**CS2820 Object Oriented Group Project Design Document**

Daniel Conway, Alex Pang, Seojin Ko

**Use Case Diagram:**





**Textual Descriptions:**

Commit:

Java program commit fileName VersionName (optional)NameOfAlternateBranchToAddTo(Must be terminal? Otherwise it’s a branch)

Requirements: Commit will allow the user to save their file to our revision control system with the option of choosing a side branch to add to instead of the main branch.

The user will use the command in the format shown above to send the required information to the program. Then, our program will store the differences between the latest file and the current local file at the end of the main branch. If a specific branch is given, it will then store the differences between the latest file of that branch and the current file in a new version at the end of that branch. Some sort of message is always stored with the version in the system, so it is required to type in a message before the version is saved. If there is not past version stored, a new “tree” is created for the file. Finally, a short summary of the version status being added is printed.

Checkout:

Java pragram checkout fileName optionalVersion:

User calls checkout command as formatted above with filName and an optional version argument. System searches DAG for file(node) that matches name and returns latest version if no version specified. Returns verion requested if version specified. To return a specific version, checkout starts at the root node, original file, and traverses the DAG, combining all of the diffs with the original file up to the designated checkout node. System downloads version to user’s local hard drive. User can make revisions and save to local hard drive. Ends when user commits or branches file

Branch:

Java program branch fileName branchFrom newVersion

The user creates a new branch following above command format. (a copy or snapshot of the repository) and it should be added to revision tree. (If it is not valid command, send error message. If it is valid command format, create a new branch and print summary.)

When it comes to rename, we check whether the name is already existing symbol or not. If it is existing symbol, send error message and if it is not, rename the branch and print summary.

Merge:

Java program merge fileName version1 version2

Requirements: User should be able to select nodes in different branches and merge the first branch into the second branch for a given file.

The user will first give the command as formatted above to the command line. Then, the system will take version1 have the next version be the version2 and add the diff of version1 and version2 to version2.

**XML Template**

<?xml version="1.0"?>

<FileName>

<Version1>

<Branch>diff stuff</Branch>

</Version1>

<Version2>

<Branch>diff stuff</Branch>

</Version2>

<Version3>

<Branch>diff stuff</Branch>

</Version3>

</FileName>

We plan to use this format to keep track of differences between versions. Diff and merge use this as described in the diagrams and textual descriptions. After getting the diff between versions, we can then store diffs by creating a new version with the branch’s diffs are stored from with the diff information stored inside of that. By doing this we can parse for the version and get the diff we want easily. Files are generated by applying (patching) diffs consecutively down the tree to the version the user wanted.