

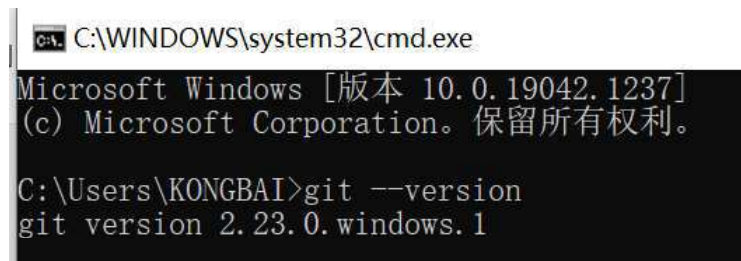
1 实验要求

- 了解配置管理工具 Git 及相应用环境;
- 熟练掌握 Git 的基本指令和分支管理指令;
- 掌握 Git 支持软件配置管理的核心机理;
- 在实践项目中使用 Gitee/GitLab/GitHub 管理自己的项目源代码。

2 安装 Git

2.1 本地机器上安装 Git

Git 版本号:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19042.1237]
(c) Microsoft Corporation。保留所有权利。

C:\Users\KONGBAI>git --version
git version 2.23.0.windows.1
```

Git 运行界面:



2.2 申请 Gitee/GitLab/GitHub 帐号

由于本人以往的开发中经常出现将项目的配置文件放在 GitHub 统一管理但是读取失败的问题, 所以本人这次实验采用的是 Gitee。

账号名称: kongbai

URL 地址:

Lab1 文件操作的仓库: https://gitee.com/edmund_lai/spl-binary-tree.git

分支操作练习仓库: https://gitee.com/edmund_lai/branch-practice.git

账号信息截图:



项目信息截图:



3 Git 操作过程

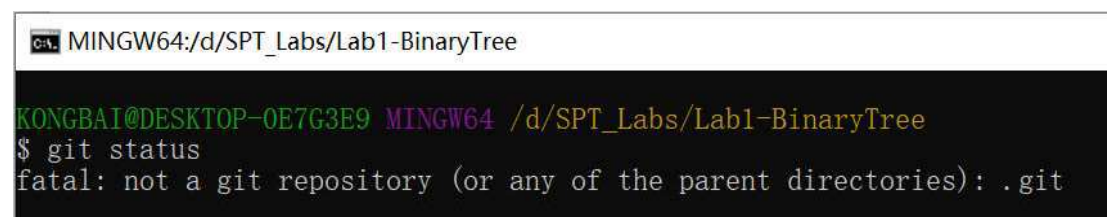
3.1 实验场景(1): 仓库创建与提交

R0: 针对 R1 和 R7, 在进行每次 Git 操作之前, 随时查看工作区、暂存区、Git 仓库的状态, 确认项目里的各文件当前处于什么状态。

指令: git status

说明: 这个指令在未进行本地仓库初始化前是无法生效的, 下面分别是本人在初始化前和初始化后执行该指令得到的结果。

初始化前:



初始化后:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    src/
    test/

nothing added to commit but untracked files present (use "git add" to track)
```

R1: 本地初始化一个 Git 仓库, 将自己在 Lab1 中所创建项目的全部源文件加入进去, 纳入 Git 管理。

指令: `git init`

执行命令:

```
MINGW64:/d/SPT_Labs/Lab1-BinaryTree

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree
$ git init
Initialized empty Git repository in D:/SPT_Labs/Lab1-BinaryTree/.git/

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$
```

执行结果: 在项目文件夹下生成一个名为.git 的隐藏文件夹, 需要将 Windows 资源管理器中的“显示隐藏的项目”勾选才能生效



R2: 提交

指令:

将文件加入暂存区: `git add 文件名`

将暂存区的文件提交到本地仓库: `git commit -m "当次提交的备注信息"`

执行结果:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git add .

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   src/cn/edu/hit/Main.java
    new file:   src/cn/edu/hit/binarytree/BinaryTree.java
    new file:   src/cn/edu/hit/binarytree/TreeNode.java
    new file:   src/cn/edu/hit/views/BinaryTreeWindow.java
    new file:   src/cn/edu/hit/views/DrawAreaPanel.java
    new file:   test/cn/edu/hit/binarytree/BinaryTreeTest.java
    new file:   test/cn/edu/hit/binarytree/TreeNodeTest.java

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git commit -m "first commit for lab1"
[master (root-commit) 2d75b36] first commit for lab1
 7 files changed, 832 insertions(+)
 create mode 100644 src/cn/edu/hit/Main.java
 create mode 100644 src/cn/edu/hit/binarytree/BinaryTree.java
 create mode 100644 src/cn/edu/hit/binarytree/TreeNode.java
 create mode 100644 src/cn/edu/hit/views/BinaryTreeWindow.java
 create mode 100644 src/cn/edu/hit/views/DrawAreaPanel.java
 create mode 100644 test/cn/edu/hit/binarytree/BinaryTreeTest.java
 create mode 100644 test/cn/edu/hit/binarytree/TreeNodeTest.java

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master
nothing to commit, working tree clean
```

R3: 手工对 Lab1 的某个文件进行修改, 查看上次提交之后都有哪些文件修改、具体修改内容是什么 (查看修改后的文件和暂存区域中相应文件的差别)。

首先手动对 Lab1 的文件进行修改, 这里为了不对原有的代码进行改动, 本人只在原来的 `TreeNode.java` 文件中添加了一行注释。

```

TreeNode.java - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

public Integer getDepth() {
    return depth;
}

public int getX() {
    return x;
}

public int getY() {
    return y;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    TreeNode treeNode = (TreeNode) o;
    return this.index.equals(treeNode.getIndex());
}

// modified for Lab2 R3 step
}

```

查看文件修改情况: `git status`

查看具体变动内容: `git diff`

执行结果:

```

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   src/cn/edu/hit/binarytree/TreeNode.java

no changes added to commit (use "git add" and/or "git commit -a")

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git diff
diff --git a/src/cn/edu/hit/binarytree/TreeNode.java b/src/cn/edu/hit/binarytree/TreeNode.java
index db10f2d..a016a90 100644
--- a/src/cn/edu/hit/binarytree/TreeNode.java
+++ b/src/cn/edu/hit/binarytree/TreeNode.java
@@ -84,4 +84,6 @@ public class TreeNode {
     return this.index.equals(treeNode.getIndex());
 }
+
+ // modified for Lab2 R3 step
+
}

```

可以发现使用 `git status` 指令后结果中将"TreeNode.java"文件被修改的结果标红了, 随后使用 `git diff` 指令, 结果中标记了"+ //modified for Lab2 R3 step"的字样, 说明前面的修改被 Git 工具发现。

R4: 重新提交

这一步的指令同 R2:

将文件加入暂存区: `git add 文件名`

将暂存区的文件提交到本地仓库: `git commit -m "当次提交的备注信息"`

执行结果:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git add .

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/cn/edu/hit/binarytree/TreeNode.java

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git commit -m "second commit for lab1"
[master 839d810] second commit for lab1
 1 file changed, 2 insertions(+)

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master
nothing to commit, working tree clean
```

R5: 再次对 Lab1 的某个文件进行修改, 重新提交

这一步的指令和前一步没有太大区别, 这里本人为了展示方便, 使用仍然使用"TreeNode.java"文件作为展示。

```
TreeNode.java - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

public Integer getDepth() {
    return depth;
}

public int getX() {
    return x;
}

public int getY() {
    return y;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    TreeNode treeNode = (TreeNode) o;
    return this.index.equals(treeNode.getIndex());
}

// modified for Lab2 R3 step
// second modified for Lab2 R5 step
}
```

使用的指令仍然是: `git add 文件名` 和 `git commit -m "提交备注信息"`, 执行结果如下:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git add .

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/cn/edu/hit/binarytree/TreeNode.java

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git commit -m "third commit for lab1"
[master 56f22bc] third commit for lab1
1 file changed, 1 insertion(+)

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git status
On branch master
nothing to commit, working tree clean
```

R6: 把最后一次提交撤销

指令: `git reset --hard HEAD^`

首先可以执行 `git reflog` 查看历史的提交版本, 执行结果如下:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git reflog_
56f22bc (HEAD -> master) HEAD@{0}: commit: third commit for lab1
839d810 HEAD@{1}: commit: second commit for lab1
2d75b36 HEAD@{2}: commit (initial): first commit for lab1
```

可以观察到包括初始化的提交在内一共提交了三次, 并且目前的 HEAD 指向的是最后一次提交。

现在执行撤销指令:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git reset --hard HEAD^
HEAD is now at 839d810 second commit for lab1
```

再次执行 `git reflog` 查看版本情况, 特别注意版本的序列号:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git reflog_
839d810 (HEAD -> master) HEAD@{0}: reset: moving to HEAD^
56f22bc HEAD@{1}: commit: third commit for lab1
839d810 (HEAD -> master) HEAD@{2}: commit: second commit for lab1
2d75b36 HEAD@{3}: commit (initial): first commit for lab1
```

观察到撤销操作后版本的序列号从"56f22bc"变成了"839d810", 这与第二次提交的序列号一致, 说明撤销操作成功。查看对应的文件, 发现第二次修改后的结果被撤回:

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    TreeNode treeNode = (TreeNode) o;
    return this.index.equals(treeNode.getIndex());
}

// modified for Lab2 R3 step 第二次修改的结果消失
}
```

R7: 查询提交记录

指令: `git log`

执行指令得到的结果:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git log_
commit 839d810df55f20d215ab1064e690b1978585a13b (HEAD -> master)
Author: Edmund_Lai <957536235@qq.com>
Date: Tue Oct 26 08:16:36 2021 +0800

    second commit for lab1

commit 2d75b365c849fdacc9004107e7eale2e4f2038c4
Author: Edmund_Lai <957536235@qq.com>
Date: Tue Oct 26 08:00:25 2021 +0800

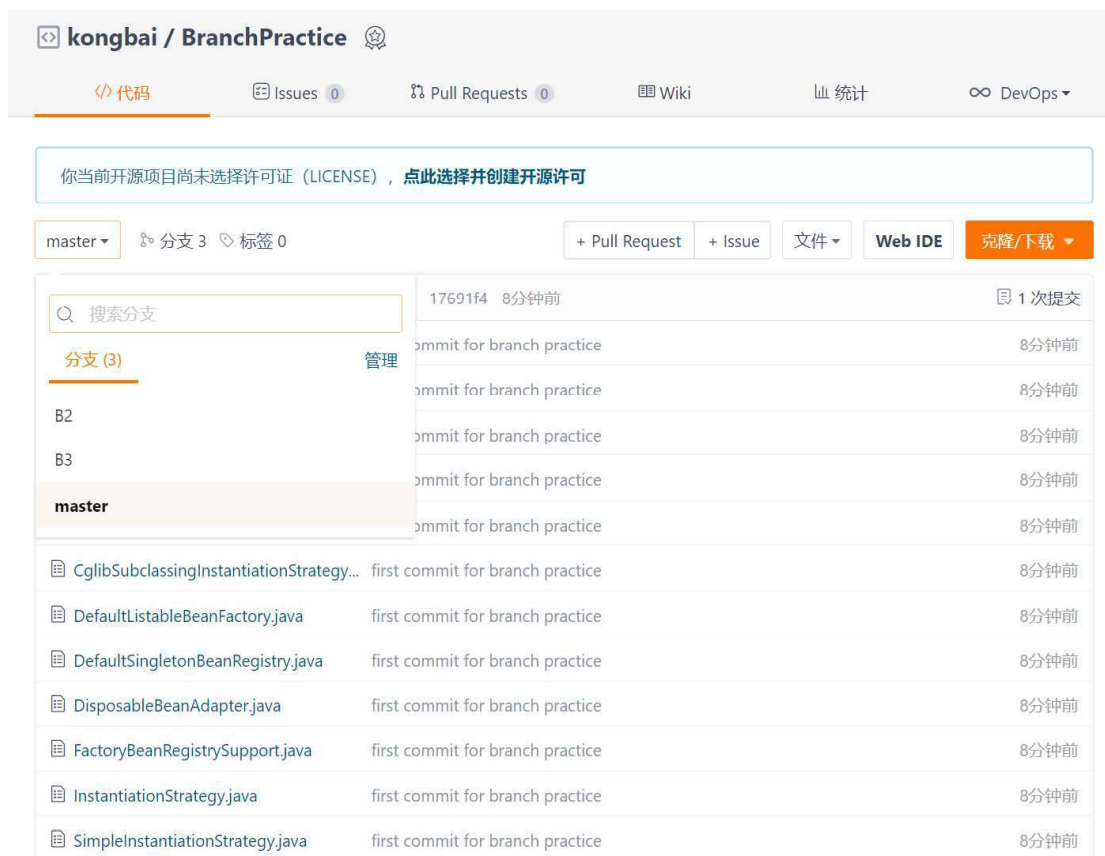
    first commit for lab1
```

可以观察到这种查询方式并不会显示撤销前的提交记录, 只会显示结果到当前 HEAD 指针指向的版本, 及该版本以前的所有提交记录。

3.2 实验场景(2): 分支管理

准备工作: 在你的 Gitee/GitLab/GitHub 上, 通过 Web 界面建立一个 Project, 将不少于 10 个文件 (程序代码、文档等) 加入进去, 形成初始分支 B1; 在 B1 基础上建立两个并行的分支 B2、B3, 手工对 B2 和 B3 上的某些文件进行不同程度的修改并提交。

完成上述的准备:



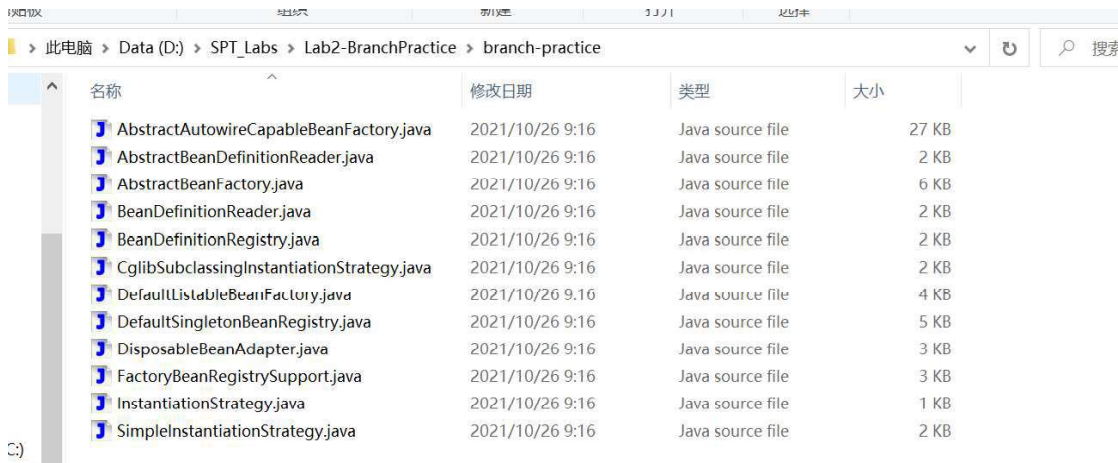
R8: 从 Gitee/GitLab/GitHub 上(URL)克隆一个已有的 Git 仓库到本地

指令: `git clone 目标 URL`

执行结果:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice (master)
$ git clone https://gitee.com/edmund_lai/branch-practice.git
Cloning into 'branch-practice'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (14/14), done.
Unpacking objects: 100% (14/14), done.
```

在执行后本地文件夹中出现所有上传到远程的文件:



R9: 获得该仓库的全部分支

指令: `git branch -a`

执行的结果如下:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (master)
$ git branch -a
* master
  remotes/origin/B2
  remotes/origin/B3
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

可以发现当前的分支是本地的 master 分支, 但是远程的 master、B2、B3 分支都已经可以在本地被查询到。

R10: 在 B2 分支基础上创建一个新分支 C4

这个操作首先需要从当前的 master 分支切换到 B2 分支, 再在这个分支的基础上创建一个新的分支。

切换到 B2 分支: `git checkout -b B2`

创建新的分支 C4 并切换: `git checkout -b C4`

执行结果如下:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B2)
$ git checkout -b C4
Switched to a new branch 'C4'

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (C4)
$
```

R11: 在 C4 上, 对 4 个文件进行修改并提交

为了方便检查, 本人直接按照 Windows 资源管理器的默认字典序对前 4 个文件进行修改。修改的方式是简单在文件中添加一行注释。



修改的结果使用 `git status` 指令查看状态:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (C4)
$ git status
On branch C4
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   AbstractAutowireCapableBeanFactory.java
    modified:   AbstractBeanDefinitionReader.java
    modified:   AbstractBeanFactory.java
    modified:   BeanDefinitionReader.java

no changes added to commit (use "git add" and/or "git commit -a")
```

说明上述的修改成功。

对于提交的过程, 使用的指令仍然是: `git add 文件名` 和 `git commit -m "提交备注信息"`, 执行这两条指令后得到如下的结果:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (C4)
$ git add .

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (C4)
$ git status
On branch C4
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   AbstractAutowireCapableBeanFactory.java
    modified:   AbstractBeanDefinitionReader.java
    modified:   AbstractBeanFactory.java
    modified:   BeanDefinitionReader.java

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (C4)
$ git commit -m "commit for C4 branch"
[C4 6b85d3c] commit for C4 branch
4 files changed, 10 insertions(+)

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (C4)
$ git status
On branch C4
nothing to commit, working tree clean
```

R12: 在 B3 分支上对同样的 4 个文件做不同修改并提交

这个要求和上一步几乎一致, 只是操作的分支不一致, 首先需要切换到 B3 分支, 使用指令 `git checkout B3`, 得到如下的结果:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (master)
$ git checkout B3
Switched to a new branch 'B3'
Branch 'B3' set up to track remote branch 'B3' from 'origin'.
```

从提示语句中可以知道已经成功切换到了从远程仓库克隆到本地的 B3 分支。现在对同样的四个文件进行修改, 这里本人仍然采用的是在文件中增加注释的方式。

在添加完注释后, 使用的指令仍然是: `git add 文件名` 和 `git commit -m "提交备注信息"`, 得到如下结果:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$ git add .

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$ git status
On branch B3
Your branch is up to date with 'origin/B3'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   AbstractAutowireCapableBeanFactory.java
    modified:   AbstractBeanDefinitionReader.java
    modified:   AbstractBeanFactory.java
    modified:   BeanDefinitionReader.java

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$ git commit -m "commit for B3 branch"
[B3 a71fa3e] commit for B3 branch
4 files changed, 10 insertions(+)

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$ git status
On branch B3
Your branch is ahead of 'origin/B3' by 1 commit.
  (use "git push" to publish your local commits)
```

R13: 将 C4 和 B3 分支合并, 若有冲突, 手工消解

合并分支指令：git merge 分支名

需要注意的是这个指令是将分支名指定的分支合并到当前所在的分支上，这里本人处在的分支是上一步的 B3，因此需要使用的指令是 git merge C4。执行后得到如下的结果：

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$ git merge C4
CONFLICT (add/add): Merge conflict in README.md
Auto-merging README.md
Auto-merging BeanDefinitionReader.java
CONFLICT (content): Merge conflict in BeanDefinitionReader.java
Auto-merging AbstractBeanFactory.java
CONFLICT (content): Merge conflict in AbstractBeanFactory.java
Auto-merging AbstractBeanDefinitionReader.java
CONFLICT (content): Merge conflict in AbstractBeanDefinitionReader.java
Auto-merging AbstractAutowireCapableBeanFactory.java
CONFLICT (content): Merge conflict in AbstractAutowireCapableBeanFactory.java
Automatic merge failed; fix conflicts and then commit the result.
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3 | MERGING)
$
```

Git 提示一共有 5 个文件产生冲突，分别是 README 文件以及之前对 C4、B3 分支分别做改动的 4 个文件，这个也是符合预期的。在产生冲突时，Git 的命令行后面会有类似这里"(B3 | MERGING)"的提示，下一步需要手动进行合并。

手动合并并没有对应的指令，需要开发者手动打开发生冲突的文件修改，这里以第一个文件" AbstractAutowireCapableBeanFactory .java"为例，打开这个文件后发现 Git 已经做了基本的编辑：

```
public abstract class AbstractAutowireCapableBeanFactory extends
<<<<<< HEAD
//这是为B3分支第1个文件添加的内容
=====
//C4分支上修改的第一个文件
>>>>>> C4
```

====提示符的上
下分别是所在分支的
内容和被合并分支的
内容

在手动处理冲突后得到如下的内容：

```
public abstract class AbstractAutowireCapableBeanFactory {

//这是为B3分支第1个文件添加的内容
//C4分支上修改的第一个文件
//现在手动处理了冲突合并的问题
```

对于剩下的 4 个文件做同样的操作，就完成了手动解决冲突的问题。

处理完冲突后提交的方式和一般提交文件到本地仓库的操作一致，使用的语句为：git add 文件名 和 git commit -m "提交备注信息"。需要注意的是一般在手动处理冲突时需要开发人员一次性处理完所有的冲突文件，也就是在 git commit 的时候一般不会指定特定的文件名。本人处理完冲突后的结果如下图所示：


```

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3|MERGING)
$ git add .
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3|MERGING)
$ git status
On branch B3
Your branch is ahead of 'origin/B3' by 1 commit.
(use "git push" to publish your local commits)

All conflicts fixed but you are still merging.
(use "git commit" to conclude merge)

Changes to be committed:
  modified:   AbstractAutowireCapableBeanFactory.java
  modified:   AbstractBeanDefinitionReader.java
  modified:   AbstractBeanFactory.java
  modified:   BeanDefinitionReader.java
  modified:   README.md

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3|MERGING)
$ git commit -m "B3 and C4 merging conflict solved"
[B3 3234326] B3 and C4 merging conflict solved

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$

```

可以发现解决冲突后命令行的提示尾部从"(B3 | MERGING)"变成了"(B3)"说明分支合并、冲突处理完毕。

R14: 查看目前哪些分支已经合并、哪些分支尚未合并。

查看已经合并的分支: `git branch --merged`

查看尚未合并的分支: `git branch --no-merged`

分别执行上述两条指令, 得到如下的结果:

```

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$ git branch --merged
  B2
* B3
  C4

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (B3)
$ git branch --no-merged
  master

```

R15: 将 C4 和 B3 合并后的分支删除, 将尚未合并的分支合并到一个新分支上, 分支名字为你的学号。

上述全部过程中一共有 4 个分支, 分别是 B1 (master) 分支、B2 分支、B3 分支、C4 分支。这一步删除了 B3 和 C4 合并后的分支, 因此需要合并的分支是 B1 和 B2 分支。

删除分支的指令: `git branch -d 分支名`

强制删除分支的指令: `git branch -D 分支名`

执行后得到如下的结果:

```

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (master)
$ git branch -D B3
Deleted branch B3 (was 3234326).

```

仍然使用分支合并指令: `git merge 分支名`, 结果如下:

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (master)
$ git merge B2
CONFLICT (add/add): Merge conflict in README.md
Auto-merging README.md
Automatic merge failed; fix conflicts and then commit the result.
```

在处理完冲突后，对当前的分支改名。

分支改名指令：`git branch -m 旧名称 新名称`

执行上述指令的结果：

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (master)
$ git branch -m master 1191000311

KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (1191000311)
$
```

发现修改完后命令行的末尾变成了"(1191000311)"，说明分支名修改成功。

3.3 实验场景(3)：远程分支管理

R16: 将本地以你的学号命名的分支推送到 Gitee/GitLab/GitHub 上

首先需要将远程仓库取一个本地的别名，否则会比较麻烦，使用的指令为 `git remote add origin 远程仓库 URL`。

在执行完上述操作后可以将本地代码推送至远程仓库。

使用指令：`git push -u 远程仓库别名 推送分支名`

执行结果如下：

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab2-BranchPractice/branch-practice (1191000311)
$ git push -u origin 1191000311
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 362 bytes | 362.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Powered by GITEE.COM [GNK-6.1]
remote: Create a pull request for '1191000311' on Gitee by visiting:
remote:   https://gitee.com/edmund_lai/branch-practice/pull/new/edmund_lai:1191000311...edmund_lai:master
To https://gitee.com/edmund_lai/branch-practice.git
 * [new branch]      1191000311 -> 1191000311
Branch '1191000311' set up to track remote branch '1191000311' from 'origin'.
```

R17: 将 R1 到 R7 各步骤得到的结果推送到 Gitee/GitLab/GitHub 上

这个操作针对的是 R1-R7 步骤中对 Lab1 项目的改动，使用的指令和上一步一样，即 `git push -u 远程仓库别名 推送分支名`，只是注意需要将当前路径切换到 Lab1 项目的路径下。执行结果如下：

```
KONGBAI@DESKTOP-0E7G3E9 MINGW64 /d/SPT_Labs/Lab1-BinaryTree (master)
$ git push -u origin master
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 8 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (28/28), 9.77 KiB | 1.95 MiB/s, done.
Total 28 (delta 2), reused 0 (delta 0)
remote: Powered by GITEE.COM [GNK-6.1]
To https://gitee.com/edmund_lai/spl-binary-tree.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

R18: 在 Gitee/GitLab/GitHub 上以 Web 页面的方式查看你的两个仓库的当前

状态。

对于分支练习的仓库，其状态如下：

你当前开源项目尚未选择许可证 (LICENSE)，[点此选择并创建开源许可](#)

1191000311 分支 4 标签 0 + Pull Request + Issue 文件 Web IDE 克隆/下载

搜索分支

分支 (4) 管理

- 1191000311
- B3
- B2
- master

15分钟前 4次提交

commit for branch practice 6小时前

commit for branch practice 6小时前

commit for branch practice 6小时前

commit for branch practice 6小时前

commit for branch practice 6小时前

commit for branch practice 6小时前

commit for branch practice 6小时前

可以发现提交的以本人学号命名的仓库已经可以在分支中找到，并且原来在本地对各种分支的操作并不会影响远程仓库的状态。

对于 Lab1 所保存的仓库，其状态如下：

你当前开源项目尚未选择许可证 (LICENSE)，[点此选择并创建开源许可](#)

master 分支 1 标签 0 + Pull Request + Issue 文件 Web IDE 克隆/下载

Edmund_Lai second commit for lab1 839d810 7小时前 2次提交

src/cn/edu/hit second commit for lab1 7小时前

test/cn/edu/hit/binarytree first commit for lab1 7小时前

```

79  @Override
80  public boolean equals(Object o) {
81      if (this == o) return true;
82      if (o == null || getClass() != o.getClass()) return false;
83      TreeNode treeNode = (TreeNode) o;
84      return this.index.equals(treeNode.getIndex());
85  }
86
87  // modified for Lab2 R3 step
88
89  }

```

可以发现在 R1-R7 步中对 Lab1 源文件的修改已经被推送到远程仓库中了。