Day 3
Saturday, July 9, 2022 12:21 PM

Git fundamentals

Version Control System
- Keep track of every change made to a project
- Provides a collaboration tool

2. Types Version Control Softwares
- Central Software Control
  - Stored in one location
  - No copies of the original
- Distributed Software Control
  - Everyone has access/can make a copy of the original files
  - Every developer can pull modify then push changes, other developers can sync those changes by pulling

Git is Distributed Software Control System

Git hosting Platforms
- We have local repositories
- To share your project with others we remote server to store these
- Popular choices for hosting repos:
  - Github
  - Gitlab
  - BitBucket
  - Kracken
- To add/update the remotely hosted repo we "push" comits
- To add/update our local repo we "pull"

Git Initializing Your Repo
- Setup your git credentials on your machine

```
Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git
$ git config --global user.name elmcgill

Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git
$ git config --global user.email ethan.mcgill2016@gmail.com
```

- first time you attempt to push you will be asked for your password, then windows/git store the credentials

Initialize our local repo with git init

```
Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git
$ git init
Initialized empty Git repository in C:/Users/Ethan/Desktop/220711-JAA/Hello-Git/.git/

Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git (main)
$ ls

Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git (main)
$ ls -a
./  ../  .git/
```

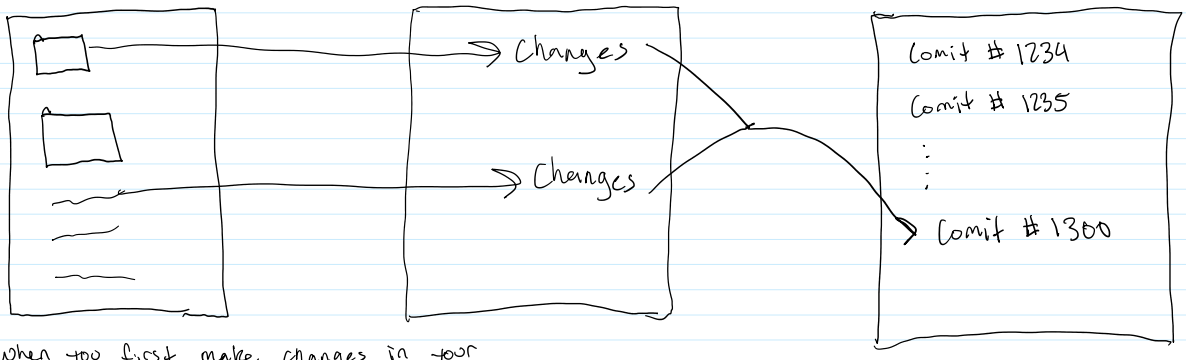
- all the information about the changes in our project will be stored in the hidden .git folder

Git Snapshotting Basics:
- We have a working flow in git

Working Directory
- folder containing the project files and .git folder

Staging Area
- Keep track of the current changes

Repository
- Stores the comits
  - Snapshots of our project



When you first make changes in your working directory git has no clue about
- to check if we have untracked files we use git status

```
Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hellogit.txt

nothing added to commit but untracked files present (use "git add" to track)
```

When you want to track a file you use git add
- git add file file...
- git add .

```
Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git (main)
$ git add hellogit.txt
warning: in the working copy of 'hellogit.txt', LF will be replaced by CRLF the next time Git touches it
Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git (main)
$ git add .
warning: in the working copy of 'src/sourcecode.txt', LF will be replaced by CRLF the next time Git touches it
```

Whenever we use git add the files/changes are added to the staging area

When you are happy with your proggress you can make a commit
- Comit is a snapshot/version of your project

To make a commit you use the command git commit

```
Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Hello-Git (main)
$ git commit -m "my first commit"
[main (root-commit) 318dcb6] my first commit
 2 files changed, 5 insertions(+)
 create mode 100644 hellogit.txt
 create mode 100644 src/sourcecode.txt
```

- We need to include a quick message describing the changes
  - Quick, concise, but descriptive
  - Commit as often as possible

Working with remote repositories
- we need git hosting sites to easily share our code
- super easy to create a remote repo
1. login to github
2. Click the "create repo" botto
3. follow directions

Using the remote Repo

- If we need to get the code for the first time
  - We will "clone" the repo with git clone

```
Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Copy-Repo
$ git clone https://github.com/220711-UTA-SH-JAA/Hello-Git.git
Cloning into 'Hello-Git'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

Ethan@DESKTOP-3OQVP3E MINGW64 ~/Desktop/220711-JAA/Copy-Repo
$ ls
Hello-Git/
```

Once we have access to our repo here is the normal flow

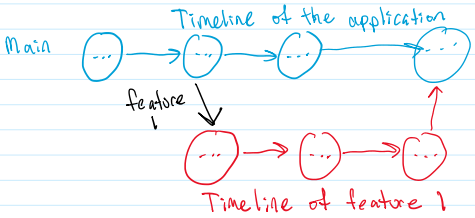
1. Create/make changes to files in our working directory
2. track the changes with git add
3. create a commit with git commit -m
4. push the new commit to the remote repo with git push
5. Teamates git poll the changes

Git Branches

Seen as separate versions of your app
- series of commits

Every Repo has a main branch "main" or "master"
- These should be where we store the current working version of our software

The idea is that we can branch off of the main branch to experiment and never break the application
- branches are typically made for implementing new features



To create a branch in git we can use one of two commands

1. git branch name
2. git checkout -b name
   - git checkout switches to the branch specified
   - -b flag creates the branch if it doesn't exist

The first time you attempt to push code in a new branch, the branch won't exist remotely

git push --setupstream origin branchname

How to combine branches
1. git merge command
2. poll requests
   - preferred method is poll requests

If multiple developers change the same file and you try to push/merge this could cause a merge conflict
- These must be fixed by choosing the preferred code then pushing

Intro To Java

What is a program
- A set of instructions for our computer to run

Programming Languages
- Used to write more complicated programs
- Theses languages give power to convert our language to one a computer will understand

Two main "levels"
- Low level
  - Closer to how computers see instructions
- High level
  - Closer to human language
- Theres is sort of a spectrum

Spetrum of programming languages (Low to high)

Machine Language
    10011010...

Assembly Language
    mv r1 r2
    ld r2 ...
    ld r3 ...
    add r2 r3 r4

C Programming Langauge
- has human readible syntax
- but memory management is still manual

Java, C#, Pythons
- Very human readible
- auto memory management

What is Java
- Programming Language
- High leveled
- Strongly typed
- Objected Oriented
- Managed by Oracle
- Compiled

Why Java
- Very large and well supported
  - 3rd most used programming
- Relatively easy to learn
  - But still powerful
  - C based syntax (human readible)
  - But abstracted away C's complexities
- Platform independent *
  - Write java code on one machine and run it any where
  - The code runs on a virtual machine installed when you install java
- Automatic memory management
- Built around/with Object Oriented Programming in mind

Object Oriented Programming
- Programming paradigm focused on constructing objects in code
- The car sitting in the driveway can be modeled in code

4 main Pillars/Principles of Object Oriented Programming

Abstraction
- Separating the implementation and the declaration of the functionality in a program/object

Polymorphism
- take on many forms
- this gives the programmer the ability to write objects that relate to each other and switch between those
- Ties in heavily with the next pillar

Inheritance
- The ability of one object to inherit traits from another object
- This will create a hierarchy
- When using inheritance this creates an is-a relationship

Encapsulation
- hiding data or behaviors from the outside world
- most straight forward because you are just making these object members private

Next week we will see specific implentations of all these in Java

Java Code files
- Java code lives in files denoted with the

of all these in Java

Java Code files
- Java code lives in files denoted with the .java extension

Java is a compiled language so our PC's can't read the language in the java files
- We must compile our java code into something machine readible

When we compile, it gets converted into byte code stored in a .class file

We compile using java tools we get when we install Java

Java provides tools such as compilers, debuggers and other libraries in the JDK, JRE, and JVM

JDK Java Development Kit

Contains everything needed to Develop a Java Application
- Includes the compiler and debugger
- Includes other kits such as the JRE

Inside of the JDK there is a script/command called Javac
- When run correctly will take a java file and output a class file
- aka how we compile java code

JRE Java Runtime Environment

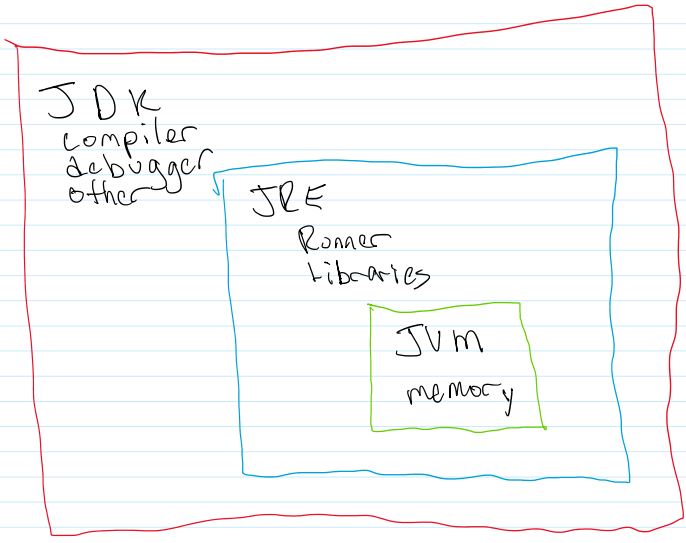
So once you have a .class file we need the JRE to run it
- the bare minimum to run a java application

To run a java program the JRE comes with the java command
- java classfile args
  - this will run and print to the console assuming the class file is setup correctly

The JRE also contains the magic behind write once run anywhere

JVM Java Virtual Machine

Is contained within the JRE and takes our .class file and converts to machine specific instructions
- Is how your program actually gets ran
- this is where the memory for your application is
- Whenever you install java the correct VM is installed for your computer



Complete Java Development Process

1. Write Java source code in .java files
2. Compile the .java file into a .class file using the JDK javac command
3. Take the main .class and run it with the java command from the JRE
4. The java command calls upon the JVM to run the java application
5. JVM runs the code and outputs to the console