1. 代码校验，**如何证明自己的代码正确性**
2. **@BeforeAll** 注解是在任何其他测试操作之前运行一次的方法。 **@AfterAll** 是所有其他测试操作之后只运行一次的方法。两个方法都必须是静态的。
3. **@BeforeEach**注解是通常用于创建和初始化公共对象的方法，并在每次测试前运行。可以将所有这样的初始化放在测试类的构造函数中，尽管我认为 **@BeforeEach** 更加清晰。JUnit为每个测试创建一个对象，确保测试运行之间没有副作用。然而，所有测试的所有对象都是同时创建的(而不是在测试之前创建对象)，所以使用 **@BeforeEach** 和构造函数之间的唯一区别是 **@BeforeEach** 在测试前直接调用。在大多数情况下，这不是问题，如果你愿意，可以使用构造函数方法。
4. 如果你必须在每次测试后执行清理（如果修改了需要恢复的静态文件，打开文件需要关闭，打开数据库或者网络连接，etc），那就用注解 **@AfterEach**。
5. *<!-- 谷歌开发工具类，极其强大 -->*<**dependency**>  
    <**groupId**>com.google.guava</**groupId**>  
    <**artifactId**>guava</**artifactId**>  
    <**version**>28.1-jre</**version**>  
   </**dependency**>
6. 谷歌guava里面提供的都是静态方法，并且处理方式上更加的快速的，编写处理更加方便，现在在集成中已经存在的有：**集合、缓存、原生类型支持、并发库、通用注解、字符串处理、I/O等等**
7. 如果你在设计和编写代码时考虑到了测试，那么你不仅可以写出可测试性更好的代码，而且还可以得到更好的代码设计。
8. “测试失败的书签”这一概念，为了能够回到上次编写代码失败的地方
9. **日志**会给出**正在运行**的程序的各种信息。在调试程序中，日志可以是普通状态数据，用于显示程序运行过程（例如，安装程序可能会记录安装过程中采取的步骤，存储文件的目录，程序的启动值等）。
10. 如 \*Simple Logging Facade for Java(SLF4J)\* ,它为多个日志框架提供了一个封装好的调用方式，这些日志框架包括 \*\*java.util.logging\*\* ， \*\*logback\*\* 和 \*\*log4j \*\*。
11. 编译器生成编译信息。然后使用如下命令开始调试程序：

\*\*jdb SimpleDebugging\*\*

1. 输入？查看jdb的相关命令
2. JMH的引入查看，并行操作和非并行操作的快与慢(基准测试)
3. **剖析和优化：**剖析器还可以帮助检测程序中的线程死锁，
4. \*\*static final\*\* 修饰的变量会被 JVM 优化从而提高程序的运行速度。因而程序中的常量应该声明 \*\*static final\*\*
5. 无论何时有可能的话，显式地设置实例为 null 表明你不再用它。这对垃圾收集器来说是个有用的暗示。
6. 关于代码风格检测的一个checkstyle-8.17-all.jar，网上有一个checkStyle的检测教程，这个暂时不适用，idea中有现成的阿里巴巴的集成
7. FindBugs的插件，IDEA中就有这个插件，用来排查：静态类型检测，排查null异常