你好，我是明明如月，一个重视方法、喜欢思考的 Java 高级开发工程师。

相信很多程序员都希望自己能够找到一些宝典，通过修炼 “打开任督二脉”，从此快速进阶成为高手。《Java 开发手册》（以下简称《手册》）就是诸多宝典之一，它几乎是每个 Java 工程师人手必备的一本参考指南。该手册包括 **编程规约、异常日志、单元测试、安全规约、MySQL 数据库、工程结构、设计规约 7 个部分** ，涵盖了 Java 开发的常见知识点。认真实践该《手册》能够帮助 Java 开发者养成好的编程习惯，帮助企业的开发团队在 Java 开发上更加高效、提高容错性、团队协作更好，并有助于提高代码的质量、降低项目维护的难度。然而很多人会遇到看过就忘，记住却不理解、不会用的困境。

另外在实际的学习和工作中，你是否遇到过如下尴尬：

1. 看《手册》等 Java 技术图书时觉得啥都懂，实战时就忘了；
2. 很多知识点，知其然而不知其所以然，面试时多问你几个为什么就 “靓仔语塞”；
3. 想通过读源码来进阶，但是容易迷失在细节中，总是半途而废；
4. 不重视需求分析，导致开发完成才意识到设计和需求有偏差；
5. 遇到问题时如果无法简单地定位原因，会优先通过百度、请教别人来解决问题；
6. 开发中遇到问题排查耗时很久，方法很原始；
7. 自己开发的项目，每次上线几乎必出 BUG，而有些同事的项目质量则很高，自己却不知道如何才能尽可能地避免。

结合自己学习和工作这么长时间的思考，将出现上述尴尬的原因归结为以下几个原因：

1. **知道很容易，懂很难，很多人把知道当做懂。**自认为掌握了就不愿意再深入学习，恰恰错过了彻底掌握该知识的最佳机会；
2. **专业基础不够扎实。** 很多人急于求成，只重视解决眼前问题，不能够未雨绸缪，巩固好专业基础，最终导致很多问题” 知其然而不知其所以然，排查问题时靠猜、靠问，而不是靠扎实的专业基础之上的推测和验证；
3. 很多人**不愿意改变学习方法，学习没培养好的编程习惯，不敢走出舒适区。**比如很多人学了很多技术，却从来没有认真仔细阅读过官方文档；比如读源码毫无章法，随心所欲，常常半途而废；
4. **态度决定一切。**很多人嘴上说要学好，但是对自己代码要求很低，总是为自己找各种理由不去学更好的方法，不去努力写更优雅的代码；
5. 在学习技术过程中，很多人**把脑力劳动当成了体力劳动，把需要思考的问题当做了纯记忆的问题**，学习和工作过程中缺乏思考。比如很多人是 “记忆” 经典图书的知识点，而不是理解知识点，导致容易遗忘，不能灵活运用。在学习很多知识点时缺乏思考，没有去搞懂是什么、不明白为什么、不知道如何去做；
6. **没有养成好的解决问题的习惯，排查问题靠猜，而不是思考和验证。**也没有主动掌握常见的排查问题的步骤和工具等。

很多人缺乏的不只是好的资料，而是学习的方法。学一样的技术，使用不同的方法，最终学习的效果截然不同。而**技术是学不完的**，如果没有科学的方法，无法很好地应对层出不穷的新技术。**每个人的成长速度是不同的**，有的人工作多年，却只有一年的技术经验；而有的人工作一年，却有超越一年的技术经验。造成这种差异的主要原因在于**学习能力。**

从 Java 新手到高手的进阶过程是一个漫长的爬坑过程。 很多同学遇到 BUG 时由于基础不扎实也没有系统地排查方法，为了解决一个小问题浪费了大把的时间。而且写出的 BUG 太多将直接或间接影响绩效，影响同事、领导对你的印象。

阻碍初学者进步的往往是一叶障目不见泰山的盲目自信，往往是一成不变学习方法。破解上述尴尬的核心在于提高学习和排错能力。

为了解决上面提出诸多尴尬，本专栏的具体应对策略如下：

1. 从学习方法主要切入点，结合源码，Java 语言规范 和 Java 虚拟机规范 等对《手册》的讲解和补充；
2. 设计者角度思考问题，很多知识点将从设计者视角去思考分析问题，更容易理解问题的根源；
3. 通过对开发中常用的思维导图、流程图和常见 UML 图的讲解，让大家可以 “大战需求分析”，前期明确需求，后期少返工；
4. 通过单元测试、Code Review 等相关知识的学习和运用，促进代码质量的提升
5. 通过独特的学习源码视角，来从正确的角度和方法来学习源码的精髓，反向促进日常的开发；
6. 结合实际的开发经验，给出相关知识点掌握不牢容易造成的坑，给出一些避坑建议。

本专栏大多数章节的结构设计如下：

1. 逻辑特色：采用 2w1h 分析方法，即是什么（what），为什么（why）和如何去做（how）的角度来学习知识；
2. 问题驱动：采用 "5w 思考法"，即不断的追问逐渐思考问题的本质，从而实现知识理解的更加深入；
3. 方法驱动：每节将使用一些学习和解决问题的方法，让大家可以掌握学习的章法；
4. 对比和类比分析：大多数章节会对知识点和类似的知识点进行对比或类比，从而找出知识之间的联系和差异，加深对知识的理解；
5. 坑点解读：讲解知识理解不到位可能造成的坑点，分析趟坑原因并给出避坑建议。

注：本专栏所涉及的 Java 源码均默认为 JDK 8 版本（特殊标注除外）。

技术是学不完的，学习能力和态度才是进阶的关键。作为一个技术人员，只有保持 “Stay Hungry， Stay Foolish” 的心态，才能够保持进取心；只有真正知道哪些才是更有价值的东西，才能真正少走弯路。

希望大家能够通本专栏的学习，改变学习技术的思维意识，**从 “学习具体内容” 为主，转变到学习 “学习的方法” 为主**；**从技术的学习者变为技术的思考者。**希望本专栏可以帮助到更多朋友加速技术成长的步伐，做一个更加专业和优秀的 Java 工程师。

}