**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 Version:

User calls checkout command as formatted above with filename. Returns version 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

A branch is created when commit is used on a version that already has a child.

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 version="1.0"?>

<FileName name="a.txt">

<current Version="1.1"></current>

<Node Version ="1.1">

<BrName>null</BrName>

<Message>null</Message>

<Parent>null</Parent>

<isBranch>F</isBranch>

<Children>

</Children>

</Node>

</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.