# java01

## Task1

- JDK:即java development kit 顾名思义就是Java开发工具包 它包括JRE 当然还有编译器 编辑器 调试工具等 就是一个完整的开发工具包
- JRE:即java runtime environment Java运行环境 包括了运行Java程序所需要的Java类库等组件 它包含JVM 有了JRE才能使java程序在设备上正常运行 当用编译器编写Java项目时就能在你的项目下看到jre目录
- JVM:即java virtual machine Java虚拟机 Java的一个优点就是兼容性好 跨平台能力强 JVM帮助在不同操作系统下能使相同的程序正常运行 输出同样的结果 此外还具有垃圾回收 内存管理等功能

> 以上是查询资料后的个人理解 总结一下就是JDK是全家桶 包括开发和运行各种功能 它包含JRE JRE只提供运行Java项目功能 不能用它来开发Java项目 而JRE包含的JVM是提供跨平台能力的

## Task2

> 配置环境变量 折腾了两天终于配上了

1. 在环境变量中添加JAVA_HOME 以我安装的jdk-21为例 变量值取jdk-21的地址
2. 之后在path变量中添加%JAVA_HOME%\bin 以便在终端里调用javac和java命令

> 在终端输入java -version检查是否配置成功 原本我是以为成功了 调用java和javac时出现了win系统后缀名被隐藏等问题 修复之后仍旧报错一个接一个 查了各种资料终于找到原因了 如下

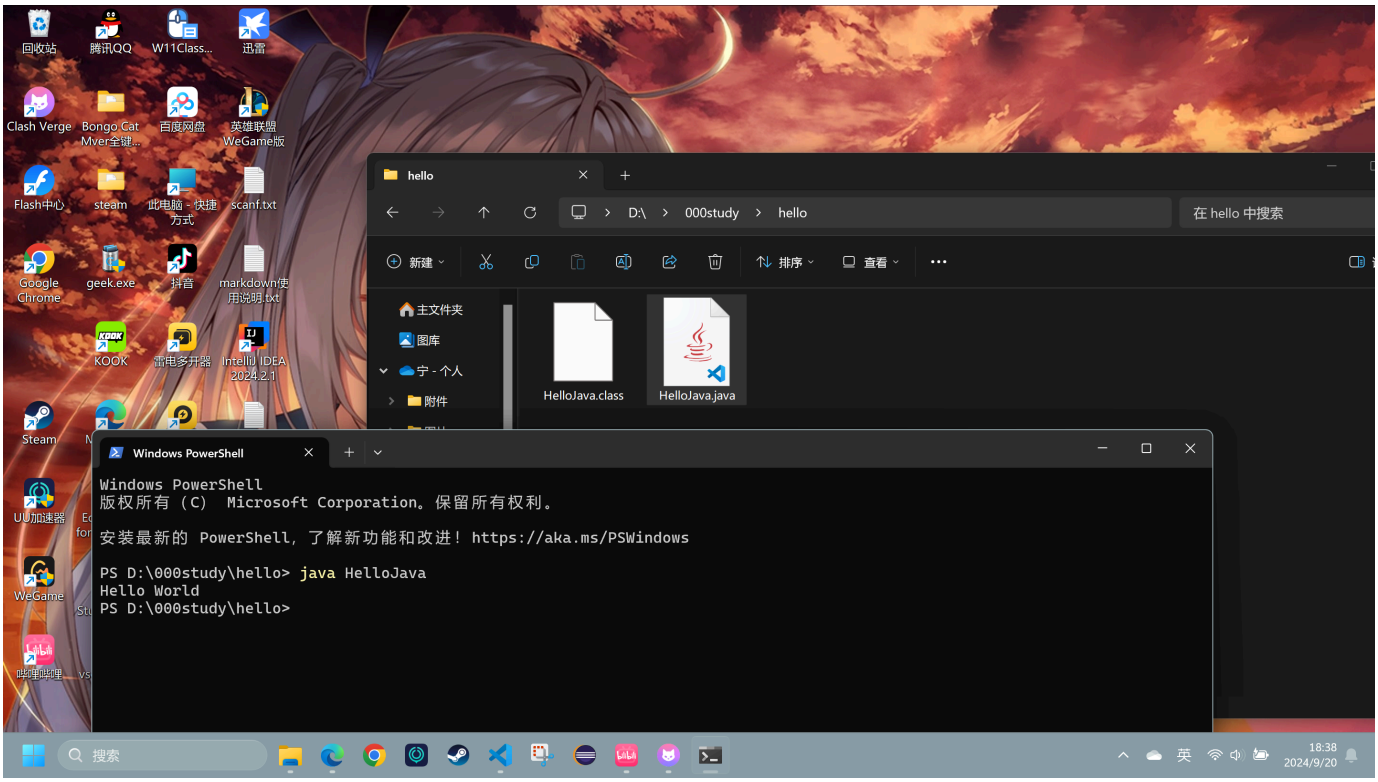
3. 新版本的jdk目录下没有jre文件夹了 所以要在终端里手动调一个出来 并且配置jre的环境变量 方法同jdk

> 配置时要注意删除旧版本的jre环境变量!!! 否则会出现jdk和jre版本不匹配而无法正常运行的问题 我查了好久才发现这个问题

- 配置环境变量后 系统就能找到jdk的可执行文件 如java javac这些 当你在命令行输入这些命令时 系统会在path中查找相应的文件 就可以执行辣!!

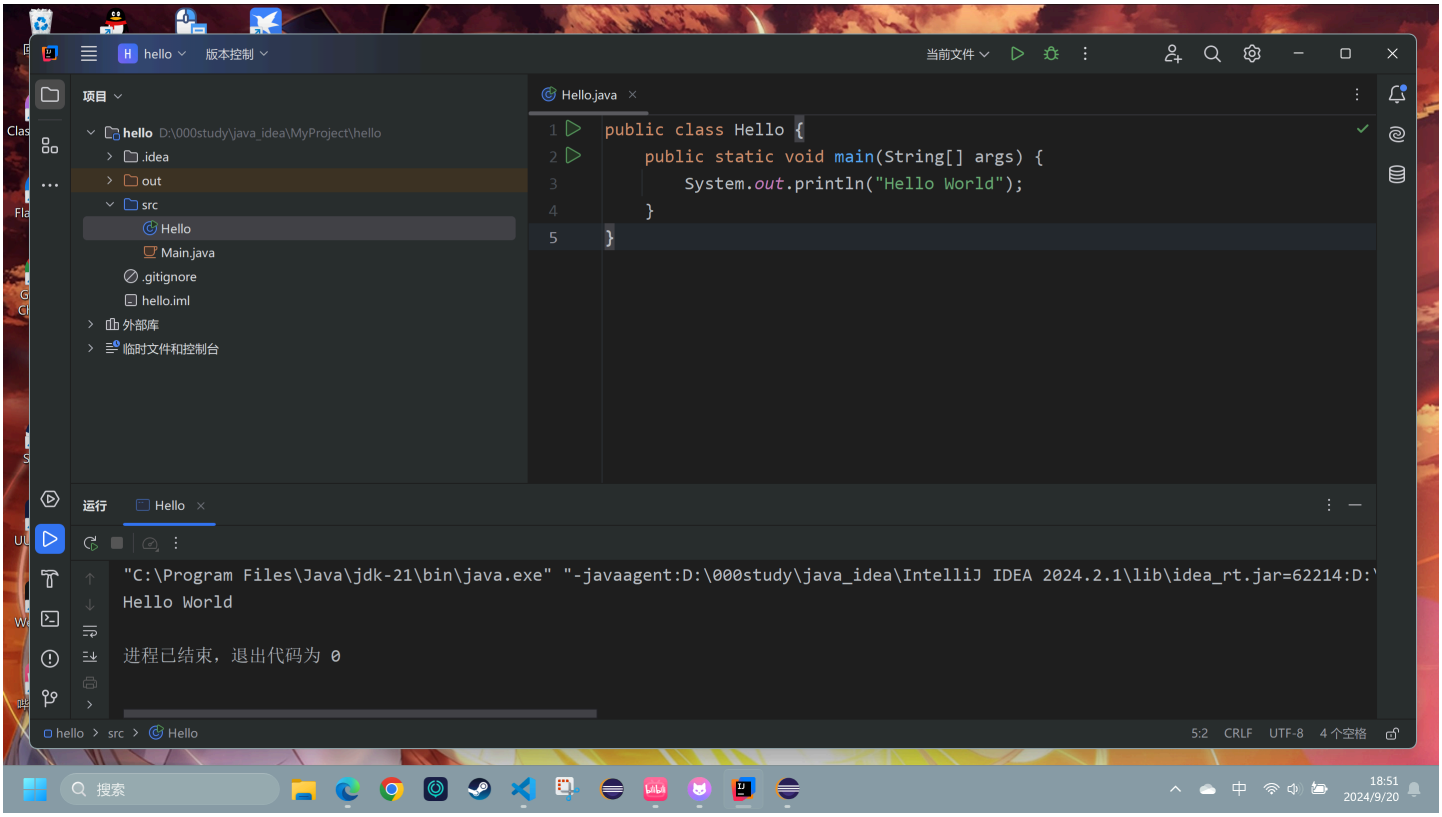
## Task3

- 涉及jdk目录下的javac.exe和java.exe文件 以及你自己目录下的.java文件和.class文件
- 运行时 先用javac将后缀名为java的文件编译 系统会自动为你生成一个class文件 再用java命令就可以执行这个class文件了




# Task4

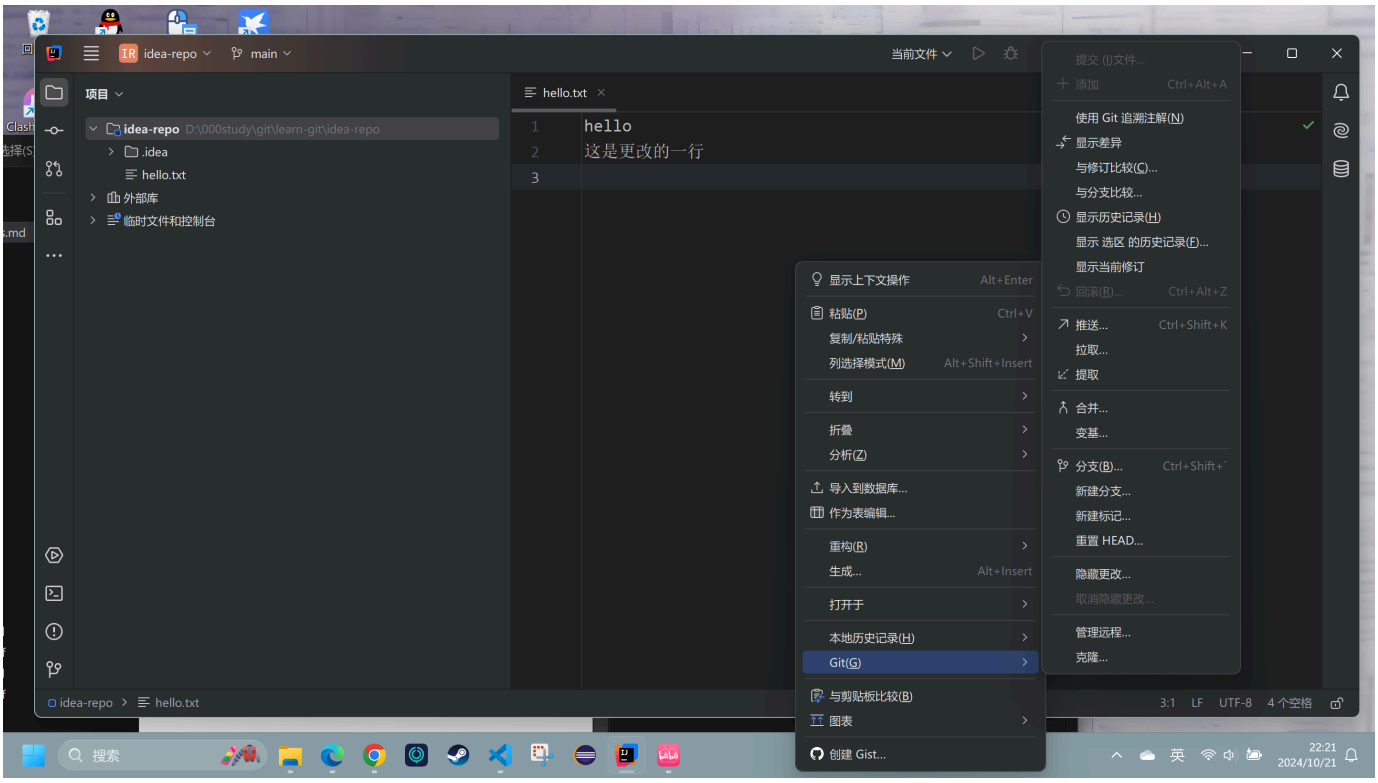
再见eclipse idea我来辣




# Plus Content

5.

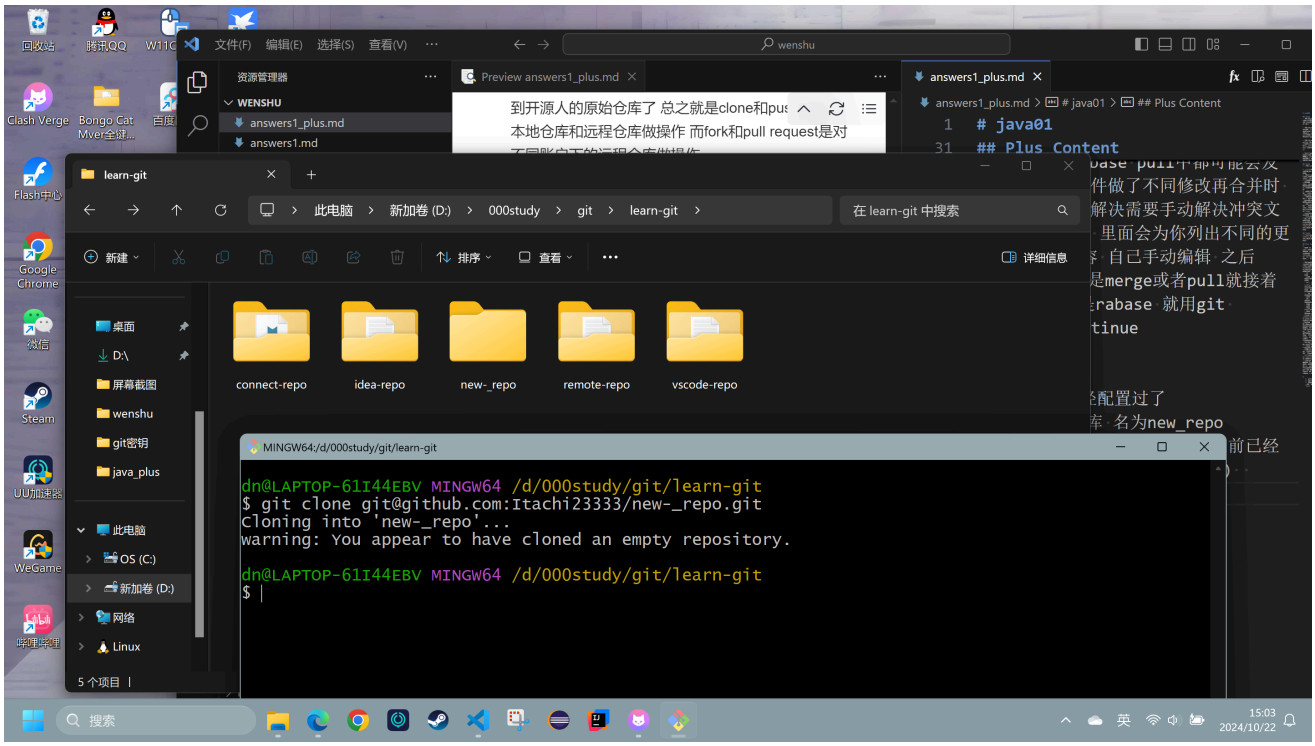
- 首先 以.开头的文件均为隐藏文件 需用ls -a才能看到 .git是本地仓库的核心文件 它包含了所有与当前git仓库相关的信息和元数据 有了这个文件 我们才能在命令行中使用git命令 它的存在意味着当前目录是一个git仓库 不能随意更改和删除 隐藏起来也体现了它的重要性
- 这里不知道是不是版本问题 我的idea上方没有索引 默认禁用了git add功能 是不是新版本都是这样的...



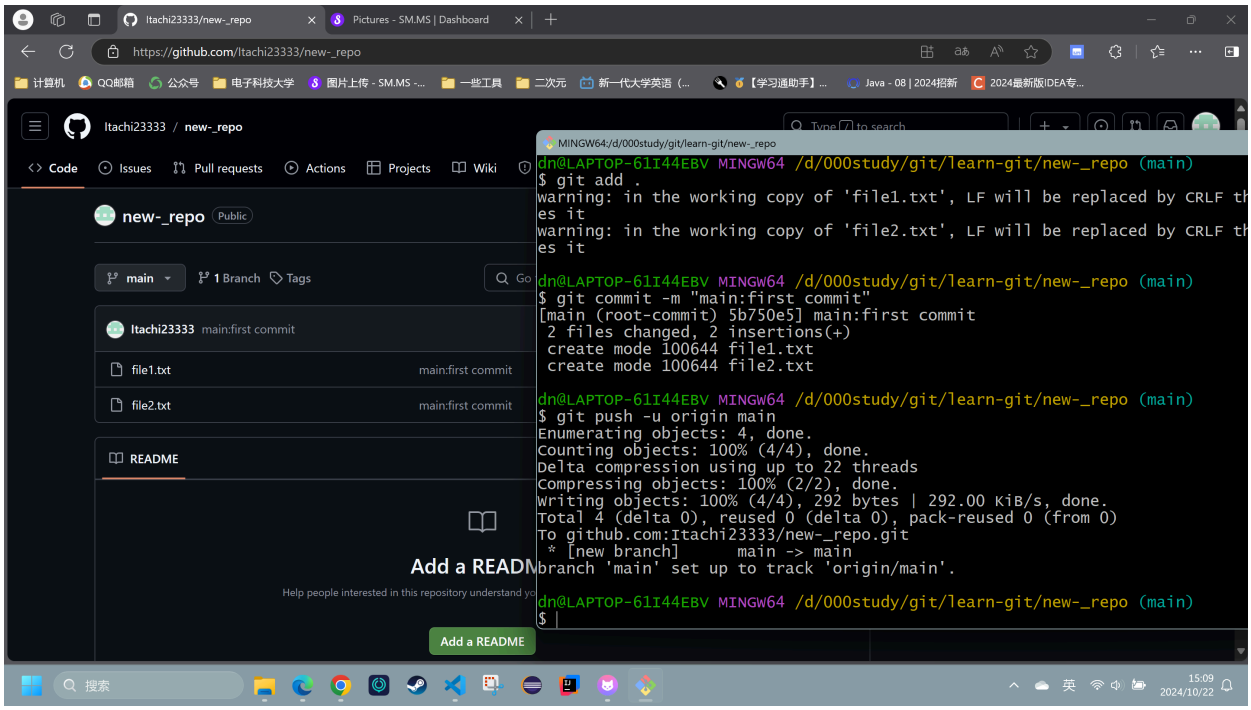

下面对这些命令做出解释
  - 添加 (git add) 将文件添加到暂存区
  - 提交 (git commit) 将所有更改提交到本地仓库(包括新添加 修改 删除文件)
  - 回滚 (git reset) 撤销更改
  - 签出 (git checkout 或 git switch) 建立新分支 命令行用git switch会更好一点 git checkout还有撤销文件更改的用途 如果文件名和分支名重的话会有歧义
  - 删除 (git rm) 删除暂存区和工作区的文件
  - 合并 (git merge) 合并分支
  - 变基 (git rebase) 也是一种分支的合并吧 把一条分支移动到另一个分支的顶端
  - 克隆 (git clone) 克隆远程仓库
  - 提取 (git fetch) 获取远程更新
  - 更新 (git pull) 获取远程更新并且自动合并 相当于先fetch再merge
  - 推送 (git push) 将本地的commit推送到远程仓库
- clone是把远程仓库克隆到本地 fork是把远程仓库克隆到自己账户里的远程仓库 push是把本地仓库的提交内容推送给与这个本地仓库相关联的远程仓库 如果是别人的开源项目 尽管你可以clone下来 得到与之相关联的本地仓库 但是你并没有push的权限 我们可以先fork到自己账户里 然后clone这个fork后的仓库到本地 更改后 push到自己的远程仓库 然后使用pull request 请求开源作者容许我的更改 对方同意后 我的修改就会合并到开源人的原始仓库了 总之就是clone和push都是对本地仓库和远程仓库做操作 而fork和pull request是对不同账户下的远程仓库做操作
- 工作区:能在文件资源管理器直接看到 做修改的区域

- 暂存区:git add可以把工作区的文件暂存到这个区域
- 本地仓库:git commit则会把暂存区的所有更改(可以是添加 删除 更改内容)一并提交到本地仓库
- 远程仓库:github等平台托管代码的仓库 在建立本地仓库和远程仓库的连接之后 就可以把本地仓库push到远程仓库了
- 在merge rebase pull中都可能会发生冲突 当对文件做了不同修改再合并时 就会出现冲突 解决需要手动解决冲突文件 打开该文件 里面会为你列出不同的更改后的文件内容 自己手动编辑 之后git add 如果是merge或者pull就接着commit 如果是rabase 就用git rebase --continue

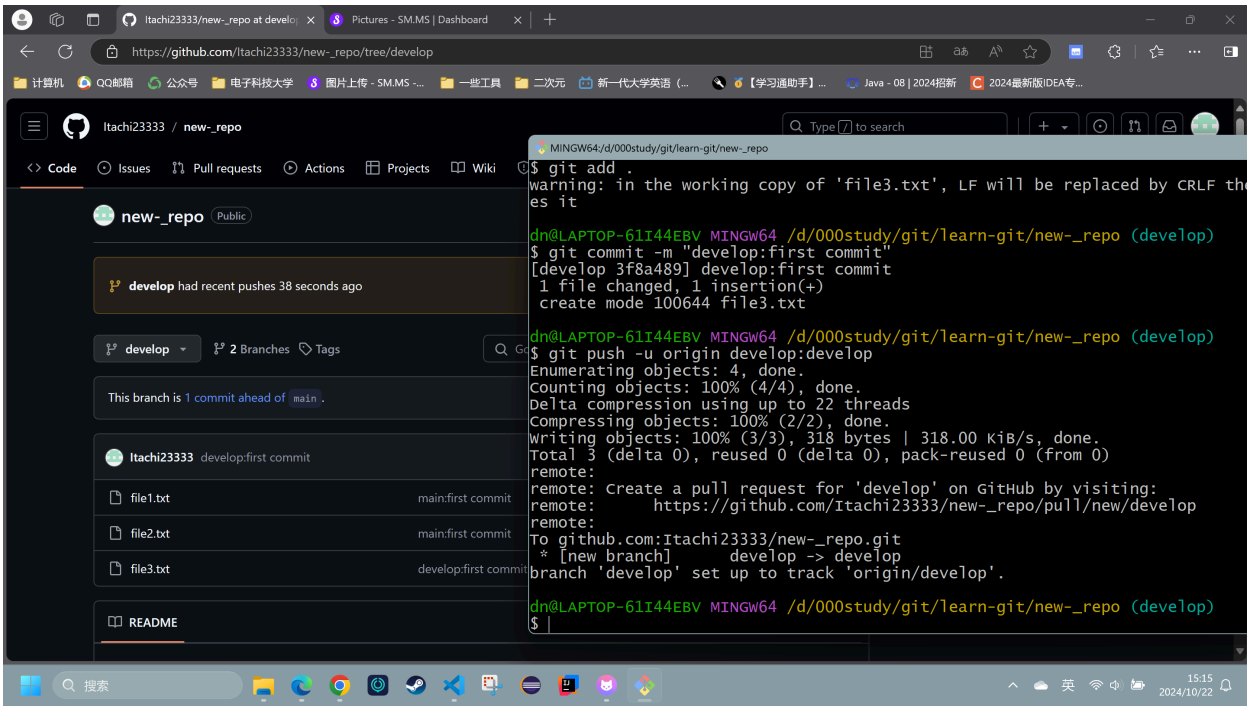
6. i. git已经配置过了
   ii. 新建仓库 名为new-_repo
   iii. 克隆到本地(SSH密钥我之前已经配置过了 这里直接克隆了就)



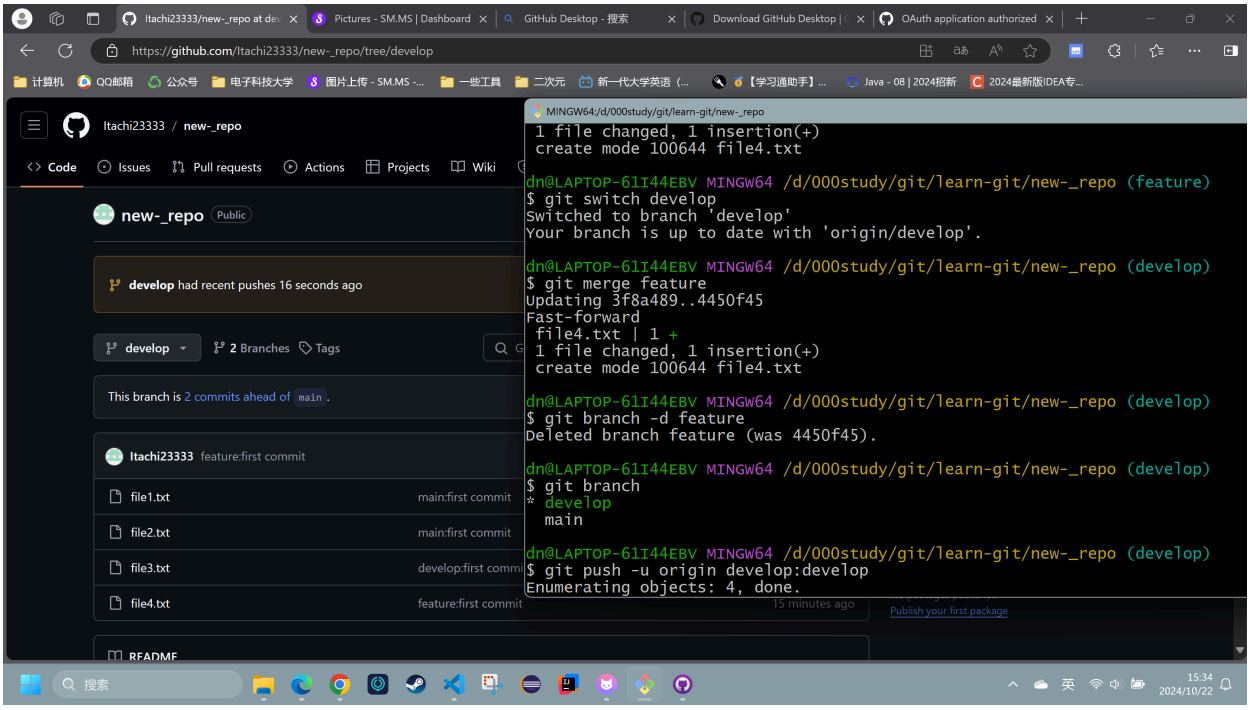

   iv. - 先再main分支上提交了两个文件



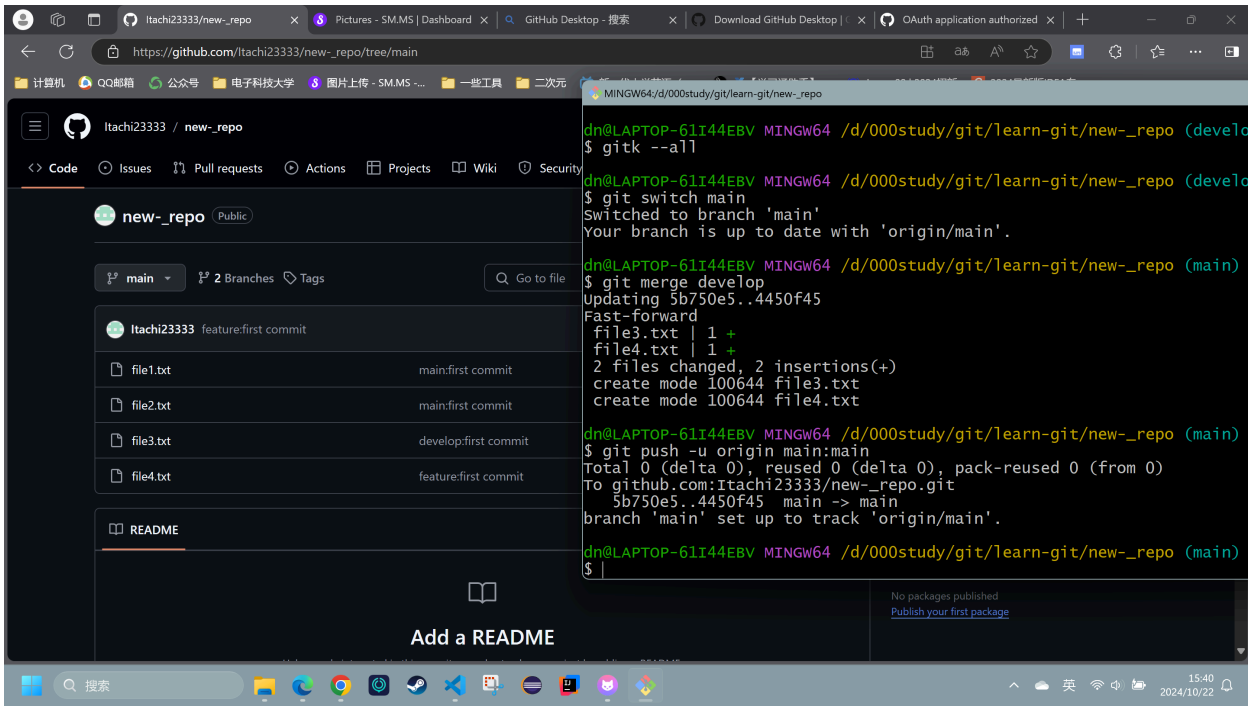

   - 创建develop分支



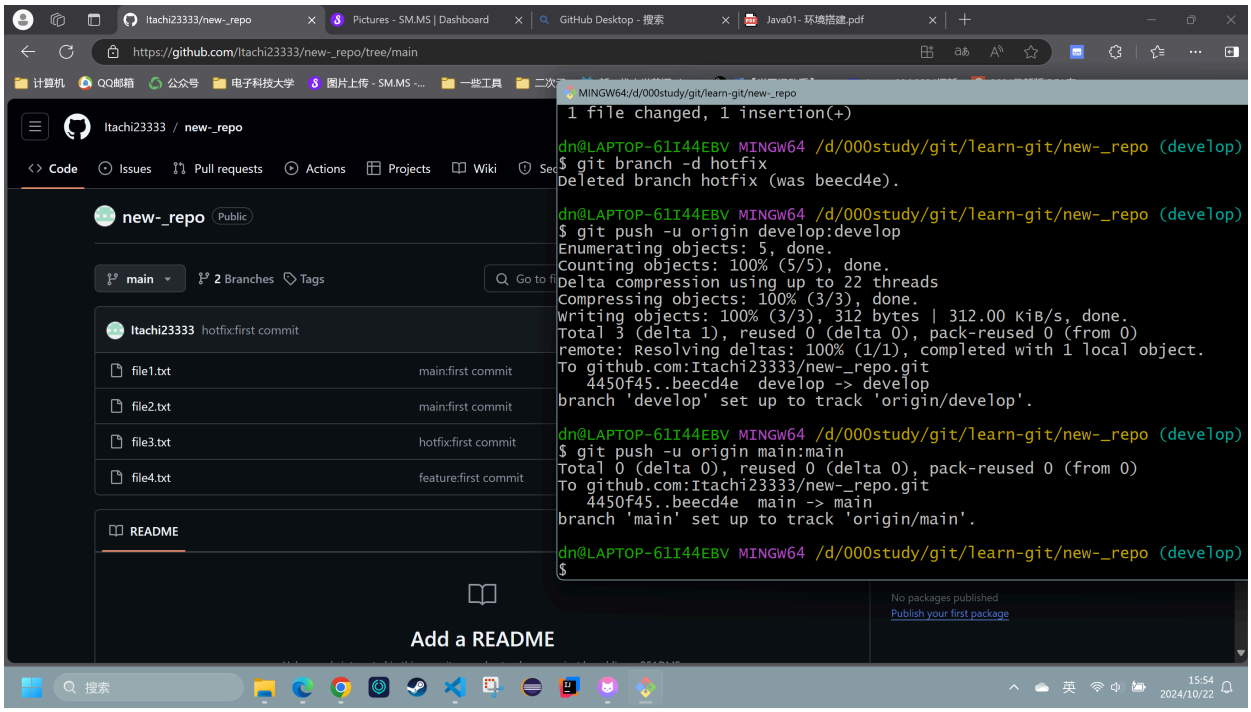

- 在feature分支上创建了file4.txt 合并到develop分支后删除feature分支 并推送到远程仓库

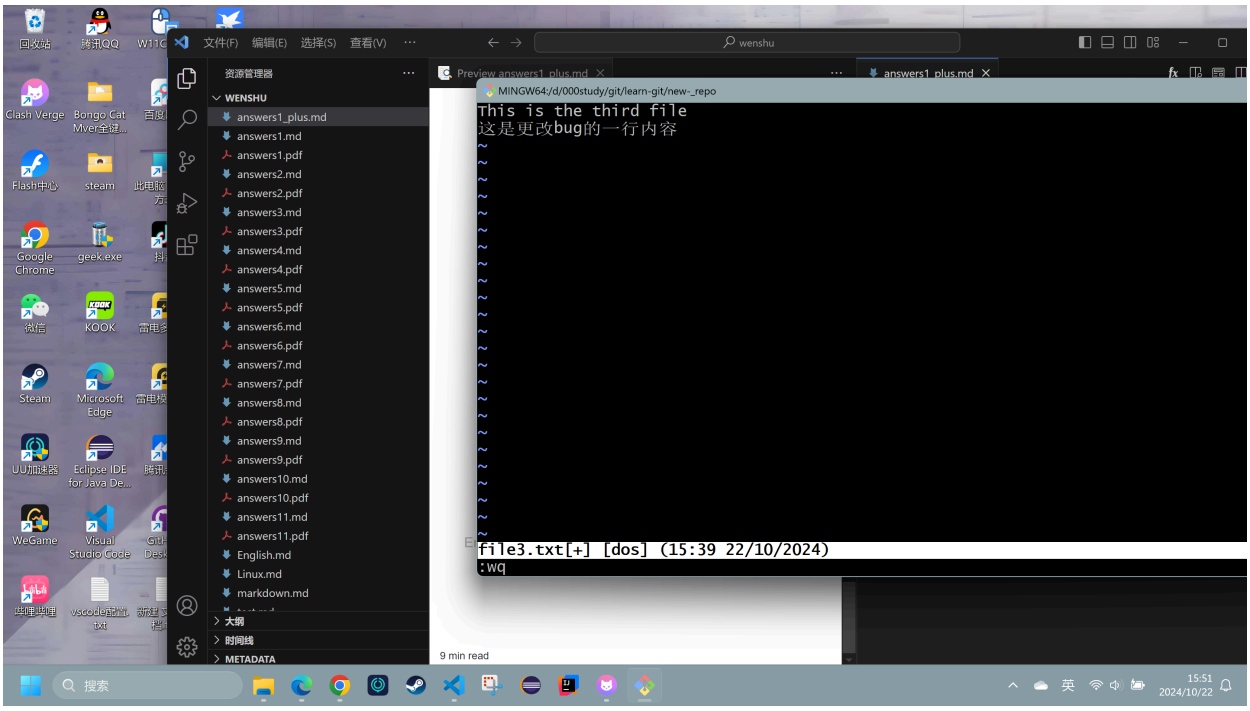



- 将develop合并到main分支上 再推送

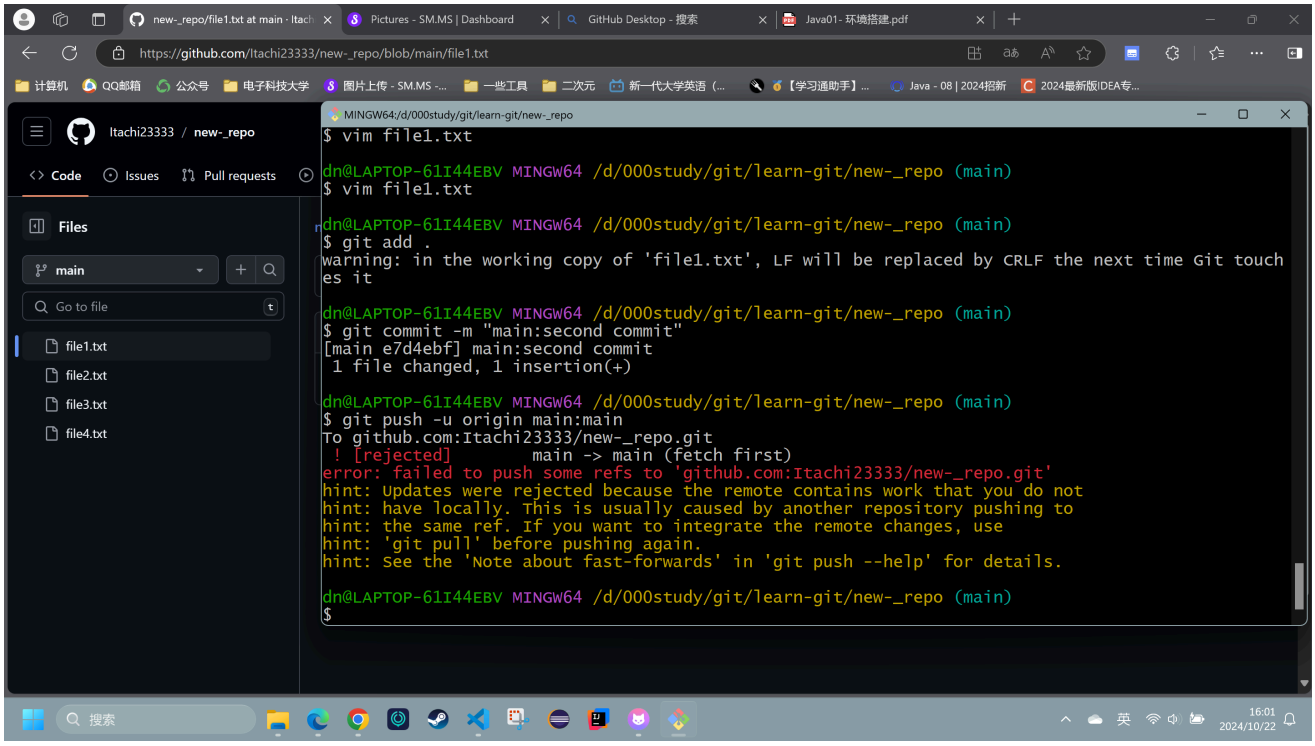


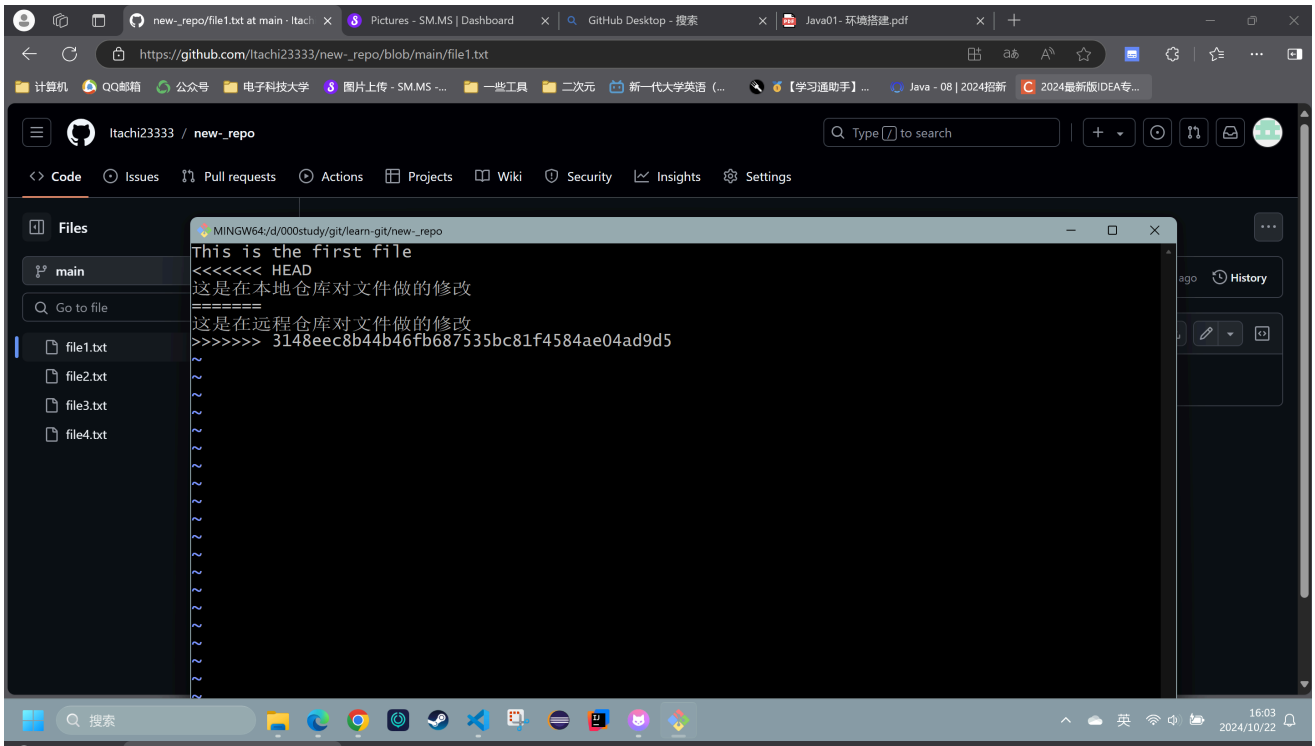

- 因为develop分支是作为一个长期存在的分支 以后开发其他功能仍然要在这里 要把hotfix上面的修改同时推送给main和develop 否则develop分支上的bug仍然存在 以后再使用时会出现意外

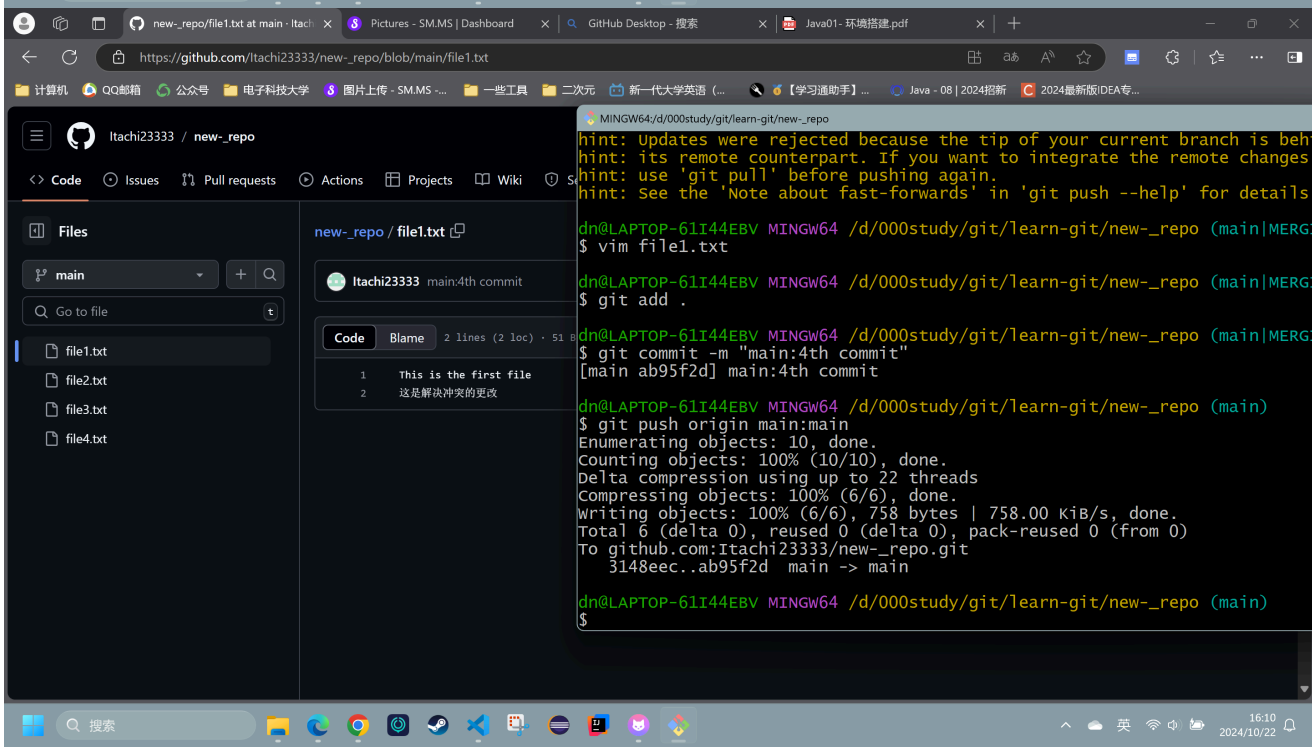





v. 这里相当于多人同时修改一个文件 就会出现冲突 手动解决后 需要add commit 最后push






vi. a. 首先创建一个远程仓库 团队成员把这个远程仓库clone到本地

b. 每个成员为自己要开发的功能创建一个新的分支 避免在main分支上直接更改

c. 注意定期提交

d. 在push之前 先pull远程仓库的最新更改 避免出现冲突

e. 需要一些成员创建一个PR 审核其他成员push中的更改 没问题后再合并到主分支中