. Tomcat 版本 Tomcat9
    13. Servlet 版本 4
13. 创建项目不是用 module

> 课件和作业暂时待定.

本节目标

1. 搭建好 java 开发环境
2. 写出 hello world

1.Java语言概述

1.1 Java是什么

Java是一种优秀的程序设计语言，它具有令人赏心悦目的语法和易于理解的语义。不仅如此，Java还是一个有一系列计算机软件和规范形成的技术体系，这个技术体系提供了完整的用于软件开发和跨平台部署的支持环境，并广泛应用于嵌入式系统、移动终端、企业服务器、大型机等各种场合。下面我们通过Java官方提供的数据来一起感受一下，Java到底有多“火”。

- 97% 的企业桌面运行 Java
- 美国有 89% 的桌面（或计算机）运行 Java
- 全球有 900 万 Java 开发人员
- 开发人员的头号选择
- 排名第一的部署平台
- 有 30 亿部移动电话运行 Java
- 100% 的蓝光盘播放器附带了 Java
- 有 50 亿张 Java 卡在使用
- 1.25 亿台 TV 设备运行 Java
- 前 5 个原始设备制造商均提供了 Java ME

1.2 为什么选择Java

通过上面的数据，同学们应该对Java目前到底有多流行有一个直观的认识了吧。这里有的同学可能会问了，上面的数据主要是Java在应用市场的流行，那么对于我们开发者，Java是否是我们开发应用程序的第一选择呢？来看下图

Dec 2017	Dec 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.268%	-4.59%
2	2		C	10.158%	+1.43%
3	3		C++	4.717%	-0.62%
4	4		Python	3.777%	-0.46%
5	6	↑	C#	2.822%	-0.35%
6	8	↑	JavaScript	2.474%	-0.39%
7	5	↓	Visual Basic .NET	2.471%	-0.83%
8	17	⇈	R	1.906%	+0.08%
9	7	↓	PHP	1.590%	-1.33%
10	18	⇈	MATLAB	1.569%	-0.25%
11	13	↑	Swift	1.566%	-0.57%
12	11	↓	Objective-C	1.497%	-0.83%
13	9	⇊	Assembly language	1.471%	-1.07%
14	10	⇊	Perl	1.437%	-0.90%
15	12	↓	Ruby	1.424%	-0.72%
16	15	↓	Delphi/Object Pascal	1.395%	-0.55%
17	16	↓	Go	1.387%	-0.55%
18	25	⇈	Scratch	1.374%	+0.19%
19	20	↑	PL/SQL	1.368%	-0.13%
20	14	⇊	Visual Basic	1.347%	-0.62%

Oct 2018	Oct 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.801%	+5.37%
2	2		C	15.376%	+7.00%
3	3		C++	7.593%	+2.59%
4	5	↑	Python	7.156%	+3.35%
5	8	↑	Visual Basic .NET	5.884%	+3.15%
6	4	↓	C#	3.485%	-0.37%
7	7		PHP	2.794%	+0.00%
8	6	↓	JavaScript	2.280%	-0.73%
9	-	⇈	SQL	2.038%	+2.04%
10	16	⇈	Swift	1.500%	-0.17%
11	13	↑	MATLAB	1.317%	-0.56%
12	20	⇈	Go	1.253%	-0.10%
13	9	⇊	Assembly language	1.245%	-1.13%
14	15	↑	R	1.214%	-0.47%
15	17	↑	Objective-C	1.202%	-0.31%
16	12	⇊	Perl	1.168%	-0.80%
17	11	⇊	Delphi/Object Pascal	1.154%	-1.03%
18	10	⇊	Ruby	1.108%	-1.22%
19	19		PL/SQL	0.779%	-0.63%
20	18	↓	Visual Basic	0.652%	-0.77%

题外话：很多同学可能会问，JavaScript和Java有什么关系呢？要说到这俩的关系，举个例子。就相当于雷锋和雷峰塔的关系或者说老婆和老婆饼的关系(没有一点关系...)，最开始JavaScript叫liveScript，当时Java太火了，于是乎liveScript更名为JavaScript借势宣传一波。。*

上图数据来自于TIOBE编程语言社区2017年12月和2018年10月最新的排行榜。TIOBE 编程语言社区排行榜是编程语言流行趋势的一个指标，每月更新，这份排行榜排名基于互联网上有经验的程序员、 课程和第三方厂商的数量。排名使用著名的搜索引擎（诸如 Google、MSN、Yahoo!、Wikipedia、YouTube 以及 Baidu 等）进行计算。同学们可以随便搜搜近几年的编程类语言排行榜，Java绝对都是当之无愧的老大哥。那么，Java究竟有什么特性能获得广大程序员的一致青睐呢？ 这里老师参考Java白皮书，列出如下11个特性：

1.2.1 简单性

Java语法是C++语法的一个“纯净版本”。这里没有头文件、指针运算（甚至指针语法）、结构、联合、操作符重载、虚基类等等。不仅如此，Java开发环境远远超出大多数其他编程语言的开发环境。

1.2.2 面向对象

什么是面向对象？ 这里我们用木匠打一个比方，一个“面向对象”的木匠始终关注的是所制作的椅子，第二位才是所使用的工具；而一个“非面向对象的”木匠首先考虑的是所使用的工具。

在Java的世界里，一切皆对象。

Java的面向对象特性与C++旗鼓相当，与C++的主要不同点在于多重继承。在Java中，取而代之的是更简单的接口概念。而且与C++相比，Java提供了更丰富的运行时自省功能。

1.2.3 分布式（微服务）

Java有丰富的例程库，用于处理像HTTP和FTP之类的TCP/IP协议。Java应用程序能够通过URL打开和访问网络上的对象，其便捷程度就好像访问本地文件一样。

1.2.4 健壮性

Java与C++最大的不同在于Java采用的指针模型可以消除重写内存和损坏数据的可能性（对于曾经花费几个小时来检查由于指针bug而引起内存冲突的人来说，一定很喜欢Java的这一特性）。不仅如此，Java编译器能够检测许多在其他语言中仅在运行时才能够检测出来的问题。

1.2.5 安全性

Java适用于网络/分布式环境。为了达到这个目标，在安全性方面投入了大量的精力。使用Java可以构建防病毒、防篡改的系统

从一开始，Java就设计成能够防范常见的各种攻击：

> - 运行时堆栈溢出。蠕虫和病毒常用的攻击手段。
> - 破坏自己进程空间之外的内存。
> - 未经授权读写文件

1.2.6 体系结构中立

编译器生成一个体系结构中立的目标文件格式，这是一种编译过的代码，只要有Java运行时系统，这些编译后的代码就可以在许多处理器上运行。Java编译器通过生成与特定计算机体系结构无关的字节码指令来实现这一特性。精心设计的字节码不仅可以很容易的在任何机器上解释执行，而且还可以动态地翻译成本地机器代码。

1.2.7 可移植性

与C/C++不同，Java规范中没有“依赖具体实现的地方”。基本数据类型的大小以及有关运算都做了明确的说明。例如，Java中的int永远是32位的整数，而在C/C++中，int可能是16位整数、32位整数，也可能是编译器提供商指定的其他大小。在Java中，数据类型具有固定的大小，这消除了代码移植时令人头疼的主要问题。

1.2.8 解释型

Java解释器可以在任何移植了解释器的机器上执行Java字节码。由于链接是一个增量式且轻量级的过程。所以开发过程也变得更加快捷，更加具有探索性。

1.2.9 高性能

尽管对解释后的字节码性能已经比较满意，但在有些场合下可能需要更加高效的性能。字节码可以（在运行时刻）动态的翻译成对应运行这个应用的特定cpu的机器码。

1.2.10 多线程

Java在当时很超前。它是第一个支持并发程序设计的主流语言。多线程可以带来更好的交互响应和实时行为。并发程序设计绝非易事，但是Java在这方面表现出色，可以很好的管理这个工作。

1.2.11 动态性

Java与C/C++相比更加具有动态性。它能够适应不断发展的环境。库中可以自由的添加新方法和实例变量，而对客户端没有任何影响。在Java中找出运行时类型信息十分简单（反射的特性，后续会学到）

1.3 Java语言发展简史

Java 语言源于 1991 年 Sun 公司 James Gosling 领导的的 Oak 项目，1995 年 Sun 公司正式起名为 Java，并提出“Write once, Run anywhere" 的口号。

1996 年 1 月 Java 1.0 发布，提供了一个解释执行的 Java 虚拟机，其时恰逢互联网开始兴起，Java 的 Applet 能在 Mozilla 浏览器中运行，被看作是未来的互联网语言。

1997 年 2 月 Java 1.1 发布，Java 语言的基本形态基本确定了，比如反射 (reflection), JavaBean, 接口和类的关系等等，一直到今天都保持一致。然而，Java 最初的一些目标，如在浏览器中执行 Applet，以及跨平台的图形界面 Awt 很快遭遇到负面的评价。

1998 年 12 月，Java 第一个里程碑式的版本，即 Java 1.2 发布了。这个版本使用了 JIT（Just in time）编译器技术，使得语言的可迁移性和执行效率达到最优的平衡，同时 Collections 集合类设计优良，在企业应用开发中迅速得到了广泛使用。Sun 公司把 Java 技术体系分成三个方向，分别是 J2SE（面向桌面和通用应用开发），J2EE（面向企业级应用开发），J2ME（面向移动终端开发）。这个分类影响非常久远，体现出主流语言设计者的思想：针对于不同的应用领域，在形态，API 集合等进行划分。

2000 年 5 月，Java 1.3 发布，这个版本中 Corba 作为语言级别的分布式对象技术，成为 J2EE 的一个技术前提。J2EE 受到 Corba 的设计的影响较大，早期 EJB 的 Home，接口和实现就是 Corba 在 C 语言的实现，被移植到 Java 语言之中。J2EE 中的 Servlet 规范获得了极大的成功，伴随着互联网的兴起，和浏览器直接通过 HTTP 协议交互的 Servlet，和众多的 MVC 框架，成为 Web1.0 的网红。

2002 年 2 月，Java 1.4 发布，Java 语言真正走向成熟，提供了非常完备的语言特性，如 NIO，正则表达式，XML 处理器等。同年微软的.NET 框架发布，两者开始了为期十几年的暗自竞争。从语言特性上来说，.NET 后发先至，一直处于优势。但 Java 依赖良好的开发者生态，绝大多数大型软件公司的使用者众多和不断贡献，以及对 Linux 操作系统良好的支持，渐渐的在服务器端获得优势地位。 2004 年 9 月，Java 5 发布，Sun 不再采用 J2SE, J2EE 这种命名方式，而使用 Java SE 5, Java EE 5 这样的名称。我认为 Java 5 是第二个里程碑式的版本。Java 语言语法发生很大的变化，如注解 (Annotation)，装箱 (Autoboxing)，泛型 (Generic)，枚举 (Enum)，foreach 等被加入，提供了 java.util.concurrent 并发包。Java 5 对于 Java 语言的推动是巨大的，特别是注解的加入，使得语言定义灵活了很多，程序员可以写出更加符合领域定义的描述性程序。

2006 年 5 月， JavaEE 5 发布，其中最主要是 EJB3.0 的版本升级。在此之前，EJB2.X 版本被广泛质疑，SpringFramework 创建者 Rod Johnson 在经典书籍“J2EE Development without EJB“中，对 EJB2 代表的分布式对象的设计方法予以批驳。EJB3 则重新经过改造，使用注解方式，经过应用服务器对 POJO 对象进行增强来实现分布式服务能力。在某种程度，可以说 EJB3 挽救了 JavaEE 的过早消亡。 2006 年 12 月，Java 6 发布，这个语言语法改进不多，但在虚拟机内部做了大量的改进，成为一个相当成熟稳定的

版本，时至今日国内的很多公司依然以 Java6 作为主要 Java 开发版本来使用。同年 Sun 公司做出一个伟大的决定，将 Java 开源。OpenJDK 从 Sun JDK 1.7 版本分支出去，成为今天 OpenJDK 的基础。OpenJDK6 则由 OpenJDK7 裁剪而来，目前由红帽负责维护，来满足 Redhat Enterprise Linux 6.X 用户的需要。

2009 年 12 月，JavaEE 6 发布，这个版本应该说是 JavaEE 到目前为止改进最大影响最深远的一个版本。因为 JavaEE5 只有 EJB3 适应了 Java 注解语法的加入，而 EE6 全面接纳了注解。CDI 和 BeanValidation 规范的加入，在 POJO 之上可以定义完备的语义，由容器来决定如何去做。Servlet 也升级到 3.0 版本，并在接口上加入异步支持，使得系统整体效率可以大幅提高。EE 划分为 Full Profile 和 Web Profile，用户可以根据自己的需要选择不同的功能集。

在此之前，Oracle 已经以 74 亿美金的价格收购了 Sun 公司，获得了 Java 商标和 Java 主导权。也收购了 BEA 公司，获得市场份额最大的应用服务器 Webogic。JavaEE 6 虽然是收购之后发布的版本，但主要的设计工作仍然由原 Sun 公司的 Java 专家完成。 2011 年 7 月，Oracle 发布 Java 7, 其中主要的特性是 NIO2 和 Fork/Join 并发包，尽管语言上没有大的增强，但我个人认为，自从 Oracle JDK（包括 OpenJDK7），Java 虚拟机的稳定性真正做到的工业级，成为一个计算平台而服务于全世界。

2013 年 6 月，Oracle 发布 JavaEE 7，这个版本加入了 Websocket，Batch 的支持，并且引入 Concurrency 来对服务器多线程进行管控。然而所有的子规范，算上可选项 (Optional) 总共有 40 多项，开发者光是阅读规范文本就很吃力了，更不要说能够全局精通掌握。JavaEE 规范的本质是企业级应用设计的经验凝结，每一个 API 都经过众多丰富经验的专家反复商议并确定。各个版本之间可以做到向后兼容，也就是说，即使是 10 年前写的 Servlet 程序，当前的开发者也可以流畅的阅读源码，经过部分代码调整和配置修改，可以部署在当今的应用服务器上。反过来，今后用 Servlet4 写的程序，浏览器和服务器通信使用全新的 HTTP/2 协议，但程序员在理解上不会有障碍，就是因为 Servlet 规范的 API 非常稳定，基本没有大的变化修改。

2014 年 3 月，Oracle 发布 Java 8，这个版本是我认为的第三个有里程碑意义的 Java 版本。其中最引人注目的便是 Lambda 表达式了，从此 Java 语言原生提供了函数式编程能力。语言方面大的特性增加还有：Streams，Date/Time API,集合的并行计算支持等，Java8 更加适应海量云计算的需要。

2018 年 3 月，Java 10 正式发布，并带来 109 项新特性。

美国当地时间2018 年9月25日，Oracle 官方宣布Java 11 (18.9 LTS) 正式发布，可在生产环境中使用！
这是自 Java 8 后的首个长期支持版本

1.4 Java是最好的语言么

不是，因为在每个领域都有更合适的编程语言。

C 语言无疑是现代计算机软件编程语言的王者，几乎所有的操作系统都是 C 语言写成的。C++ 是面向对象的 C 语言，一直在不断的改进。

JavaScript 是能运行在浏览器中的语言，丰富的前端界面离不开 Javascript 的功劳。近年来的 Node.js 又在后端占有一席之地。

Python 用于系统管理，并通过高性能预编译的库，提供 API 来进行科学计算，文本处理等，是 Linux 必选的解释性语言。

Ruby 强于 DSL（领域特定语言），程序员可以定义丰富的语义来充分表达自己的思想。Erlang 就是为分布式计算设计的，能保证在大规模并发访问的情况下，保持强壮和稳定性。

Go 语言内置了并发能力，可以编译成本地代码。当前新的网络相关项目，很大比例是由 Go 语言编写的，如 Docker、Kubernetes 等。

编写网页用 PHP，函数式编程有 Lisp/Scala，编写 iOS 程序有 Swift/Objective-C。

一句话概括，能留在排行榜之上的语言，都是好的语言，在其所在的领域能做到最好。

那么，Java 语言到底有什么优势可以占据排行榜第一的位置呢?

其一，语法比较简单，学过计算机编程的开发者都能快速上手。

其二，在若干了领域都有很强的竞争力，比如服务端编程，高性能网络程序，企业软件事务处理，分布式计算，Android 移动终端应用开发等等。

最重要的一点是符合工程学的需求，我们知道现代软件都是协同开发，那么代码可维护性，编译时检查，较为高效的运行效率，跨平台能力，丰富的 IDE，测试，项目管理工具配合。都使得 Java 成为企业软件公司的首选，也得到很多互联网公司的青睐。

没有短板，容易从市场上找到 Java 软件工程师，软件公司选择 Java 作为主要开发语言，再在特定的领域使用其他语言协作编程，这样的组合选择，肯定是不会有大的问题。 所以综合而言，Java 语言全能方面是最好的。

我们来一起看看Java社区的概况：

Java 是一门开放的语言，其开源社区也是参与者众多。最有名的应当数 Apache 社区，目前已经拥有近 200 个顶级项目，其中绝大多数是 Java 语言项目。在 Java 生态圈中，具有重要地位的如 **Spring**、Ant、Commons、**Tomcat**、Xerces、**Maven**、 Struts、Lucene、ActiveMQ、CXF、Camel、**Hadoop** 等等。很多技术时代，一大批 Java 项目加入，如 Web 时代的 Velocity、Wicket；JavaEE 相关的 Tomee、OpenJPA、OpenWebBeans、Myfaces；WebService 时代的 jUDDI、Axis、ServiceMix；Osgi 时期的 Flex、Karaf；大数据时代的 HBase、Hive、**ZooKeeper**、Cassandra；云时代的 Mesos、CloudStack 等等。

涉及到软件开发的方方面面，可以说当今几乎所有的中型以上 Java 应用中，都会有 Apache 开源项目的身影。国内最早参与 Apache 社区的以国外软件公司国内研发团队成员为主，如红帽、IONA、Intel、IBM 研发中心等。如今国内互联网公司和软件公司也不断的参与，特别是开始主导一些 Apache 项目，如 Kylin 等。

JBoss 开源社区，包含了 50 多个 Java 开源项目，其中有 Hibernate、Drools、jBPM 等业界知名开源项目，也有 Undertow、Byteman、Narayana 等名气不算大，但绝对是相应领域业界的顶级优秀项目。当前 JBoss 开源社区主要以企业应用中间件软件为主，RedHat 是主要的技术贡献力量。

Eclipse 开源社区，之前主要是包含 Eclipse IDE 的项目，后来也逐步进行多方面的扩展，比如 OSGi，服务器等，目前一些知名 Java 项目，如 Jetty、Vertx 等都是 Eclipse 开源组织成员。此外 IOT 目前是 Eclispe 的一个重点方向，在这里可以找到完整的 IOT Java 开发方案。

Spring 开源社区，以 SpringFramework 为核心，包括 SpringBoot、SpringCloud、SpringSecurity、SpringXD 等开源项目，在国内有广泛的应用场景。

所以说，Java不仅仅是一门编程语言，它是一个综合的技术体系，是面向对象思想的规范。

1.5 Java开发环境安装

- 可能是Windows下最简单的Java环境安装指南
- Linux下JDK的安装（多种方式）
- Mac下JDK的安装
- Java 软件 https://pan.baidu.com/s/1X7zPb-YT11xR_UDqjN-oJw 提取码:r471

初识Java的main方法

2.1 main方法示例

```
public class Test{

    public static void main(String[] args){

        System.out.println("Hello,Java");
        System.out.println("My name is:"+args[0]);
    }
}
```

如上展示的就是最简单的一个Java程序，可能同学们看到后一头雾水，可以说，Java的main方法应该是当前主流编程语言中最“长”的。

通过上述代码，我们可以看到一个完整的Java程序的结构，Java程序的结构由如下三个部分组成：

> 1.源文件（扩展名为*.java）：源文件带有类的定义。类用来表示程序的一个组件，小程序或许只会有一个类。类的内容必须包含在花括号里面。

> 2.类：类中带有一个或多个方法。方法必须在类的内部声明。

> 3.方法：在方法的花括号中编写方法应该执行的语句。

> 总结一下：类存在于源文件里面；方法存在于类中；语句存在于方法中。

好了，代码编写完了，如何让它“运行”起来呢？

2.1 运行Java程序

Java是一门半编译型、半解释型语言。先通过javac编译程序把源文件进行编译，编译后生成的.class文件是由字节码组成的平台无关、面向JVM的文件。最后启动java虚拟机来运行.class文件，此时JVM会将字节码转换成平台能够理解的形式来运行。

> - JRE(Java Runtime Environment):Java运行时环境，包含了JVM，Java基础类库。是使用Java语言编写程序运行的所需环境。
> - JDK(Java Development Kit):Java开发工具包，提供给Java程序员使用，包含了JRE，同时还包含了编译器javac与自带的调试工具Jconsole、jstack等。

Java程序运行需要经过编译，运行两个阶段。

- 编译：javac 命令
- 运行：java 命令

3.初识Java简单语句和语法

通过以上我们对main方法的剖析和之前同学们C语言的基础，下面我们来看一段代码：

```
import java.io.*;
import java.util.Arrays;
import java.util.List;
public class FirstDemo {
    public static void main(String[] args) {
        int age = 2;
        String name = "66";
        Dog dog = new Dog(name,age);
        int x = age - 1;
        if (x > 1){
```

```
            dog.play();
        }else {
            dog.bark();
        }
        int[] numList = new int[]{1,3,5,7,9};
        // 这是一段注释
        // JDK8新增的lambda表达式
        new Thread(()->System.out.println("hello world")).start();
        try {
            // 如果文件没找到则会抛出异常
            InputStream in = new FileInputStream(new File("//xx.txt"));
        } catch (Exception e) {
            e.printStackTrace();
        }
        // JDK1.8新增的Stream数据流
        List<Integer> list = Arrays.asList(1,2,2,4,5,6,7,10);
        list.stream()
                .filter(e -> e % 2 ==0)
                .distinct()
                .forEach(System.out::println);
    }
}
class Dog {
    private String name;
    private Integer age;
    public Dog(String name,Integer age) {
        this.name = name;
        this.age = age;
    }
    public void play() {
        System.out.println(this.name + "want to play");
    }
    public void bark() {
        System.out.println(this.name + "汪汪汪");
    }
}
```