Lab 1

IntelliJ, Git (Setup)

Welcome to CS61B!

We’re going to do some cool things here :)

CS61B in a Nutshell

This is a data structures course!

- We'll be covering a variety of topics in this course: debugging, linked lists, hash tables, asymptotics, and a lot more!

In this course, we'll also be focusing on efficiency of writing and running programs.

Introductions!

Announcements

Announcements

Homework 0A is due Friday, 8/30 at 11:59 pm

Lab01 is due Friday, 8/30 at 11:59 pm

Homework 0B is due Tuesday 9/3 at 11:59 pm

Pre-Semester Survey is due Friday 9/6 at 11:59pm

Project 0 is due Friday 9/6 at 11:59 pm

Beyond these deadlines, please make sure to read through the syllabus in its entirety and ensure that you understand the policies (you will need to read through it for lab01).

Announcements - What to expect during lab

We use `oh.datastructur.es` to queue up. Questions are welcome!

We tend to lead you to the answer, rather than give it to you right away. Today is setup, so we'll be more direct, but we expect you to think more in future labs!

Additionally, we generally only take lab-related questions in lab sections. For help on other assignments, go to office hours, which usually runs concurrently alongside labs.

Java Basics

Python: HelloWorld

```
print(“Hello World!”)
```


Java: HelloWorld

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

So, what's special about Java? Let's walk through this!

Java: HelloWorld

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

This denotes that you're creating a class. Everything in Java lives in a class!

Java: HelloWorld

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The name of the class is HelloWorld.

Java: HelloWorld

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The method name here is main, where the code to print “Hello World!” is executed.

Java: HelloWorld

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

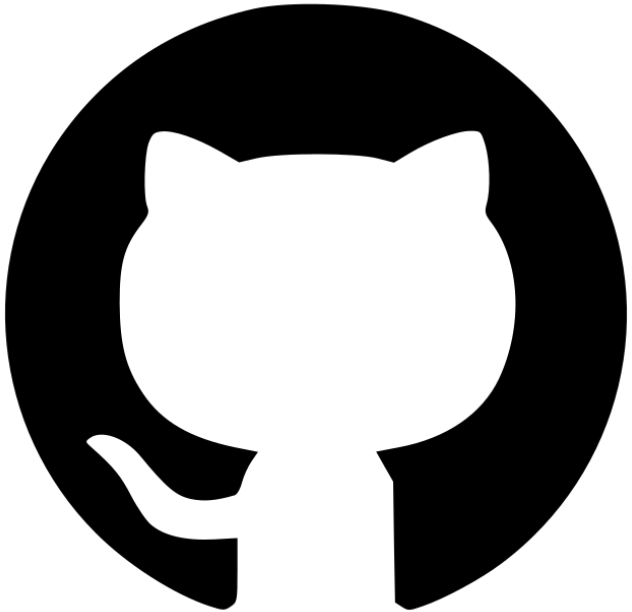
This is the statement to print “Hello World!”. Notice that it ends with a ; (used to separate statements).

Java: HelloWorld

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The brackets denote sections of code (i.e. the method belongs inside of the class).

Git(hub)



Git

Git is a distributed version control system.

Git

Git is a distributed version control system.

- What does that mean?
  - Version Control System: Tracks all changes to files over time.
  - Distributed: Every developer's computer stores the entire history of the project.
- The entire history of a project is called a **repository.**

Git

Git is a distributed version control system.

- What does that mean?
  - Version Control System: Tracks all changes to files over time.
  - Distributed: Every developer's computer stores the entire history of the project.
- The entire history of a project is called a **repository.**

Throughout this course, you'll be using Github to host all your work and to submit your assignments. And with Git, you'll be able to push all your work onto Github.

The Main Git Commands You Should Know

`git init:` Initializes a git repository in current directory.

`git add <file_name>:` Adds a file to staging (to be committed).

`git commit -m "Message":` Saves all staged files into a commit (like a snapshot of your current repository)

`git status:` Displays state of the repository and staging area (tracked, untracked files and changes).

`git log:` Displays the history of our *committed* history.

Git Commands Continued

`git restore:` Restores files to their versions in the most recent commit.

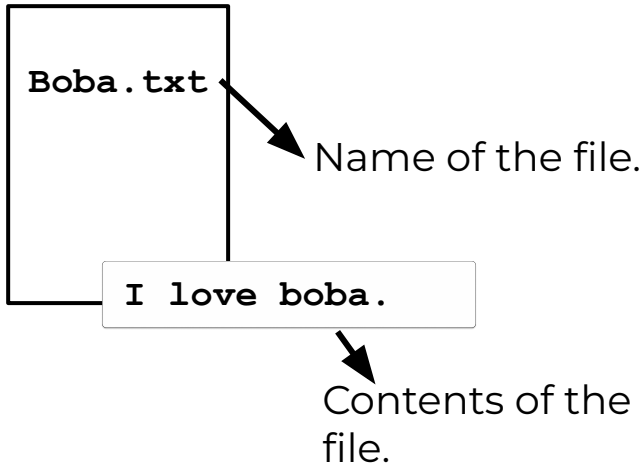
`git restore --source=[commitID]:` Restores files to their versions in the given commit.

`git push <repo> <branch>:` Takes commits on your local computer and pushes it into the remote repository (git push origin main)

`git pull <repo> <branch>:` Pulls any changes from the remote repository onto your computer (git pull skeleton main)

Visualization: https://git-school.github.io/visualizing-git/#free-remote

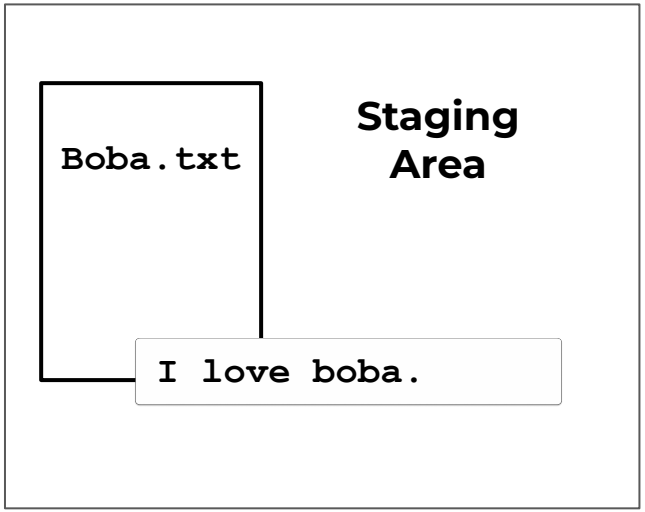
Local Repository



Let's say we have a file called **`Boba.txt`** in your local repository (you can think of your repository as another file on your local computer).

Remote Repository

Local Repository



Let's run the following:
`git add Boba.txt`

We've now added the file to the staging area (aka, to be committed).

Remote Repository

Local Repository

Let's run the following now:
`git commit -m "Done!"`

At this point, we've taken a snapshot - the files in the staging area have been tracked by our local repository.

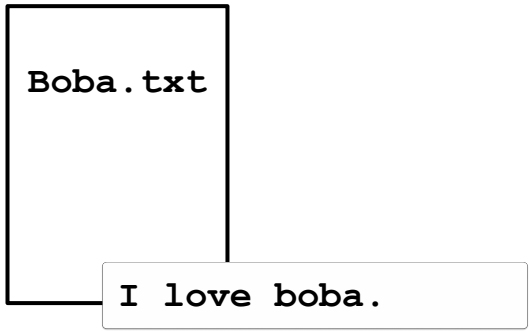
Remote Repository

Local Repository

In the event that we somehow lose our laptop, we want to be sure we saved our work and progress - so we add our work into a remote repository:

```
git push origin main
```

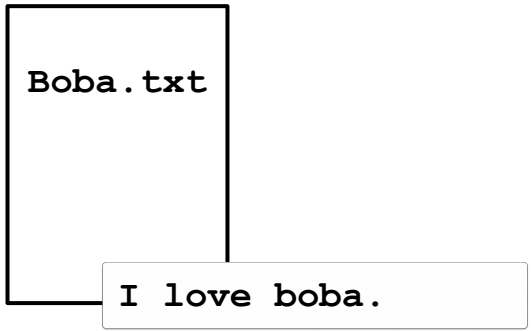
Remote Repository



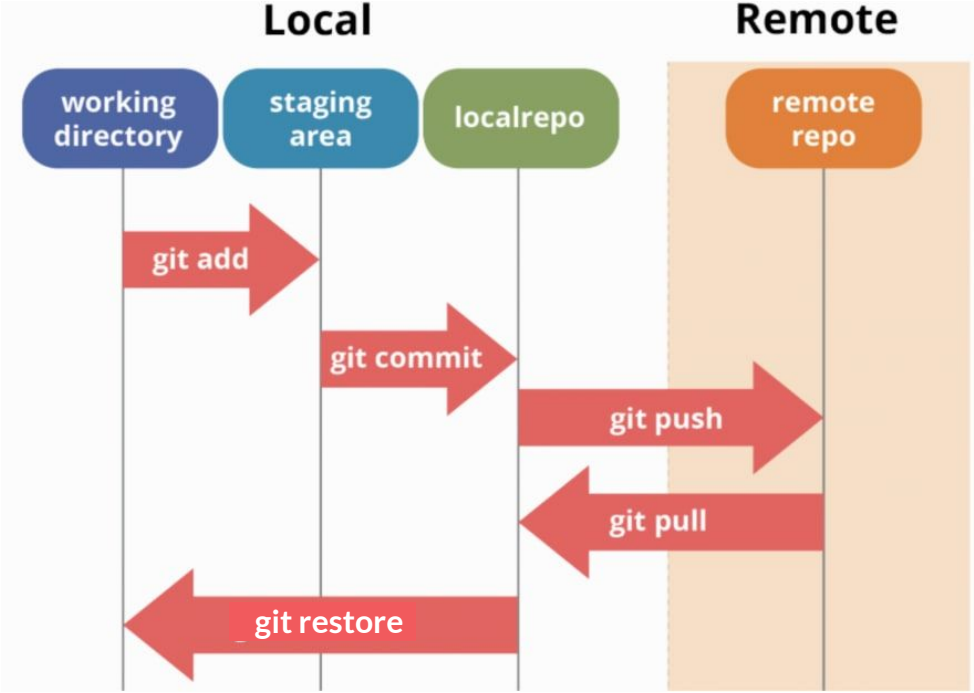
Local Repository

At this point, we've saved our progress into our remote repository! Whoo! If we ever want to get something from our remote to local repository, we would use **`git pull [repo] [branch]`**

Remote Repository



The Basics of Git



Lab Overview

An Overview

Lab 1 is due Friday, 8/30 at 11:59 pm.

- Make sure to read lab 1 carefully! There's a lot of information, so it might be difficult to digest on first pass. That's okay! Go at your own pace and we'll be here to help :)

Deliverables:

- **`Arithmetic.java`**
- **Your lab setup! Make sure you've set up IntelliJ and have completed the setups detailed in the lab (including authentication with Beacon - more details in the spec).**

Some Tips for Setup

You can run through this if you're having trouble with setup (check out the FAQ too!).

- When opening up your assignment through IntelliJ, make sure you select the lab folder itself, not the repository (ex. open lab01 instead of fa24-s*** directory)
  - Right click on the src folder → Mark Directory as → check that the src folder is marked as Sources Root
  - Right click on the tests folder → Mark Directory as → check that the tests folder is marked as Test Sources Root
- Make sure you have the proper settings configured:
  - Go to File → Project Structure and check the following (in IntelliJ Ide):
    - **Check that the language level is correct (version 17+)**
    - **Check that there is a path for Compiler Output (default is out folder)**
    - **Check that you have the library-fa24 imported under Libraries tab**
- If you think your setup is correct, then try going to File → Invalidate Caches and restart the IntelliJ IDE. That may help if you're unsure where certain errors might be showing up.