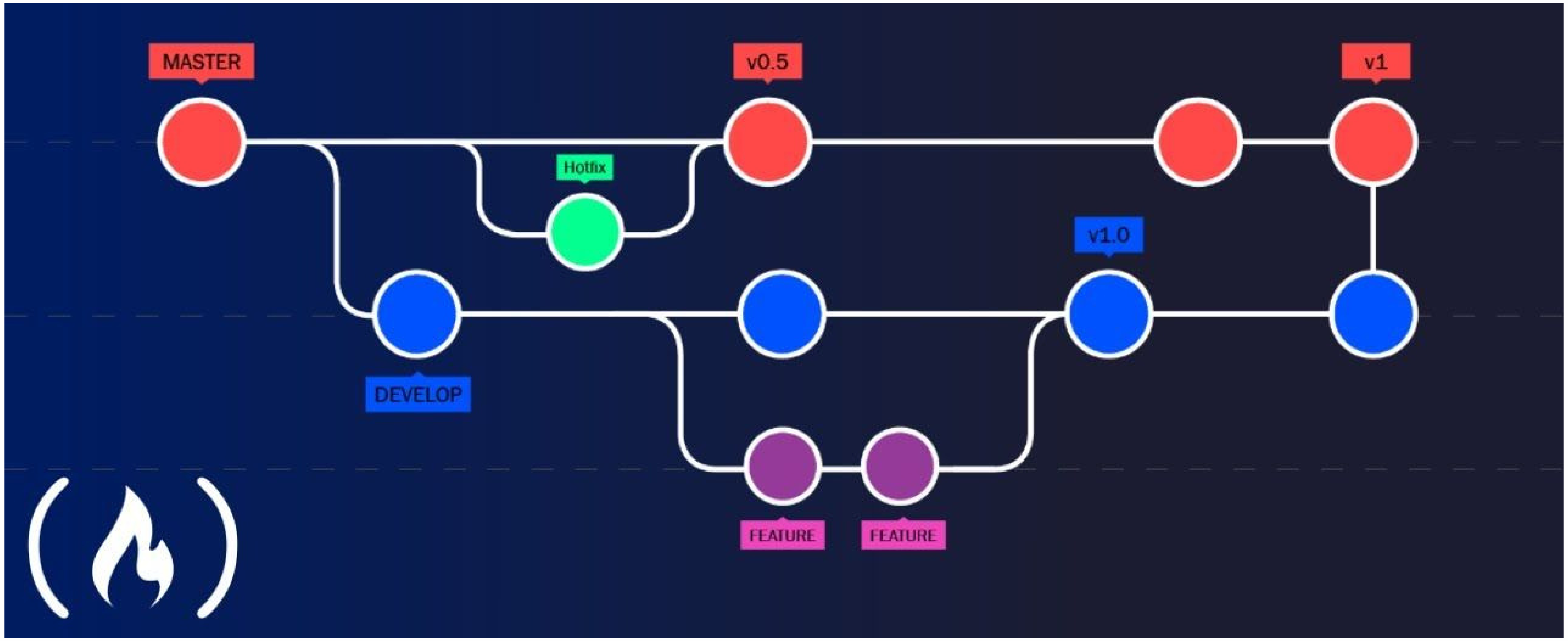
Senay Tilahun

5004 FINAL PROJECT

GITLET

A version control system



Agenda

1. Overview of Gitlet
2. Design Classes/Data Structures
3. Internal Structures
4. Commands: now and future
5. Demo
6. Reflections & Lessons Learned
7. OOP Ideas used
8. Q & A
9. Citations

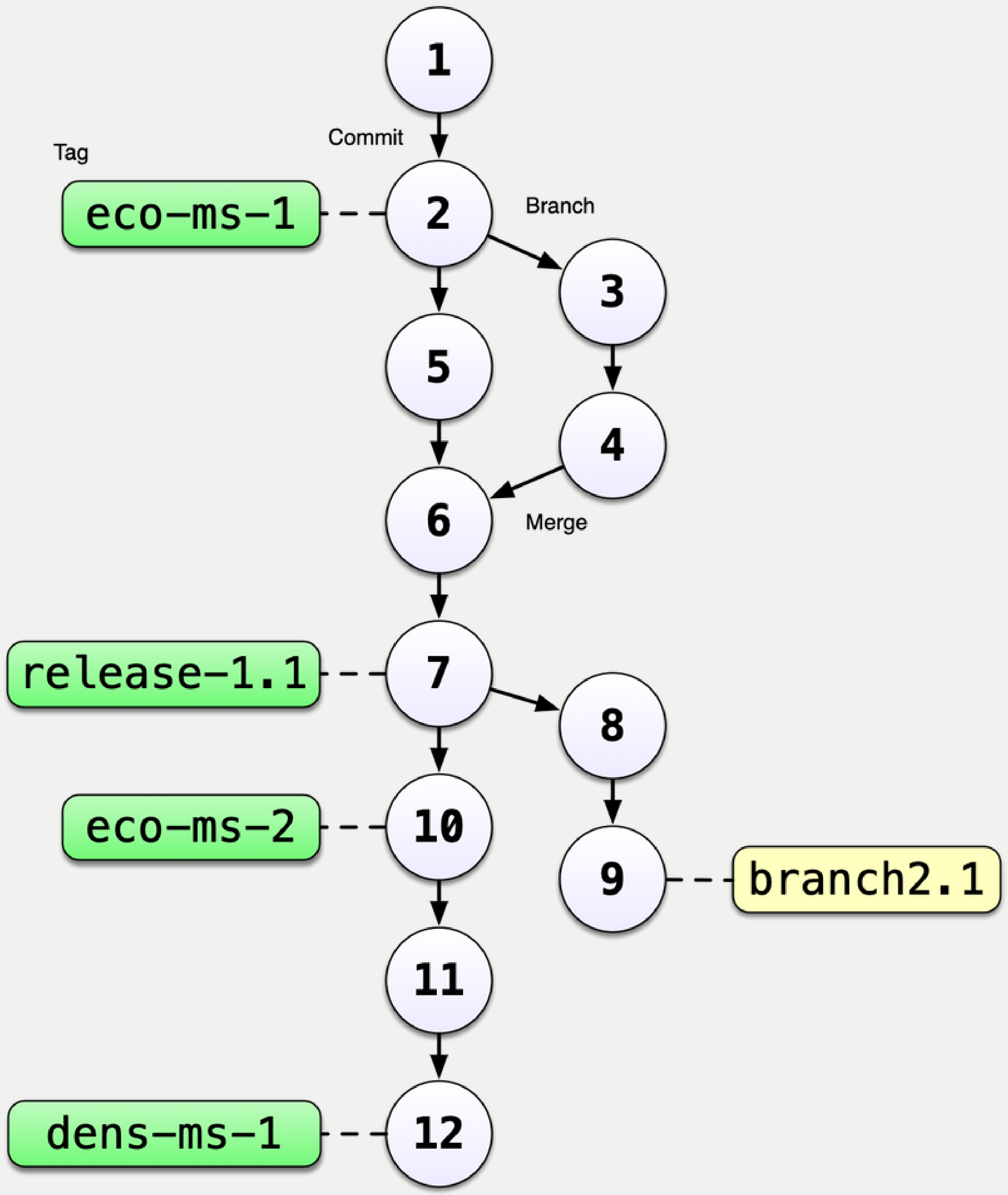
Implementing a version control system (VCS) that mimics the basic features of Git

Functionality -

- Commits - saving contents of entire directories of files
- Checking out - restoring versions of commits
- log - viewing history of backupis
- add/rm - staging files for tracking and updates

Key idea -

- keep track of the different versions of complicated projects/collaborations
- save a coherent set of files at the same time
- each commit is a snapshot of the entire project at one time



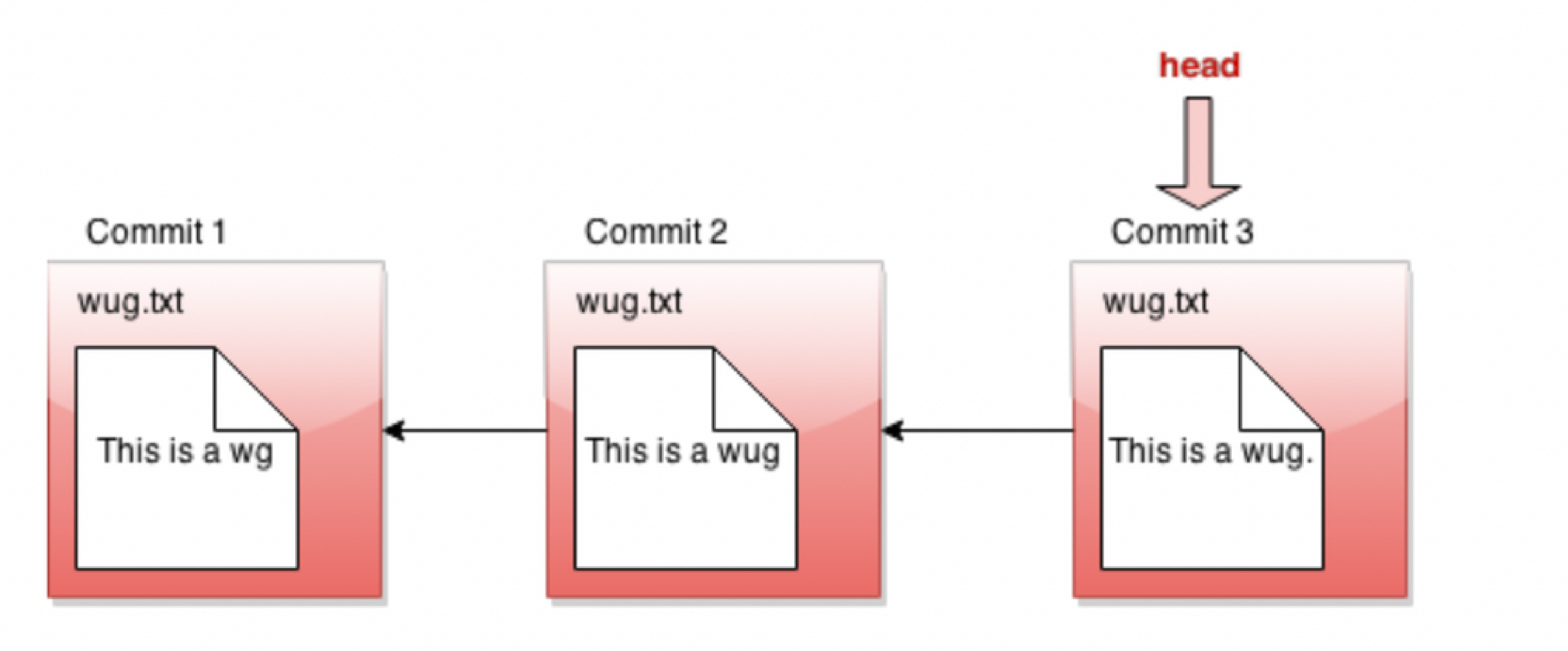
COMMIT TREE

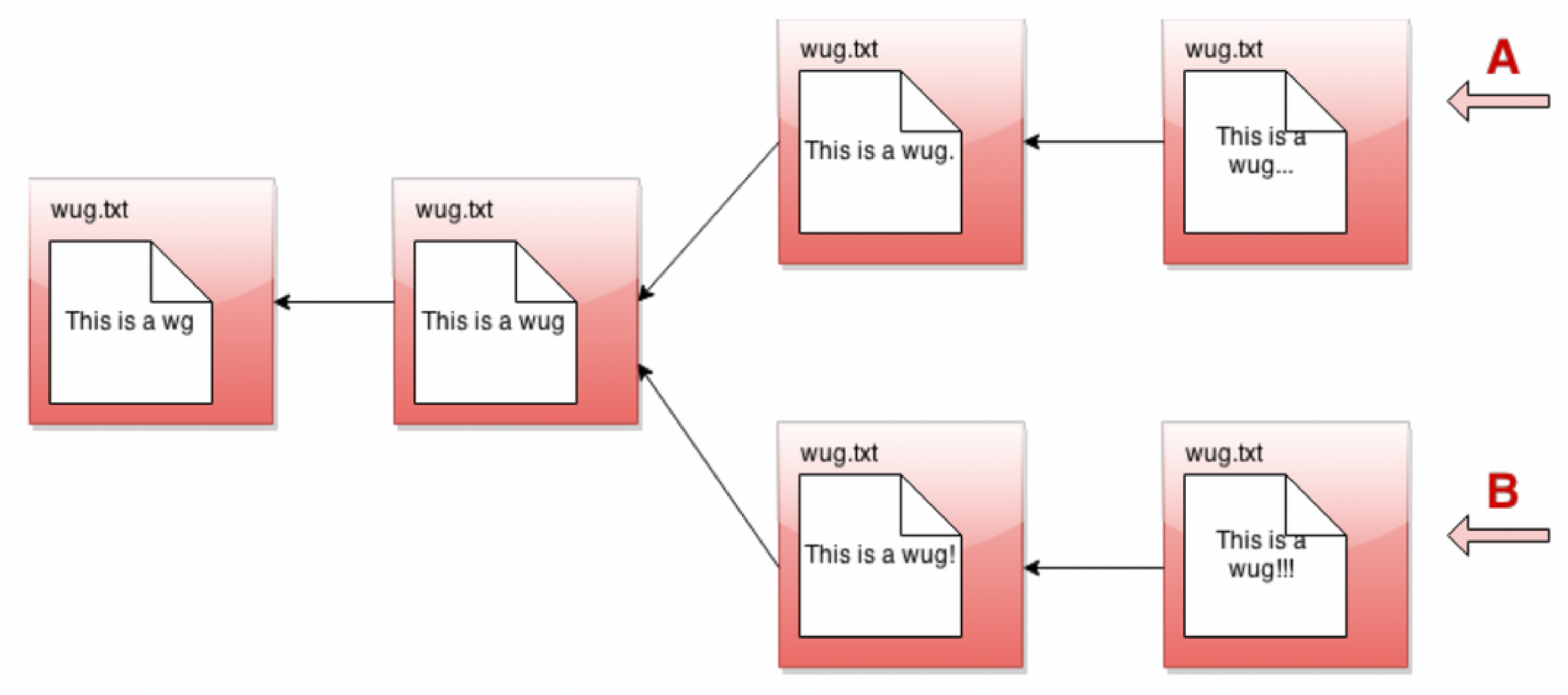
Commit tree -

- when we create a new commit, it is linked to its parent and the head pointer is updated
- if we explore different changes at the same time - branch, we can explore those
- we will have one active branch/pointer - the head pointer

A commit tree is immutable

- Once a commit node has been created, it can never be destroyed or changed at all
- We only add things to the commit tree
- We can't modify existing commits, only add new ones





Design

Classes

Main ->

- Gets user arguments validates, parses, and executes the correct command

AbstractUserCommand->

- Abstract Class for identifying user commands (more for future use)

UserCommand ->

- Gets the user command and validates correct arguments

GitLet ->

- class (Model), and has methods to handle the execution of the main commands

GitletObjects->

- handles functionality for two main objects in gitlet; blobs and commits

GitFileManage -> has methods for file management/directories/files

- GitFileCommand-> file management for the Gitlet class
- GitFIleObjects-> file management for GitletObjects class

Utility Classes -> These are utility classes to help with the main functionality

- Utility -> has various utility methods
- Valid -> Validation class that I use to validate user input

Data Structures

LinkedList -> to store the parent commits
HashMap -> to store filenames to Ids
Objects ->

- blobs ->
  - content of the files
- commits ->
  - log message
  - timestamp
  - files changed
  - id

Internal Structures

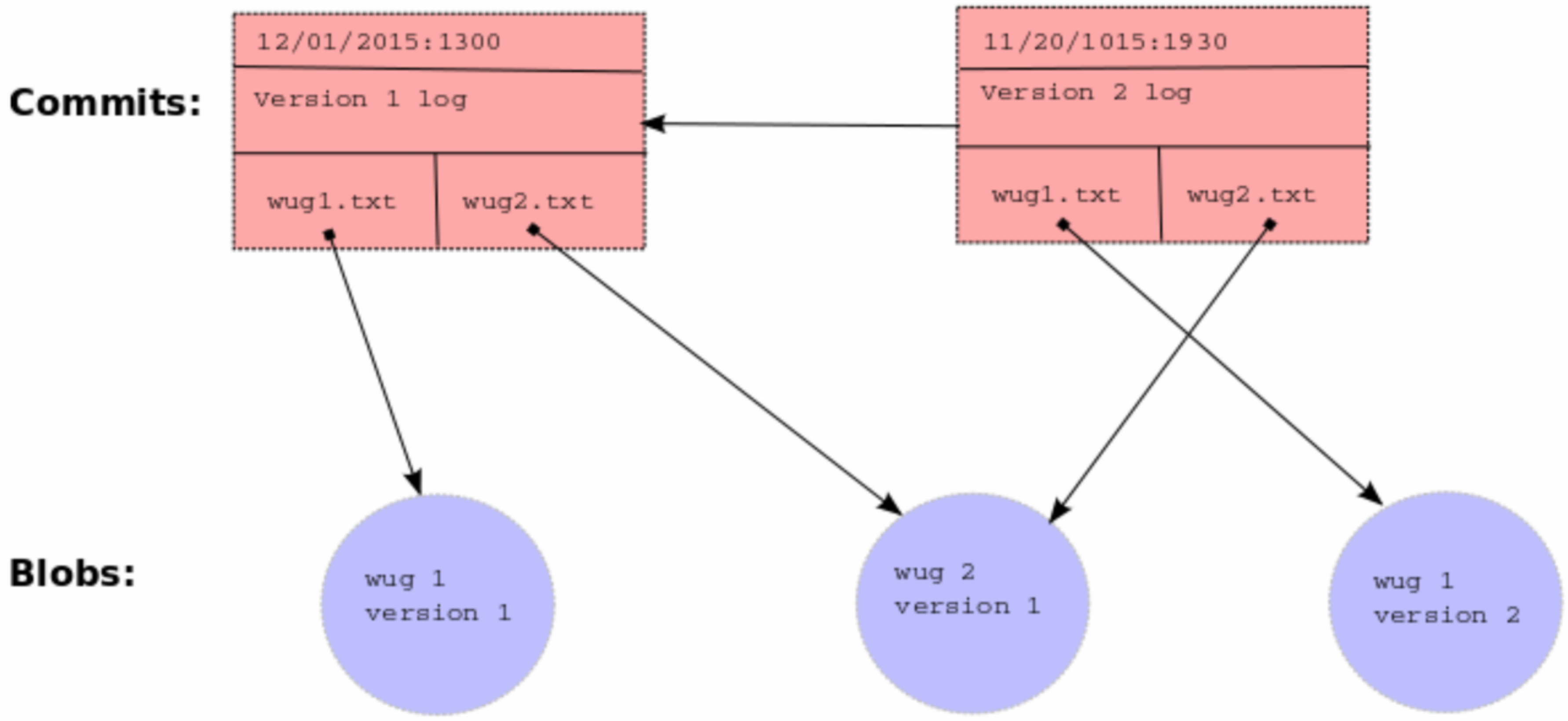
Simplified git

blobs - contents of our files

Commit - commit tree incorporated into commits - for gitlet we will have one flat directory of plain files for each repository

Commit metadata - timestamp and log message
Commit instance variables

- log message, timestamp,
- mapping of file names to blob references
- parent reference



Gitlet Commands

GITLET COMMANDS IMPLEMENTED - BASIC FEATURES OF GIT

- init
- add
- commit
- log
- V2 - improvements

  checkout
  branch
  rm-branch
  merge
  reset
  status
  rm

Code Examples

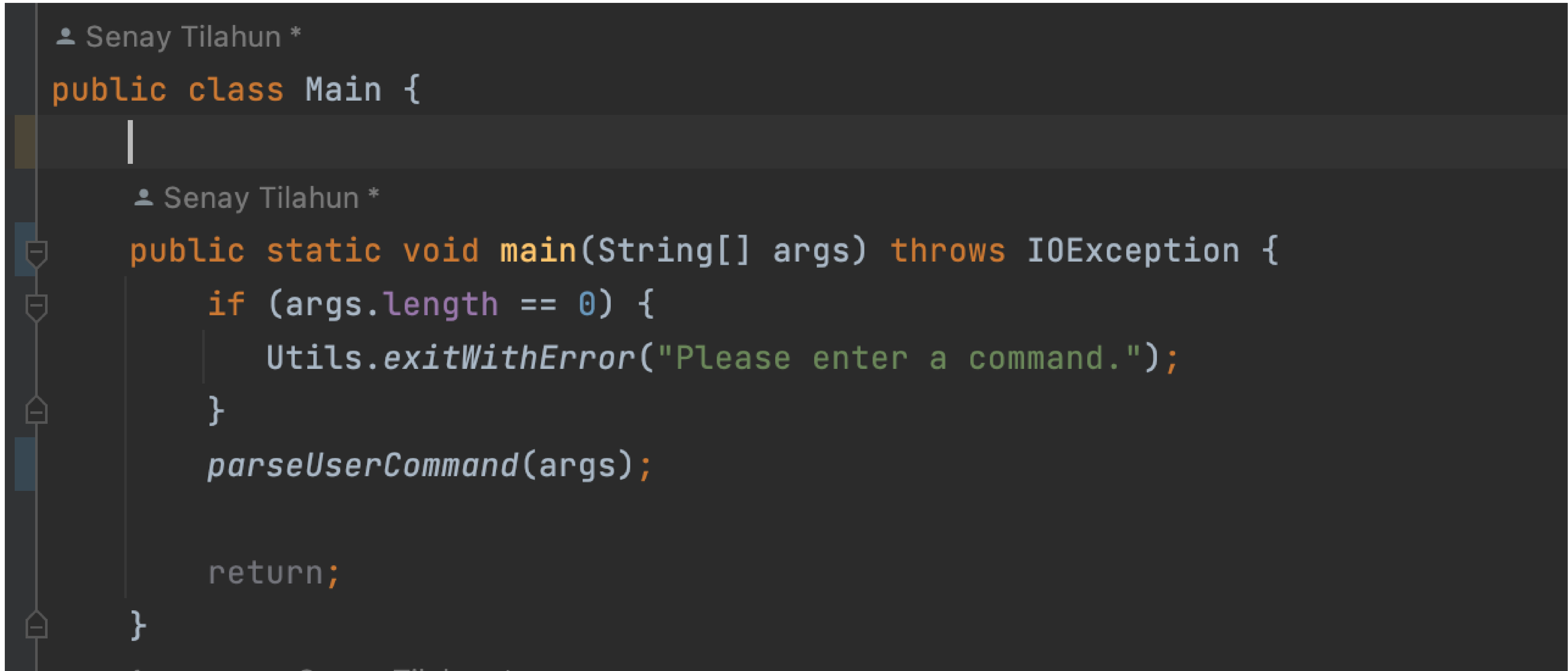
Main class

Used to get arguments from command line from user

Parse user input

Validate

Execute proper command

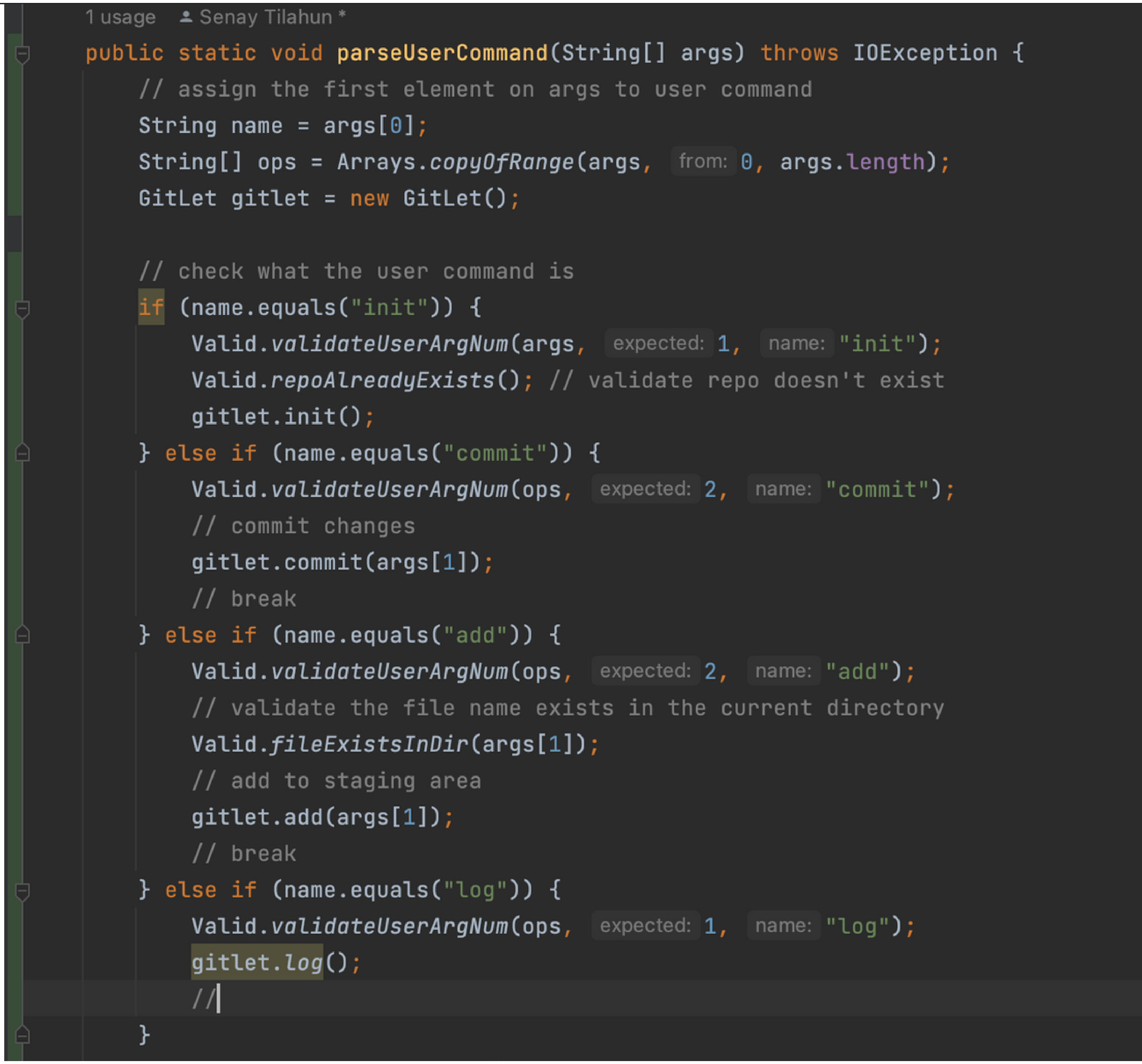


```java
public class Main {

    public static void main(String[] args) throws IOException {
        if (args.length == 0) {
            Utils.exitWithError("Please enter a command.");
        }
        parseUserCommand(args);

        return;
    }
```

parseUser Command

Method for parsing

- Gets the command from user input
- Validates the number of arguments
- Validates command specific checks
- Execute correct commands

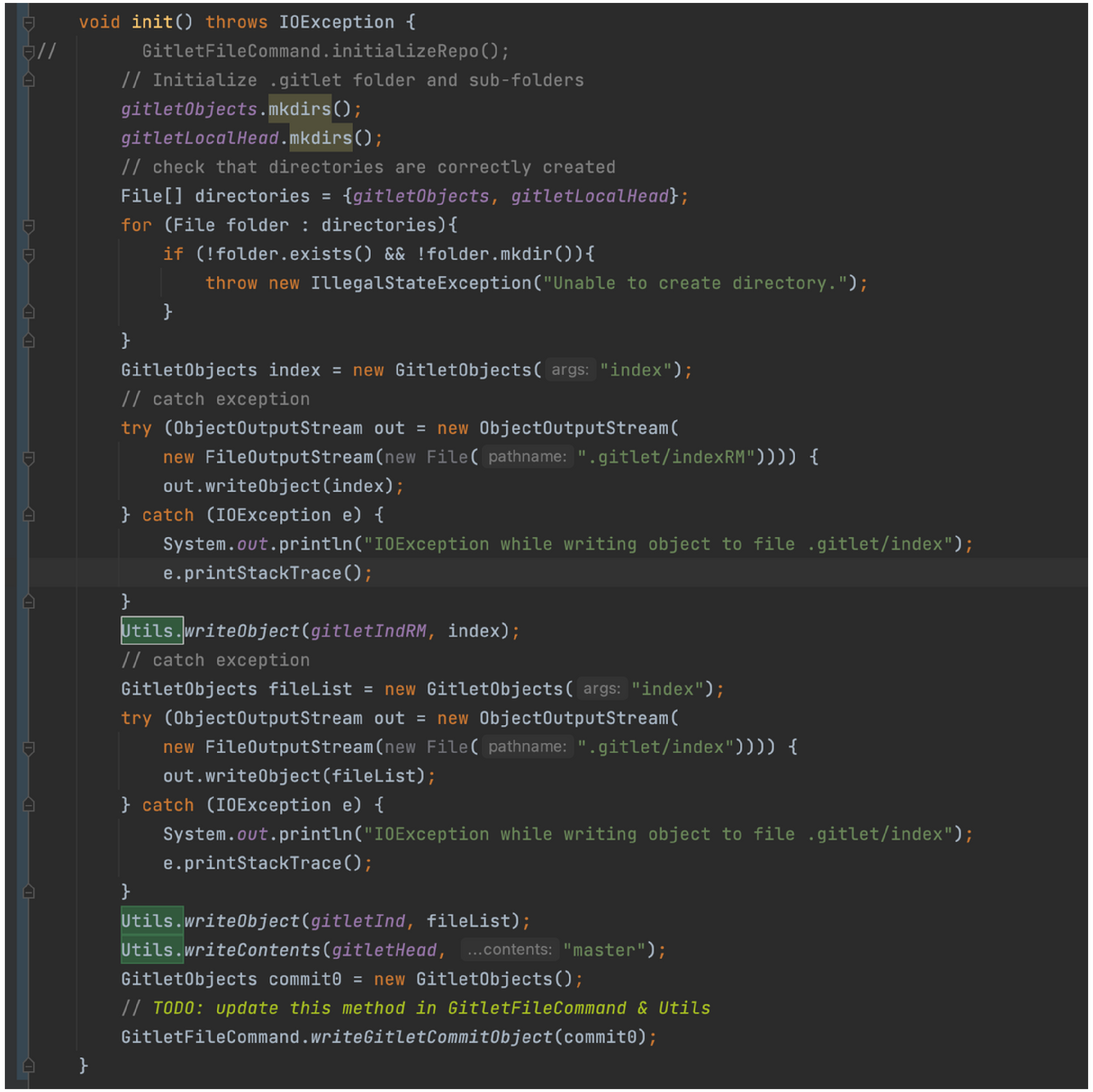


```
1 usage  Senay Tilahun *
public static void parseUserCommand(String[] args) throws IOException {
    // assign the first element on args to user command
    String name = args[0];
    String[] ops = Arrays.copyOfRange(args, from: 0, args.length);
    GitLet gitlet = new GitLet();

    // check what the user command is
    if (name.equals("init")) {
        Valid.validateUserArgNum(args, expected: 1, name: "init");
        Valid.repoAlreadyExists(); // validate repo doesn't exist
        gitlet.init();
    } else if (name.equals("commit")) {
        Valid.validateUserArgNum(ops, expected: 2, name: "commit");
        // commit changes
        gitlet.commit(args[1]);
        // break
    } else if (name.equals("add")) {
        Valid.validateUserArgNum(ops, expected: 2, name: "add");
        // validate the file name exists in the current directory
        Valid.fileExistsInDir(args[1]);
        // add to staging area
        gitlet.add(args[1]);
        // break
    } else if (name.equals("log")) {
        Valid.validateUserArgNum(ops, expected: 1, name: "log");
        gitlet.log();
        //
    }
```

Gitlet - init

Initialize local

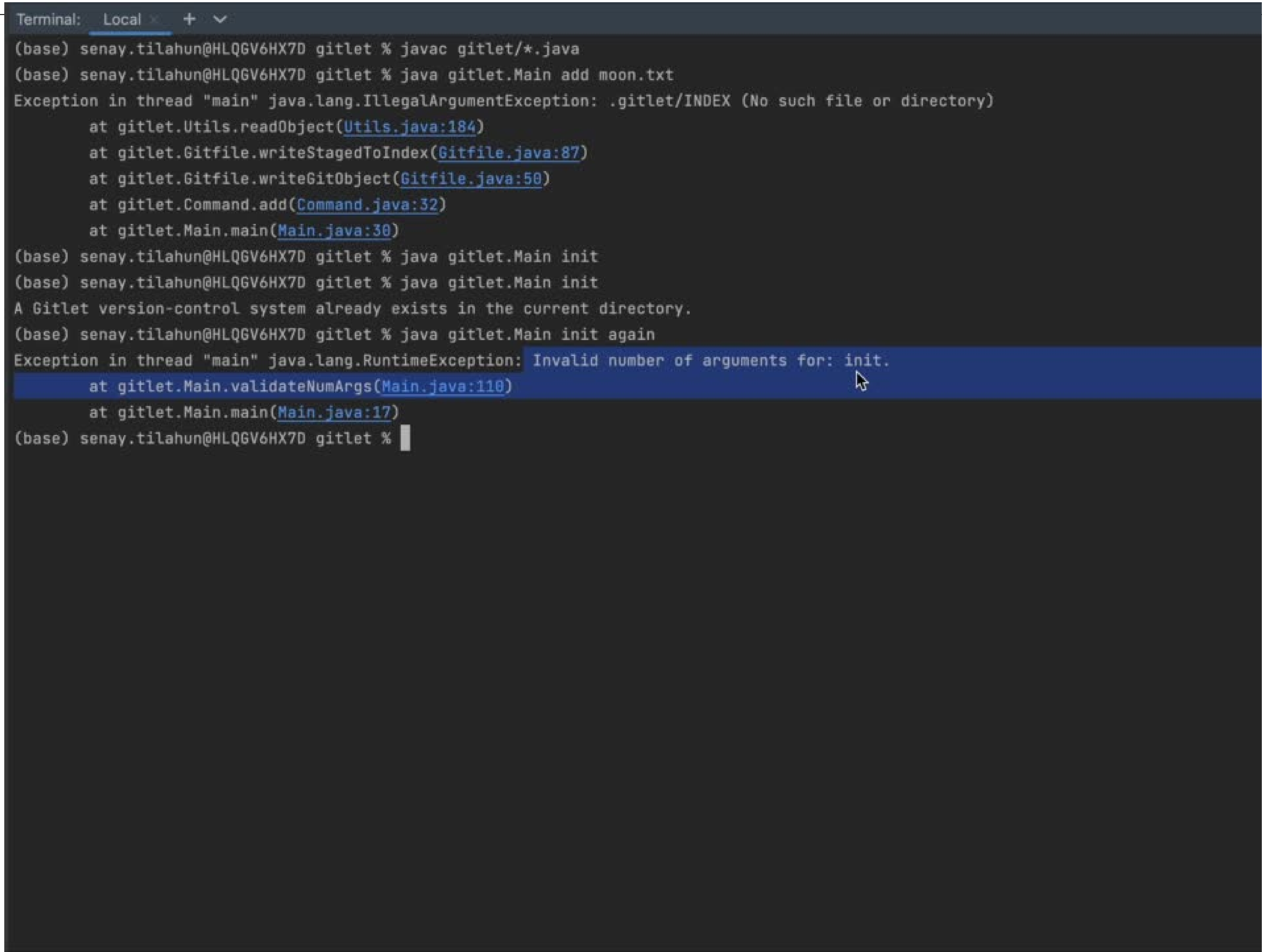
- Creates .gitlet folder
- Creates sub-folders
- Validates folder creation
- Creates initial commit



```
    void init() throws IOException {
//        GitletFileCommand.initializeRepo();
        // Initialize .gitlet folder and sub-folders
        gitletObjects.mkdirs();
        gitletLocalHead.mkdirs();
        // check that directories are correctly created
        File[] directories = {gitletObjects, gitletLocalHead};
        for (File folder : directories){
            if (!folder.exists() && !folder.mkdir()){
                throw new IllegalStateException("Unable to create directory.");
            }
        }
        GitletObjects index = new GitletObjects( args: "index");
        // catch exception
        try (ObjectOutputStream out = new ObjectOutputStream(
            new FileOutputStream(new File( pathname: ".gitlet/indexRM")))) {
            out.writeObject(index);
        } catch (IOException e) {
            System.out.println("IOException while writing object to file .gitlet/index");
            e.printStackTrace();
        }
        Utils.writeObject(gitletIndRM, index);
        // catch exception
        GitletObjects fileList = new GitletObjects( args: "index");
        try (ObjectOutputStream out = new ObjectOutputStream(
            new FileOutputStream(new File( pathname: ".gitlet/index")))) {
            out.writeObject(fileList);
        } catch (IOException e) {
            System.out.println("IOException while writing object to file .gitlet/index");
            e.printStackTrace();
        }
        Utils.writeObject(gitletInd, fileList);
        Utils.writeContents(gitletHead, ...contents: "master");
        GitletObjects commit0 = new GitletObjects();
        // TODO: update this method in GitletFileCommand & Utils
        GitletFileCommand.writeGitletCommitObject(commit0);
    }
```

Demo

Demo



```
(base) senay.tilahun@HLQGV6HX7D gitlet % javac gitlet/*.java
(base) senay.tilahun@HLQGV6HX7D gitlet % java gitlet.Main add moon.txt
Exception in thread "main" java.lang.IllegalArgumentException: .gitlet/INDEX (No such file or directory)
        at gitlet.Utils.readObject(Utils.java:184)
        at gitlet.Gitfile.writeStagedToIndex(Gitfile.java:87)
        at gitlet.Gitfile.writeGitObject(Gitfile.java:50)
        at gitlet.Command.add(Command.java:32)
        at gitlet.Main.main(Main.java:30)
(base) senay.tilahun@HLQGV6HX7D gitlet % java gitlet.Main init
(base) senay.tilahun@HLQGV6HX7D gitlet % java gitlet.Main init
A Gitlet version-control system already exists in the current directory.
(base) senay.tilahun@HLQGV6HX7D gitlet % java gitlet.Main init again
Exception in thread "main" java.lang.RuntimeException: Invalid number of arguments for: init.
        at gitlet.Main.validateNumArgs(Main.java:110)
        at gitlet.Main.main(Main.java:17)
(base) senay.tilahun@HLQGV6HX7D gitlet %
```

Relfections

Importance of Design process ->

- In the future, I want to spend more time upfront on the design so the project flows better

Underestimated complexity of project->

- Initially wanted to do 12 commands, but had to settle for 4/5 for v1, and add rest in v2 improvements

Java File management complexities ->

- Very hard concept for me to understand - everything is a file
- Even folders are files, so the distinction is important and needs time to understand

Early Feedback ->

- I need to show my work much earlier to someone and get feedback on my project

Improve MVC architecture ->

- for the V

OOP Ideas used

Encapsulation

Abstraction

Inheritance

A little bit of MVC Architecture

Not OOP, but from 5004 - Recursive Data Structures

Q & A

CITE

Sources

1. https://www.youtube.com/watch?v=AuADFzc7moY&t=253s
2. https://inst.eecs.berkeley.edu/~cs61b/sp20/materials/proj/proj3/index.html#
3. https://www.youtube.com/watch?v=tc4LnmhZusc
4. https://inst.eecs.berkeley.edu/~cs61b/sp20/materials/proj/proj3/design.html