利用 jlink/jaotc 工具，可以把一个 Java 程序编译成可执行文件，在 Java9 推出时，可能只有 java.base 模块支持 AOT。

安全方面的加强。引入新的摘要算法 SHA-3，内置 ALPN 使得更好的支持 HTTP/2 协议，提供 DTLS（数据包传输层安全性协议），可以保证 UDP 数据传输的安全，PKCS12 格式替代原有的 JKS 成为 keystore 的默认格式。

此外，统一 JVM 日志 (Unified JVM Logging)，多版本共存 jar(Multi-release jar files)，接口内部的私有方法 (Interface provate method) 等也是非常重要的新特性。

7.与其他语言的对比，Java 的优势

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，编写 iOS 程序有 Swift/ObjectiveC。

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。

8.Java 未来方向的展望

如今的 Java，已经在功能上相当丰富了，Java 8 加入 Lambda 特性，Java 9 加入模块化特性之后，重要的语言特性似乎已经都纳入进来。如果要说值得考虑的一些功能，我觉得有以下几点：

1. 模块化改造完毕之后，可能会出现更多专业的 JDK 发行软件商，提供在功能方面，比如针对于分布式计算，机器学习，图形计算等，纳入相关的功能库作为文件。这样专业行业客户可以选择经过充分优化后的 JDK 版本。
2. Java 语义上对“模式匹配”有更强的支持，如今的 switch 语句能力还是比较欠缺，可以向 Erlang， Scala 等语言借鉴。
3. 多线程并发处理，Java 做的已经很好了。不过我个人觉得可以在多进程多线程配合，以及语言级别数据管道表示上，可以进行改造和优化。
4. JDK9 会有 HTTP/2 client 端的能力，但毫无疑问会有更多更好的三方库出现，JDK 可以和这些三方库通力合作，提供一个更好 API 界面和 SPI 参考实现。
5. 目前 Java 在云计算方面遇到的最大问题还是占用内存过大。我个人认为从两个方面来看：
   * 如果该应用的确是长时间运行的服务，可以考虑结构清晰的单体结构，算下来总的内存消耗并不会比多个微服务进程占用的更多。
   * 微服务应用，未来可以采用编译成本地代码的方式，并使用优化过的三方库，甚至本地 so 文件，减少单个进程的过多内存占用。
6. 安全框架更加清晰，SPI 可以允许三方库提供更强大更高效的安全功能。
7. JavaEE 方向则有更多的改进的地方：
   * EJB 重构目前的 Corba 分布通信基础，参考 gRPC 进行远程系统调用。
   * 分解 EJB 规范，把 JVM 进程相关的特性，如注入 / 加强 / 事务 / 安全都统一到 CDI 规范中；对 EJB 进行裁剪，保留远程访问特性和作为独立执行主体分布式对象能力。
   * 加强 JMS 和 MDB，媲美 Akka 目前的能力。
   * JaxRS 适度优化，不必要依赖 Servlet，或者适度调整，来提供更大的能力。
   * JPA 借鉴 JDO，以及融入一部分特性，做到对 NoSQL 更良好的支持。

9.一些个人的心得和经验分享

软件业有个 Hype Cycle 模型，有很多技术受到市场的追捧而成为明星，也有些身不逢时而备受冷漠。

1. EJB 是一个广泛被误解的技术，在企业应用分布式计算方面，EJB 给出了非常完备的技术体系。只是目前所有的应用服务器都实现的不够好。对于目前打算转型微服务设计的架构师，EJB 也是一个非常值得学习借鉴的技术。
2. Java 的慢是相对的，有些是当前实现的不够好。比如原来有人对 Java 的网络 IO 性能提出质疑，然而稳定的 Netty 框架出现后，就没有人再怀疑 Java 处理网络 IO 的能力了，甚至在 JDK8 中自身的 NIO 也相当出色。要知道 Java 为了实现跨平台能力，采用的是各个操作系统的一个公共能力子集，而且其设计哲学就是给出 API 框架，实现是可以自行实现和加载服务的。
3. Java 在处理界面方面，Swing 和 Swt 表现可圈可点（Idea 和 Eclipse 分别采用的图形基础库），JavaFX 已经运用到很多的行业软件上。在浏览器界面表现上，SpringMVC 在模板渲染页面方面使用者最多；GWT 似乎使用者不多，但基于 GWT 的 Vaddin 在国外企业中用户众多，而且很多服务器管理软件也用 GWT 写成；JSF 也在企业软件中得到广泛使用，状态信息直接在后端进行管理，配合 js 前端框架，可以充分发挥各种技术的优势。
4. CDI 规范和 SpringFramework 在服务器程序中作用类似，Spring 是一套设计优良，完备的框架，CDI 具有更强的可扩展性。通过对注解的语义定义，一家公司可以维护一套自己的组件描述语言，来做到产品和项目之间的软件快速复用。CDI 是定义软件组件内部模型的最佳方式，只可惜了解的软件工程师实在太少。
5. 微服务架构在互联网应用，快速开发运维管理方面，配合容器技术使用，有很强的优势。但并不是所有的应用场景都适合微服务：强事务应用系统，采用单体结构的软件体系设计，更容易从整体方面维护，也能获得更优的性能。Java 语言无论在微服务还是单体结构，都有成熟稳定的软件架构供选择使用。