

# 零基础学 Java



# 1

## 什么是学习一门语言

- 语法和基础
- 类库
- 工具
- 底层和进阶
- 超越代码：能力

## 语法和基础：

- 数据类型
- 表达式，语句，代码块
- 面向对象：封装，继承和多态
- 线程和异常处理
- 标准类库

## 类库

- jar包：类库是以 jar 包的形式发布的。jar 包是 Java 中组织多个 class 文件的方式。其本质是一个 zip 压缩包。后缀名是 jar。
- 类库是以 jar 包发布的、可以完成某个功能的一个或者多个 jar 包
- 本章后面会有一个小节专门介绍一些常用的类库

## 工具

- maven：构建和打包。丰富的插件可以组成复杂的构建过程
- git：源代码版本控制
- IDE
- Jenkins：自动化持续集成

## 底层 - 内存相关

- 内存堆（heap）和线程栈
- JMM
- GC（garbage collection）：  
GC log 分析和 GC 调优

## 底层 - JVM 和 Java 规范相关

- JVM + Java 字节码
- Class 文件格式
- Class Loader

## 进阶

- 多线程和线程安全
- 程序可读性
- 优化

## 能力（不仅限于某种编程语言）

- 本能 + 直觉：对问题的难度、复杂性和需要的时间有一个直觉的反应。能够快速且准确的判断问题的难点和可能出问题的地方
- 分解 + 架构：拿到一个问题的时候，可以将问题分解为某种编程语言 / 技术可以解决的子模块 / 功能
- 解决问题：任何工作中的问题。从需求分析到架构设计，从接口设计到代码实现，从文档到测试，从脑洞大开到靠谱创新，从内存使用到 CPU 占用，从 bug 可以重现到 bug 难以重现，从集成到联调，从数据到环境，从数据库到消息队列，从在线到离线，从开发到部署，从测试环境的问题到生产环境的问题，从单台机器的问题到集群分布式的问题，从上线到维护，从技术支持到 hotfix，从性能到监控，从技术栈更新到系统完全重构。只有想不到，没有做不到。“报菜名”一样说这么多是为了让大家有一个感性的认识，程序员不只是写代码



# 2

## Java 平台简介

- Java 是什么：语法之外
- Java 是什么：字节码和 class 文件
- Java 是什么：Java 虚拟机（JVM）
- Java 是什么：规范

# Java 是什么： 语法之外

- 我们之前都在学 Java 的语法，写 Java 的程序，但这并不是 Java 的全部
- 符合 Java 语法的源文件通过 `javac` 命令被编译成 `class` 文件，又可以通过不同操作系统平台下的 `java` 命令运行相同的 `class` 文件，得到相同的结果。在后面支撑着完成着一整套操作的系统才是完整的 Java
- `class` 文件也是 Java 和别的语言不一样的地方。所有的 Java 程序的进程名字都是 Java。但是如果是一个 C 语言编写的程序，进程名字可以是程序本身决定的。而且 C 语言编写的程序，可以直接编译链接生成 `exe` 文件（Windows），可以直接运行。但是 Java 程序编译出来是 `class` 文件，运行还要靠 `java` 命令，这一切的背后到底是什么？

# Java 是什么： 字节码和 class 文件

- bytecode: 字节码。类似于 CPU 的指令集，是 Java 程序跨平台的基础。字节码也是 Java 规范中的内容。
- class 文件: Java 源文件可以生成 class 文件，class 文件就是字节码组成的。只要有合适的编译器。别的语法也可以生成 class 文件，在 JVM 上运行
- 执行字节码的是 Java 虚拟机（JVM）。直观的感受就是我们一直用的 java 命令。JVM 是跨操作系统的，和平台无关的。所以这也就是为什么大家说 Java 可以一次编写，到处运行（write once, run anywhere）
- 基于 JVM（或者说字节码）的更多语言: Scala, Groovy, Kotlin

# Java 是什么：Java 虚拟机（JVM）

- JVM Specification: Java 虚拟机规范。核心功能是执行字节码，即 Class 文件的内容。我们一直启动的 java 命令，就是符合 java 虚拟机规范的
- 更多虚拟机：既然 JVM 有一个规范，那么只要满足这个规范，就可以是 Java 虚拟机。Azul 的 Zulu 就是一个 Oracle 之外的 提供 JDK / Java 虚拟机的厂商，可以免费使用，商业版也比较便宜，支持所有主流操作系统。同样出自 Azul 的 Zing 虚拟机是业界公认的性能最好的 JVM 虚拟机。最近阿里也开放了自己的 JDK 版本 Dragonwell
- 在规范之外，每个厂商都可以在 JDK 的发行版里提供不同的工具

# Java 是什么： 规范

- Java 最核心的是规范，包括 JVM 的规范（其中就包含我们之前提过的 JMM 规范），包括字节码规范等。OpenJDK 实现了这些规范，并在 GPL V2 协议下开源
- Java 是一个开放的平台。可以通过 JSR（Java Specification Requests）的形式向 JCP（Java Community Process）提出对 Java 规范的修改。修改包括 Java 的方方面面，可以小到一个语法糖，大到一个 JVM 的改动。比如我们之前提到的自动拆箱装箱，concurrent 包，annotation（新的语法和 class 文件格式），lambda（需要增加新的字节码指令）等都是通过 JSR 的方式推进，并增加到新版本的 Java 中的
- 正因为 Java 是一个开放的平台，所以才会生机勃勃

# 3

## Maven 概念简介

- classpath
- 上古时代的做法
- jar 包的仓库
- maven 客户端

- classpath: 顾名思义, classpath 就是类路径。它是 java 命令在执行 Java 程序的时候, 寻找 class 的路径。它可以是文件夹或者 jar 包的路径的组合。一般执行 java 命令的时候都会配以 classpath 参数, 参数是这个程序用到的 所有 jar 包路径组合
- 我们之前没有指定是因为 java 命令默认会把当前路径加入到 classpath。而我们之前运行的 HelloWorld 程序没有使用别的 jar 包, 所以不指定也没有问题。

- 看例程：使用 guava 包，将包加入 IDE 的项目中。运行时使用 java 命令的 classpath 参数传递 jar 包路径。注意不同操作系统下的分隔符
- 问题：发布 jar 包不方便，寻找 jar 包不方便，使用 jar 包不方便，编译打包自己的程序不方便，升级 jar 包不方便，jar 包之间的依赖管理更是无法想象的困难和繁琐
- maven：解决以上所有问题的利器，而且可以做的更多



- maven 有两部分，首先是服务器端，叫做 maven repo，或者 nexus server。它将所有的 jar 包放在一个仓库里
- 所有 jar 包都发布到这个仓库。需要用到某个 jar 包，就去这个仓库下载
- 仓库里每个 jar 包，都有唯一的 id。这个 id 是由三部分组成的：group id, artifact id 和 version
- 为了避免每次都从服务器下载 artifact（jar包），maven 会把下载好的 artifact 放在本地的文件夹，这个就叫做 local repo

- 如果一个项目 ChatRoom 依赖某个 jar 包，比如 guava，那么就把 guava 的 id 加入到自己的依赖里，maven 客户端就可以通过 id 找到并使用 guava 了
- 同时，maven 的依赖是传递的。如果使用 maven 发布这个 jar 包到 maven repo，maven 还会记住 ChatRoom 的 jar 包依赖于 guava。如果有别的项目依赖 ChatRoom，那么它将自动依赖 guava，无需再次声明。

# 4

## Maven 的安装和配置

- Maven 客户端安装
- Maven 客户端配置（可选）

- apache maven 主页
- apache maven 下载: <https://maven.apache.org/download.html>
- apache maven 安装: <https://maven.apache.org/install.html>
- maven 的安装
- 测试 maven 安装: 在命令行执行 `mvn -v`

# Maven 客户端配置（可选）

- maven repo 的镜像
- 阿里云的 maven repo 镜像: <https://yq.aliyun.com/articles/78124>
- 使用阿里云的镜像: 将项目下的 settings.xml 文件拷贝到 home 目录下的 .m2 文件夹下。（如果没有这个文件夹就尝试使用命令行创建）
- IntelliJ 的 maven settings 配置

# 5

创建一个简单的 Maven 项目

# 创建一个简单的 Maven 项目

- 导入现有的 maven 项目
- 创建新的 maven 项目，增加依赖和代码
- 搜索 artifact: <https://maven.aliyun.com/mvn/search>
- 认识 pom.xml 文件。使用 maven 这个文件是我们最常打交道的文件了

# 6

一个从 pptx 文件中抽取文字的小工具



# 一个从 pptx 文件中抽取文字的小工具

- 通过搜索引擎，找到需要的 jar 包，加到 pom.xml 文件的 dependencies 里

- 根据文档学习如何使用：

<http://poi.apache.org/components/slideshow/index.html>

<http://poi.apache.org/components/slideshow/xslf-cookbook.html#GetShapes>

- 解决其余问题

# 7

## Maven 常用命令和插件

## ● 命令

- maven 构建中的几个主要的 phase: compile test package install
- mvn clean install 或者 mvn clean install -U
- mvn dependency:tree

## ● 插件

- 插件是什么: maven 其实是一套框架, 所有的具体任务都是插件完成的。除了核心的编译打包插件, 还有非常多的别的目的的插件。
- 打出 fatjar 的插件

# 8

IntelliJ 更多功能介绍

## ● 快捷键：

- Edit : Find: Find In Path, Replace In Path, Find Usage
- View: Recent Files
- Navigate: Back, Forward, Last Edit Location, Next Edit Location, Implementations, Class..., File Structure
- Code: Override Methods..., Generate..., Reformat Code
- Refactor: Rename, Extract Method

## ● 插件

- Maven Help
- Markdown support
- Grep Console

# 9

## 值得学习的类库简介

- 工具类库
- 框架类库

- 工具型类库即 util 型类库，用来完成 JDK 自带的类库中没有提供的方便的功能，比如我们之前说过 concurrent 包之前就是以独立的类库出现的
- 翘楚 + 基础：
  - 纯工具：后起之秀 guava，一个库中包含很多方面的工具。老牌 apache common，其中 lang，io 和 common 可以重点关注。网址为 <https://commons.apache.org/>
  - 测试：junit，testng
  - 日志：slf4j + logback
  - 序列化：avro，protobuf
  - JSON 处理：Jackson，Gson
  - http：apache http component，网址为 <http://hc.apache.org/httpclient-3.x/index.html>

- 框架型类库要解决的一般是一个系统的复杂的问题。框架型类库一般都有一套标准（比如接口，配置等）需要遵守。用起来不像工具型那样拿过来用一下就完事儿了。所以一般用某种框架的时候，都会说开发什么什么程序。比如，如果使用 Spring 框架，一般会说开发 Spring 程序 / 应用
- 翘楚：
  - 应用开发框架：Spring + Spring Boot
  - REST API 开发：Swagger Codegen + Swagger UI + <https://editor.swagger.io>
  - 网络框架：Netty
  - ORM 框架：Hibernate, MyBatis / iBatis



# 10

如何在 Stack Overflow 上提问  
才不会被骂

# 如何在 Stack Overflow 上提问才不会被骂

- stackoverflow 是个什么网站？
- 提问三要素：
  - 环境描述：操作系统，JDK 版本，软件 / 工具版本等可能和问题相关的信息
  - 问题描述：描述要解决的问题和思路。可能要解决的问题或者思路本身就有问题。如果问题和程序执行的错误有关，则要把错误信息描述详尽，包括 exception trace 等
  - 代码 + 配置：和问题可能相关的代码和配置
- 一个核心：要自己先思考，先尝试解决。**问题要清晰明了有条理，要让想回答问题的人能够重现所说的問題**。而不是甩手掌柜一样不清不白的问：XX 怎么搞？XX 应该怎么弄？XX 程序运行出错怎么改？池建强老师在极客时间 app 上的卖桃者说有一期专门聊这个，说的非常好，强烈建议大家去听听

# 11

## 浅谈程序设计

- 理解要解决的问题：一开始提键盘就写代码没问题，开始就是要多写代码培养感觉。慢慢得要开始思索，先把问题想清楚，再写代码
- 将问题分解为模块：理解了问题，就可以在大脑里构建这个问题的解决方案了。使用不同的模块，来解决问题，让不同的模块负责不同的功能
- 高内聚，低耦合：模块内部高内聚，聚焦解决模块所要解决的问题；模块与模块之间要低耦合，交互的接口要简单清晰，避免模块内部的细节出现在模块之间的交互中

# 12

## 游戏小程序功能定义

- 游戏为打怪冒险类。主角为一个质子，模拟粒子间通过撞击、量子纠缠（我瞎编的）等方式，吸收别的粒子，获取别的粒子的装备，一步步构造出分子的故事
- 游戏为字符界面游戏，通过字符输入操作游戏。可以支持扩展到 GUI 以及联网多人游戏
- 对战方式：默认情况下随机遇敌。通过撞击打败敌人，可以获取对方物品。通过量子纠缠击败对方，可以将对方粒子纳入自己队伍中
- 可以使用道具，发动量子纠缠攻击就需要道具

# 13

## 游戏小程序设计和模块划分

# 游戏小程序设计和模块划分

- 游戏 Context
- 事件模块
- 输入输出模块
- 物品模块
- 游戏主流程模块
- 工具模块



# 14

## 游戏小程序代码分析

- 按照模块看源代码

# 15

使用 Swagger 创建一个Spring  
Boot 的 Web 服务

# 使用 Swagger 创建一个Spring Boot 的 Web 服务

- 直接实操

# 16

结束语

感谢

# 终身学习

莫欺

少年穷



种

子

非 科 班

必 备

