CS319 Spring 2021 Git Lab Assignment

Prerequisites

- Git
- GitHub account

Setup

Make sure you have correctly set your Git full name and email. Please write your GitHub email as a Git email.

Notes

- Make sure that you push all your local branches to your GitHub repo.
- You can use the following command to print your Git history graph in terminal:

```
git log --all --graph --decorate --oneline
```

Grading

Submission time	Percentage cut
15:30 - 17:20	0%
17:21 - 17:40	10%
17:41 - 18:00	30%

No submission will be accepted after 18:00.

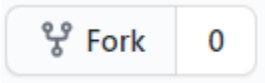
Background story

Note: You don't need to write code or use Git in this section. This section gives you an overview of the scenario. The scenario will be provided to you via a Github repository. The actual assignment starts on page 2.

Imagine you are a software engineer in an institution where your team has been assigned to implement a module that will allow searching students by their attributes. Initially, your team has written a method (in StudentSearch class) that checks whether there is a student with the given student ID. You have noticed that the code has a bug that allows empty lists to be passed to the method, which is expected to throw an exception. To solve the issue, you have created a branch called **bug-fix** to implement the fix. While you were implementing the fix, your team member fixed the bug without notifying you. When trying to merge the **bug-fix** branch to the **master** branch, you got a merge conflict.

Part 1 - Forking, Resolving Merge Conflicts (25 Pts)

Please Fork the given GitHub repository using the Fork button on the top-right section of the page: https://github.com/jelgun/Lab



Next, you need to **clone** the project to your local computer. Because the clone command only keeps the **master** branch in your local repository you won't see the **bug-fix** branch initially. To have the **bug-fix** branch in your local repository, you need to **checkout** to the **bug-fix** branch. Git will automatically match it with **origin/bug-fix** branch which is a remote branch. Then, **checkout** to the **master** branch again. After these operations, you should see the Git history as follows:

```
* (HEAD -> master, origin/master, origin/HEAD) create Main class
* throw exception for empty list
| * (origin/bug-fix, bug-fix) throw exception for empty list
|/
* add studentExists method
* create Student class
* init
* add .gitignore
```

Make sure that you are currently on the **master** branch. Now try to merge the **bug-fix** to the **master** branch. You should get an error saying that the merge is not possible due to the conflicts. Open the **src/StudentSearch.java** file with your code editor and resolve the conflict (codes in both parts are correct but you need to keep only one of them). Your code should be similar to the following if you keep the snippet from the master branch:

```
public boolean studentExists(ArrayList<Student> students, String id) throws Exception {
   if (students.isEmpty()) {
       throw new Exception("Students list should not be empty!");
   }
   for (Student student: students)
       if (student.getId().equals(id))
           return true;
   return false;
}
```

Next, **commit** the changes and **push** them to the remote server. The Git graph should be similar to this:

```
* (HEAD -> master, origin/master, origin/HEAD) Merge branch 'bug-fix'
|\
| * (origin/bug-fix, bug-fix) throw exception for empty list
* | create Main class
* | throw exception for empty list
|/
* add studentExists method
* create Student class
* init
* add .gitignore
```

Part 2 - Branching, Fast-forward Merging (20 Pts)

You have been assigned to implement a method that returns a student who has the given name. To do so, create a new branch called **findOne** and **checkout** to it. Now open the **src/StudentSearch.java** file, add the following method to the class and save the file.

```
public Student findOne(ArrayList<Student> students, String name) throws Exception {
   for (Student student: students)
       if (student.getName().equals(name))
           return student;

   throw new Exception("There is no student with the given name!");
}
```

Commit the changes with a message "*implement findOne method*". Next, merge the **findOne** branch to the **master** branch using **fast-forward merge** and **push** the changes to the remote. The Git graph should be as follows after the operations:

```
* (HEAD -> master, origin/master, origin/findOne, origin/HEAD, findOne)
implement findOne method
* Merge branch 'bug-fix'
|\
| * (origin/bug-fix, bug-fix) throw exception for empty list
* | create Main class
* | throw exception for empty list
|/
* add studentExists method
* create Student class
* init
* add .gitignore
```

Part 3 - GitHub Issues and Pull Requests (40 Pts)

Your next task has been assigned. You need to implement a method that will return all the students whose name matches the given name. Go to your GitHub repository and create a new **Issue** with a title "*requesting findAll feature*" (if you do not see the Issues tab in the project menu, tick the checkbox in **Settings>Options>Features>Issues**). Now, in your local repository, create a new branch called **findAll** and **checkout** to it. Add the following code snippet to the **StudentSearch** class:

```
public ArrayList<Student> findAll(ArrayList<Student> students, String name) throws
Exception {
   ArrayList<Student> result = new ArrayList<Student>();
   for (Student student: students)
       if (student.getName().equals(name)) {
           result.add(student);
       }

   if (result.isEmpty()) {
       throw new Exception("There is no student with the given name!");
   }
   return result;
}
```

Create a **commit** with a message "*implement findAll method*". **Push** the **findAll** branch to your **remote** repository. Create a **Pull Request** that will merge **findAll** branch to the **master** branch, and **link** the **Issue** that you created earlier to this pull request. Suppose that your team member reviewed your code and told you to add some comments to your code. Using the same branch (**findAll)**, in your local repo, add some comments to your code and then **commit** the changes. Then, **push** your changes to the **remote** repository. After these operations, in the **Pull Request** page, you should see 2 commits. Then, complete the pull request by clicking **Merge pull request**. Next, as you have made changes in the remote branch, **pull** the changes to your local repo. Your Git history graph should be similar to this:

```
* (HEAD -> master, origin/master, origin/HEAD) Merge pull request #2 from
jelgun/findAll
|\
| * (origin/findAll, findAll) add comments
| * implement findAll method
|/
* (origin/findOne, findOne) implement findOne method
* Merge branch 'bug-fix'
|\
```

```
| * (origin/bug-fix, bug-fix) throw exception for empty list
* | create Main class
* | throw exception for empty list
|/
* add studentExists method
* create Student class
* init
* add .gitignore
```

Part 4 - Pulling (15 Pts)

Now open the GitHub page of your forked repository. Make sure that you are on the master branch. Click the "**Add a README**" button to create a readme file. Write your **full name** and **student number** to that file and click the "**Commit new file**" button. Return back to your local repository and checkout to the **master** branch. **Pull** the changes from remote repo to your local repo. You should see the README file in your local repo. Next, open the **Main.java** file and add the following method:

```
public static void main(String[] args) {
   Student student1 = new Student("student1", "2222", "st1@gmail.com");
   StudentSearch search = new StudentSearch();
   ArrayList<Student> students = new ArrayList<Student>();
   students.add(student1);
   try {
       System.out.println(search.findOne(students, "student1").getId());
   } catch (Exception e) {
       System.out.println(e.getMessage());
   }
}
```

Commit the changes with a message "*add an example execution"* and **Push** them to the remote repo. Final Git history graph should be similar to this:

```
* (HEAD -> master, origin/master, origin/HEAD) add an example execution
* Create README.md
* Merge pull request #2 from jelgun/findAll
|\
| * (origin/findAll, findAll) add comments
| * implement findAll method
|/
* (origin/findOne, findOne) implement findOne method
* Merge branch 'bug-fix'
|\
| * (origin/bug-fix, bug-fix) throw exception for empty list
```

```
* | create Main class
* | throw exception for empty list
|/
* add studentExists method
* create Student class
* init
* add .gitignore
```

Submission

This is the end of the assignment. Make sure that you have pushed all the changes and branches to the GitHub. Please send your **public** repository link and full name to the following email address with your student ID as the subject line: **elgun1999@gmail.com**