GitLit Design Document

Deadlines:
Design document worth 4 points is due in-lab Monday, July 9.
Project due Saturday, July 21

Compose a design document that specifies all the classes in your design. For each class:

- Include a name.
- Give a natural language descriptions of the class's responsibilities.
- Enumerate the data and functionality it will encapsulate.
- For each kind of data, specify in natural language the exact data type.
- For each kind of functionality, specify in natural language the inputs and outputs and their exact data types and the properties of the intended output.

In addition to details about individual classes, write a natural language description of how the classes interact, describing their connectors. Are they communicating with events, function calls, timers, or other kinds of mechanisms? If it helps clarity, create diagrams to specify these interactions.

Remember that the purpose of this document you're writing is not to satisfy the instructors. It's intended to help you organize, plan, and streamline your Gitlet implementation, so focus on writing a document that's useful to you.

<u>Classes</u>
Name:Tree
Abstract data type would be list
Commit class within tree
Prev and Next nodes are pointed to by each commit
Each Commit object stores metadata and pointers to necessary blobs
Pointer named head pointing to the current branch
Branch objects (similar to sentinel node) that point to beginning and end of branch
(allowing for merge functionality)
Head pointer points to current branch

Name: Blobs
Individual Blobs:
Each blob will have text stored as String
Will be Snapshot of the actual file

BlobMab:
Blobs are organized by a new map for each file added. Each term in the map points to the blob with an updated version of the file
Key of each element will be its SHA 1 Hash

Order of BlobMap entries will be tracked

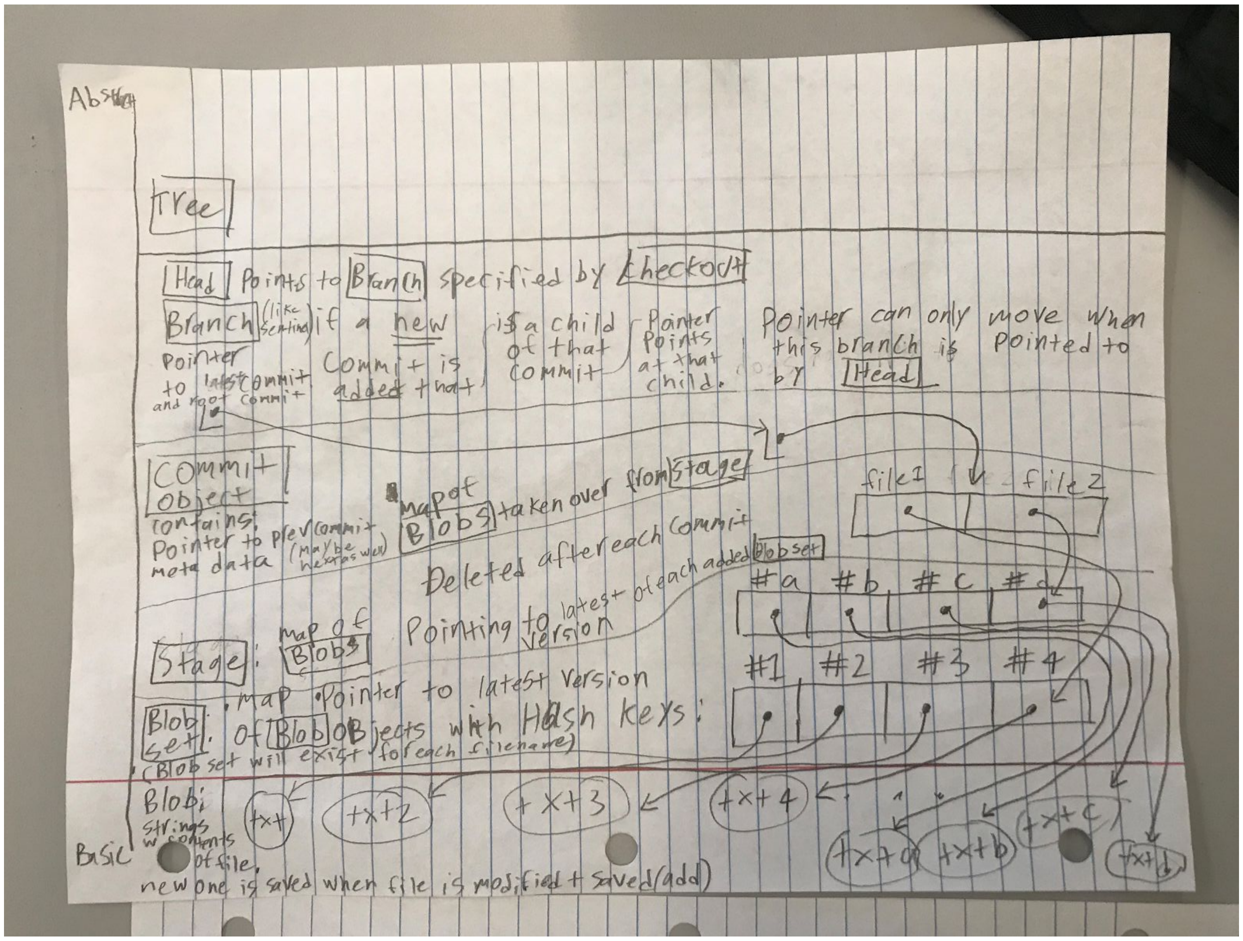
Overview:
Will have a constructor where the input will be text name and inside there is a SHA 1 hash code generator and the name will be the generator code
Organized by an array where each file gets an array and a blob in the array

Name: Commits
The data structure would be similar to DLLists where commits are nodes and we can have commit type variables that connects them to parent commits and to later commits
It will have a variable name of type String that is the user's inputs name (and other metadata)
Will have variables of type blob where they point to the blobs that the stage area is pointing to

Name: Branch
Likely to be nested in Tree class
Branches are as a pointers to commits
Double pointer similar to sentinel that will track the beginning and end of a branch
Current branch pointed to by head

Class: Staging area
Place where you add all the files to commit
Access it when committing
List where each term will point to the latest version of the file the user inputs (i.e the latest member of that file's BlobMap)

Visual representation of our draft design


Abstract
Tree
Head points to Branch specified by checkout
Branch (like setting) pointer to latest commit and root commit
if a new commit is added that is a child of that commit pointer points at that child.
Pointer can only move when this branch is pointed to by Head
Commit object contains: pointer to prev commit, meta data (maybe hash as well)
map of Blobs taken over from Stage
Deleted after each commit
Stage: map of Blobs pointing to latest of each added Blob set version
Blob set: map of Blob objects with hash keys; pointer to latest version
(Blob set will exist for each filename)
Blob: strings w contents of file. new one is saved when file is modified + saved(add)
file1
file2
#a #b #c #d
#1 #2 #3 #4
txt
txt2
txt3
txt4
txta
txtb
txtc
txtd
Basic

Delegation (timeline):

7/12 - 7/18 *Eli/Alec Work from 11AM - 5PM *Won/Soungbae can come at 3
(work on everything together)
First Gradescope submit (7/18)
Fix code/bugs (7/19 - 7/21)
DUE 7/21

The Commands:
*line counts
FINISH BY

Init (20) - **7/12**
Add (20) - **7/12**
Commit (40) - **7/13**

Rm (5) - **7/14**
Log (5) - **7/14**
Global-log (5) - **7/14**
Find (5) - **7/14**
Status (15) - **7/15**
Checkout (20/20/20)- **7/16**
Branch (5) - **7/17**
RM-branch (5) - **7/17**
Merge (70) - **7/18**