Lab - 4: Version Control

Team Members:

Jay Sanjaybhai Patel (jy451478@dal.ca)

Kenil Kevadiya (kn486501@dal.ca)

Date:

February 08, 2024

Subject:

Software Development Concepts

Professor:

Michael McAllister

Part 1 - History

1. How many constants were added between versions 4 and 5?

→ We can check it with the "**git diff ab8277e fda09b3**" (V4 and V5) command. A total of 4 constants were added between version 4 and 5.

- private static final int BOLD = 3;
- private static final int UNDERLINE = 4;
- private static final int ITALICIZE = 5;
- private static final int LISTITEM = 6;

2. In which version was the method "main" separated into its own file?

→ In Version 4, the main method has been separated into another Java file. Until Version 3, it resided within the class named "htmlTranslator"; however, it has now been moved to a separate class named "convert".

→ When we use the "**git log --oneline**" command, it displays all commits along with their messages. In Version 4, the commit message was "Split main() from the rest of the code. Include code to split the body into paragraphs/lists".

→ After that, we executed "**git diff 2944661 ab8277e**" (V3 and V4) to observe the changes, revealing the creation of a new class containing the main method.

3. How many versions of the code have been checked in? How can you tell?

→ **Five versions** were checked in total. We checked them using the "**git log**" command.

4. In which version did the external documentation file appear in the repository?

→ For this we will go through each commit using **"git diff"** command and we see that in the 3rd version external documentation file was added.

- **git diff 98011b1 feec348 :** Some changes in htmlTranslator.java
- **git diff feec348 2944661 :** External_documentation.txt was introduced and again some modifications happened in htmlTranslator.

→ Another way, to find out in which version the external documentation file appear in the repository is;

- **git log --follow External_documentation.txt**

Part 3 – Branch

5. Which imports were added between versions 9 and 10?

→ We used the **“git diff 772bd8d 128550c”** command to find the number of imports between version 9 and 10. A total of 2 new imports were added.

- **import java.util.HashMap;**
- **import java.util.HashSet;**

Broadening Questions

1. When working alone on a project, how frequently should you commit your code to a version control system? Explain why.

→ When I work on my personal project, I begin by implementing the brute force approach and ensure that it produces correct outputs for appropriate inputs. Then, I push this brute force version as version 1 and proceed to optimize each functionality individually.

→ This guarantees that each commit is meaningful and represents a complete, thoroughly tested section of work.

→ After optimization, I push the next version. This method allows me to manage the development path of my personal projects effectively. If I make a mistake during the optimization process, I can always revert to the brute force version.

→ Furthermore, committing after completing a chunk of work encourages excellent coding standards, which improves overall project quality.

2. When working in a team, how frequently should you commit your code to a version control system? Explain why.

→ When we work in a team, committing code depends on many factors such as size of team, number of features, project complexity and many more factors.

→ Committing frequently ensures that team members can easily collaborate on the same codebase.

→ Changes that rely on the work of others or affect their work should be committed as soon as possible to ensure smooth integration and clarity for team members.

→ Version control systems serve as a backup mechanism for your code. By committing frequently, you're essentially backing up your work regularly. This minimizes the risk of losing code due to system failures.

→ Regular commits keep the project's progress moving forward and team members aware of changes in the code. Ensure that each commit is meaningful, well-tested, and does not disturb the work of others, hence promoting project stability and collaboration.

→ Each commit message serves as documentation for the changes made. Regular commits with specific messages provide a clear record of the development cycle, making it easier for team members to understand the development of the codebase over time.

3. Why might you create branches for your project in your version control system?

→ Branches allow developers to work on new features or changes to the codebase without affecting the main or production code. And while working in a team, we have to divide work among individuals, with each person working on different functionalities.

→ Consequently, each team member makes their own copy of the software from the remote repository to their local environment to begin developing their respective features. Therefore, Branches facilitate collaboration among team members by enabling multiple developers to work on different features or fixes concurrently.

→ If they work directly on the main branch and something goes wrong, the entire codebase could become corrupted. Moreover, if someone accidentally deletes the entire database, it could be disastrous.

→ Therefore, it is necessary for each team member to work on their own copy and then submit merge requests when ready.

→ Moreover, branches allow the management of multiple versions of the software, and it simplifies development efforts and ensures effective project collaboration.