### 代码泛读结对练习

#### 阅读源码

阅读源码，对我们的编码能力提升还是比较大，就像我们搞研究，是需要看一些文献。所以阅读源码多了，我们在代码能力、代码功底就会有很大的提升。

我们在阅读代码前，首先是需要对你所阅读的代码有一个整体的认知，就比如我们阅读Hadoop源码。2008年1月，Hadoop成为Apache顶级项目。到现在2022年，已经发展了14年了，所以我们正确的认5识到，我们行业顶级精英，天才级别的大神，写了14年的Hadoop项目，我们想花费多长时间来去读懂、读透。

显然我们想一周、两周、两个月来读懂读透，这是不大可能的，所以这是为什么很多的同学，在阅读源码的时候，还没有开始就已经结束的原因，很多是因为没有正确认知项目的发展和积累。这有点像我们刚会识字或者写作文，就让我们去看四大名著，而且像快速看完，这显然是达不到的。

在我们对源码有了整体的认识后，接着我们就要具体的去研读源码，那么该如何阅读源码，阅读源码的步骤：

1.首先了解项目背景

前面我们已经涉及到，项目背景是非常重要的，比如Hadoop，我们是否对他所了解。Hadoop为何产生，是为了解决在大数据量的情况下，单机很难计算和处理的数据的情况下，所以产生了价格和成本都非常昂贵的超级计算机。所以有的人就想如何通过廉价的普通的计算机来实现计算大数据量，所以Hadoop应运而生。Hadoop又分为Hdfs、Yarn等组件，当然这里面又会细分，我们了解的越详细，对我们阅读源码越方便和快捷。

2.了解项目功能、结构

了解项目功能、结构，比如哪些是通用部分，哪些是功能部分。所以这里我们需要一定猜测，这个猜测我们同样需要去验证。有的大神称其为““正向推导+验证””，这里其实也和我们的学习方法和思维关联。我们在学习比如当前的源码，你的学习思路是什么？是一直跟着文件或者文档的思路去学习，还是自己提前有一定的想法，然后去文档或者文件中去验证自己的想法，这二者的学习效果和速度是不一样的。

3.调试代码

调试代码，很多同学卡在了这里，因为跟踪代码的时候，跳来跳去，就整蒙了。要么不知道为何会跟踪到这里，要么跟踪到这里，不认识它，总之很多同学到这里有的就放弃了。

所以在这里我们该如何破解这个关卡？

1.掌握整个代码的流程走向

2.掌握类之间的主次、依赖关系

3.摸清架构层次，整体把控

4.阅读由易到难

先阅读基础模块，然后阅读依赖较多的模块。我们在刚开始阅读文档，就想去攻克难点，这样会非常容易让我们放弃，所以其实这也是做事的策略和方法。

5.总结

我们在阅读完源码后，去总结从代码深度，到类与类之间的关系，总结出代码的逻辑和实现逻辑。这样达到我们读懂源码的目的。

1. 代码泛读（2 周）
2. 阅读别人的代码作为开发人员是一件经常要做的事情。
   1. 一个是学习新的编程语言的时候通过阅读别人的代码是一个最好的学习方法
   2. 另外是积累编程经验。如果你有机会阅读一些操作系统的代码会帮助你理解一些基本的原理。
   3. 还有就是在你作为一个质量保证人员或一个小领导的时候如果你要做白盒测试的时候没有阅读代码的能力是不能完成相应的任务。
   4. 如果你中途接手一个项目的时候或给一个项目做售后服务的时候是要有阅读代码的能力的。
3. 代码泛读的主要目标是在较短的时间内快速掌握该软件的功能和概要设计，包括软件的体系架构、模块划分等。
4. 规格：对于应用型程序，要先大致整清楚它的使用方法。
5. 静态结构：弄清楚代码的基本静态结构。如：包构成、类构成等。要弄清楚每个部分的核心职责
6. 抽取出几个典型的应用场景：要在典型场景下考察上面的静态结构是如何发挥作用的。典型场景下用到的接口往往就是关键的接口，要整清楚，他们的定义和作用。也要整清楚，典型场景下数据流的变迁。弄清楚代码的时序。
7. 关注进程、线程的结构。比如：都是什么时候开始、什么时候结束的，在上述典型场景下都负责干什么。

二、微人事

* 1. 微人事是一个前后端分离的人力资源管理系统，项目采用 SpringBoot+Vue 开发，项目加入常见的企业级应用所涉及到的技术点，例如 Redis、RabbitMQ 等
  2. 项目地址：<https://github.com/lenve/vhr> 或者

<https://gitee.com/lenve/vhr> <https://gitee.com/lenve/vhr2.0>

* 1. 功能特点：

实现人事管理信息系统，包括员工资料、人事管理、工资管理、统计管理、系统管理等;

三、小米便签社区开源版（Community edition of XM notepad/MIUI notes)

* 1. 小米便签是小米公司 MIUI 团队开发的一款手机 App。Android 编程是目前非常热门的一项编程技术，便签也是大家广为熟悉的一款日常使用软件<https://github.com/MiCode/Notes>
  2. 下载链接<https://codeload.github.com/MiCode/Notes/zip/master>
  3. 构建参考：[https://www.learpnerhub.net/#/spaces/146/resources-docs/6353](https://www.learnerhub.net/#/spaces/146/resources-docs/6353)
  4. 功能特点：
     1. 快速新建，自动保存，自由分类;
     2. 与Google Task同步，方便将本地便签上传、谷歌工作表中的工作项同步到本地;
     3. 独特的清单模式，简单方便的管理事务；
     4. 闹钟提醒，发送便签到桌面、快速分享.
     5. 文本中对邮箱、电话号码、网址的识别，方便快速操作

### 练习要求

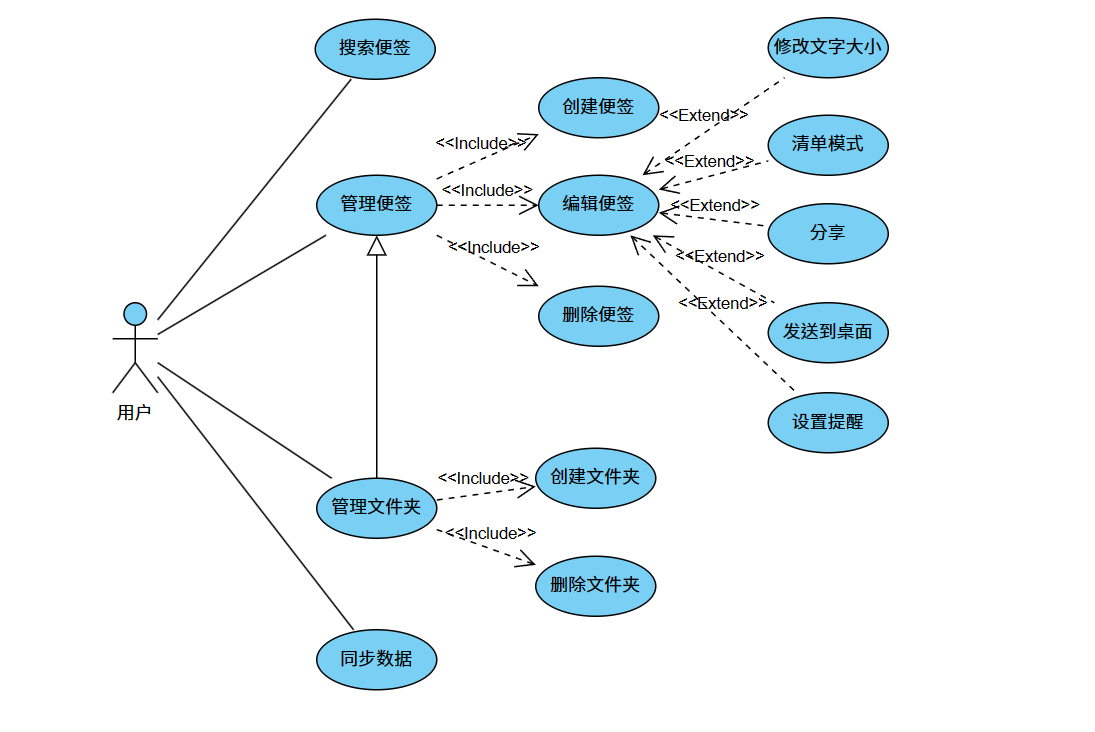
1. 要求 **2** 人一组（实验报告给出结对的学号和姓名，如果是一人尝试和AI结对），采取结对的方式完成对整个项目代码的泛读。
2. 每个人都必须给出实验报告，标明共同完成部分和自己完成部分。
3. （20分）在微人事或者小米便签（二选一）尝试搭建环境运行项目，给出运行过程的**动态展示**（gif或者视频）

视频已经上传到课堂派

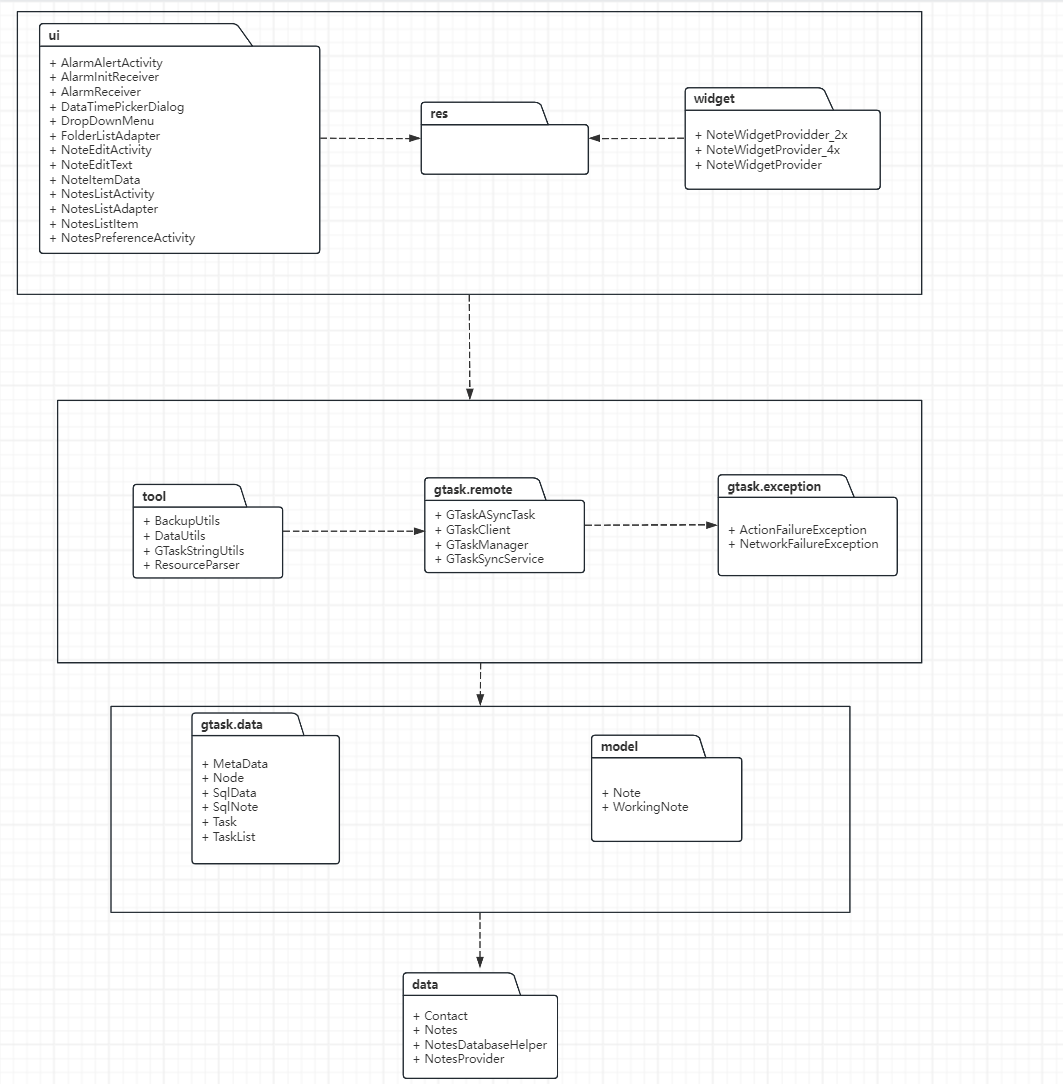
1. （10分）源码组织方式（给出仓库地址）：
   1. 创建针对本作业的项目和软件版本库，在版本库中建立“src”和“doc”两个文件夹，分别存储软件系统的源代码和报告文档
   2. 建立master、develop以及成员分支(a\_branch)，将当前版本存入master目录下
   3. 实践操作参考：组长组员两个人协同开发：组长负责维护开发分支dev，组员向dev上传提交；当dev测试合适后，组长有唯一权限向master上传作为最终结果。
      1. 远程仓库有master和dev两个分支
      2. 组长本地有master和dev分支，分别关联对应的远程分支
      3. 组员本地只有一个分支，关联远程dev分支【可以选择clone某一个远程分支到本地】
      4. 具体开发流程是：
         1. 组长和组员分别在各自的本地dev分支开发，有阶段性成果后push到远程dev【若有冲突，解决冲突再合并】
         2. 当开发完成、结果稳定后，组长将本地的master和dev分支merge，再把master分支push上去

仓库地址：https://github.com/WanQuanBuDongBianCheng/SoftwareEngineering.git

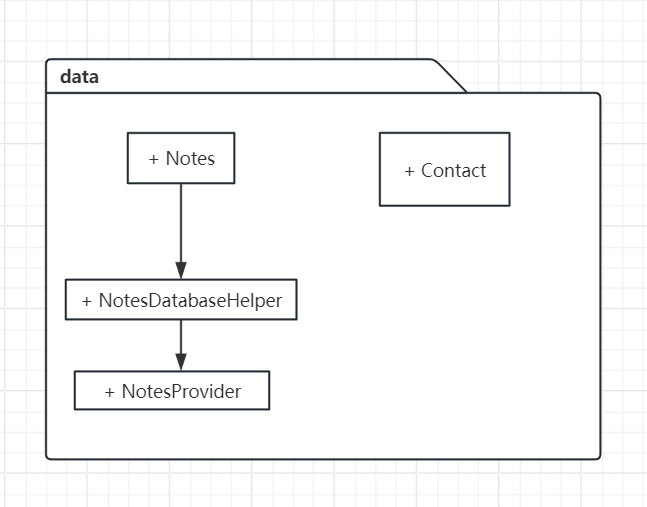
1. （40分）通过泛读微人事或者小米便签（二选一）代码给出：
   1. **用例图**：通过用例图来描述微人事系统（或小米便签）的主要功能以及它们之间的关系；

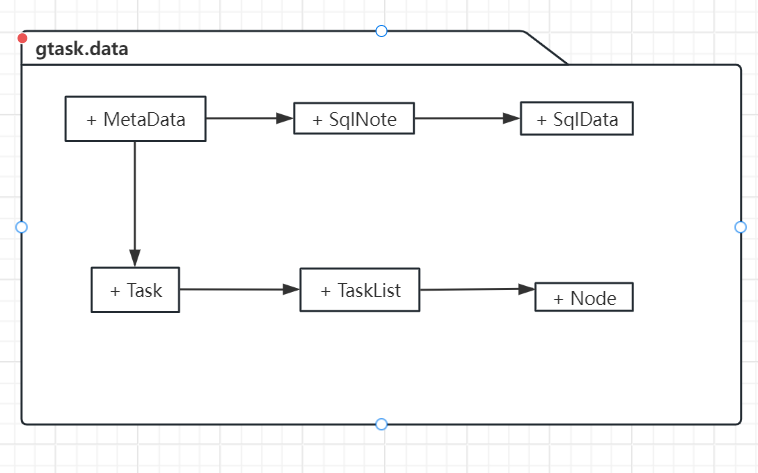


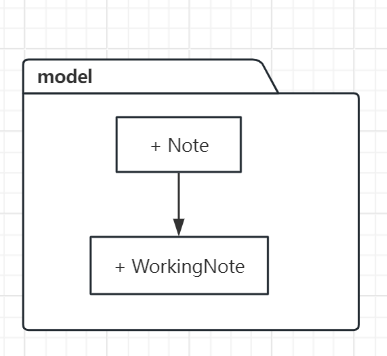
* 1. **体系结构图（包图）**：通过绘制体系结构图来了解整个软件的总体设计思路；

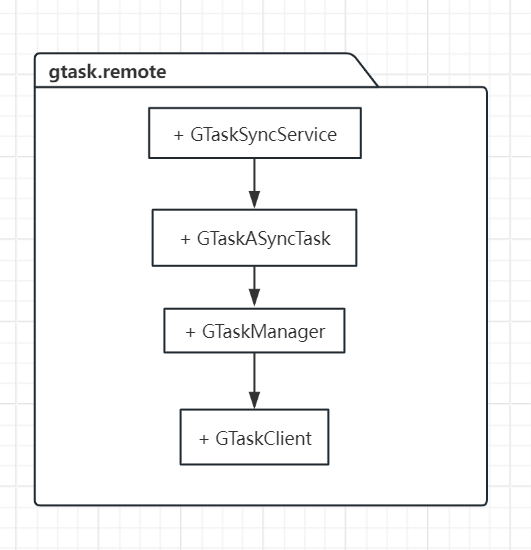


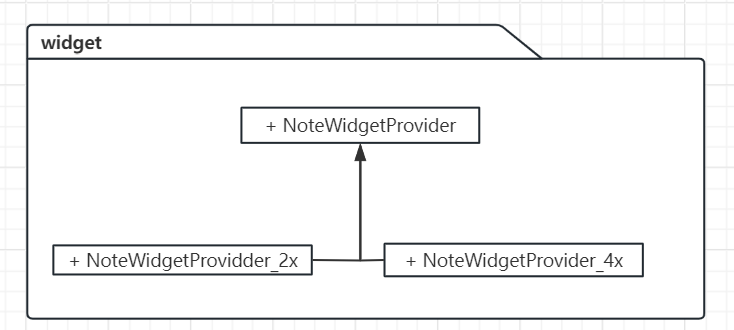
* 1. **类之间的调用关系图**：通过绘制类之间的调用关系图来掌握微人事（或小米便签）的具体设计；











* 1. **核心类**的主要作用：通过给出核心类的主要作用来进一步加深对软件设 计的理解。

1. NoteEditActivity：创建便签、删除便签
2. DataUtils：移动便签
3. NoteListActivity：新建文件夹、查看文件夹、删除文件夹、修改文件夹、导出
4. GTaskSyncService：同步便签
5. NotesListActivity：搜索便签
6. NoteEditActivity：修改字体大小
7. WorkingNote：修改背景颜色、进入清单模式
8. NoteEditActivity：发送到桌面、设置提醒、分享
9. Contact：识别电话号码
10. WorkingNote：保存便签
11. （30分）给出两人的具体的分工描述（例如，哪些部分由谁完成，哪些部分谁做领航员，谁担当驾驶员，具体参考课本4.5.4节如何结对编程），请描述结对的感受，是否1+1>2？如果是和AI结对，给出和AI的交互过程。





本次作业我选择和AI结对，AI不算是一个人而是一个工具，使用这个工具肯定是可以实现1+AI>1的，只要能用好AI，肯定是比一个人埋头苦干搜索问题效率更高。

这次泛读代码的过程中，我直接将代码块复制了发给AI让他帮我解释，虽然不能一下子精读代码，但这样的方法在面对大段的代码时，可以更快的对它们有一个基本的认识，这样在仔细研究时候可以更有目的性。如果自己直接看代码想要很快弄懂的话，对我来说浏览这么多单词并且要联系起来还是比较头疼的。

但是AI不能解决所有问题，问它问题也需要有一定的提问技巧，问题问的不够具体的话它可能就无法给到想要的答案。对AI的使用还是需要一些时间去熟悉的。当真正能把AI用到极致时，我觉得也能实现1+AI>2。