CS319 Spring 2022 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.
- Unless otherwise stated, use **fast-forward merge**.
- If you merge, pushing right after the merging operation will reflect merge to "`origin`". Pushing right after the merge will not change your final result, because you will do lots of pushes anyway :). Do not worry if you forget to push. You can push any time.
- You can use the following command to print your Git history graph in terminal:

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

Questions

If you have any questions, you can ask that on the #git-lab Slack channel. The TAs will respond to your question over there. If there is an issue that you might not want to discuss publicly, please send a private message to the TAs via Slack. In case the problem persists, you will be called to a 1-on-1 Zoom breakout room to help you with your issue. Please **do not send messages** to the Zoom chat as we will not be using it.

Grading

No submission will be accepted after 12:30.

Background story

Note

You will **neither write code nor 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 the next page.

Your Role

Lead Software Engineer at Microsoft

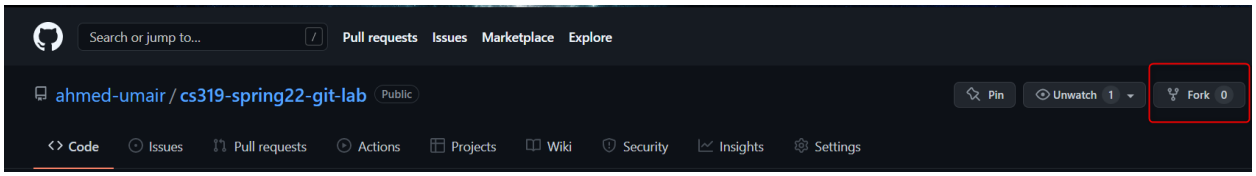
Background

You have come a long way in your career and have made your way towards the top as a Lead Software Engineer at Microsoft. Unfortunately, Microsoft is not the same company it used to be anymore. They have lost a huge portion of their codebase as a part of a data breach. Since they were neither using any version control system like Git nor any backup strategies, there is no way to recover the lost code. As a result, they are looking up to the few remaining talented and experienced engineers like you to help them (re) develop their codebase.

You have been tasked with developing the Windows Calculator app.

Part 1 - Git Basics (20 Pts)

~~> Please **fork** the given GitHub repository using the Fork button on the top-right section of the page: https://github.com/ahmed-umair/cs319-spring22-git-lab



~~> Clone **your forked repository** to your local.

➔ Your colleague Bill from the software test team informs you that the NewCalculator class is almost complete but they need you to try it out by using it to Add, Subtract and Multiply in the main method in Main.java. He tells you that it's a bit urgent and that you should not bother with branches since it is a small step. Against your best judgement, you decide to comply with Bill's request and set to work directly on the main branch.

Note: In real projects, please avoid working on the main branch at all costs. Each bug fix and new feature should be developed on their own branch and then merged into the main branch when complete. For the purposes of this exercise though, you may proceed.

~~> Make sure you are on the **main** branch

~~> Add the following lines of code to the *main* method in the `Main.java` file as below.

```
public static void main(String[] args) {
        NewCalculator calculator = new NewCalculator();

        // Try division
        calculator.Divide(100, 12);
        System.out.println(calculator.getLastResult());

        // Try addition
        calculator.Add(10, 35);
        System.out.println(calculator.getLastResult());

        // Try subtraction
        calculator.Subtract(50, 15);
        System.out.println(calculator.getLastResult());
}
```

~~> Commit the changes with the message “add evaluation code in main method”. Push the commit.

➔ Your Git history should look like this right now.

```
* 81bffda (HEAD -> main, origin/main, origin/HEAD) add evaluation code in main method
* 2b531a4 add basic features
* 0100ec6 Initial commit
```

Part 2 - Multiple Branches and Fast Forward Merging (30 Pts)

➔ After pushing the previous code in a hurry, you realize that the calculator is missing a very crucial feature: Multiplication! You decide to add this feature, in the proper manner this time (i.e. using a feature branch)

~~> Open a new branch called “**`multiply`**”. Checkout to this branch.

~~> Add the “`Multiply(double x, double y)`” method to the NewCalculator class in “`NewCalculator.java`” file as shown below.

```
public double Multiply(double x, double y){
        this.lastResult = x * y;
        return this.lastResult;
    }
```

~~> Commit the changes with the message “add multiplication feature”.

~~> Add the following lines to the “`main`” method in “`Main.java`”

```
        // Try Multiplication
        calculator.Multiply(10, 30);
        System.out.println(calculator.getLastResult());
```

~~> Commit these changes with the message “add evaluation code for Multiply”

~~>Push to the upstream **multiply** branch. (Note that git may warn you that your branch has no upstream branch. You should read the warning and push accordingly.)

➔ Your Git history should look like this right now.

```
* d14b078 (HEAD -> multiply, origin/multiply) add evaluation code for Multiply
* 038f4f7 add multiplication feature
* 81bffda (origin/main, origin/HEAD, main) add evaluation code in main method
* 2b531a4 add basic features
* 0100ec6 Initial commit
```

➔ Now that you have implemented the Multiply feature in the proper way, let's merge it back to the main branch to include it in the stable version of our Calculator app.

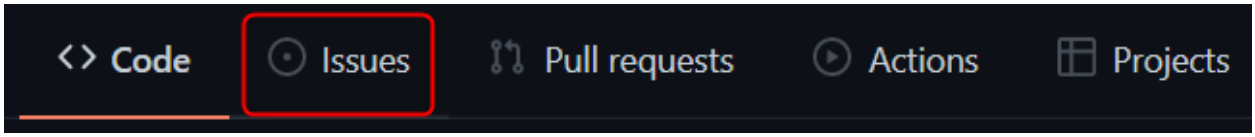
➔ Merge your **multiply** branch to your **main** branch. Push the changes.

➔ Your Git history should look like this right now.

```
* d14b078 (HEAD -> main, origin/multiply, origin/main, origin/HEAD, multiply) add evaluation
code for Multiply
* 038f4f7 add multiplication feature
* 81bffda add evaluation code in main method
* 2b531a4 add basic features
* 0100ec6 Initial commit
```

Part 3 - GitHub Issues, Pull Requests and Merge Conflicts (50 Pts)

- ➔ So far you have worked on our calculator app in bits and pieces. In this section, we will combine all major features of GitHub to solve a persisting issue in our Calculator app.
- ➔ If you look carefully at the `Divide()` method in the `NewCalculator` class, there is a glaring error in it. It does not catch the case where the divisor is 0, which would lead to an unwanted outcome (the output would be INFINITY or NaN, which we want to avoid). You decide to fix it and follow the proper Git Workflow from beginning to end in order to set a good example for your peers at Microsoft.
- ➔ The first step is to open an 'issue' about this bug on GitHub. Navigate to the GitHub page of your forked repository, switch to the '**Issues'** tab and create a new issue with the title "Divide by zero goes unchecked in NewCalculator.java". You can add a short explanation of the issue in the description. **If you can't find the 'Issues' tab, you might need to enable it from the repository settings.**



- ➔ Assign the issue to yourself.
- ➔ Add the "bug" label to this issue.

~~> Open a new branch called "`zero-check`" from the up-to-date main branch. Checkout this branch.

~~> Rewrite the "`Divide(double dividend, double divisor)`" method in "`NewCalculator.java`" file as below. The extra piece of code catches if the divisor is zero and emits an error message as well as returns 0 instead of returning INFINITY or NaN.

```
public double Divide(double dividend, double divisor) {
        if (divisor == 0){
            System.err.println("Attempted to divide by zero");
            return 0;
        }
        this.lastResult = dividend % divisor;
        return this.lastResult;
    }
```

~~> Commit the changes with the message “catch divide by zero”. Push the commit. (Note that git may warn you that your branch has no upstream branch. You should read the warning and push accordingly.)

- ➔ After you have pushed your fix to the `zero-check` branch, you are considering creating a pull-request and merging it into the main branch when your colleague Bill comes to you once more with one of his requests. He is in need of a hotfix again and he wants it as soon as possible on the main branch without any additional branches being involved. You try reasoning with him on why that’s such a bad idea but he does not listen. Since Bill is senior to you, you begrudgingly cave in and decide to entertain this one last request, promising yourself that you will never do it again.
- ➔ What Bill wants you to fix is the Divide method again. It’s apparent that whoever made it was not doing their best work. Either way, it is now your responsibility to apply yet another fix to this Calculator app’s Divide method
- ➔ The problem is as follows: The original author used the modulus ‘%’ operator instead of the division operator which is `/` in the Divide method. Unless this is fixed, the Divide method will keep returning the remainder instead of the quotient of its parameters every time it’s called.

~~> Checkout the **main** branch

~~> Rewrite the “`Divide(double dividend, double divisor)`” method in “`NewCalculator.java`” file as below.

```
public double Divide(double dividend, double divisor){
        this.lastResult = dividend / divisor;
        return this.lastResult;
    }
```

~~> Commit the changes with the message “fix division operator”. Push the commit.

- ➔ Your Git history should look like this right now.

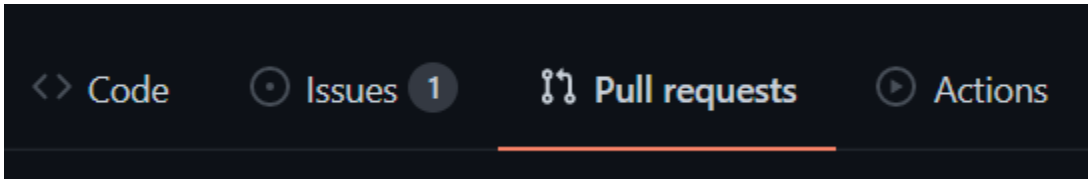
```
* b192455 (HEAD -> main, origin/main, origin/HEAD) fix division operator
| * ae43af7 (origin/zero-check, zero-check) catch divide by zero
|/
* d14b078 (origin/multiply, multiply) add evaluation code for Multiply
```

```
* 038f4f7 add multiplication feature
* 81bffda add evaluation code in main method
* 2b531a4 add basic features
* 0100ec6 Initial commit
```

➔ Finally done with all of Bill's troublesome requests, you decide to get back to the fix that you were originally working on and complete it. You were pondering over creating a merge request to merge your **zero-check** branch to the **main** branch

~~> Make sure all the changes you made `zero-check` locally have been pushed to GitHub
~~> Create a Pull Request on GitHub that will merge the "`zero-check`" branch to the "`main`" branch. (**Pull Request should be created with the branches in your forked repository. You may see branches of the repo ahmed-umair/cs319-spring22-git-lab. Ignore those branches and operate on your own ones.**)

➔ (Be careful, do not act without reading this completely) You decide to merge your code because your colleagues are on vacation (this is a bad practice). In practice, pull requests by one person should always be reviewed by at least one other team member before being merged. This time, you make an exception and proceed with merging



➔ After creating the Pull Request, you will GitHub indicating that the pull request cannot be merged directly. This means that we have run into a merge conflict. We need to resolve it.

~~> Before proceeding with a merge of any sort, assign the pull request to yourself.

~~> Link this pull request to the "Divide by zero goes unchecked in NewCalculator.java" issue that you created earlier. This helps preserve traceability across issues encountered and the code changes made to fix them and will come in handy in the future.

~~> Now we can proceed to the manual merging part. Click on “Resolve Conflicts”. You will see that there is a conflict marker on the `Divide` method in the “`NewCalculator.java`” file. The version coming from the zero-check branch is different from the version coming from main. The zero-check branch has the zero check code. The main branch has the correct division operator. We need the final version to have both these fixes. You need to resolve this so that the final version of the Divide method looks like this:

```
public double Divide(double dividend, double divisor) {
        if (divisor == 0){
            System.err.println("Attempted to divide by zero");
            return 0;
        }
        this.lastResult = dividend / divisor;
        return this.lastResult;
    }
```

~~> Once you resolve the conflict, please proceed with the merge and commit it. You will then be taken back to the Pull Request screen. Proceed with merging the pull request. After this, make sure that the issue linked to this pull request is closed.

➔ Your Git history should look like this right now. Since you created and applied the pull request on GitHub, your local might not have the changes. Run **git pull** before to get new changes on your local. Then you can check this graph.

```
*   19768e6 (HEAD -> main, origin/main, origin/HEAD) Merge pull request #3 from
ahmed-umair/zero-check
|\
| *   d583100 (origin/zero-check) Merge branch 'main' into zero-check
| |\
| |/
|/|
* | 39a13e3 fix division operator
| * ae43af7 (zero-check) catch divide by zero
|/
* d14b078 (origin/multiply, multiply) add evaluation code for Multiply
* 038f4f7 add multiplication feature
* 81bffda add evaluation code in main method
* 2b531a4 add basic features
* 0100ec6 Initial commit
```

Phew, finally you are done with restoring the Calculator app. Now you are going to hand it over to the front-end team for them to complete the design work and make a presentable app. And you promise yourself to never entertain any requests by Bill that break the proper Git Workflow.

The End!

Here is Bill’s response on live TV when he finds out what you think about him.

Submission

This is the end of the assignment. Make sure that you have pushed all the changes and branches to GitHub. Send the following in the email.

- **Notes**
    - Replace X with your section number.
    - Example subject lines: “CS319-Lab-22113344”,
- **Subject**: CS319-Lab-<YourStudentID>
- **To**: umair.ahmed@bilkent.edu.tr
- **Body**:
    - Link to the repo: <Paste your git URL>
    - Link to your pull request: <Paste your pull request URL>
    - Full Name: <You full name>