Kister, Nicole

Version 1.0

cs 301 Game Framework guide

An all-inclusive guide to the Game Framework

Contents

[Introduction 1](#_Toc11945990)

[Git and GitHub 2](#_Toc11945991)

[What is Git? 2](#_Toc11945992)

[How to Install Git 2](#_Toc11945993)

[Flavors of Git 2](#_Toc11945994)

[Git Bash 2](#_Toc11945995)

[GitHub Desktop 5](#_Toc11945996)

[Git IDE Integration 6](#_Toc11945997)

[Git Essentials 6](#_Toc11945998)

[Git Commands (For Bash Users) 7](#_Toc11945999)

[What is GitHub? 8](#_Toc11946000)

[Creating a GitHub Account 8](#_Toc11946001)

[Creating a Repository – Android Studio 8](#_Toc11946002)

[Using a Repository 9](#_Toc11946003)

[Android Studio 12](#_Toc11946004)

[What is Android Studio? 12](#_Toc11946005)

[How to Download 12](#_Toc11946006)

[Creating a New Project 13](#_Toc11946007)

[Cloning a Repository 16](#_Toc11946008)

[Layout 19](#_Toc11946009)

[Features 21](#_Toc11946010)

[Committing, Pushing, and Pulling 27](#_Toc11946011)

[The Game Framework 32](#_Toc11946012)

[What is the Game Framework? 32](#_Toc11946013)

[Where to Find the Framework 32](#_Toc11946014)

[Classes in the Framework 32](#_Toc11946015)

[Classes in the actionMessage Package 33](#_Toc11946016)

[Classes in the animation Package 33](#_Toc11946017)

[Classes in the gameConfiguration Package 33](#_Toc11946018)

[Classes in the infoMessage Package 33](#_Toc11946019)

[Classes in the utilities Package 34](#_Toc11946020)

[Advanced Features of the Framework 34](#_Toc11946021)

[Setup Phase 34](#_Toc11946022)

[Custom Game Rules 34](#_Toc11946023)

[Your Game! 35](#_Toc11946024)

[Cloning the Base Repository 35](#_Toc11946025)

[Which Classes do I Need to Inherit/SubClass? 35](#_Toc11946026)

[Help! Everything is Broken! 36](#_Toc11946027)

[IDE Errors 36](#_Toc11946028)

[GitHub Errors 36](#_Toc11946029)

[Debugging Tips 36](#_Toc11946030)

[FAQ 37](#_Toc11946031)

# Introduction

Hi! Welcome to CS 301, Object Oriented Design. In this game you will turn a board game into an Android app. This guide is intended to help walk you through some of the key components of this framework and how you can use it to make your CS 301 experience great.

This guide assumes you are familiar with basic programming terms and concepts such as inheritance, methods, and objects.

If you already know something in a section, or feel you don’t need to read it – go to the next one. This guide is designed to be easy to pick and up and read from any section.

Hopefully you find this guide useful and a pleasant addition to your CS 301 experience.

One final note- this guide **IS NOT** a replacement for going to class. Please go to class. This guide is a supplement for the in-class experience.

# Git and GitHub

This section will guide your through Git installation, use, and how to use GitHub.

## What is Git?

Git is a form of version control that is widely used. It is free and open-source, meaning you can use it for any and every project you work on. There are other version control options available, however Git is widely used in the industry so it is important to begin learning how to use it now.

Git is NOT the same as GitHub. They are two separate tools. GitHub will be explained in a few sections.

## How to Install Git

Git can be installed from the Git website: <https://git-scm.com/>

Download the version that is appropriate for your operating system. The standard installation should be fine, and I would recommend only having Git available in the Bash shell, not in the command line for Windows users. There is also documentation on the Git website you could read if you are interested.

Git is installed by default on most Mac and Linux computers.

## Flavors of Git

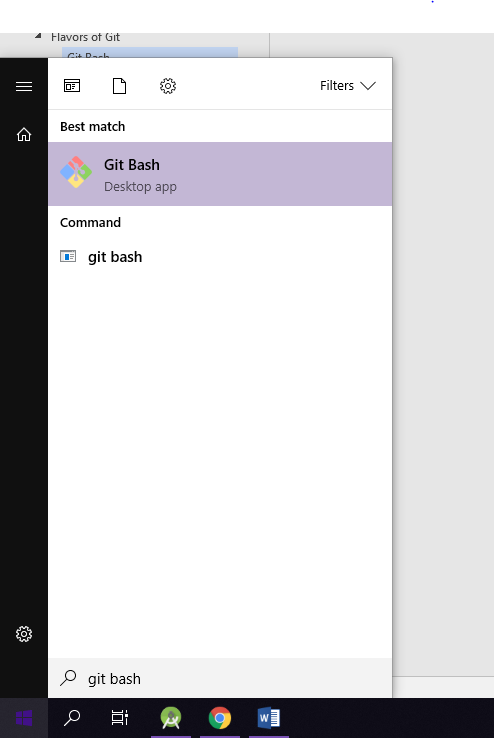
There are a few different ways to use Git for your projects. For the purposes of CS 301, you will be using Git in an IDE (Integrated Development Environment), so feel free to skip directly to that section below. I have provided information about the other two ways to use Git in case you are interested in learning more.

### Git Bash on Windows

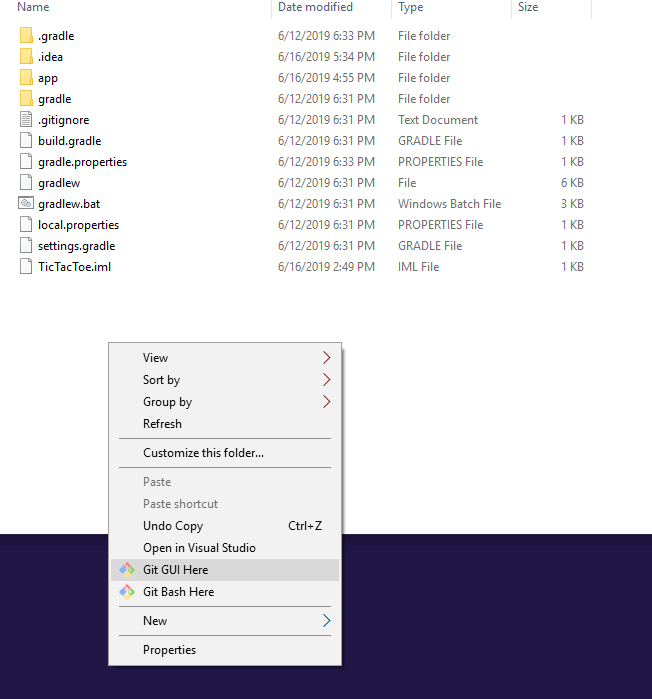
Git Bash is the standard terminal provided when you download Git. It performs the same functions as the other two options (discussed below), however it is all done through command line-like functions. If this thought scares you- do not fear! See the other two sections below.

The Git Bash terminal can be accessed one of two ways:

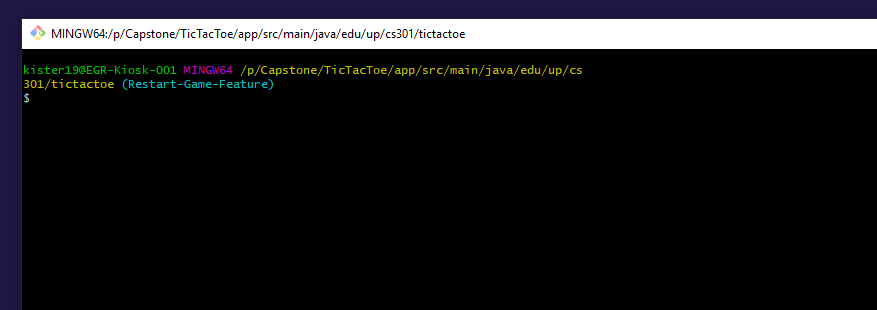
1. Typing Git Bash into the search of your computer



1. Navigating to a folder, right-clicking, and selecting ‘Git Bash here’

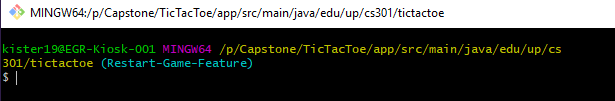


Either selection will result in the Git Bash opening. It looks like this:



The information present here can be broken down like this:

File Path



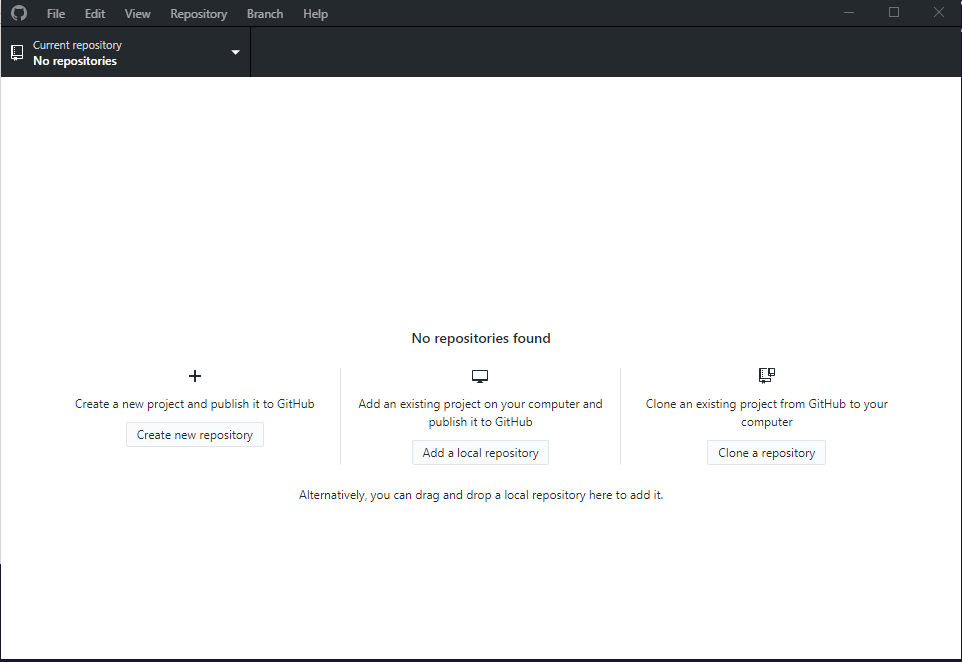
Branch You Are Currently On

To use Git from the Bash terminal is done by using commands. There is a separate section on Bash commands a few pages ahead.

### GitHub Desktop (Windows and Mac)

GitHub Desktop is a tool created by GitHub to use Git without the scary command line interface. It performs all of the same actions but has a GUI to provide ease of use. GitHub Desktop must be downloaded separately. The link to download this tool is here: <https://desktop.github.com/>

When opening the application for the first time it will ask you to login to your GitHub account (see the section below for how to create an account) and then it will bring up a GUI for you to interact with:



From here you can create or clone a repository. This option is a good choice if you don’t have an IDE and are cautious about the command line.

### Git IDE Integration

If you are working in an IDE (which you will be for CS 301), it will most likely have Git integration built into it. You still need to have Git installed to use it here. Every IDE has its Git features in a different location, so take a moment to learn where they are in your IDE of choice and how to use it. Usually the Git features have GUIs to lead you through the Git process, so there will not be a command-line interface here.

## Git Essentials

Version control works by recording changes to files in a project over time and can give the team members the opportunity to discuss changes and ensure only the changes they want in the project.

Here are some key components to Git and version control:

**Repository:** A project

**Branch:** A branch is a copy of the code. Usually the default branch is called Master, and other branches can be created and named differently.

**Commit:** When you make changes to a project, you need to make sure these changes are saved. You can do this by committing the changes. When you commit changes, you will need to provide a message to let others know about the changes you made. You can commit as many changes as you’d like on your branch.

**Push:** Pushing means publishing your committed changes to the branch you are working on. This lets other people using your repository know you have made changes and what these changes are

Standard Git Flow:

Make Changes

Commit Changes

Push Changes

(one or more commits)

Please read the ‘Good Git Practices’ section for more detailed information on the best practices for your project that uses Git version control.

## Git Commands (For Bash Users)

If you choose to use the Git Bash terminal the follow commands are what you will use to perform actions such as committing and pushing:

|  |  |
| --- | --- |
| Command | Description |
| git add | Snapshots the file in preparation for versioning. |
| git commit | Records the file snapshots permanently in version history.  Can be used with the –m parameter to pass in a message on the command line:  git commit –m “Initial commit” |
| git push | Uploads all local branch commits to GitHub |
| git pull | Downloads changes and incorporates them into your local branch |
| git branch | Lists all the local branches in the current repository |
| git checkout | Switches to the specified branch and updates the directory |
| git status | Displays the names and file paths of the files that have changed since the last commit in red, the files that have been added and are ready to commit in green.  It also displays the number of commits this local branch is ahead/behind of the remote one. |

More commands are listed here: <https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>

## What is GitHub?

GitHub is an online webhosting service for the Git version control software: <https://github.com/>

GitHub uses Git to track and record project changes in a web format. Sometimes employers will look at your GitHub to see what projects you are working on/have worked on during the job interview process.

GitHub allows teams to collaborate easily with their projects and review/record changes.

## Creating a GitHub Account

If it is your first time using GitHub, you will need to create an account. You can do this by selecting the ‘Sign Up’ button on the home page.

Since you are a student you are eligible for the GitHub student pack. Ask your professor how to get this because it is a great pack for free while you are a student.

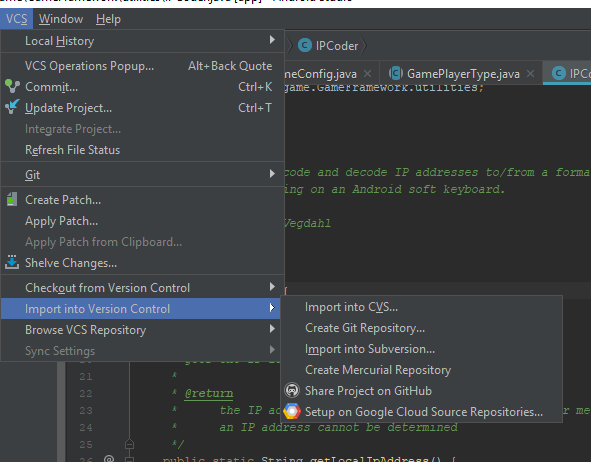
A word of advice- I would create your GitHub account with your UP email and add your personal email to the account as well. This allows you to have one account that you can use even after your time at UP. To add another email, go to the settings in your account and select the emails tab. From here you can add another email (I would add your personal email) and select which one is the primary email. Keep your UP email as the primary for now, and when you graduate you can switch it to your personal email. This also allows you to work on personal projects with your personal email instead of your UP email.

A word of advice- Choose your GitHub username wisely. Employers will sometimes look at your GitHub profile to see what projects you work on. You can also put your GitHub profile on your resume if you have some interesting projects visible. If your username is something like SuperAwesomeCoder426345234, that might rub some employers the wrong way. Choose something professional like your first and last name, or a combination of the two.

A word of advice- If you choose to put your GitHub profile on your resume, make sure the projects that are visible have good code and are semi-professional.

## Creating a Repository – Android Studio

To create a repository using Android Studio simply select the ‘Share Project on GitHub’ option in the menu after creating a new project.



This will prompt you to login and select a repository name. After doing this it will share your repository on GitHub.

## Using a Repository

When using a repository, it is best to create a new branch, make changes, and then request the changes to be merged back into master:

Pull Request to merge to Master

Make Changes

Commit/Push Changes

Branch from Master

Another way to view this flow:

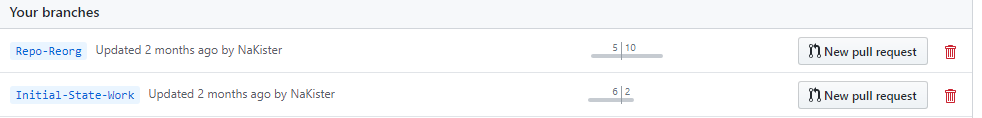
Master

Pull request and merge back to master

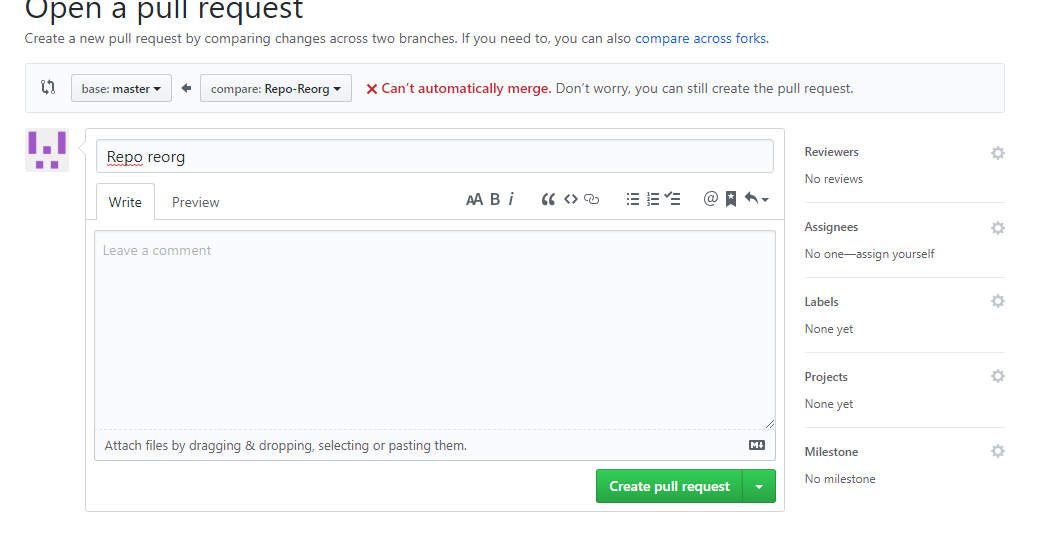
Make changes, commit, and push

Make your branch

A pull request tells the other members in your team that you are ready to merge your changes back into the master branch. You can create a pull request by selecting the option next to your branch:

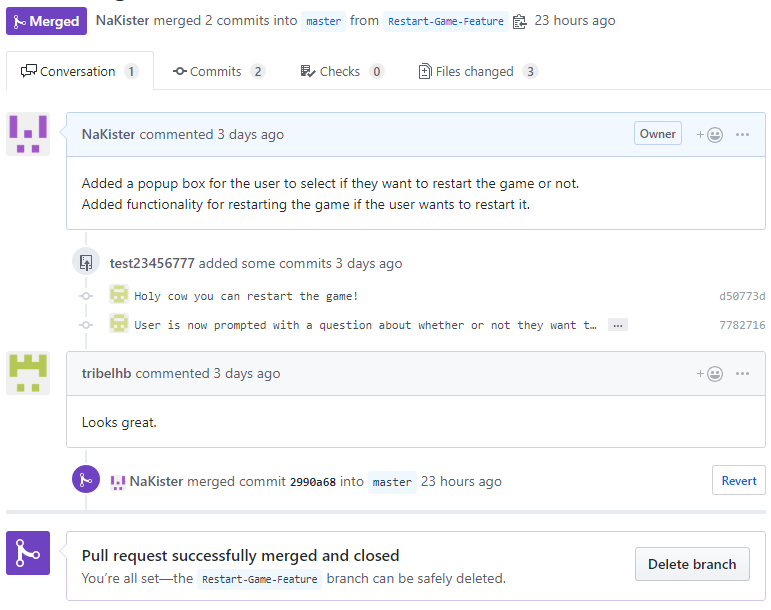


This will take you to the main pull request page. Here you can add more info in the request and request reviewers.



Make sure you have selected master as the base:

  
Once you team members approve the request you can merge the changes into the master branch. You can then automatically delete the branch.



# Android Studio

This section will guide you through using the IDE Android Studio. For the purposes of your project you may wish to skip to the **Cloning a Repository** step.

## What is Android Studio?

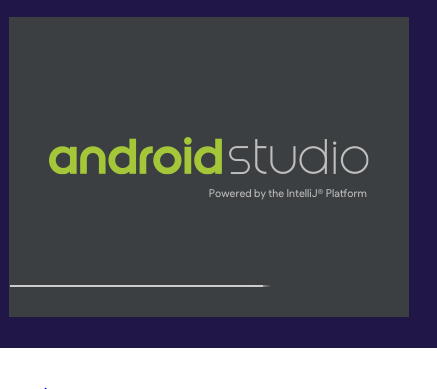
Android Studio is the Integrated Development Environment (IDE) specifically made for Android application development. It is made by JetBrains, the creators of CLion and IntelliJ, so if you have used one of these tools before this IDE should feel familiar.

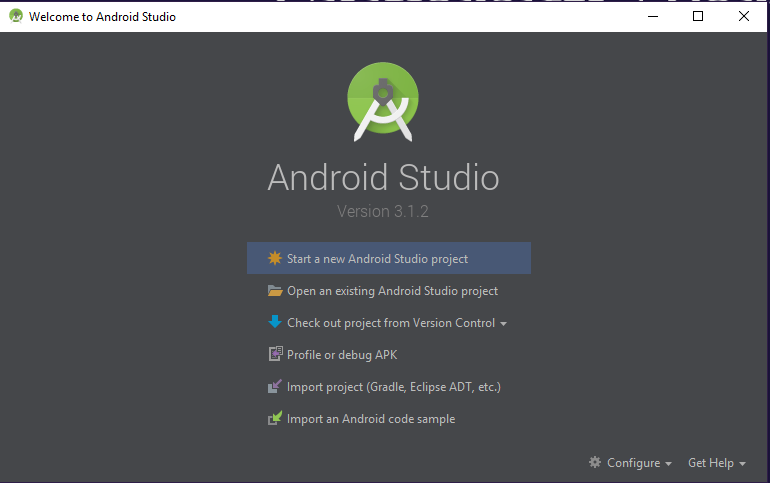
## How to Download

Android Studio can be downloaded at the following site: <https://developer.android.com/studio>

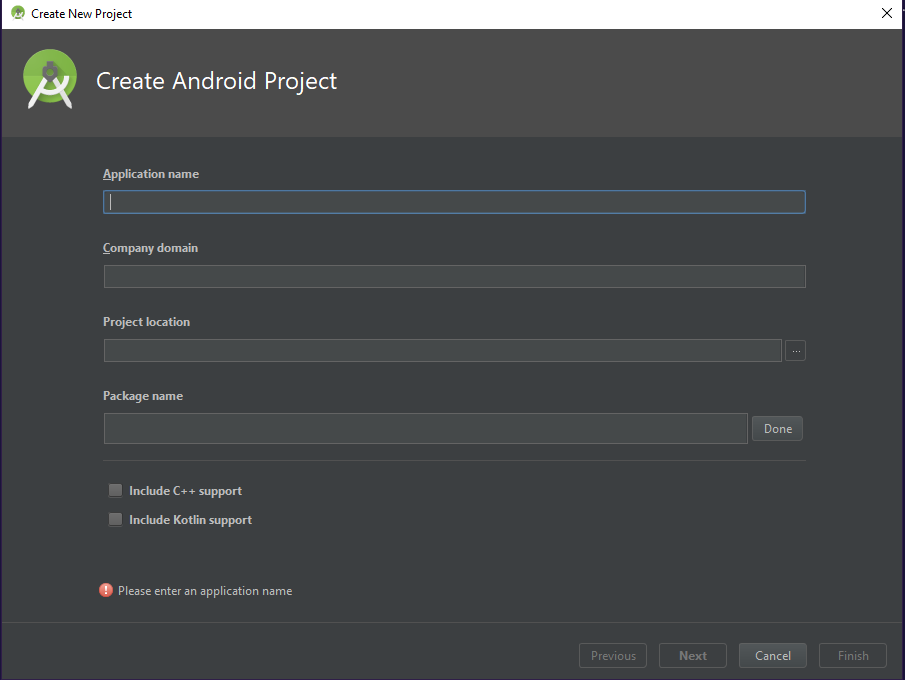
If you don’t meet the specifications recommended for the IDE, or you don’t want to download Android Studio the school computers all have the software on them. I would strongly encourage you to download the IDE on your personal computer as it will make debugging easier if you need to go see your professor.

## Creating a New Project

Upon launching Android Studio, you will be presented with the launch screen. Don’t be alarmed if the IDE takes a long time to load- this is normal

It is common to see the above loading screen for a while. Faster computers will have an easier time loading the IDE. 

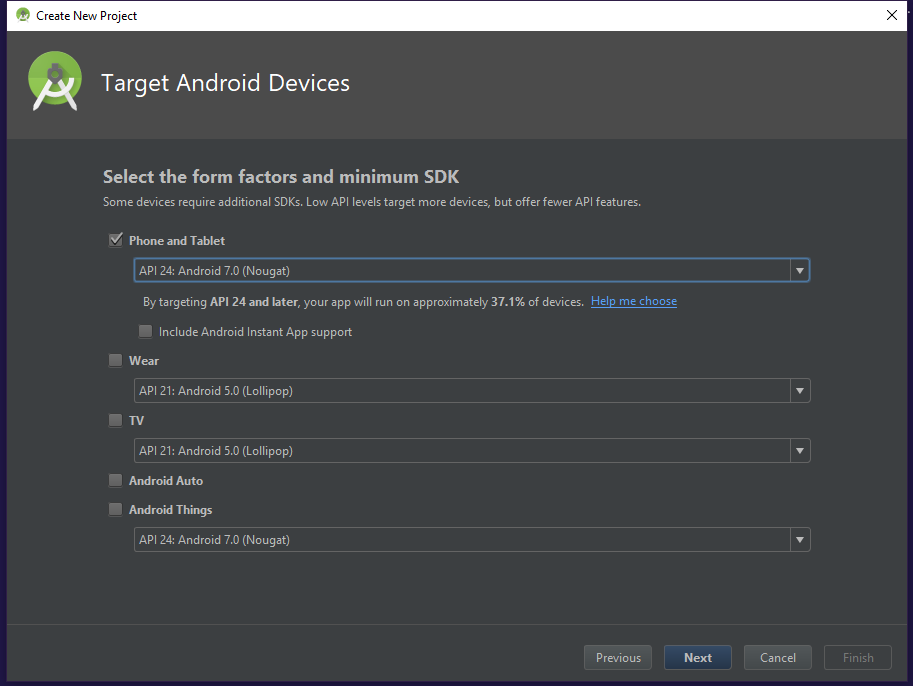
From the launch screen (above) choose the ‘Start a new Android Studio Project’ option. This will take you to the customization screen:

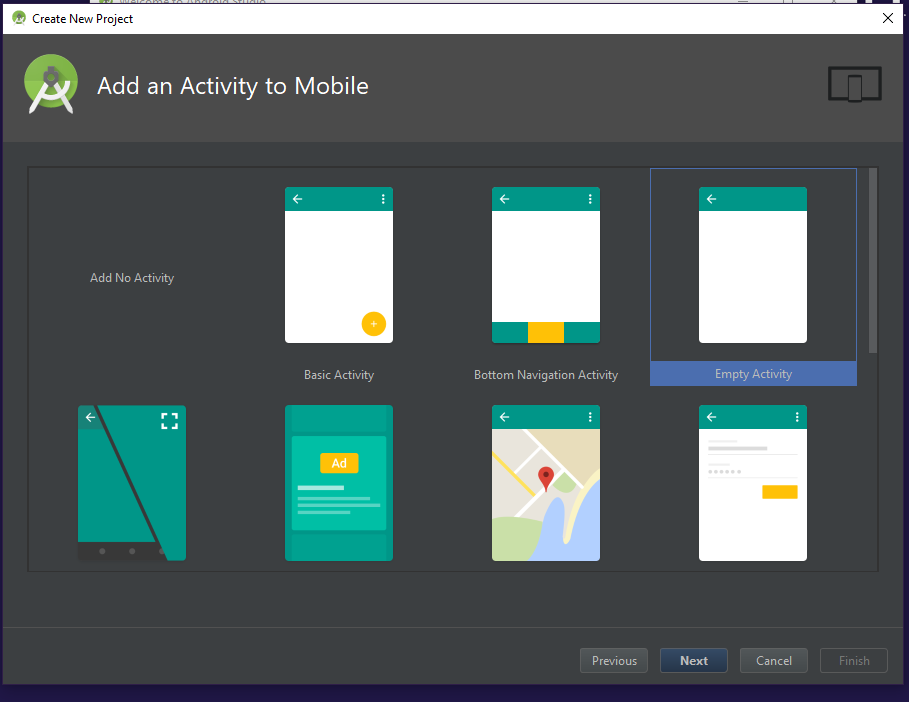


From here fill in the following fields:

**Application Name**, **Project Location**, and **Package Name**. The application name is what the project will be called, the location is where it will be stored, and the package name is the name of the default package when starting the project (if you don’t know what a package is- ask your professor). The package name can be changed later so don’t worry too much about this.

Next you will need to specify the target SDK for the project. Please use at least API 24, which is for Android 7.0 (Nougat). This is the minimum SDK for the tablets you are using. You can set it to be lower, however this is not recommended.

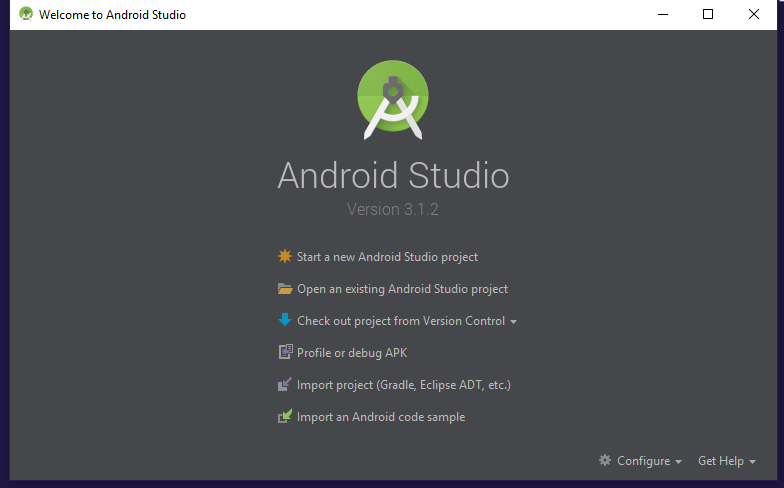
Once you have selected the target SDK the next screen has you select which type of activity you want. Depending on what your project is, you can select an activity accordingly. I usually like to select the Empty Activity.



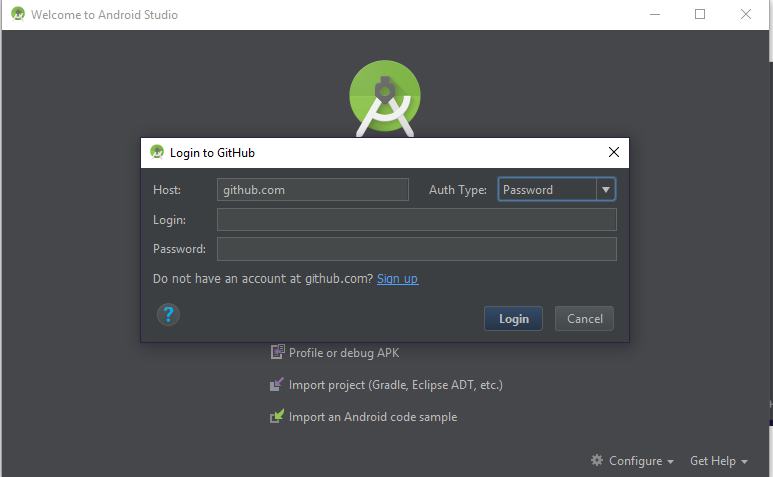
The final screen has you pick the name of the main activity: just leave this as the default. Click ‘Finish’ once you are done. Congrats! You have created your own new Android project!

## Cloning a Repository

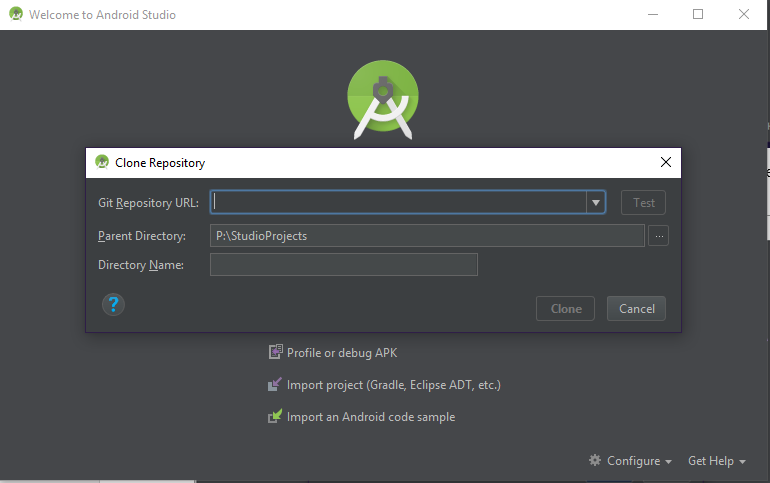
Cloning a repository from Android Studio is simple and straightforward. You will be cloning a repository as part of your project. On the launch page click the ‘Checkout a project from Version Control’ option and select GitHub.



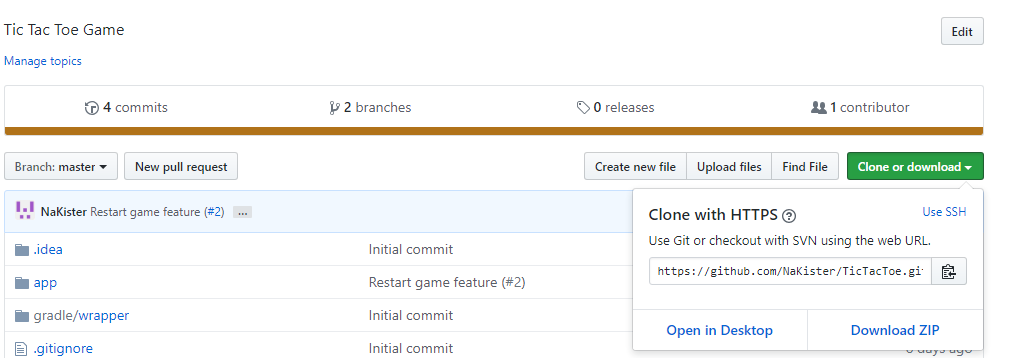
On the next screen change the ‘Auth Type’ to password and enter your GitHub username and password:



Once you have logged in you will be prompted with this screen:



You need to paste in the repository URL in the top box. The URL for your repository is located here:



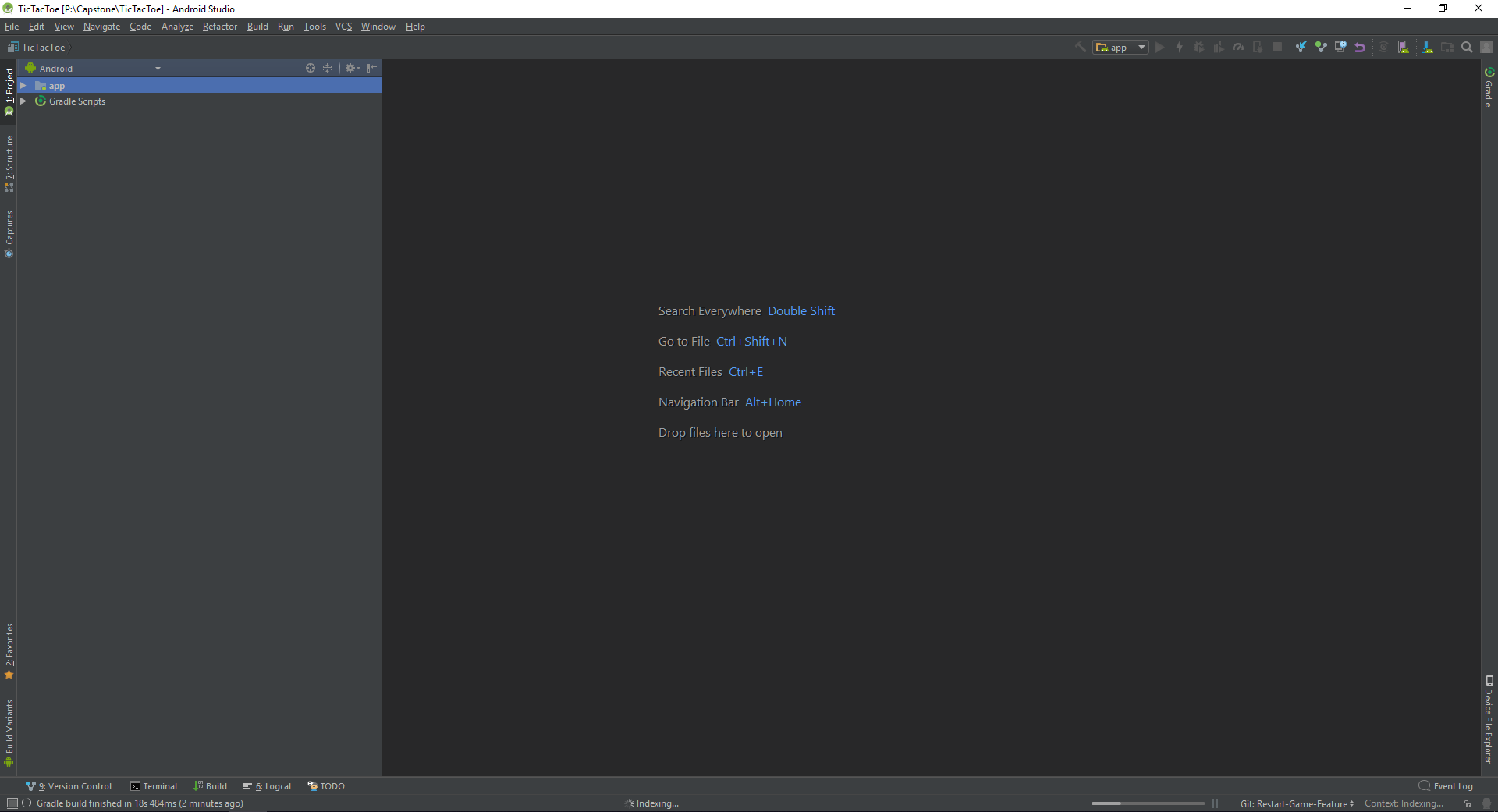
Paste this URL into the top box in the Android Studio UI. Choose the directory you wish to clone the repository to (the second box) and then the name you wish to save the repository as on your machine.

After the three boxes are filled out you can clone the repository. Once it is done cloning you can open the project.

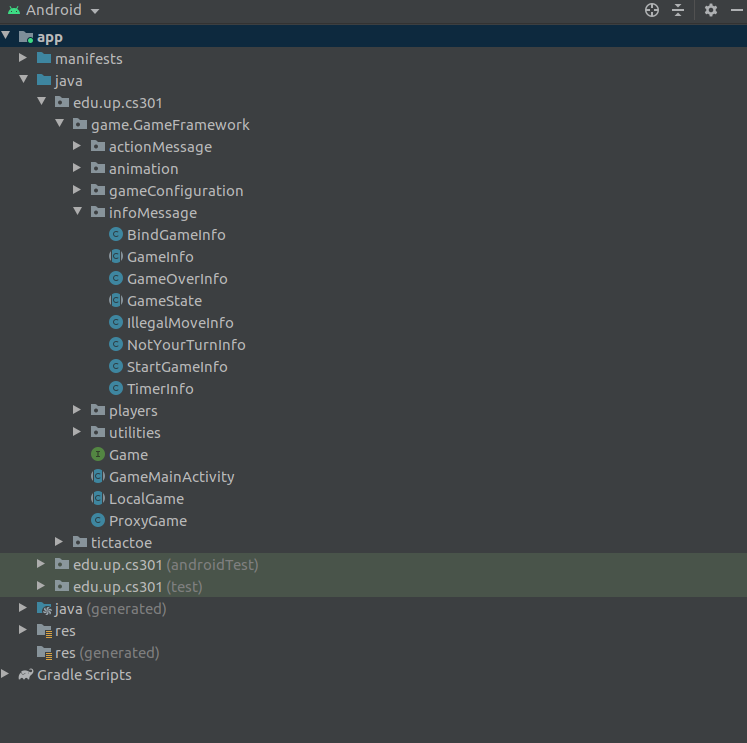
## Layout

Once you have your project cloned, you can open your project at any time by selecting the ‘Open an existing Android Studio project’ option.

Here is what the IDE looks like when you open a project:

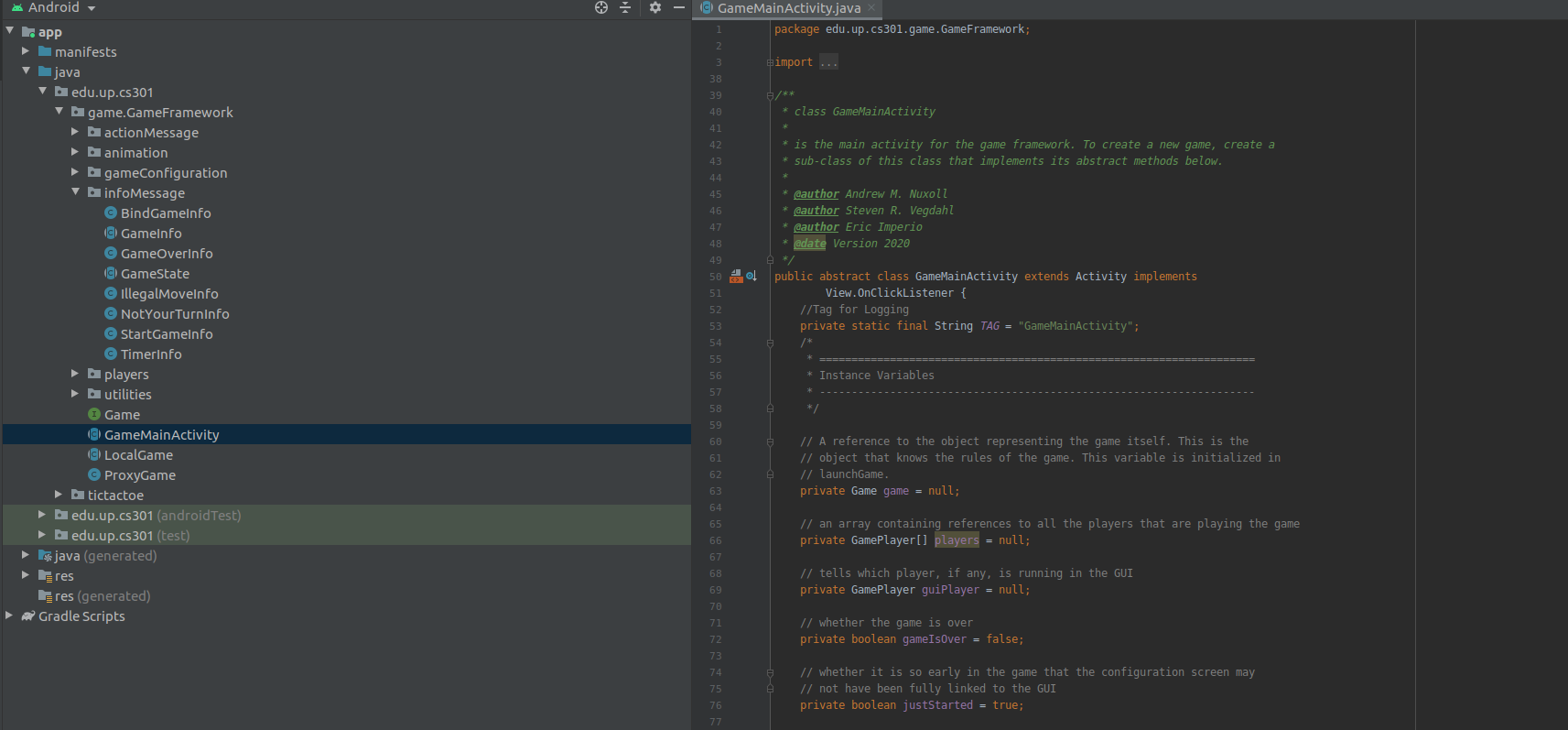
There are two main windows: the project layout and the code viewer.

On the left-hand side is the project viewer. This contains the layout of the classes and resources in the project.



The project viewer can be viewed in a variety of options; I’ve found that the Android and Project layouts are the best.

The main window is the code-viewer. Here is where the code is displayed. Double-clicking a class or resource will bring it into the main code viewer.



Double-clicking the name of the class in the code viewer will collapse the layout view to make the code section bigger.

This is the main layout with the default settings. As you become more comfortable with the IDE you will find a layout that works best for you.

The menu at the top of the IDE has some helpful features that will be explained in the next section.

The menu bar at the bottom has options to view the build status, the version control status, and the logging display.

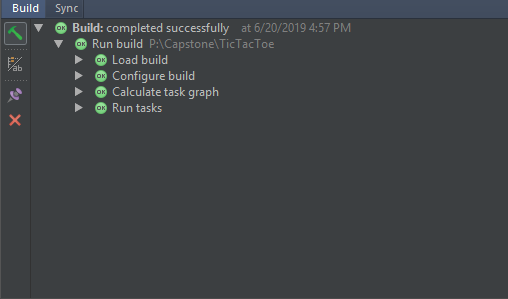


## Features

In addition to having a real-time compiler, there are many useful features built into Android Studio:

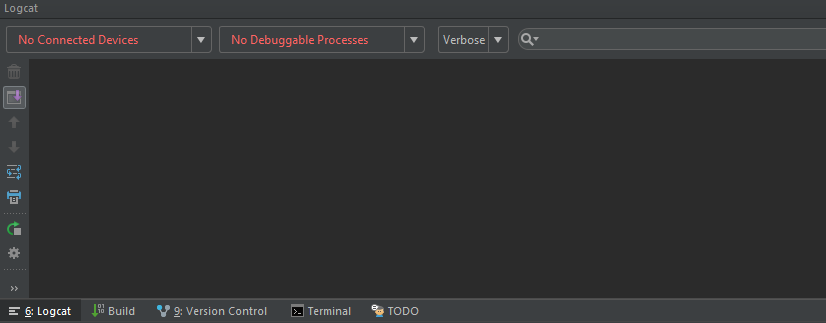
**Build Tab**

This tab displays the information of the current build:



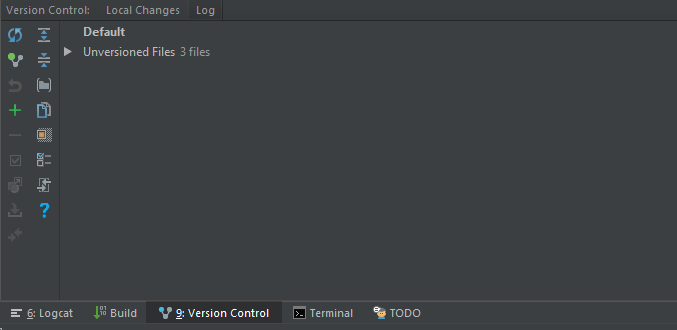
**LogCat**

This menu displays the logging information when the logger is running during app execution. When your app is running and you have logger statements in your code they will appear here:



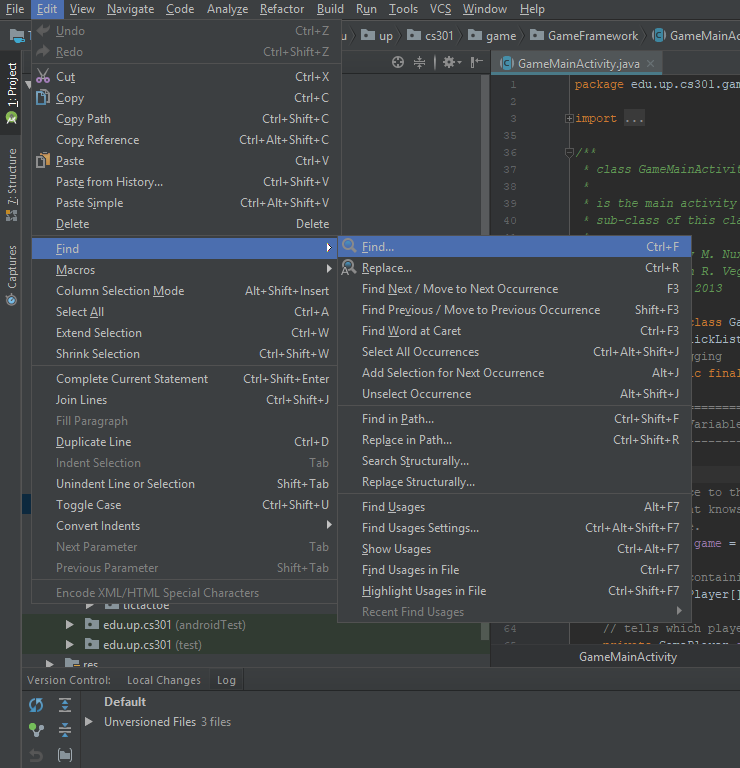
**Version Control**

This menu displays the files you have changed on your branch and any files that are un-versioned. It can help you decide which files to push:



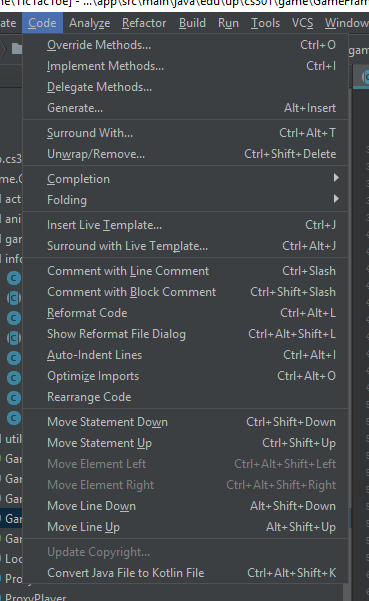
**Find/Replace**

This is helpful for finding and replacing pieces of code. This can be located under the EDIT menu at the top, or by the keyboard shortcut CTRL+F for a local find and CTRL+SHIFT+F to search the entire project.



**Code**

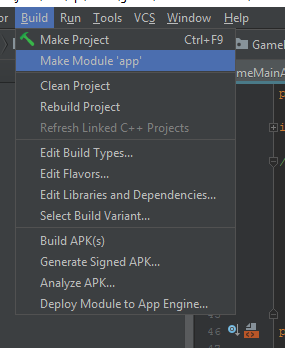
The code menu option has many different options to help you.



Some of these features are **Comment with Line Comment** and **Auto-Indent Lines**

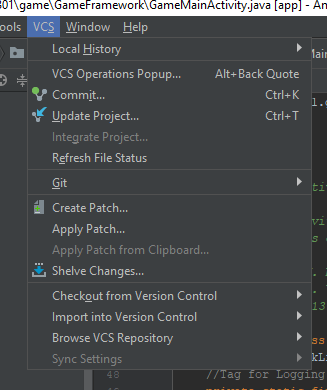
**Build**

This menu option can be helpful if your build fails. Cleaning and rebuilding the project can be a good option if something has gone wrong.



**VCS**

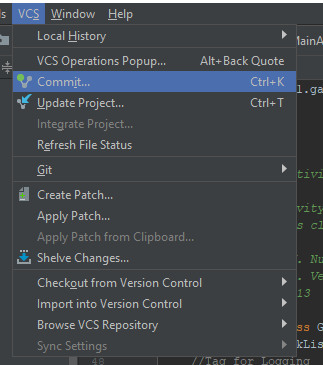
This is the version control area of the IDE. It will be explained further in the next section.



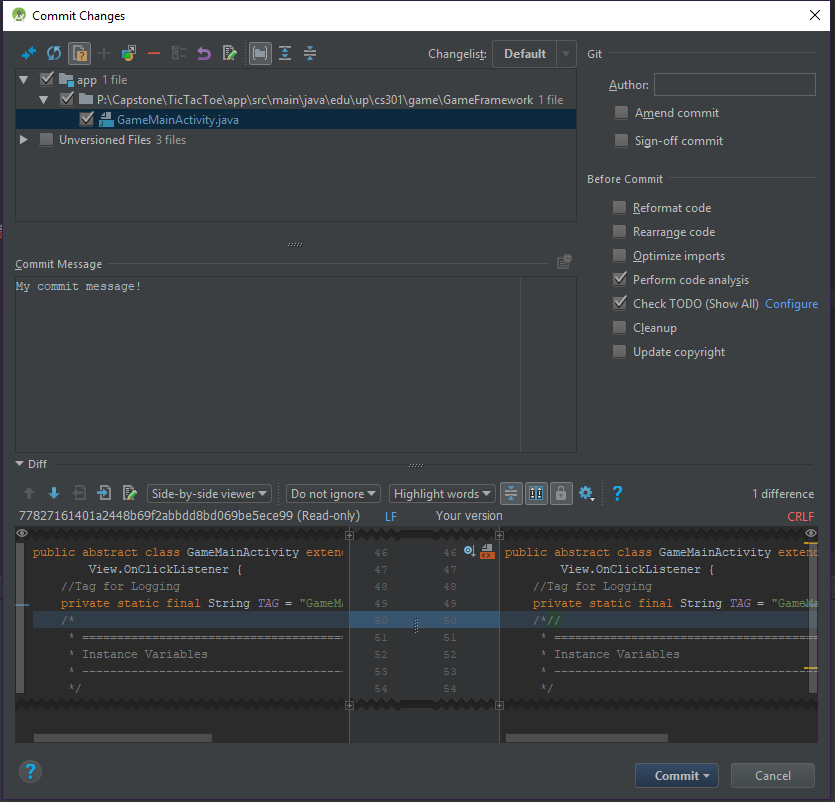
## Committing, Pushing, and Pulling

Committing, pushing, and pulling are some of the most important features you will be using in the IDE.

To commit changes, select the Commit option from the menu:

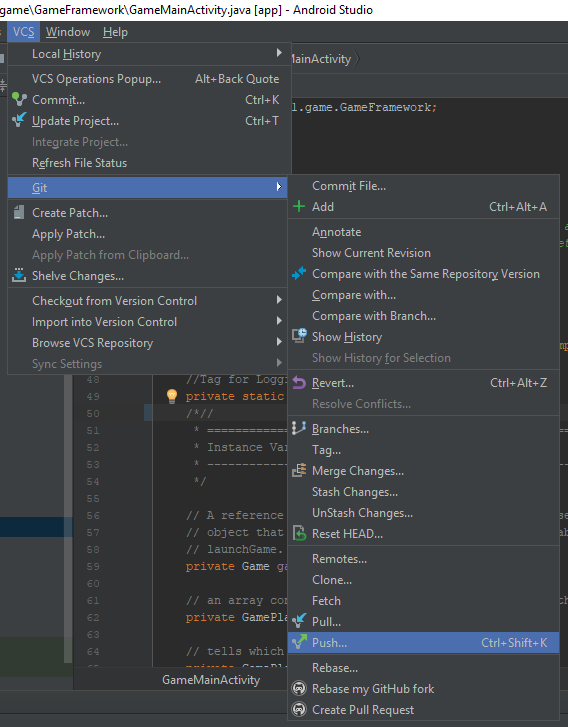


This will bring you to a menu that lists all the changes you have made. Select which changes you would like to commit in this menu and put a message in the message box.

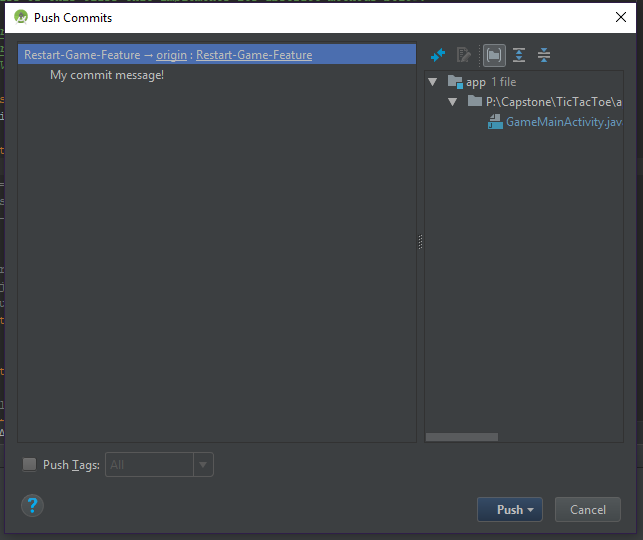


Once you have committed the changes you will need to make sure you push them. You can make more changes and commit again, but remember to push your changes in order for them to be updated on GitHub.

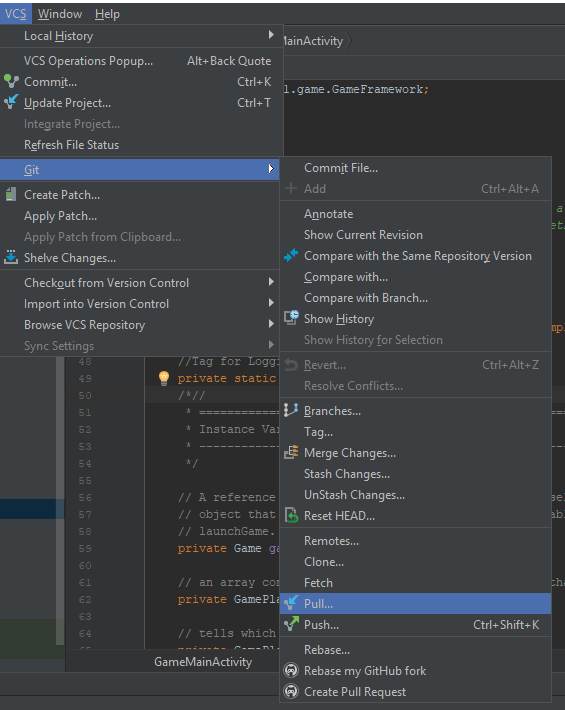
To push your committed changes, go to the Push option in the VCS menu:



Selecting this option will bring up the Push menu. It will display your commits and their details. From here you can select the Push option to push your changes to GitHub.



To pull changes from the remote branch, select the Pull option in the VCS menu:



This will update your local copy with the most up-to-date changes in the remote version of your project on GitHub.

# The Game Framework

This section will guide you through the classes provided to you are part of the framework. You will most likely not need to edit these files.

## What is the Game Framework?

The game framework is the foundation provided to you in order to build your game. These classes provide the basis of building your game, from network play to the startup screen.

## Where to Find the Framework

The framework is on GitHub. Which version of the framework you will use is dependent on your professor. See what your professor wants you to use before cloning the framework repository. Either way you will clone the framework repository from GitHub.

## Classes in the Framework

Classes in the main framework package are the main classes you need to interact with to create your game. You should extend these classes expect for ProxyGame. These include:

|  |  |
| --- | --- |
| Class | Description |
| Game | An interface that represents the game |
| GameMainActivity | Contains methods and functionality for the setup screen and setup. |
| LocalGame | Contains methods and functionality for enforcing game rules and other actions. |
| ProxyGame | For implementing network play –DO NOT EXTEND |

## Classes in the actionMessage Package

This package contains framework actions that are basic for game functionality. You will not need to edit these files.

|  |  |
| --- | --- |
| Class | Description |
| EndTurnAction | An action that represents the end of a player’s turn |
| GameAction | The basic action all other actions inherit from |
| GameOverAckAction | An action that represents the acknowledgement that the game is over |
| MyNameIsAction | An action that represents the player telling the game what its name is. |
| ReadyAction | An action that represents the player telling the game it is ready to begin playing |
| TimerAction | And action that represents a timer tick |

## Classes in the animation Package

The classes in this package are used for animations in the game. You will not need to edit these files.

|  |  |
| --- | --- |
| Class | Description |
| AnimationSurface | A surface view for the animator |
| Animator | The actual animator for the game framework |

## Classes in the gameConfiguration Package

The classes in this package are used for game configurations. You will not need to edit these files.

|  |  |
| --- | --- |
| Class | Description |
| GameConfig | For user specified configurations for playing the game |
| GamePlayerType | A type of game player |

## Classes in the infoMessage Package

The classes in this package are used for sending information to players and the game state. You will not need to edit these files.

|  |  |
| --- | --- |
| Class | Description |
| BindGameInfo | Sends information to the game that the player is ready to play the game |
| GameInfo | The base class for all information classes |
| GameOverInfo | Sends information to the player that the game is over |
| GameState | A representation of the current state of the game |
| IllegalMoveInfo | Sends information to the player that the move they tried to make is illegal |
| NotYourTurnInfo | Sends information to the player that is isn’t your turn |
| StartGameInfo | Sends information about the start of the game |
| TimerInfo | Sends information about a timer tick |

## Classes in the players Package

The class in this package is the base for the players you will add for your game. You should extend these classes expect for ProxyPlayer. These include:

|  |  |
| --- | --- |
| Class | Description |
| GameComputerPlayer | Contains methods for implementing the computer player |
| GameHumanPlayer | Contains methods for implementing the human player |
| GamePlayer | Contains methods and information present in all players |
| ProxyPlayer | For implementing network play – DO NOT EXTEND |

## Classes in the utilities Package

Classes in this package are features that are built in that can be used in your game. You do not need to edit these files.

|  |  |
| --- | --- |
| Class | Description |
| EqualityMethods | Provides methods for comparing arrays. Make sure any of your own objects passed to these methods have equals methods. |
| FlashSurfaceView | For animation help |
| GameTimer | The game timer |
| IPCoder | For coding/decoding IP addresses |
| Logger | Can be used for advanced logging for debugging |
| MessageBox | Creates a popup box on the UI |
| NetworkObjectPasser | For passing objects over the network |
| Saving | Provides Saving methods for the Framework. You shouldn’t need to call these yourself. Also provides separators for any toString methods you write (Anything that is part of the GameState will need a toString method). |
| Tickable | For handling game timer ticks |

## Advanced Features of the Framework

### Setup Phase

This is an optional feature if your game has a setup phase portion. In your LocalGame you can set the number of setup phases you wish to have and then the game will be in a setup phase for the number of turns you specified. Once these turns have elapsed the game will start as normally.

If you have other questions about setup phase, ask your professor.

### Custom Game Rules

This is an optional feature that will allow you to add custom rules to your game, such as how many accusations one can have in Clue before losing. In order to setup custom rules you will have to edit the xml for the main setup menu as well as the MainActivity specific to your game in order to complete this for your game.

If you have other questions about the custom game rules, ask your professor.

# Your Game!

## Cloning the Base Repository

To clone the repository, follow the steps in the Android Studio section: ‘Cloning a Repository’. The base repository you will need to clone depends on your professor. Ask your professor which repository to use as the basis of your game.

## Which Classes do I Need to Inherit/SubClass?

The main classes you need to Inherit are:

* GameMainActivity
  + For setting the default configuration of your game
* LocalGame
  + The enforcer of the game rules
* GameState
  + The state of your game
* MoveAction
  + The actions that your players will take in your game
* GameComputerPlayer
  + For creating the computer players
* GameHumanPlayer
  + For creating the human player and GUI

Inheriting these classes and making game-specific versions of these classes will be the beginning of creating your game.

# Help! Everything is Broken!

Sometimes stuff doesn’t work- and that’s okay! Here are some things we’ve noticed while working on this project:

## IDE Errors

Sometimes Android Studio will have issues compiling or loading packages. When this happens it is best to rebuild the project. If these issues persist, try closing and reopening Android Studio. If issues still occur after trying these steps please see your professor. Something else may be happening that may require further investigation.

## GitHub Errors

If you have a Git error, it is usually due to a conflict. This means you have edited files that someone else has edited. Merge conflicts can be resolved by following these steps:

1. Checkout the Master branch
2. Pull the Master branch
3. Checkout your branch
4. Merge Master into your branch
5. When conflicts occur, use the UI to select which changes to keep.

Once you have resolved the merge conflicts you can continue to develop as normal.

If other Git issues happen, please see your professor.

## Debugging Tips

When debugging it can be helpful to place statements to print information at different parts of the code. Using the Logger provided by the framework is a good option because it will not only print onscreen if you want it will also print in the LogCat menu in the IDE.

# FAQ