M1 - Team Charter and Planning, Version Control and TDD

Part 1 - Team Page

On Canvas, go to *People > cs2340_team* and click "visit group homepage" from your team's drop-down menu to go to your team's Canvas page. You are to create a team page that will serve as an easy way for your TAs to get your information. You should come up with a unique team name that we will use for the rest of the semester to refer to you. Be sure to include the team number as well; here's an example team page: Team 0: The Burdells. You must include your team emails here, so everyone knows how to get a hold of you. You may also include other team information.

This basic information set will work for now. You will expand the page as the course progresses; for example, you will also put a link to your project GitHub repository here in a later assignment.

Part 2 - Team Contract

Download the CATME Team Charter form from the Files section of Canvas. Fill it out for your team. You may submit this electronically as a Word (DOC/DOCX) or Adobe (PDF) file to the Assignments page.

CATME-Team-Charter.docx

Submission Instructions

- Submit your Team Charter via the Assignments page. This is a group assignment, so only one team member will need to submit the charter for the whole group.
- Ensure that your team's Project Page is published and accessible here

Part 3 - Team Availability

Complete the Team Availability Assignment located here **by 9/15/2021 EOD**.

Part 4 - Version Control, TDD

Purpose

When working in larger teams, it becomes imperative to collaboratively manage the source code. Version control allows multiple developers to work synchronously on a project, saving time and money. There are two primary types of version control systems in use: central and distributed repositories. Distributed version control is more common in industry today, so we will use a distributed versioning system called Git. This milestone will introduce you to the basics of Git. If you want to learn more about the Central vs Distributed version control, a relevant article can be found in the Helpful Resources tab.

This milestone will also introduce you to *test driven development* (TDD). In the past CS 2340 students have had trouble writing test cases as TDD was taught at the end of the class. With this milestone, you will get your feet wet by following a JUnit tutorial and then by writing your own JUnit tests. By building the habit of writing tests early and often, you will save countless hours of debugging when a new addition to your project suddenly breaks everything!

Helpful Resources:

- Git is described in:
  - http://git-scm.com/doc
- Git tutorials:
  - https://try.github.io/levels/1/challenges/1 [interactive tutorials]
  - https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line [merge conflicts]
  - https://chris.beams.io/posts/git-commit/ [commit message guidelines]
  - https://docs.github.com/en/github/using-git/ignoring-files [gitignores]
- JUnit documentation/tutorials:
  - https://junit.org/junit5/ [docs]
  - https://junit.org/junit5/docs/current/user-guide/ [official guide]
  - https://www.tutorialspoint.com/a_practical_guide_to_junit_5/index.asp
- Editor guides:
  - https://www.jetbrains.com/help/idea/discover-intellij-idea.html [IntelliJ]
  - https://code.visualstudio.com/docs/editor/versioncontrol [vscode]
  - https://dev.to/iggredible/working-with-vim-and-git-4nkh [vim]
- Centralized vs Distributed Version control:
  - https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs

For project hosting, you are **required** to use https://github.gatech.edu/

Task

1. **Setup**

   Setup Git on your machine with the desired client. *Please use Git from the command line and refrain from using GUI versions of Git, like ones packaged with GitHub, IntelliJ, VSCode, etc*. It is important to learn how to use Git using the command line as that is what is expected in industry. If you have trouble with Git, feel free to come to TA office hours for help.

   The recommended IDE for this assignment, and this class, is IntelliJ.

   One team member will need to create a new private repository on **the GT Enterprise GitHub (link above)** and clone the repository onto their machine. This repository will be for M1 only and is not for your course project repository.

Download the M1 resource file (M1.zip) and unzip it. Add the unzipped files over the command line (git add, commit, and push) to the repository (again, only one person needs to do this).

Whoever created the repository will need to add the rest of their team as collaborators so that everyone is able to access and clone the repository.

Create a new branch at this point called "original" and push this branch to the online repository. The purpose of this branch is to retain an original unmodified set of code.

Be sure that "original" **remains as the unmodified version** and that changes to the repository are present on the main (master) branch.

2. **Edit, add and commit files as detailed below**

Firstly, if you are using IntelliJ or another project manager, you might notice additional directories and files that have been added in after you created the project using the M1 files. These files (iml files) and directories (out/ and .idea/) are created by IntelliJ to manage your local version of the project and should not be tracked by git. Create a gitignore to ensure the out/ directory, the .idea/ directory, and all iml files are not tracked.

If you examine the src/main/java directory, you will see the **MyStack.java** file containing a faulty implementation of a stack data structure**.**

Each person should create a branch named after their own name (bob, sally, buzz...) and checkout their branch.

Each person on the team should add their own separate, unique JUnit test verifying this stack implementation. These unit tests should be added to the **MyStackTest.java** file located in the src/test/java directory.

If you are on a four-person team, you will only need to implement four JUnit tests.

Modify **MyStackTest.java** file by adding your JUnit tests. Modify **MyStack.java** file if needed.

Commit your changes and push. All commits should have a commit message, and you should make sure ***all*** your commits have **non-default** messages. Useful commit messages are essential for proper use of version control. Commit messages should be **concise** descriptions of your changes and in **present tense**. Then, you should create a pull request to merge your code from your development branch into the master (**not** original) branch.

3. **Add and remove files**

Each team member should add a text file on their branch to the top-level directory labeled readme.p*n*.txt where the p*n* would be p1, p2, p3, p4 or p5 based upon which person you are for the assignment. The contents of the file should include your name and email. Each team member should delete the text file useless.p*n*.txt where p*n* is p1, p2, p3, p4 or p5 based on your team member number. *Do not delete the wrong files!* Then, you should create a pull request to merge your changes on your individual branch into the main/master branch.

Roll back changes by checking out your original branch from an earlier step. Verify that none of the changes you have made are in the original branch project version you checked out.

Criteria

Your M1 will be graded as a group upon the following criteria:

- Team page is created, properly linked to, and provides team member names and information
- Team charter is submitted and is properly filled out
- M1 imported into a private Gatech GitHub repository
- **original** branch has been created and no changes have been made to it
- Changes to **MyStackTest.java** file committed and present on the main/master branch
- Each team member has written the required test cases for the **MyStack.java** methods on their individual branches and has made a pull request to merge them into the main/master branch
- If needed, changes to **MyStack.java** file have been committed on the individual's branch and a pull request has been made to merge those changes into master
- **readme.p*n*.txt** files added
- **useless.p*n*.txt** files deleted
- All commits have proper messages
- Individual branches created and used