/\*

As long the team uses the same class packages, class names, and libraries, I believe each team member can user their own preferred IDE. Preferably, for the team to use the same tools to work on the project, below are the tools to get started working on the project. Let louitek know if you need help with configuration because once eclipse is downloaded you will have to update the paths to jdk 17 and openjfx. Also after you do a git clone, you have to clean, refresh and rebuild project.

1. download gitbash : <https://git-scm.com/downloads>
2. download github desktop : <https://desktop.github.com/>
3. download java jdk 17 : <https://jdk.java.net/17/>
4. download java fx release 17 : <https://gluonhq.com/products/javafx/>
5. download eclipse IDE : <https://www.eclipse.org/downloads/>

\*/

Git can be difficult to work with, therefore keep the below warnings in mind when working on the project. They will help in avoiding overwrites to code changes, merge conflicts, dirty branches and avoiding tedious manual labor.

1. when you are editing make sure you are making changes to your local branch not the master/main branch. You should only update master/main branch with your final changes.
2. make sure you are on the correct branch, meaning you are not editing master/main branch or working on the wrong project.
3. make a copy of the original branch before “git push”, then if somethings goes wrong, it will be easy to go back to the previous working state.
4. let the team know you’re about to commit and push changes to avoid conflict.
5. Before you push changes make sure your local branch and main are up to date and your code changes compiles and work.
6. after a team member push changes to master, its good practice to pull the new updated branch to your local working directory. You should be with the most recent update to the project.
7. Avoid simultaneous edits of the same part of the project as another team member because there will conflicts when executing a git merge or pull command. Therefore, Avoid editing the same class, at the same time as another team member. Best to wait for the team member to finish making his/her changes and push to those changes to main branch. Then you will pull or merge the updated code changes to your local branch and continue working.
8. Make sure your code changes are saved before git pull or merge with the main branch.
9. The main goal is to avoid your changes from being overwritten or lost, avoid conflicts, and your git push code changes are able to compile and execute.

#git clone master **#to clone the project main/master repository**

$git status **#will display if you are working on Main/Master or local branch**

**if you are on the wrong branch then**

$git checkout correct-branch

**to pull changes from main/master branch to your local branch keep in mind your current changes on your local branch might get overwritten if not save properly for instance push your changes**

$git git pull branch name

**example: pulling master into local branch**

$git pull origin/master

**joining two branches together make sure the branches don't have any conflicts**

$git merge branch-name

adding your changes to your local branch on github

first step:

$git add .

second step:

$git commit -m "descriptive text"

third step:

$git push

fourth step verify the changes have been push by going to the github project online repository.