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 99 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 GitHub Page 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

Part 3 - Team Availability

Complete the Team Availability Assignment located here.

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 4 - Version Control, TDD

Purpose

When working in larger teams, it becomes imperative to manage the source code as multiple developers work on it. 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 most 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, an 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, 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 edition to your project suddenly breaks everything!

Helpful Resources:

- Git is described in: http://git-scm.com/doc
- There is a nice interactive tutorial for Git here: https://try.github.io/levels/1/challenges/1
- How to write Git Commit Messages: https://chris.beams.io/posts/git-commit/
- Centralized vs Distributed Version control: https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs
- JUnit documentation: https://junit.org/junit5/
- JUnit tutorials: https://www.tutorialspoint.com/junit/index.htm

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 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.

   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.

   Download the M1 resource file (M1.zip) and unzip it. Add the unzipped files (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 the 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 done on the main (master) branch.

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

   If you examine the src directory, you will see the files **Person1.java, Person2.java, Person3.java, Person4.java,** and **Person5.java**.

   Each person on the team should choose ONE of these so that everyone is working on a different file. If you are on a four-person team, you will only need to modify four of the person files.

   Each person should create a branch named for their name (bob, sally, buzz...). Checkout this branch.

Now examine your individual file and complete the required TODO. The Javadoc comment will explain what must be done.

Commit your changes and push. Commit messages should be **concise** descriptions of your changes and in **present tense**. Then merge your code from your development branch into the master branch.

2. Add and remove files

Each team member should add a text file on the master branch to the top-level directory labeled readme.pn.txt where the pn would be p1, p2, p3, p4 or p5 based upon which person you are for the lab. The contents of the file should include your name and email. Each team member should delete the text file useless.pn.txt where pn is p1,p2, p3, p4 or p5 based on your team member number. Do NOT delete the wrong files!

Make sure all your commits have messages. Useful commit messages are essential for proper use of version control.

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:

- M1 imported into git
- Changes to PersonN.java file committed and pushed to master
- Readme file added
- useless.txt file deleted
- All commits have messages
- Individual branch created and used
- Merged some changes into master branch
- "original" branch created and has no changes
- Your repository is located on the Georgia Tech Enterprise GitHub.
- Everyone has written the required test cases for their PersonN methods