Introduction to Programming, IDEs, and Eclipse!

MIST352

THIS IS YOUR GUIDE TO PREPARE YOUR COMPUTER TO CODE FOR MIST352

Table of contents

What is Programming?

- Programming is the art of instructing computers to perform tasks.
- It involves providing clear and structured commands that a computer can understand and execute.
- In today's technology-driven world, programming is a fundamental skill with diverse applications.
- We write “source code” of programs, the computer translates those into something “cool”.

Source Code Files: Fancy Text with Extensions

- At its core, source code is composed of plain text files.
- These files contain human-readable instructions written in programming languages.
- The choice of extension, such as .java for Java or .py for Python, indicates the language used.

Introduction to IDEs

- An Integrated Development Environment (IDE) is a software suite that combines various tools to aid developers in creating and managing code.
- IDEs offer features like code editing, debugging, version control, and more, all within a unified interface.

The Need for IDEs

- While basic text editors can be used for programming, IDEs offer a more efficient workflow.
- They provide tools that enhance productivity, such as code completion, error checking, and integrated debugging, making development smoother and faster.

Eclipse IDE and Its Significance with Java

- Eclipse is a widely-used IDE known for its robust support of Java development.
- Its rich features, such as project management, code navigation, and plugin integration, make it a favored choice among Java developers.
- Download and install Eclipse

Eclipse as a Java Compiler

- Eclipse not only serves as an IDE but also includes a built-in Java compiler.
- This compiler translates human-readable Java code into bytecode, which is executed by the Java Virtual Machine (JVM).

Compilers vs. Interpreters

- Compilers and interpreters are tools for translating code into machine-readable instructions.
- Compilers, like the one in Eclipse, convert code to bytecode before execution.
- Interpreters, on the other hand, execute code line by line, as seen in languages like Python.

Benefits of Using Eclipse for Java

- Eclipse's benefits include code auto-completion, real-time error checking, and powerful debugging tools.
- Its integrated nature simplifies coding and debugging, ultimately leading to more efficient and reliable software development.

Eclipse's Advanced Features

- Eclipse offers advanced features like code templates and refactoring.
- Code templates allow for quick generation of commonly used code patterns, while refactoring tools help maintain code quality during development.
- It also helps us detecting syntax errors.

Debugging with Eclipse

- Eclipse's debugging tools enable step-by-step code execution, breakpoints, and variable inspection.
- These features are invaluable for identifying and resolving issues within your code.

Eclipse Marketplace and Plugins

- The Eclipse Marketplace hosts a variety of plugins that extend Eclipse's functionality.
- These plugins cater to different development needs, such as:
  - web development
  - database management, and more.

Introduction to Version Control

- Version control are computer systems that track changes to code over time, enabling collaboration and safeguarding against mistakes.
- They ensure that developers can work on projects simultaneously without overwriting each other's work.

Git and GitHub

- Git is a distributed version control system that tracks changes locally.
- GitHub, a platform built around Git, offers remote repositories for collaborative development, code review, and issue tracking.

GitHub Integration with Eclipse

- Eclipse seamlessly integrates with GitHub, simplifying version control within the IDE.
- This integration streamlines tasks like committing changes, branching, and merging.
- Using this feature is OPTIONAL in our class. Instead, we will use the command line to link your projects to your GitHub account.

Collaborative Development with Eclipse

- Eclipse's GitHub integration enhances collaborative development.
- Developers can work on the same project, manage changes, and resolve conflicts efficiently, promoting teamwork and code stability.

Eclipse's User Interface

- Eclipse's user interface is divided into perspectives, views, and editors.
- Perspectives customize the workspace for specific tasks, while views and editors provide tools for code editing, project management, and more.

Eclipse Shortcuts and Productivity Tips

- Eclipse offers numerous keyboard shortcuts and productivity tips for efficient coding.
- These shortcuts expedite tasks like navigation, code selection, and compilation.

Eclipse for Other Languages

- While renowned for Java development, Eclipse supports various programming languages through plugins.
- Developers can tailor Eclipse to their preferred language, such as Python, C++, and more.

Eclipse Community and Resources

- Eclipse boasts an active and vibrant community.
- Online tutorials, forums, and documentation provide assistance and insights for both newcomers and experienced developers.

Alternatives to Eclipse

- While Eclipse is prominent, other IDEs like IntelliJ IDEA and Visual Studio Code offer unique features.
- Developers often choose IDEs based on personal preferences and project requirements.

IDEs in Professional Software Development

- In professional software development, IDEs play a pivotal role.
- They streamline coding, debugging, and testing, leading to faster development cycles, fewer errors, and improved software quality.

Future Trends in IDEs

- IDEs are evolving with emerging technologies.
- AI-powered code suggestions, automated refactoring, and enhanced collaboration tools are shaping the future of IDE development.

Conclusion

- In this presentation, we explored the world of programming, IDEs, and the Eclipse IDE.
- IDEs like Eclipse are essential tools for modern developers, providing an integrated environment for efficient and collaborative software development.

What is GitHub?

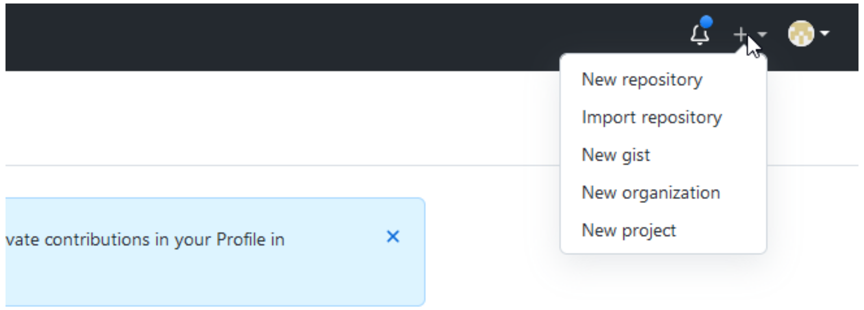
- So, we write code locally using Eclipse, how can we allow others to see this code?
- How can we allow others to even contribute to this code?
- The answer is the cloud.
- GitHub is a “cloud” service that allows you to save code and maintain different “version” of the code
- GitHub is a cloud based version control system.

How does GitHub work?

- Simple, it allows you to have an online repository to save –and share- any data you want
- Mostly used for coding projects
- In our class, you will be submitting all of your homeworks, in-class tasks, and other coding assessment to GitHub.
- You need to create an account using your mix account
  - Then you can create a repository, see next slides.

Create GitHub repository for MIST352

1. Go to GitHub.com and create an account using your mix account.
2. Click on the (+) sign on the right top corner of your repository and click New Repository as shown below.



vate contributions in your Profile in

Create GitHub repository for MIST352

3.Name your repository EXCATLY like this:

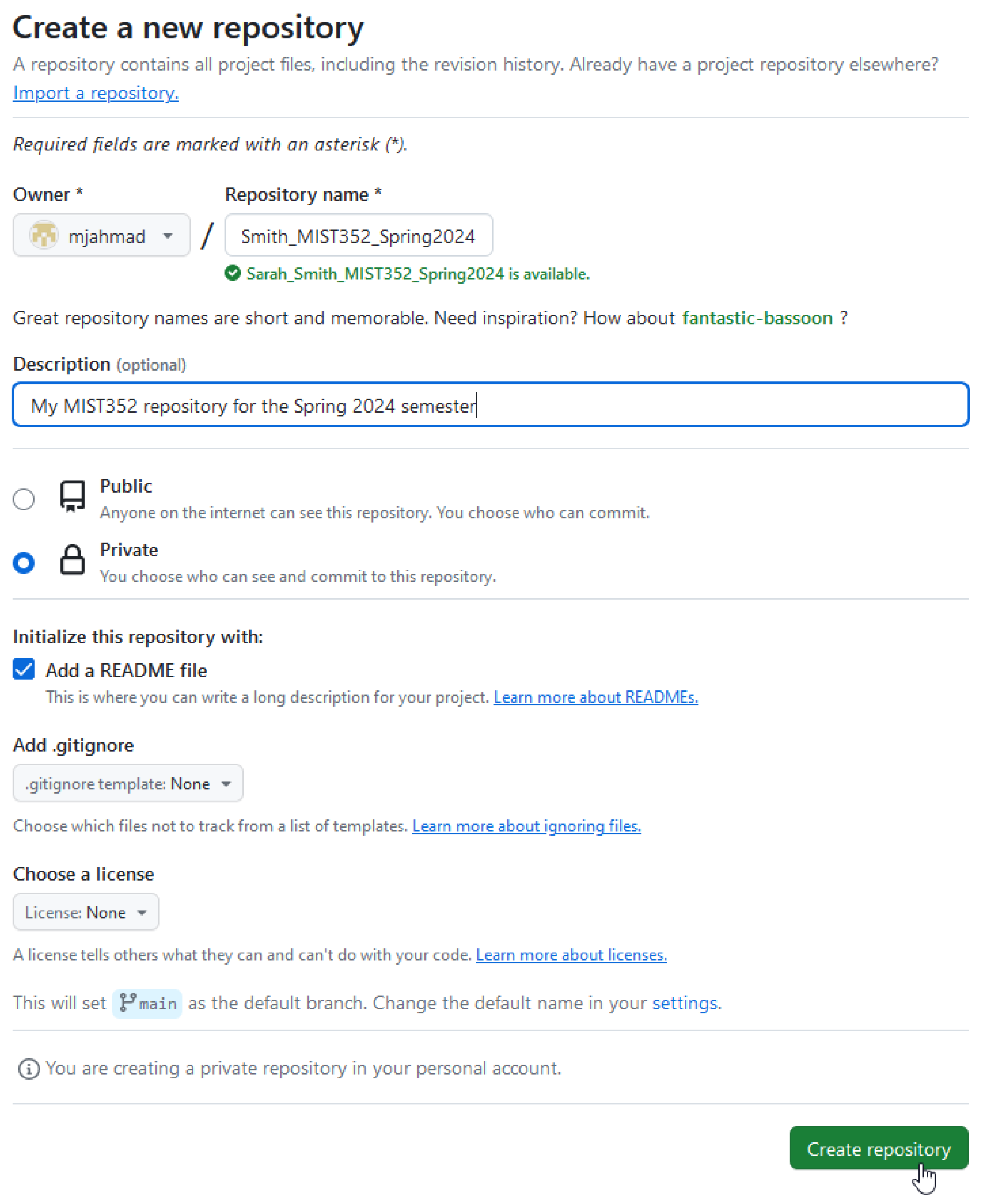
[First Name]_[Last Name]_MIST352_Spring2024

So if your name is Sarah J. Smith, then your repository should be named

Sarah_Smith_MIST352_Spring2024

4. Make your repository private

5. add a README file. And then create repository.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner *: mjahmad / Repository name *: Smith_MIST352_Spring2024

Sarah_Smith_MIST352_Spring2024 is available.

Great repository names are short and memorable. Need inspiration? How about fantastic-bassoon ?

Description (optional): My MIST352 repository for the Spring 2024 semester

( ) Public
Anyone on the internet can see this repository. You choose who can commit.

[✓ Private]
You choose who can see and commit to this repository.

Initialize this repository with:

[✓] Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

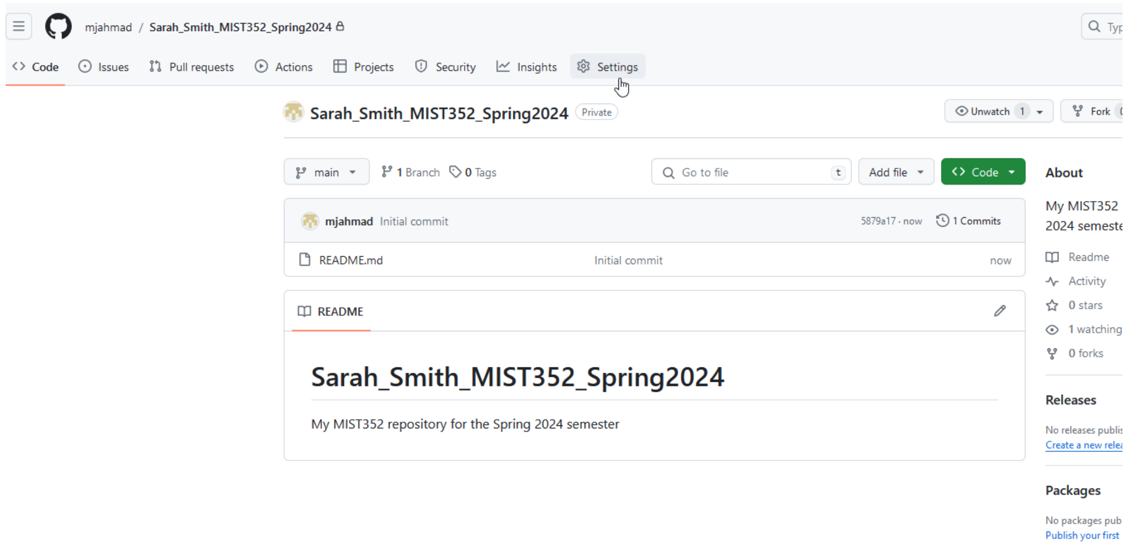
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

You are creating a private repository in your personal account.

Add collaborators to your GitHub

- Next, You need to add Dr. Ahmad as a collaborator.
- This is the only way Dr. Ahmad can se you and grade your code.
- Go to settings and add (mahmad2@mix.wvu.edu) as a collaborator



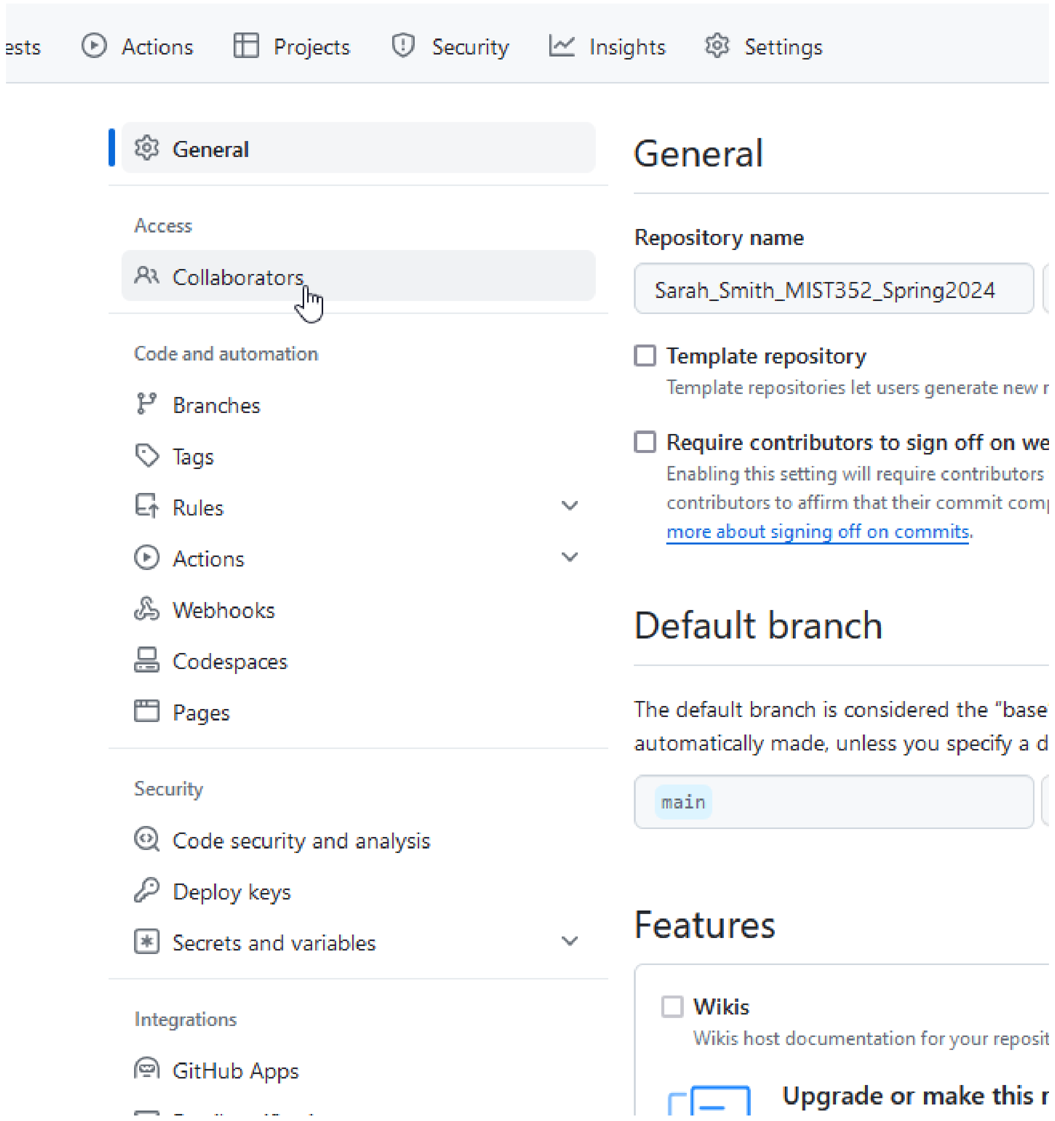
Sarah_Smith_MIST352_Spring2024

My MIST352 repository for the Spring 2024 semester

My MIST352 2024 semeste

No releases publi
Create a new relea

No packages pub
Publish your first



Repository name: Sarah_Smith_MIST352_Spring2024

☐ Template repository
Template repositories let users generate new r

☐ Require contributors to sign off on we
Enabling this setting will require contributors contributors to affirm that their commit comp more about signing off on commits.

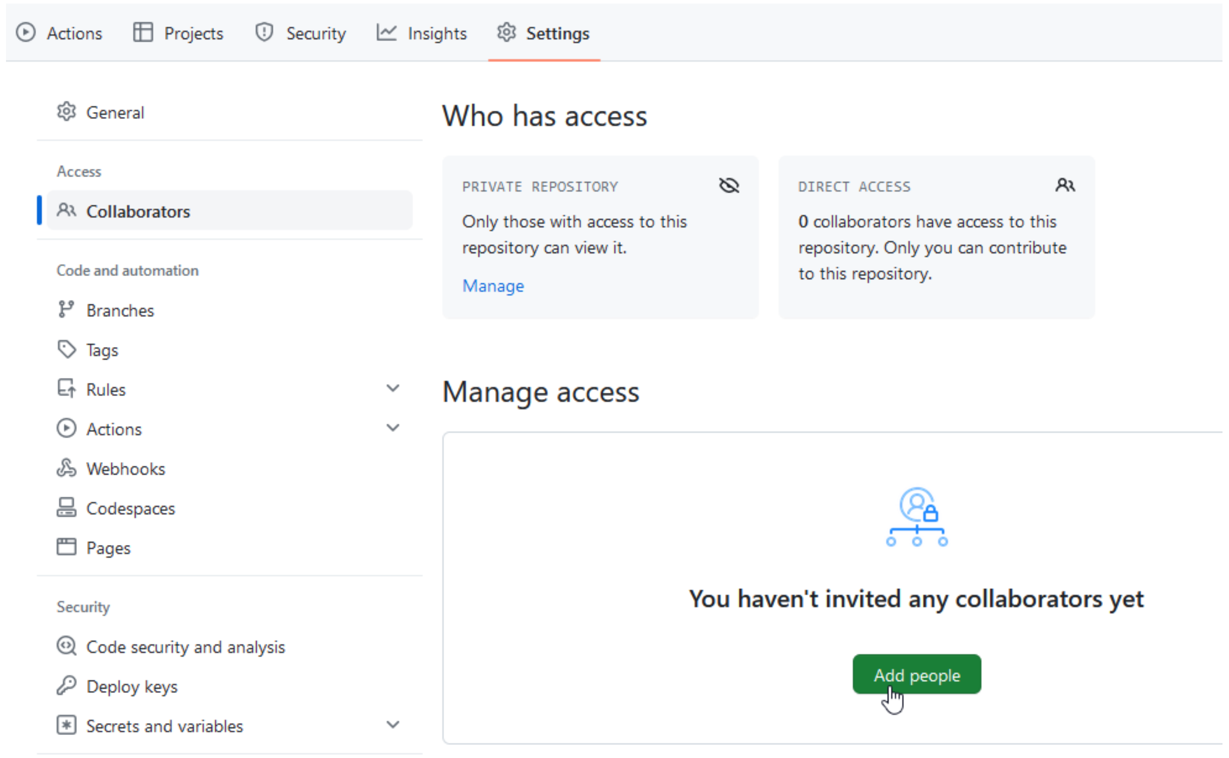
The default branch is considered the "base automatically made, unless you specify a d

main

☐ Wikis
Wikis host documentation for your reposit

Upgrade or make this r

Adding collaborators



Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

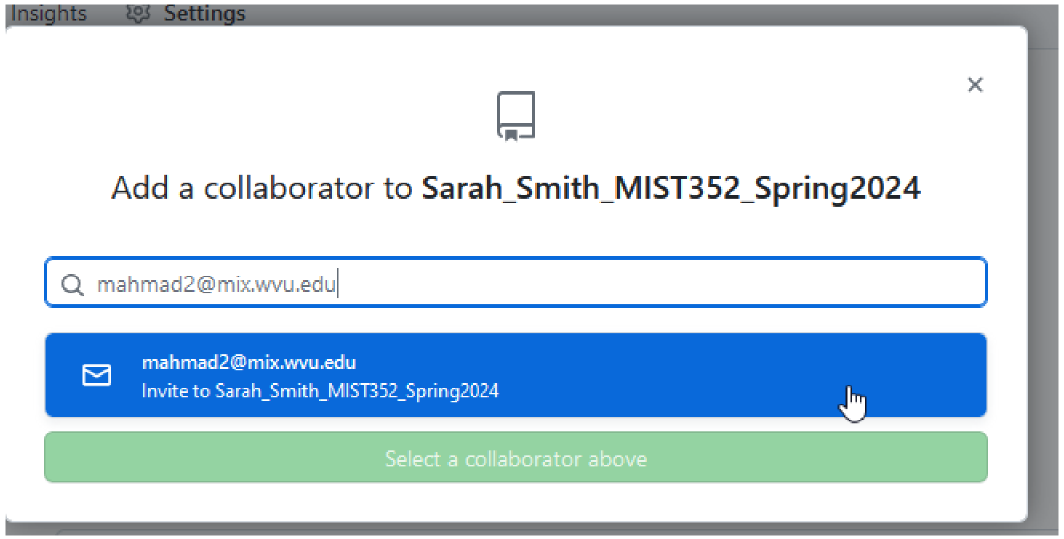
Manage

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet



Add a collaborator to **Sarah_Smith_MIST352_Spring2024**

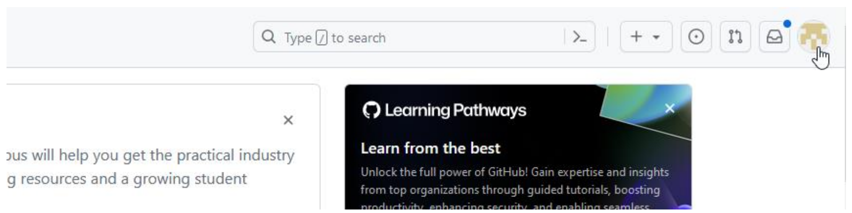
mahmad2@mix.wvu.edu

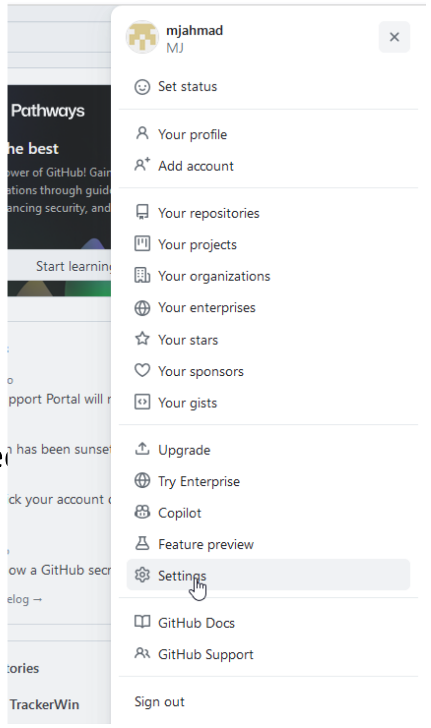
mahmad2@mix.wvu.edu
Invite to Sarah_Smith_MIST352_Spring2024

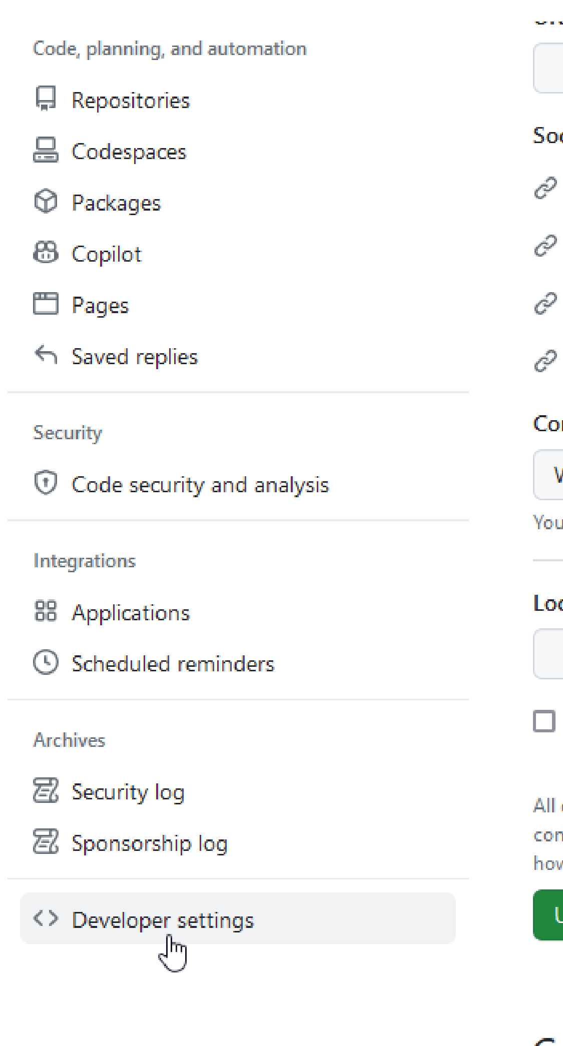
Obtain GitHub personal token code

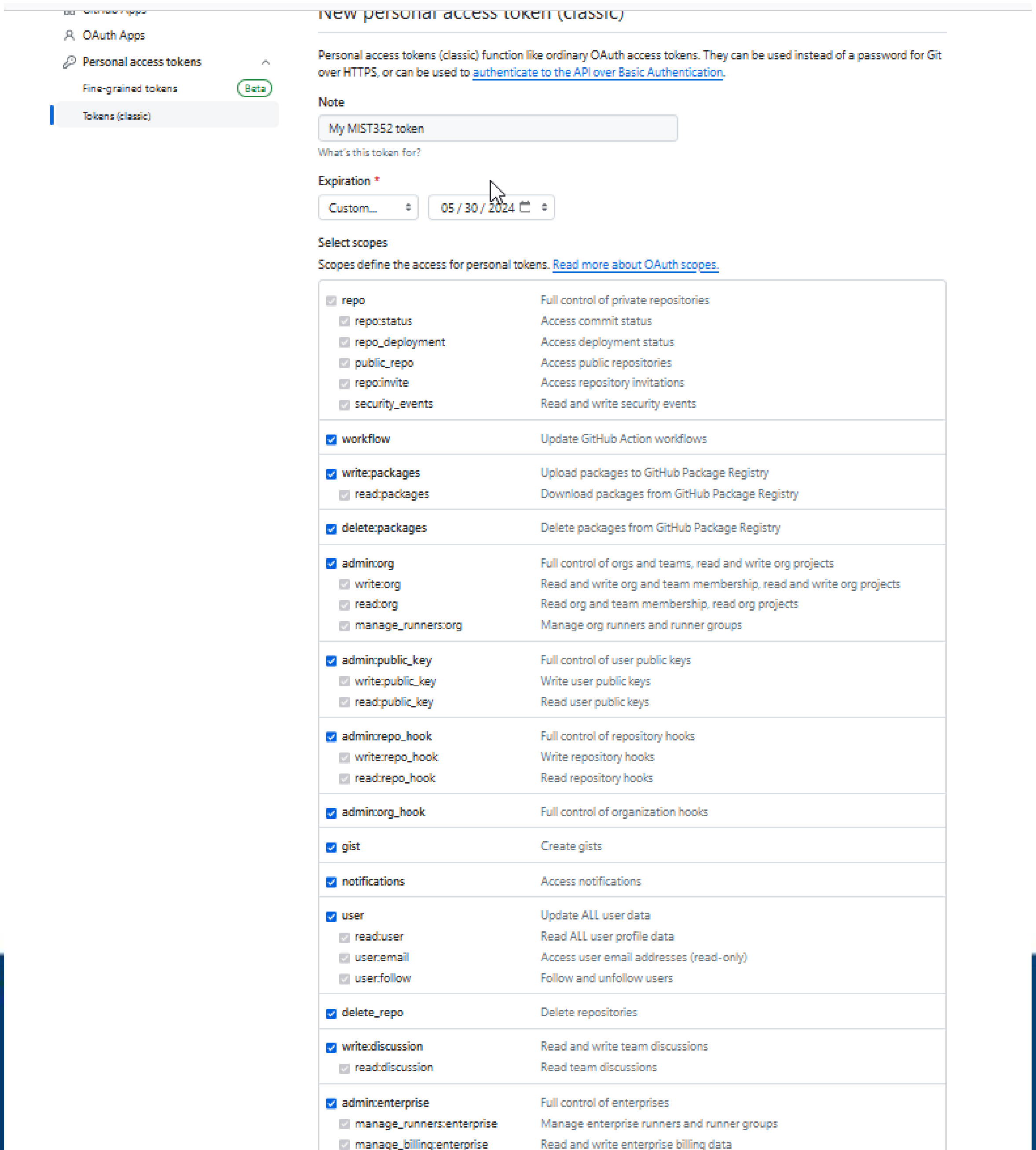
Next, we need to obtain a token from GitHub to allows us to interact with the online repository.

- On GitHub, go to your account setting
- Scroll down and go to Developer settings
- Go to Personal access tokens
- Go to Tokens (classic) -> Generate new token
- Go to Generate new token (classic)
- Set date to 5/30/2024
- Check all boxes -> Generate Token
- SAVE your personal token in a secured file
  - You will never see this token again. If you lose, then you will nee token.
  - See next slides for screenshots


Type / to search
Learning Pathways
Learn from the best
Unlock the full power of GitHub! Gain expertise and insights from top organizations through guided tutorials, boosting
ous will help you get the practical industry
g resources and a growing student







New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

My MIST352 token

What's this token for?

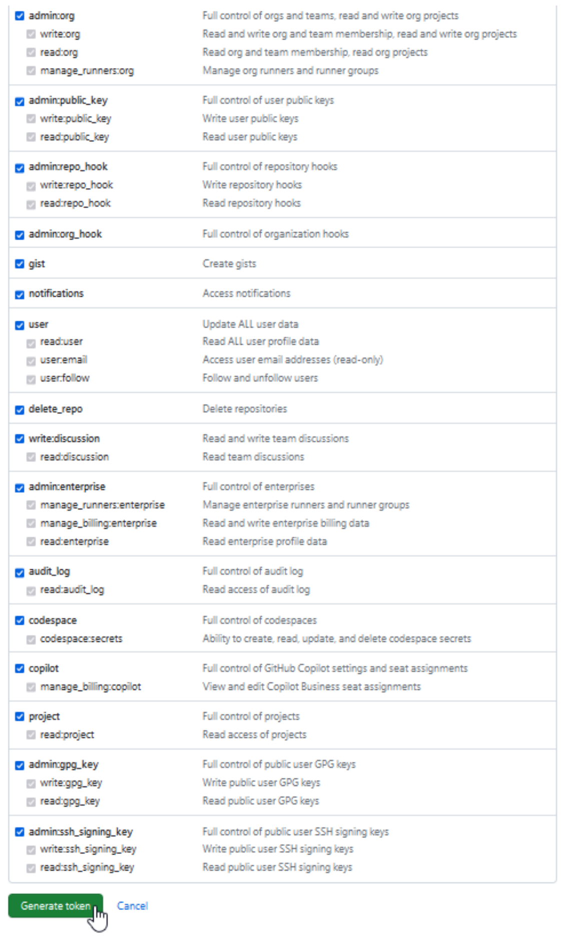
Expiration *

Custom... 05 / 30 / 2024

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

	repo	Full control of private repositories
	repo:status	Access commit status
	repo_deployment	Access deployment status
	public_repo	Access public repositories
	repo:invite	Access repository invitations
	security_events	Read and write security events
✓	workflow	Update GitHub Action workflows
✓	write:packages	Upload packages to GitHub Package Registry
	read:packages	Download packages from GitHub Package Registry
✓	delete:packages	Delete packages from GitHub Package Registry
✓	admin:org	Full control of orgs and teams, read and write org projects
	write:org	Read and write org and team membership, read and write org projects
	read:org	Read org and team membership, read org projects
	manage_runners:org	Manage org runners and runner groups
✓	admin:public_key	Full control of user public keys
	write:public_key	Write user public keys
	read:public_key	Read user public keys
✓	admin:repo_hook	Full control of repository hooks
	write:repo_hook	Write repository hooks
	read:repo_hook	Read repository hooks
✓	admin:org_hook	Full control of organization hooks
✓	gist	Create gists
✓	notifications	Access notifications
✓	user	Update ALL user data
	read:user	Read ALL user profile data
	user:email	Access user email addresses (read-only)
	user:follow	Follow and unfollow users
✓	delete_repo	Delete repositories
✓	write:discussion	Read and write team discussions
	read:discussion	Read team discussions
✓	admin:enterprise	Full control of enterprises
	manage_runners:enterprise	Manage enterprise runners and runner groups
	manage_billing:enterprise	Read and write enterprise billing data



✓	admin:org	Full control of orgs and teams, read and write org projects
	write:org	Read and write org and team membership, read and write org projects
	read:org	Read org and team membership, read org projects
	manage_runners:org	Manage org runners and runner groups
✓	admin:public_key	Full control of user public keys
	write:public_key	Write user public keys
	read:public_key	Read user public keys
✓	admin:repo_hook	Full control of repository hooks
	write:repo_hook	Write repository hooks
	read:repo_hook	Read repository hooks
✓	admin:org_hook	Full control of organization hooks
✓	gist	Create gists
✓	notifications	Access notifications
✓	user	Update ALL user data
	read:user	Read ALL user profile data
	user:email	Access user email addresses (read-only)
	user:follow	Follow and unfollow users
✓	delete_repo	Delete repositories
✓	write:discussion	Read and write team discussions
	read:discussion	Read team discussions
✓	admin:enterprise	Full control of enterprises
	manage_runners:enterprise	Manage enterprise runners and runner groups
	manage_billing:enterprise	Read and write enterprise billing data

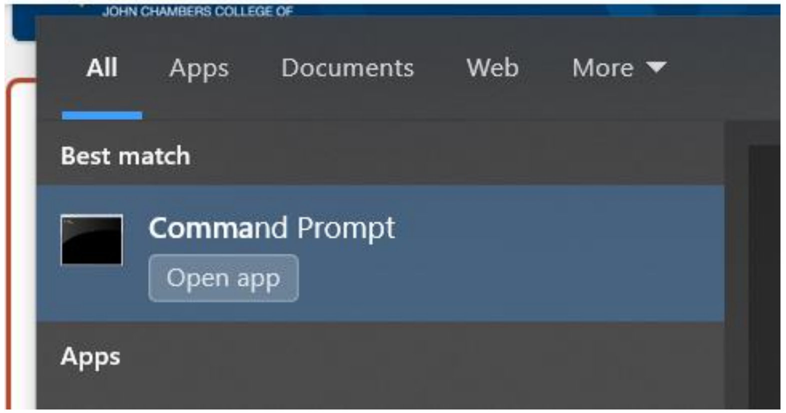
	read:enterprise	Read enterprise profile data
✓	audit_log	Full control of audit log
	read:audit_log	Read access of audit log
✓	codespace	Full control of codespaces
	codespace:secrets	Ability to create, read, update, and delete codespace secrets
✓	copilot	Full control of GitHub Copilot settings and seat assignments
	manage_billing:copilot	View and edit Copilot Business seat assignments
✓	project	Full control of projects
	read:project	Read access of projects
✓	admin:gpg_key	Full control of public user GPG keys
	write:gpg_key	Write public user GPG keys
	read:gpg_key	Read public user GPG keys
✓	admin:ssh_signing_key	Full control of public user SSH signing keys
	write:ssh_signing_key	Write public user SSH signing keys
	read:ssh_signing_key	Read public user SSH signing keys

Git, one last thing

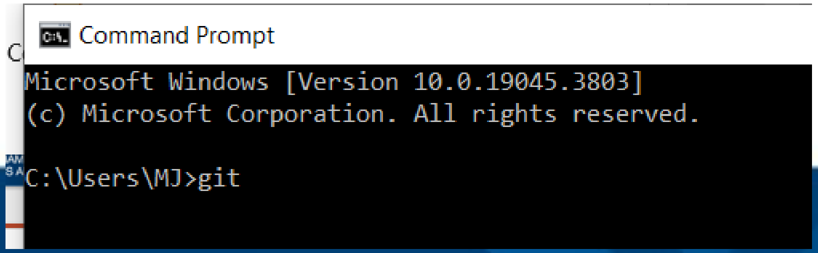
- Now we need an application to allow us to synch data and code from the online repository (GitHub) to our local computers.
- Git does that.
- Before you download Git, check whether you have it or not. See next slide
- Download and install Git.

Check whether you have Git or not

- Go to the command line (on windows) or the terminal (on mac).
- Go to menu and type command line

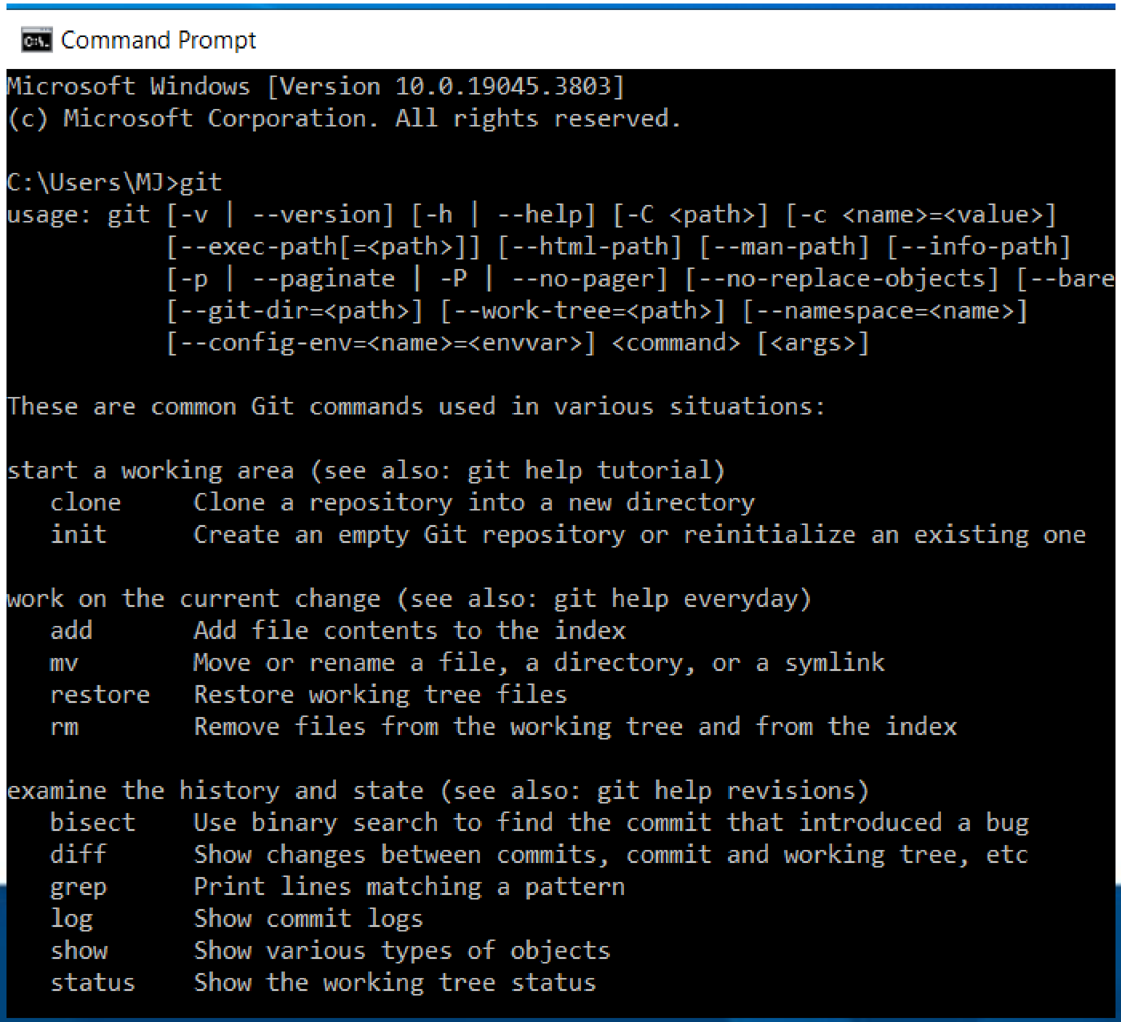


- In the command line, type `git` and hit enter



```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MJ>git
```

- If you have git, you should see the screen below, otherwise you will see “not found” or “not recognized”:



Command Prompt

```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MJ>git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status
```

If Git is not installed

- Download Git and install
- If asked for credentials, provide your GitHub user name and the token you obtained.
- Once installed, check from the command line/ terminal again using the command in the previous slide.

PUTTING THE WHOLE THING TOGETHER
PREPARE YOUR COMPUTER FOR MIST352 CODING

Create folder for MIST352

- Now you should have Eclipse and Git installed, and GitHub configured, we need to create the proper folders on your computer.
- Although you may skip these steps, you are highly encouraged to flow this structure.
- On your computer, navigate to the desktop or any other location and create a folder named MIST352_Spring2024

The MIST352_Spring2024 folder

- Inside this folder, you will eventually add two repositories:

1. The MIST352 class's repository: This is where Dr. Ahmad upload all source codes related to this class (https://github.com/mjahmad/MIST352.git)
2. Your own repository: This is where you should keep all of your codes, submit assessments, etc..

Next slides will show you how to clone each.

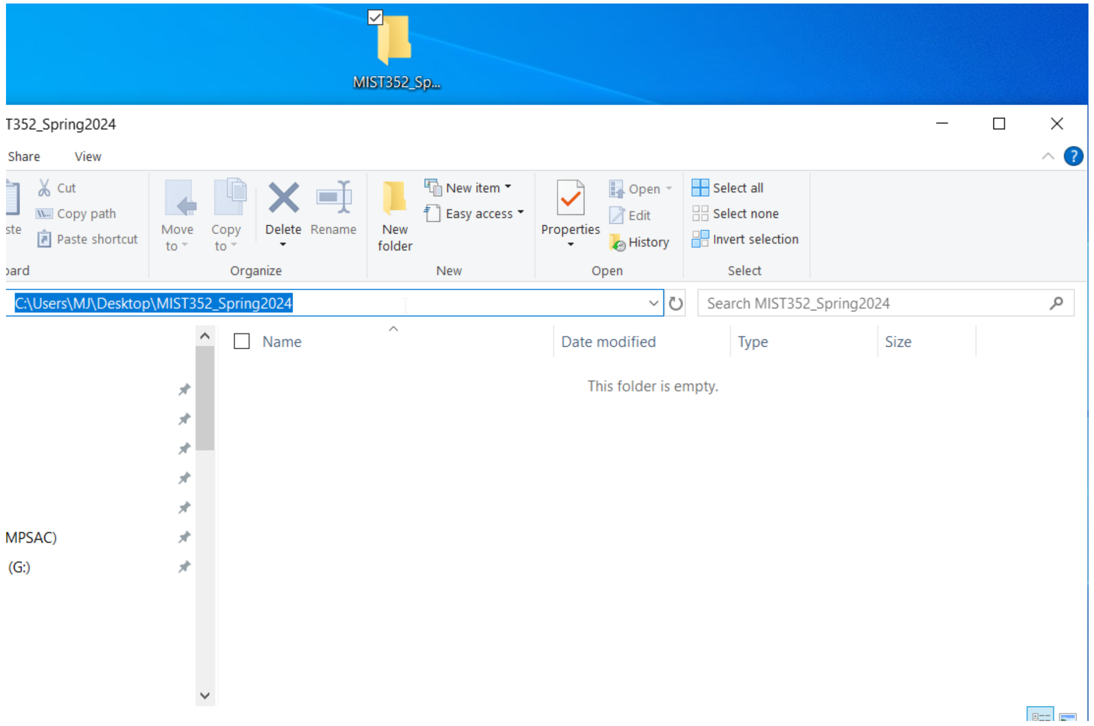
Clone the MIST352 repository

- As stated earlier, this is the class's GitHub repository.
- We will use git commands in the command line/ terminal to clone each of these two repositors into our local MIST352_Spring2024 folder.
  - Each repository will be in a separate folder
- Go to your command line/ terminal and navigate to the location of the MIST352_Spring2024 folder
- To navigate, see the next slide

Navigate to local folder from the command line/ terminal

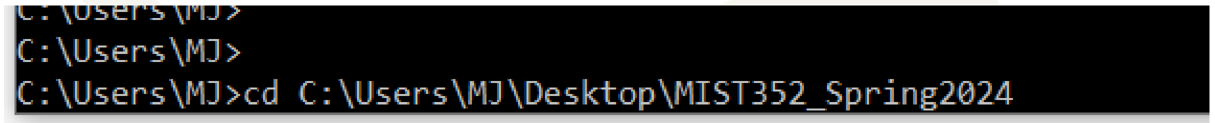
- First, we need to know where the folder is, to do that, open that folder and click on the address bar
- Now you can see that the folder is on my (C:\Users\MJ\Desktop\MIST352_Spring2024) directory.

- **Yours will be different of course.**
- Copy this link and go to command line/ terminal

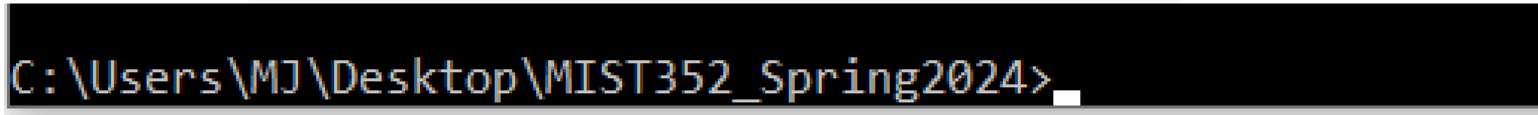


Navigate to local folder from the command line/ terminal

- Once in the command line/ terminal type: cd [location you copied in previous slide]
- cd stands for “change directory”
- The command line/ terminal should now be inside the MIST352_Spring2024 folder as shown in the second screenshot.



```
C:\Users\MJ>
C:\Users\MJ>
C:\Users\MJ>cd C:\Users\MJ\Desktop\MIST352_Spring2024
```



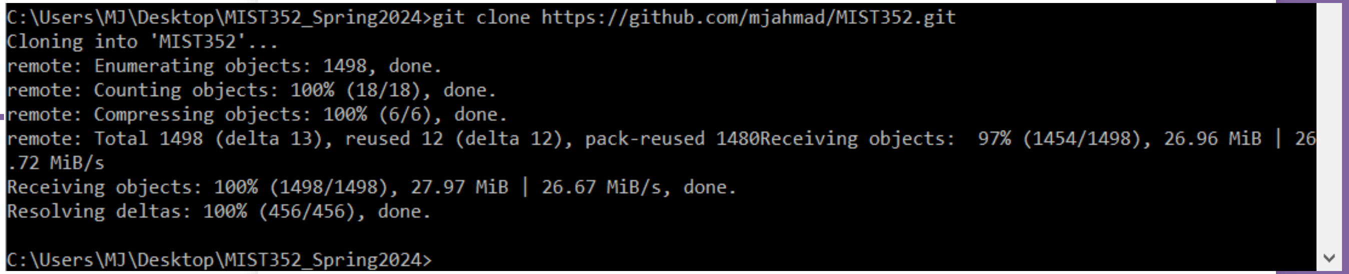
```
C:\Users\MJ\Desktop\MIST352_Spring2024>
```

Clone repositories

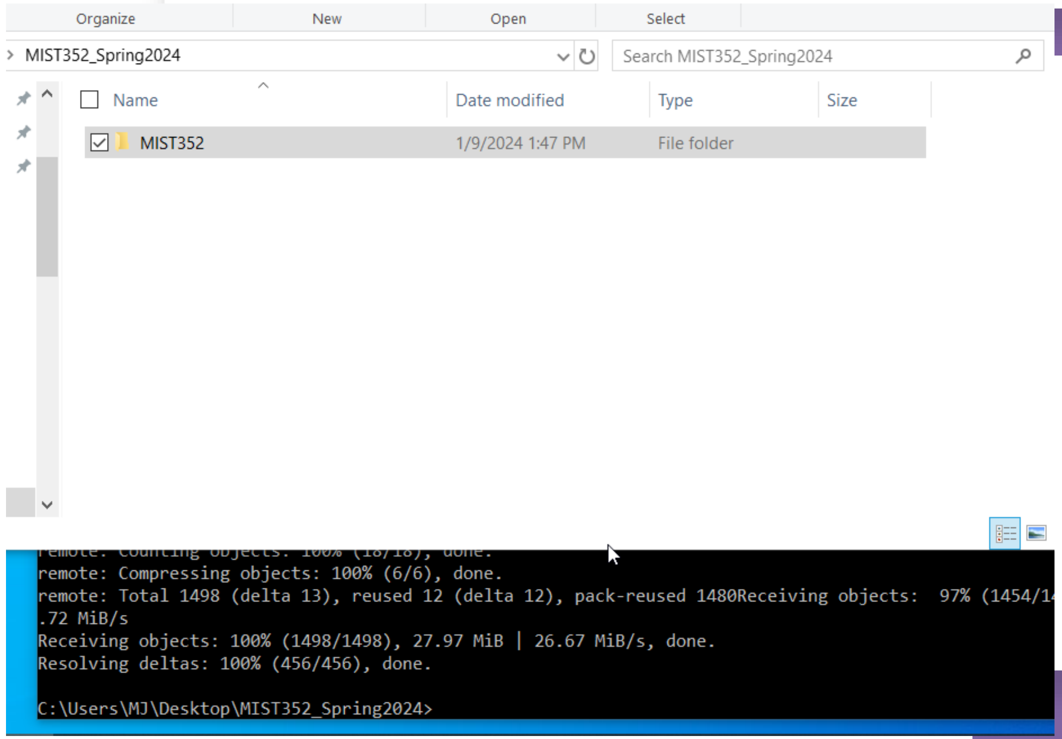
- Once your command line/ terminal is inside the correct directory, type:

git clone
https://github.com/mjahmad/MIST352.git

Now, open the MIST352_Spring2024 folder, you should see the MIST352 folder there



```
C:\Users\MJ\Desktop\MIST352_Spring2024>git clone https://github.com/mjahmad/MIST352.git
Cloning into 'MIST352'...
remote: Enumerating objects: 1498, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 1498 (delta 13), reused 12 (delta 12), pack-reused 1480Receiving objects:  97% (1454/1498), 26.96 MiB | 26
.72 MiB/s
Receiving objects: 100% (1498/1498), 27.97 MiB | 26.67 MiB/s, done.
Resolving deltas: 100% (456/456), done.

C:\Users\MJ\Desktop\MIST352_Spring2024>
```

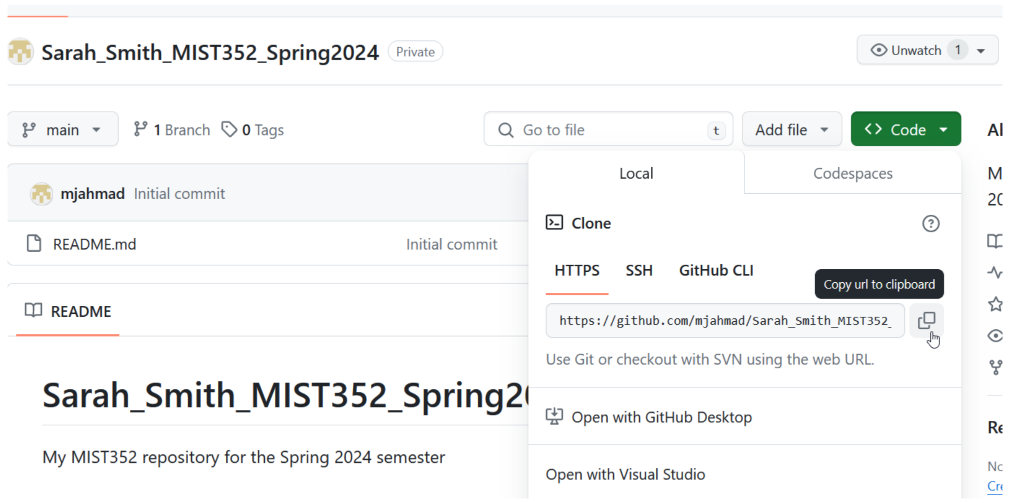


Name	Date modified	Type	Size
MIST352	1/9/2024 1:47 PM	File folder	

```
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 1498 (delta 13), reused 12 (delta 12), pack-reused 1480Receiving objects:  97% (1454/1
.72 MiB/s
Receiving objects: 100% (1498/1498), 27.97 MiB | 26.67 MiB/s, done.
Resolving deltas: 100% (456/456), done.

C:\Users\MJ\Desktop\MIST352_Spring2024>
```

Clone your own repository

- Go to your GitHub's account online and obtain the clone link.
- Copy the code and go back to the command line/ terminal
- Make sure –again- that the command line / terminal is setting inside the MIST352_Spring2024 folder
- Type `git clone [link you copied for your repo]`
- Now, you should see two folders inside the MIST352_Spring2024 folder
  - MIST352
  - Your own



mjahmad Initial commit

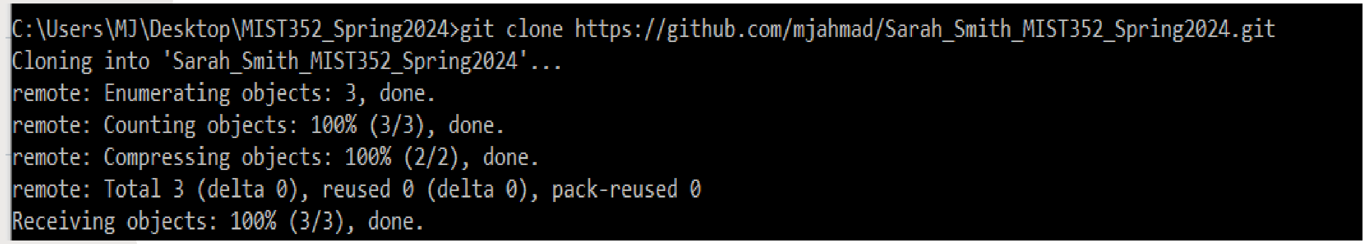
README.md Initial commit

https://github.com/mjahmad/Sarah_Smith_MIST352_

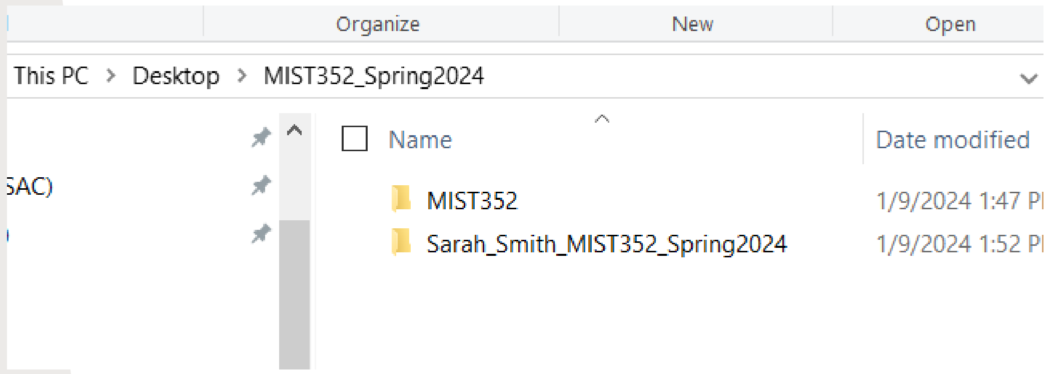
Use Git or checkout with SVN using the web URL.

Sarah_Smith_MIST352_Spring2

My MIST352 repository for the Spring 2024 semester



```
C:\Users\MJ\Desktop\MIST352_Spring2024>git clone https://github.com/mjahmad/Sarah_Smith_MIST352_Spring2024.git
Cloning into 'Sarah_Smith_MIST352_Spring2024'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

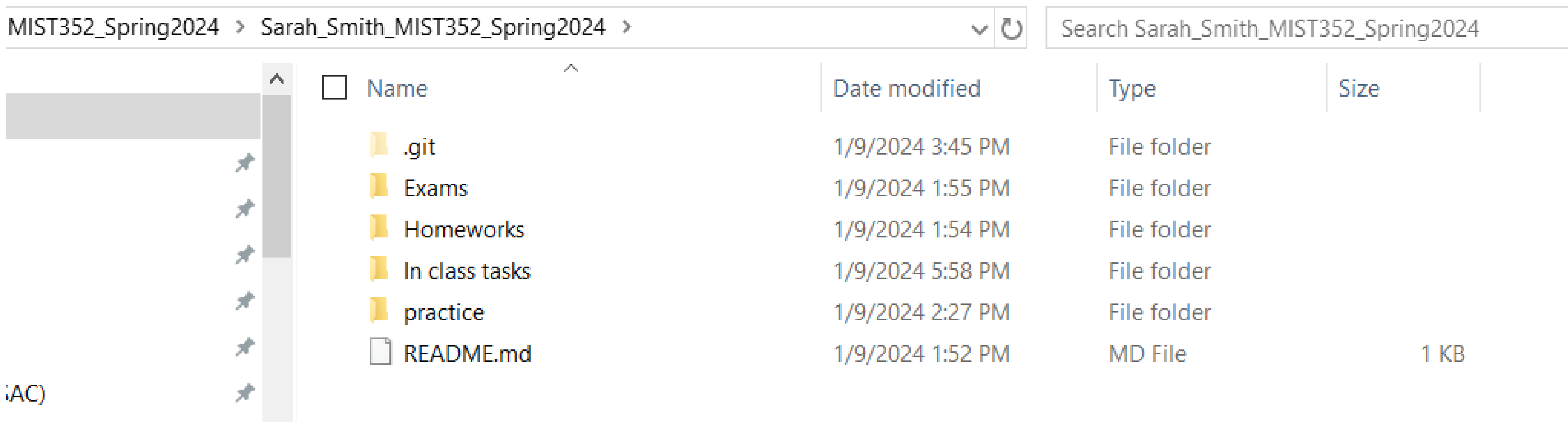


Name	Date modified
MIST352	1/9/2024 1:47 PI
Sarah_Smith_MIST352_Spring2024	1/9/2024 1:52 PI

Prepare your local folder

- Go to your local repository folder and create the following folders (exactly as they are named)

1. Exams
2. Homeworks
3. In class tasks
4. practice



MIST352_Spring2024 > Sarah_Smith_MIST352_Spring2024 >

Search Sarah_Smith_MIST352_Spring2024

Name	Date modified	Type	Size
.git	1/9/2024 3:45 PM	File folder	
Exams	1/9/2024 1:55 PM	File folder	
Homeworks	1/9/2024 1:54 PM	File folder	
In class tasks	1/9/2024 5:58 PM	File folder	
practice	1/9/2024 2:27 PM	File folder	
README.md	1/9/2024 1:52 PM	MD File	1 KB

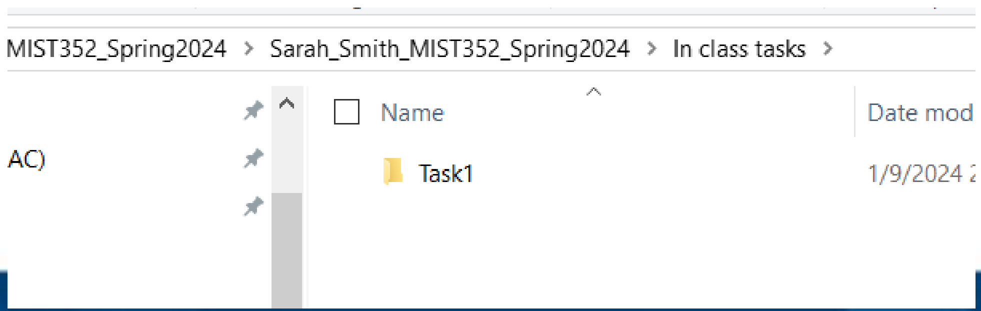
(you will other things in this folder, just ignore them).

Create your first java project

- Now you have your computer ready to starting coding and pushing code to GitHub.
- The git command/ operation “push” means: take the changes I made to my local repository, and push it to the cloud (synch it).
- We will be using a set of git commands to do that.
- First, lets create a java project

Create your first java project

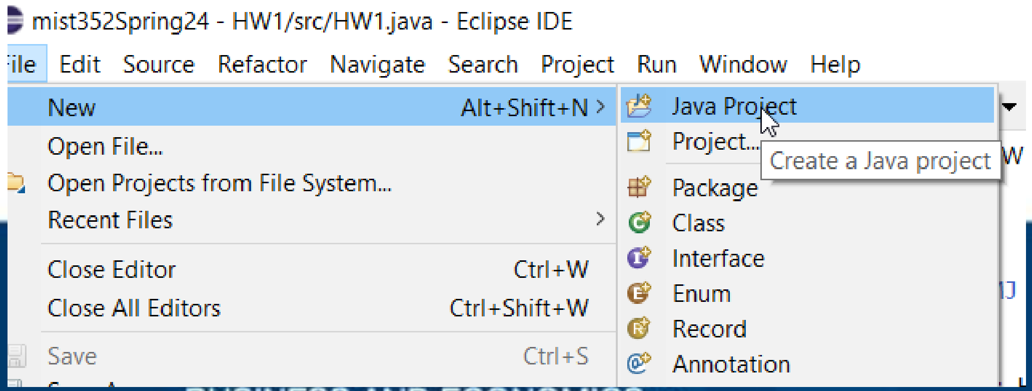
- The easiest way is to first create a folder for the project you want to create, then launch eclipse to start coding.
- Go to the *In class tasks* folder and create a folder named Task1
- From now on, when being asked to create a new java project, create a folder for that project first, before coding with eclipse.


MIST352_Spring2024 > Sarah_Smith_MIST352_Spring2024 > In class tasks >

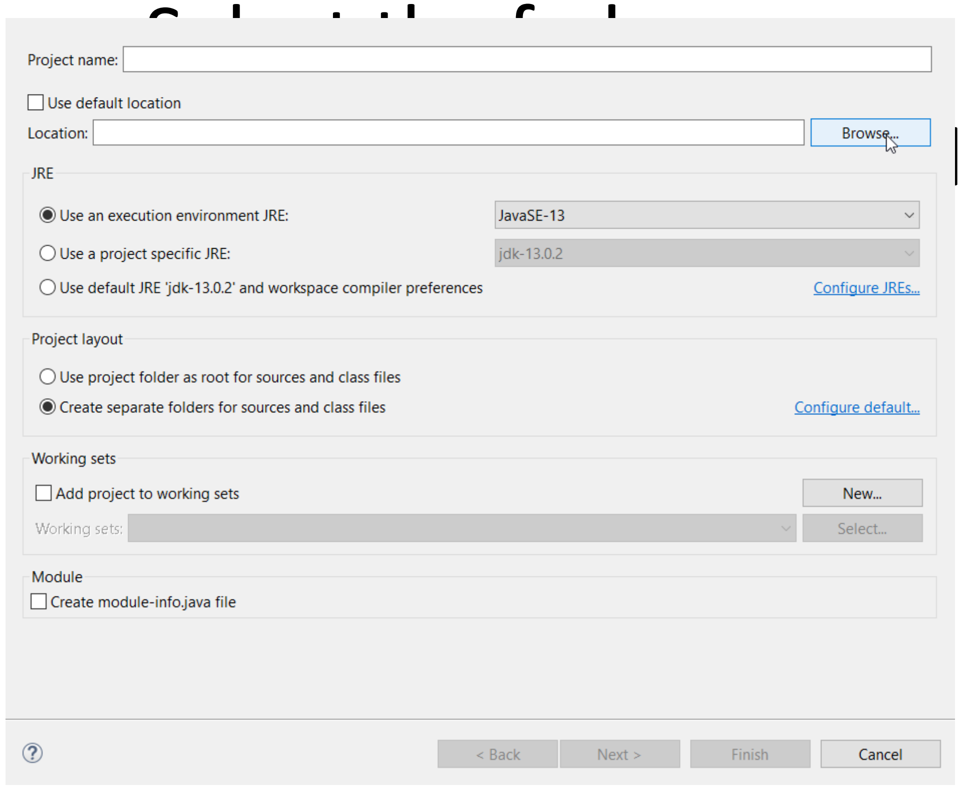
Name	Date mod
Task1	1/9/2024 2

Create your first java project

- Start Eclipse -> leave default options
- Go to File -> New ->Java Project
- Uncheck (Use default location) and brows to the folder you created for this project.
  - – In this case browse to Task1 folder


mist352Spring24 - HW1/src/HW1.java - Eclipse IDE
ile Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Java Project
Project...
Create a Java project
Package
Class
Interface
Enum
Record
Annotation

Create your first java project



Project name:

[ ] Use default location

Location: Browse...

JRE

(●) Use an execution environment JRE: JavaSE-13

( ) Use a project specific JRE: jdk-13.0.2

( ) Use default JRE 'jdk-13.0.2' and workspace compiler preferences — Configure JREs...

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files — Configure default...

Working sets

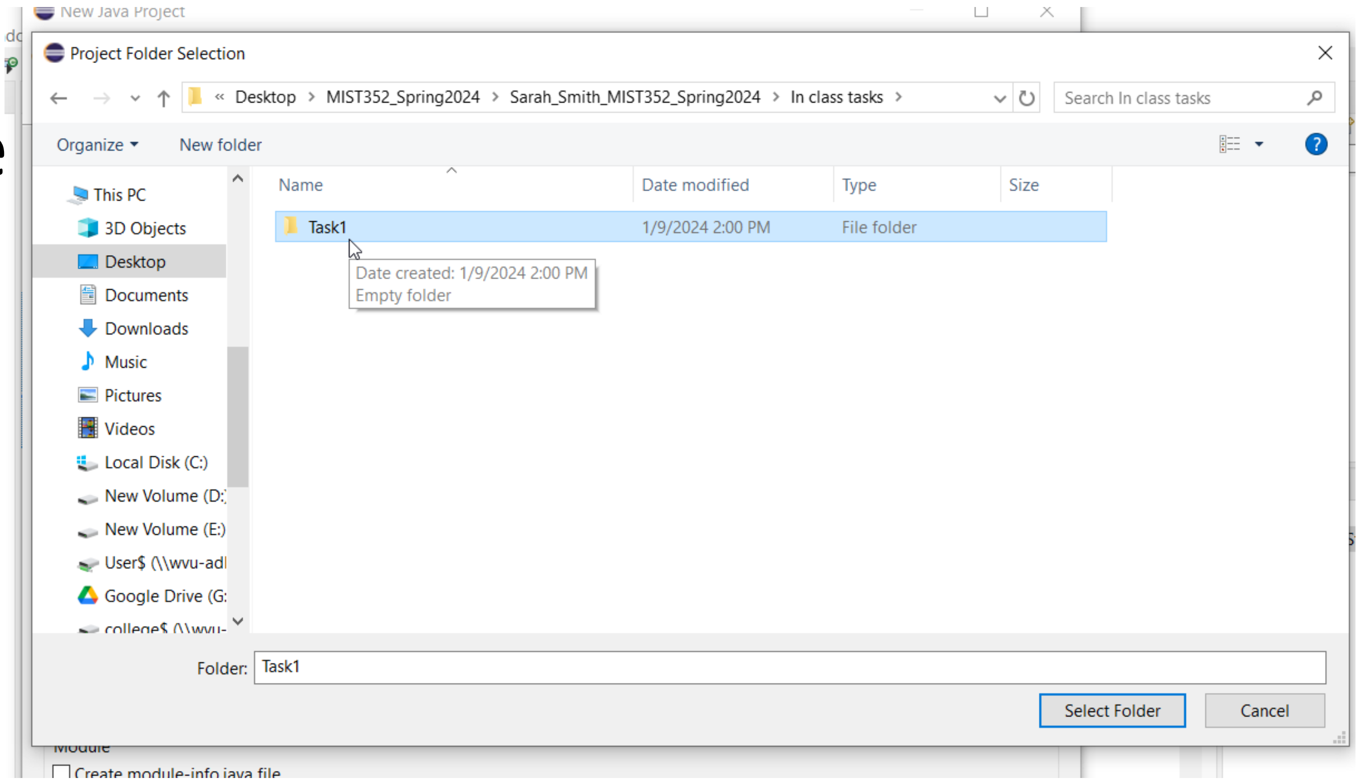
[ ] Add project to working sets — New...

Working sets: — Select...

Module

[ ] Create module-info.java file

< Back | Next > | Finish | Cancel



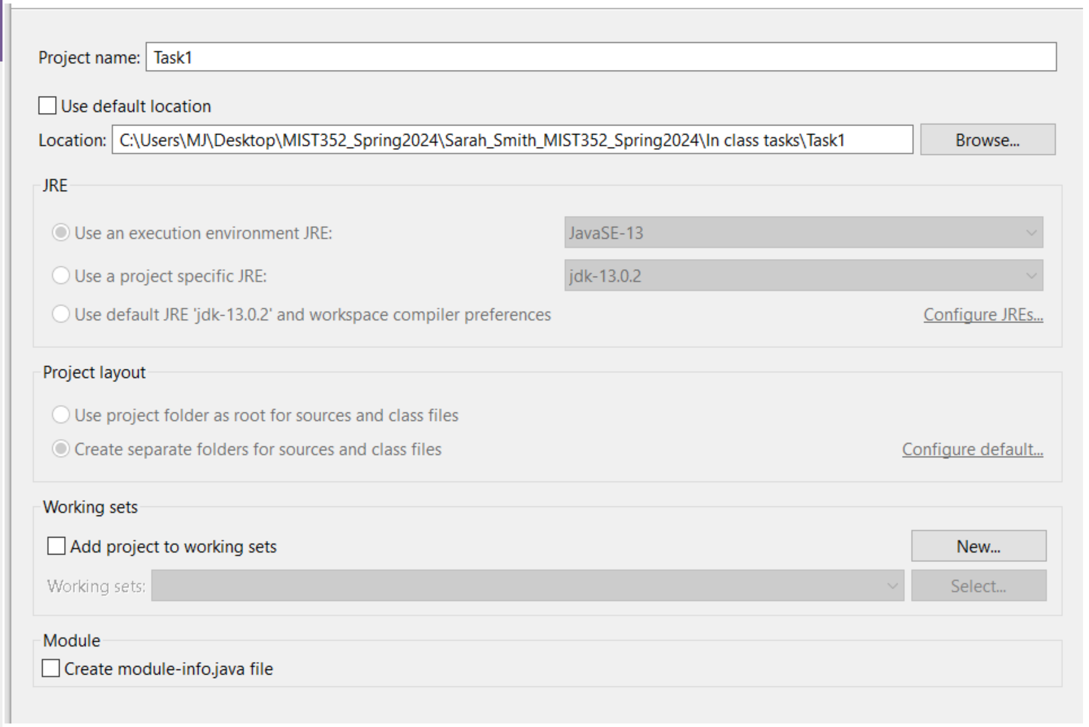
Project Folder Selection

Desktop > MIST352_Spring2024 > Sarah_Smith_MIST352_Spring2024 > In class tasks

Name	Date modified	Type	Size
Task1	1/9/2024 2:00 PM	File folder	

Date created: 1/9/2024 2:00 PM
Empty folder

Folder: Task1


Project name: Task1

☐ Use default location

Location: C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024\In class tasks\Task1 Browse...

JRE

● Use an execution environment JRE: JavaSE-13

○ Use a project specific JRE: jdk-13.0.2

○ Use default JRE 'jdk-13.0.2' and workspace compiler preferences Configure JREs...

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files Configure default...

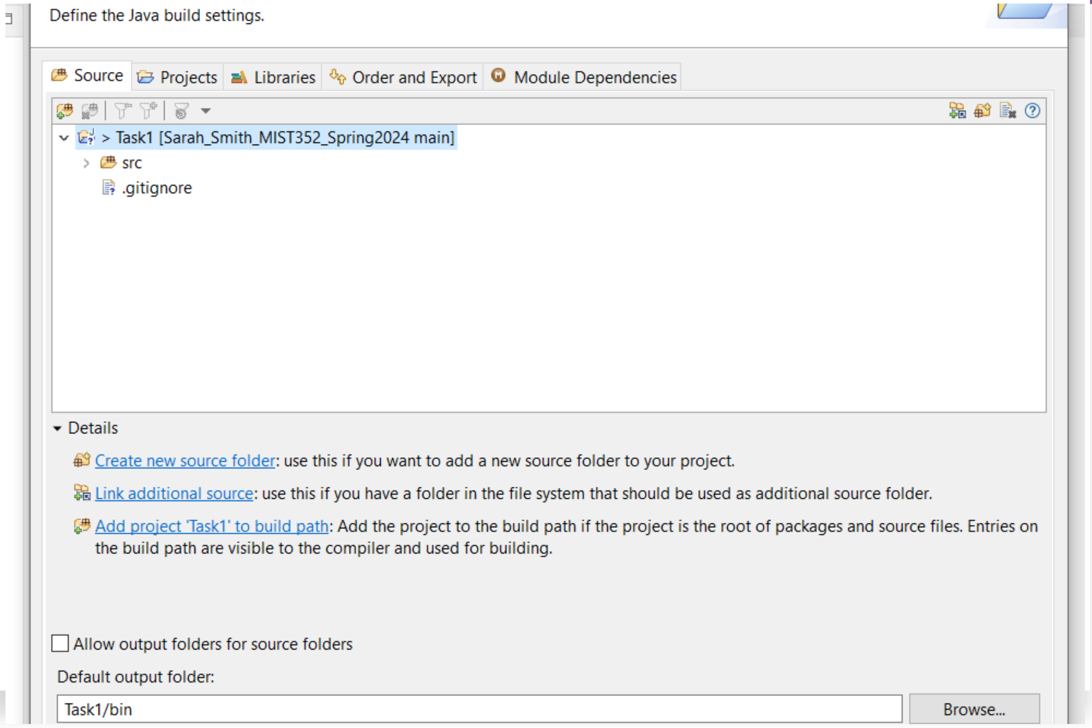
Working sets

☐ Add project to working sets New...

Working sets: Select...

Module

☐ Create module-info.java file


Define the Java build settings.

Source | Projects | Libraries | Order and Export | Module Dependencies

```
> Task1 [Sarah_Smith_MIST352_Spring2024 main]
    src
    .gitignore
```

Details

Create new source folder: use this if you want to add a new source folder to your project.

Link additional source: use this if you have a folder in the file system that should be used as additional source folder.

Add project 'Task1' to build path: Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

☐ Allow output folders for source folders

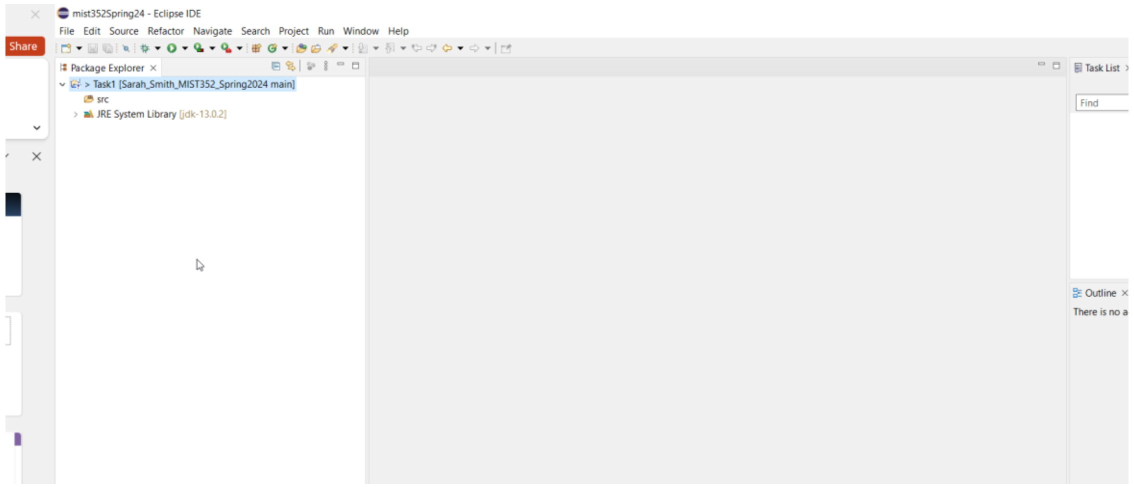
Default output folder:

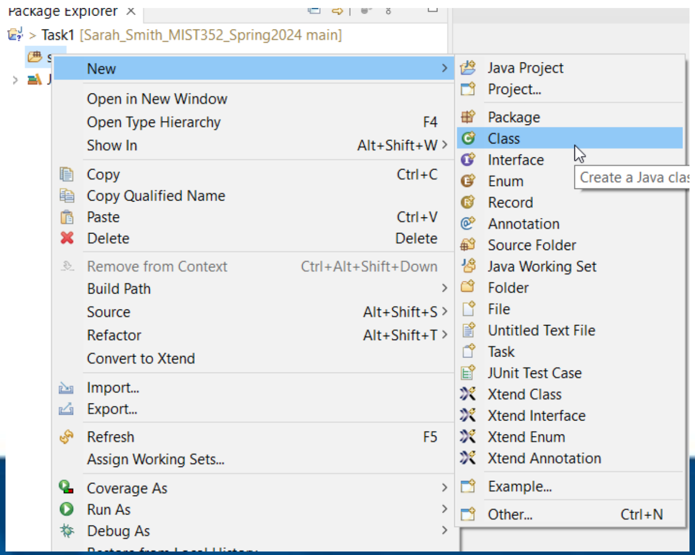
Task1/bin Browse...

- Hit Next, then Finish

Adding java class to project

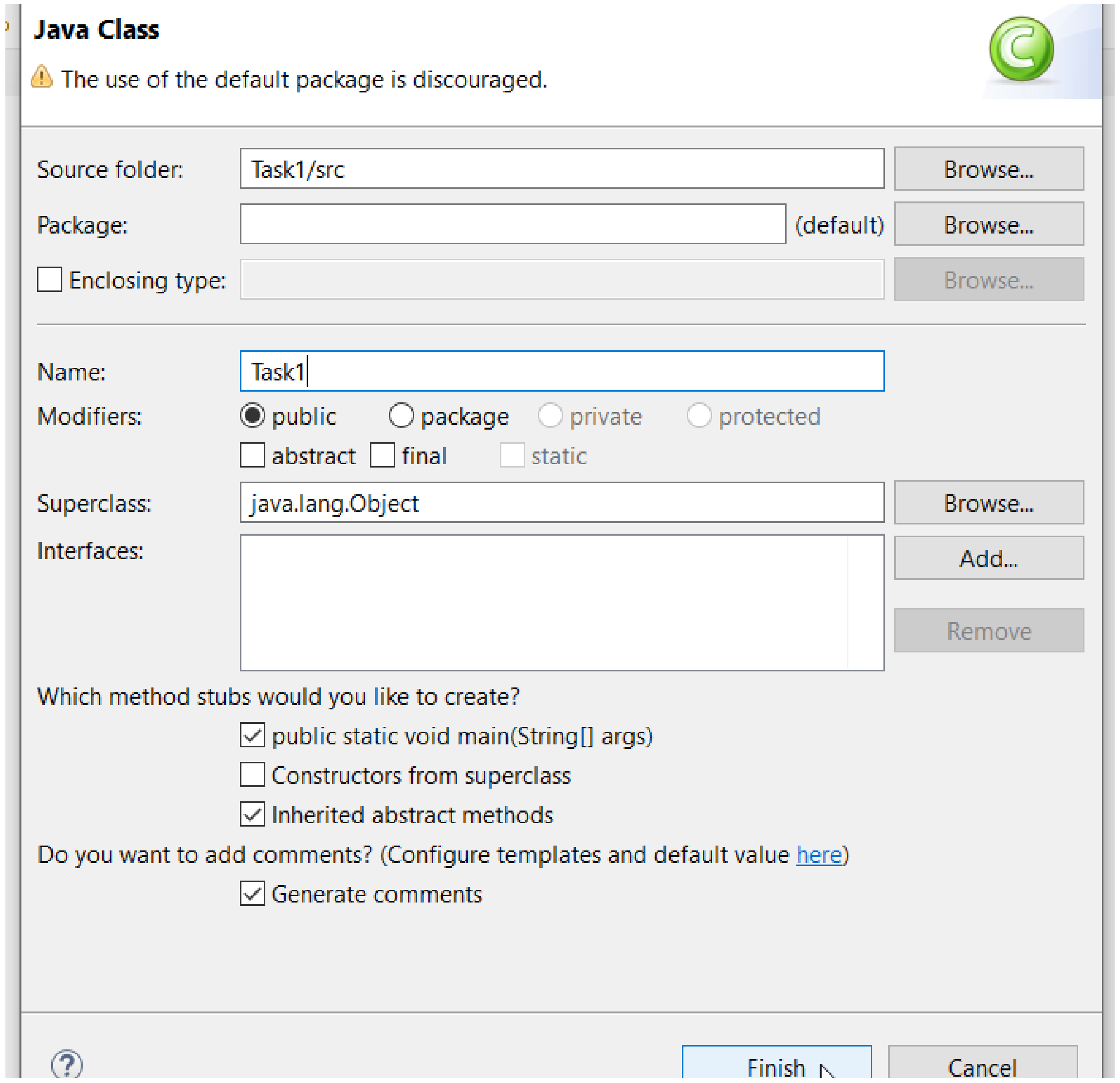
- Once the project is created, we need to start adding java files.
- Java files are where we actually write the code
- On Eclipse' package explorer, expand the Task1 folder -> right click on the src folder -> New -> Class


mist352Spring24 - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Share
Package Explorer
Task1 [Sarah_Smith_MIST352_Spring2024 main]
src
JRE System Library [jdk-13.0.2]
Task List
Find
Outline
There is no a



Adding java class to project

- Nam the files similarly to the project name (this might change in the future)
- Check the options as given in the screenshot
- Hit Finish


Java Class

⚠ The use of the default package is discouraged.

Source folder: Task1/src

Package: (default)

☐ Enclosing type:

Name: Task1

Modifiers: [✓ public] package private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stub would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

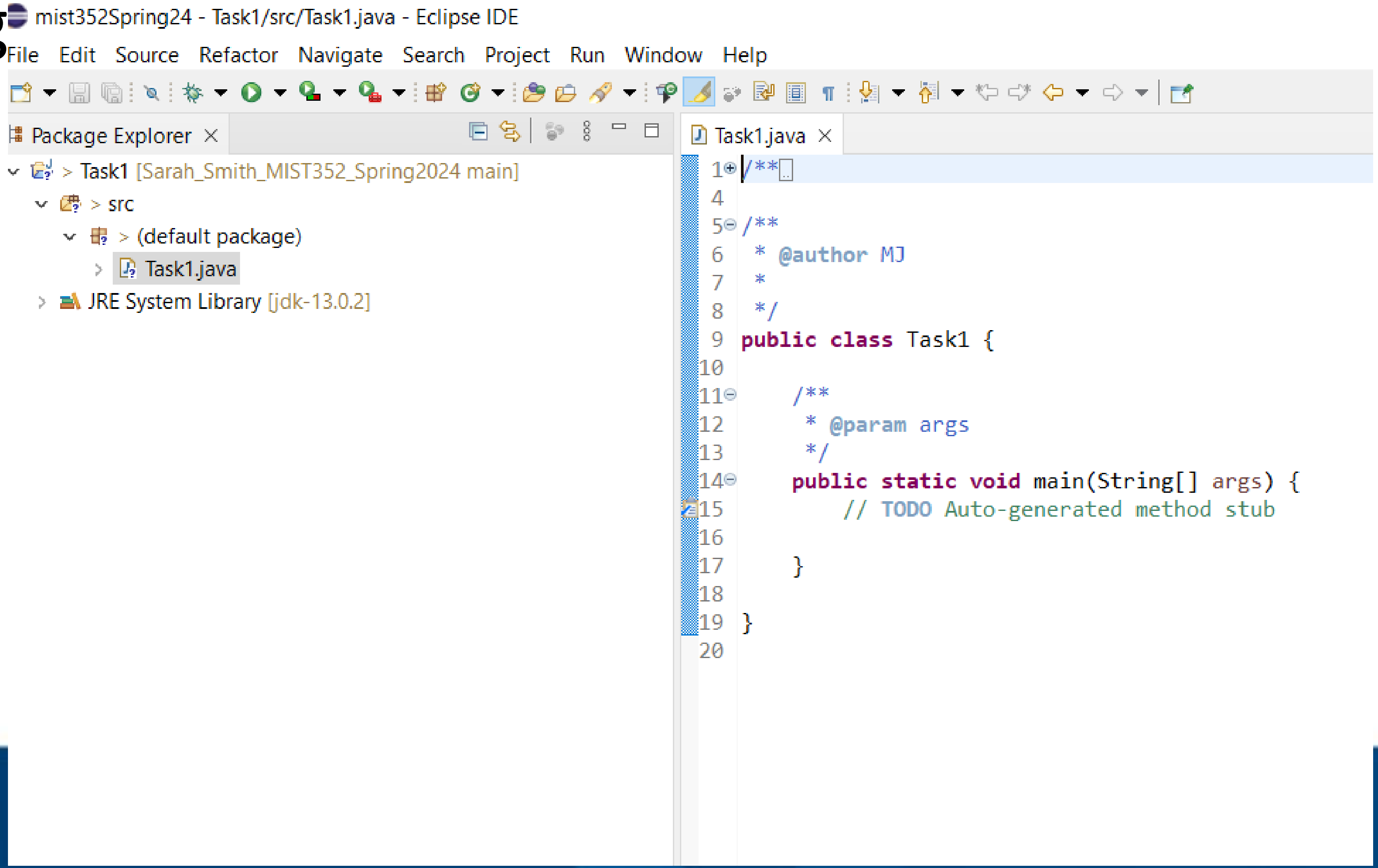
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☑ Generate comments

Welcome to Eclipse

- Now, we start coding

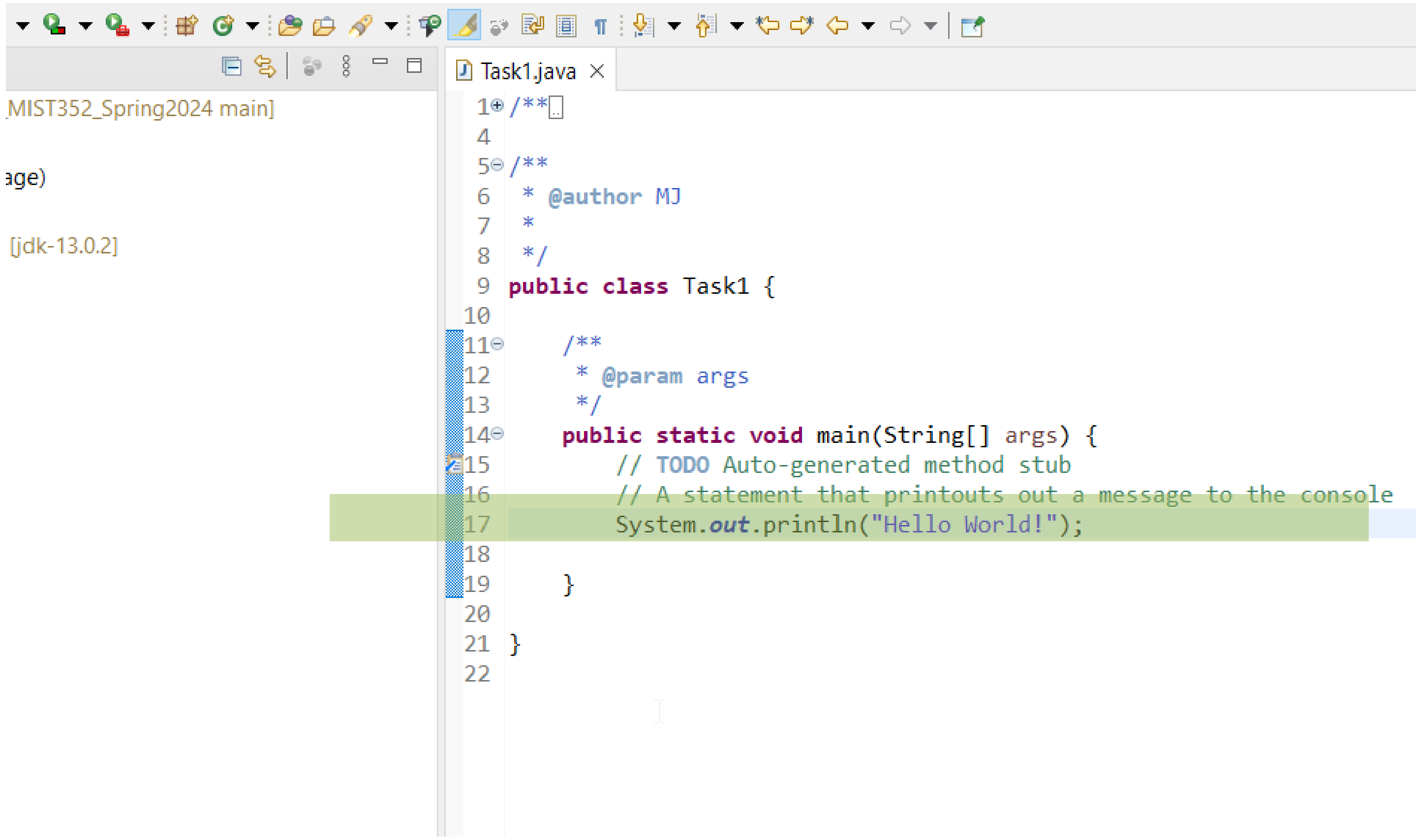


mist352Spring24 - Task1/src/Task1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Task1 [Sarah_Smith_MIST352_Spring2024 main]
src
(default package)
Task1.java
JRE System Library [jdk-13.0.2]
Task1.java

```
/**

/**
 * @author MJ
 *
 */
public class Task1 {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Write your first program

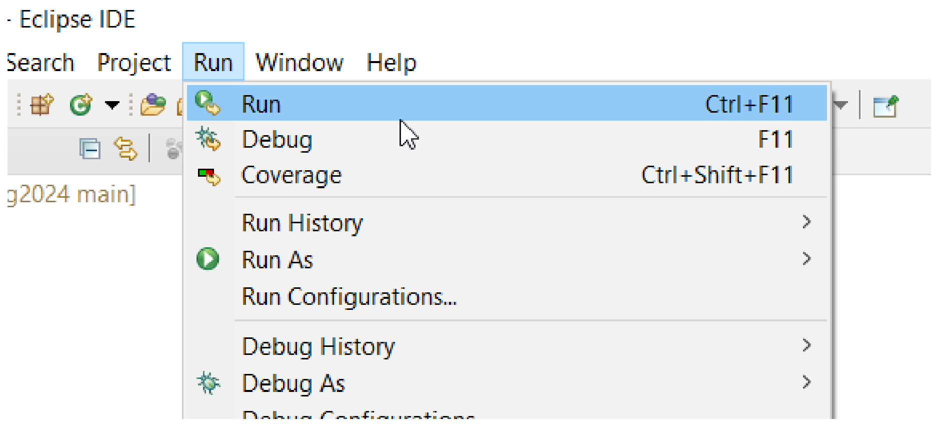
- For now, your code should go inside the two {} that belong to "public static void…." sentence
- Write the command in line 17 in the screenshot
  - This is case sensitive
  - No place for any mistakes

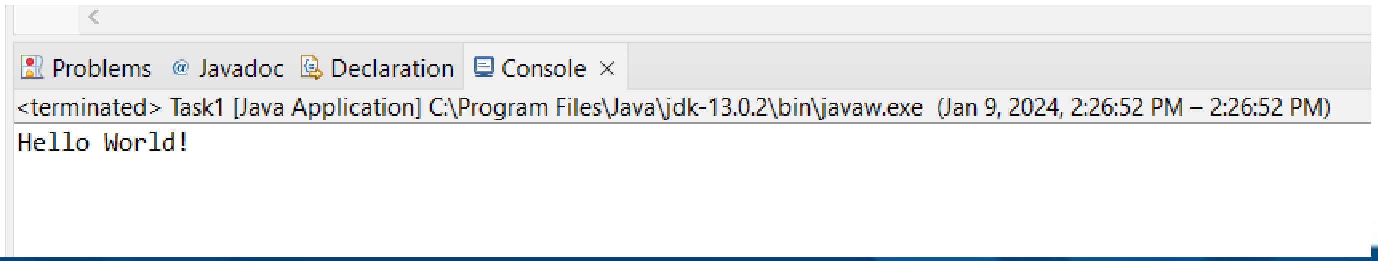


```
 1⊕/**
 4
 5⊖/**
 6  * @author MJ
 7  *
 8  */
 9 public class Task1 {
10
11⊖    /**
12     * @param args
13     */
14⊖    public static void main(String[] args) {
15        // TODO Auto-generated method stub
16        // A statement that printouts out a message to the console
17        System.out.println("Hello World!");
18
19    }
20
21 }
22
```

Run you first program

- Run your program
- You should see an output on your console





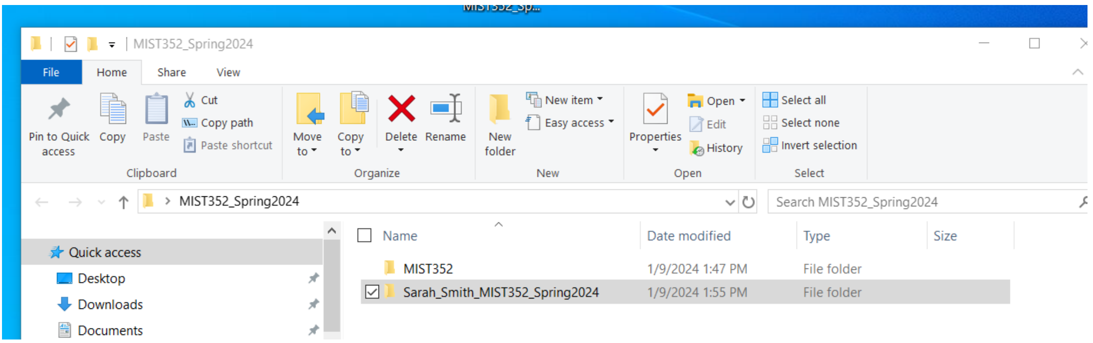
```
<terminated> Task1 [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe  (Jan 9, 2024, 2:26:52 PM – 2:26:52 PM)
Hello World!
```

Sync “push” project to GitHub

- There are multiple ways to push yoiur projects/ changes to GitHub
- The easiest is to use Git commands from the command line/ terminal
- Open command line/ terminal and navigate to the repository you want to sync.
- See next slide

Navigate to your local repo

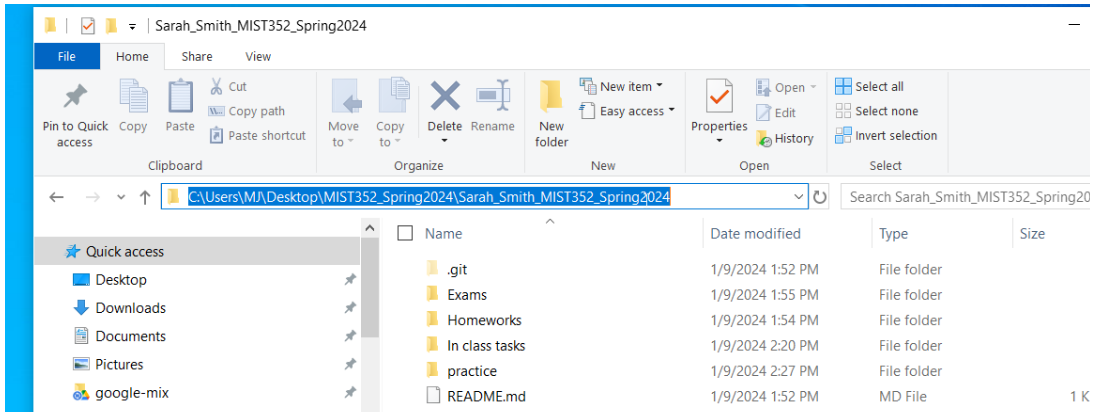
- Go to your local repository
- Copy its location as shown below



MIST352_Spring2024

Search MIST352_Spring2024

Name	Date modified	Type	Size
MIST352	1/9/2024 1:47 PM	File folder	
Sarah_Smith_MIST352_Spring2024	1/9/2024 1:55 PM	File folder	



Sarah_Smith_MIST352_Spring2024

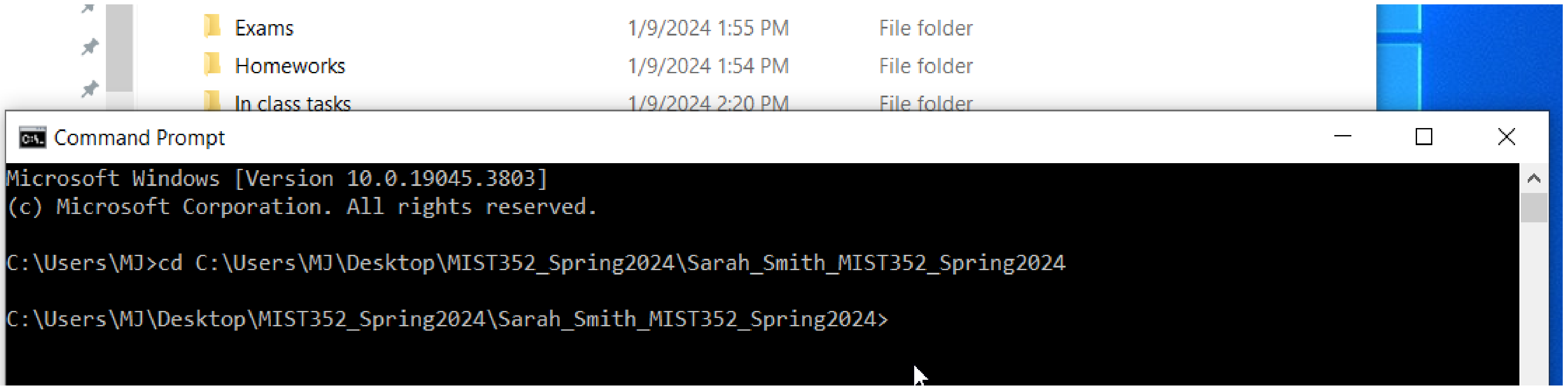
C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024

Search Sarah_Smith_MIST352_Spring20

Name	Date modified	Type	Size
.git	1/9/2024 1:52 PM	File folder	
Exams	1/9/2024 1:55 PM	File folder	
Homeworks	1/9/2024 1:54 PM	File folder	
In class tasks	1/9/2024 2:20 PM	File folder	
practice	1/9/2024 2:27 PM	File folder	
README.md	1/9/2024 1:52 PM	MD File	1 K

Navigate to your repo from command line/ terminal

- Now type cd [location you copied in the previous slide]



Exams 1/9/2024 1:55 PM File folder
Homeworks 1/9/2024 1:54 PM File folder
In class tasks 1/9/2024 2:20 PM File folder

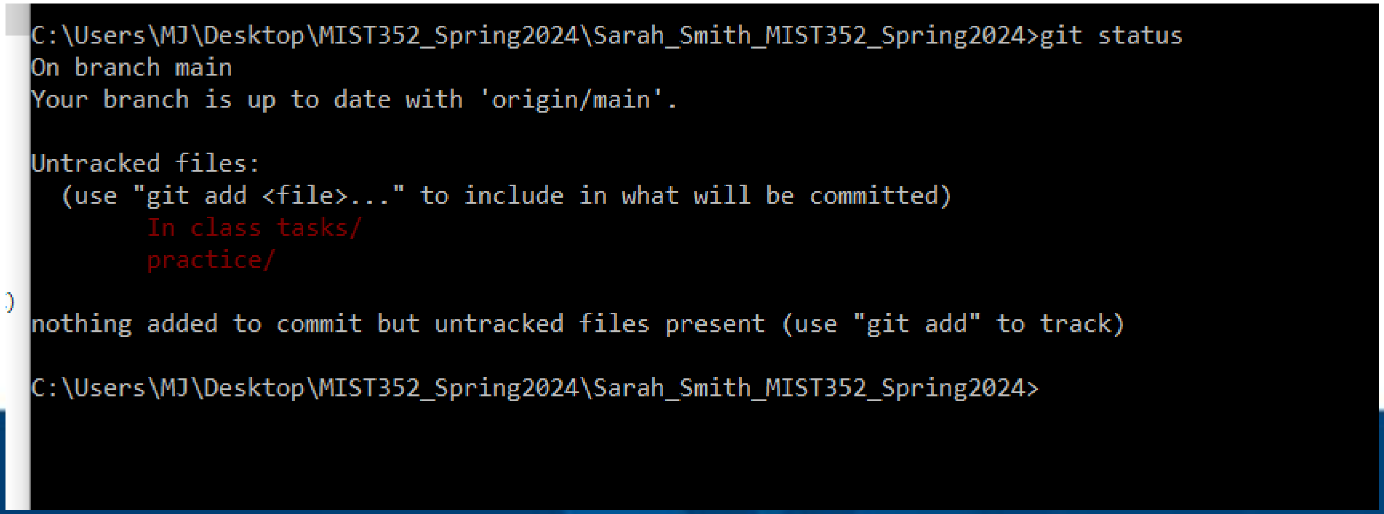
Command Prompt

```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MJ>cd C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024

C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>
```

Push changes to GihHub using git

- Now run the following commands in order

1. `Git status`: this shows you the changes made to your local repository.
   – Red means these files are added locally, but thy are not present on GitHub (remotely)

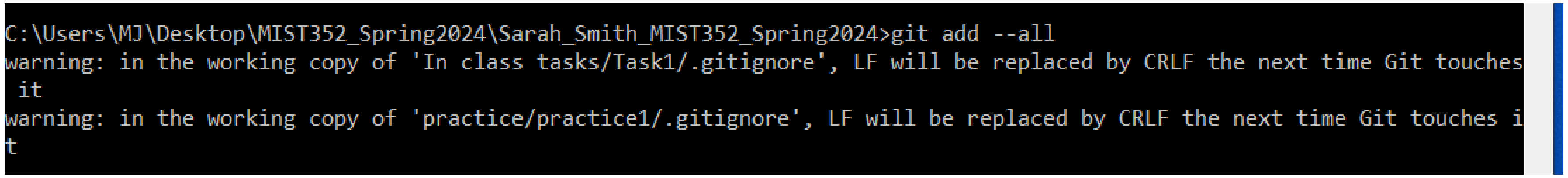


```
C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        In class tasks/
        practice/

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

C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>
```

Push changes to GitHub using git

- Now run the following commands in order:

`2. Git add -all`: this prepares changes to be pushed remotely by adding them for the next step



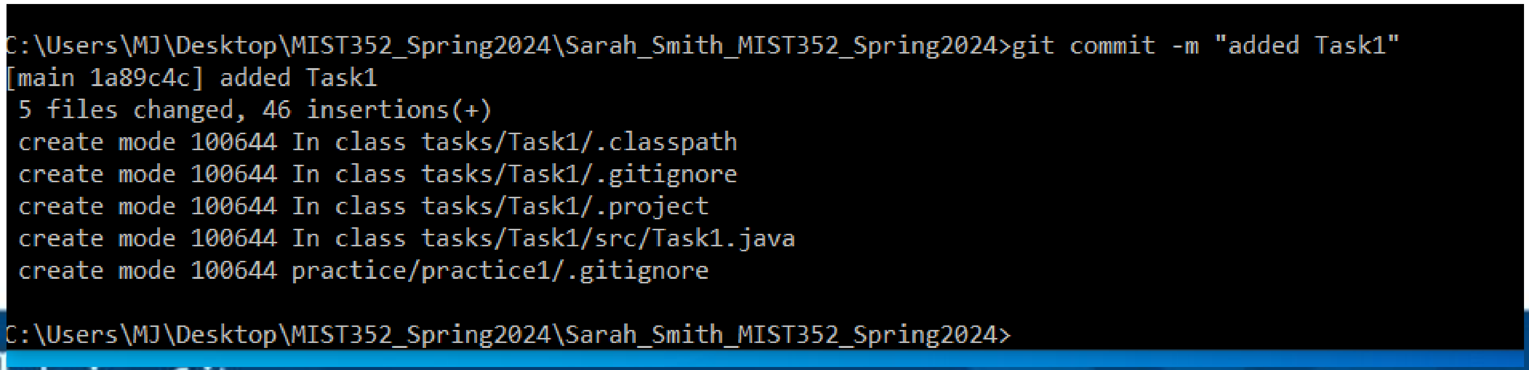
```
C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>git add --all
warning: in the working copy of 'In class tasks/Task1/.gitignore', LF will be replaced by CRLF the next time Git touches
 it
warning: in the working copy of 'practice/practice1/.gitignore', LF will be replaced by CRLF the next time Git touches i
t
```

3. Run the command "git status" again, do you see anything different?

Push changes to GitHub using git

- Now run the following commands in order

`4. Git commit –m “added Task 1”:` This is named a *commit*. Basically you are asking git to commit the changes. You should always provide the description of the changes you made between the “”. In this case, I added Task1 code, so I would type something like added Task1.

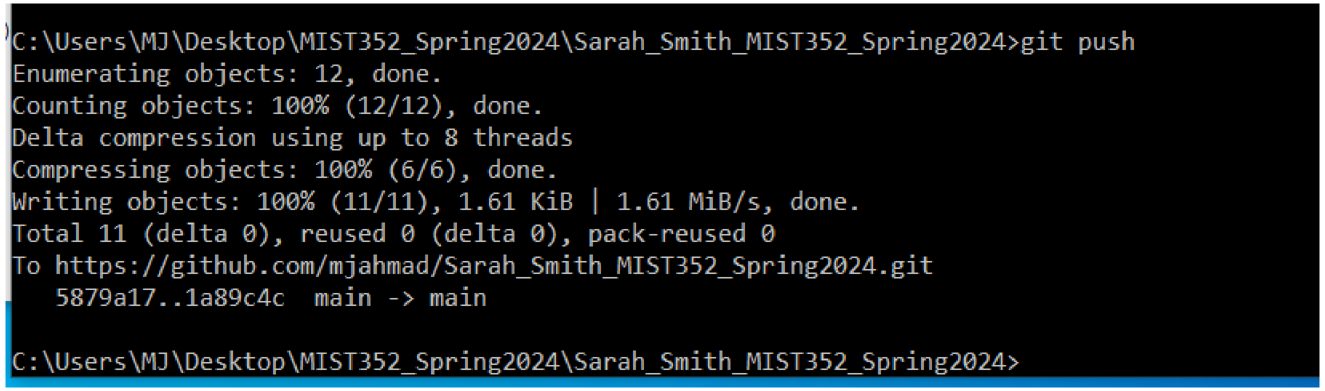


```
C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>git commit -m "added Task1"
[main 1a89c4c] added Task1
 5 files changed, 46 insertions(+)
 create mode 100644 In class tasks/Task1/.classpath
 create mode 100644 In class tasks/Task1/.gitignore
 create mode 100644 In class tasks/Task1/.project
 create mode 100644 In class tasks/Task1/src/Task1.java
 create mode 100644 practice/practice1/.gitignore

C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>
```

Push changes to GitHub using git

- Now run the following commands in order

`4. Git push:` This ”pushes” the changes to GitHub. It syncs your local repo to GitHub.



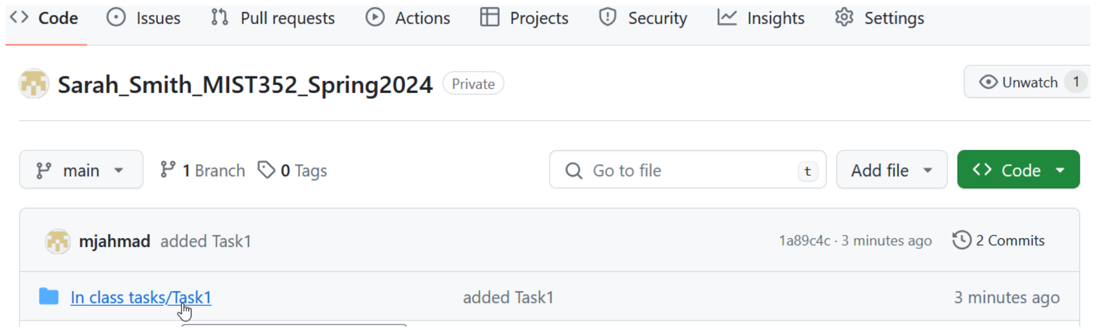
```
C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (11/11), 1.61 KiB | 1.61 MiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mjahmad/Sarah_Smith_MIST352_Spring2024.git
   5879a17..1a89c4c  main -> main

C:\Users\MJ\Desktop\MIST352_Spring2024\Sarah_Smith_MIST352_Spring2024>
```

Now see what happens when you try `git status`

How do you verify and ensure your code is pushed correctly?

There are several ways:

1. Go to GitHub and check your repository, do you see the project/ data you added? Do you see the commit?

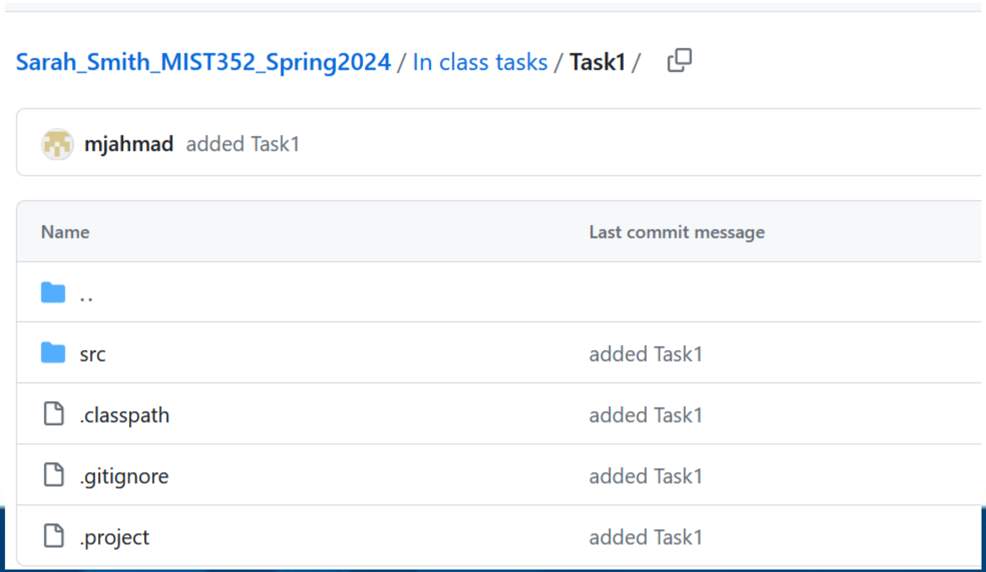


Sarah_Smith_MIST352_Spring2024 Private

1 Branch 0 Tags

mjahmad added Task1 1a89c4c · 3 minutes ago 2 Commits

In class tasks/Task1	added Task1	3 minutes ago



Sarah_Smith_MIST352_Spring2024 / In class tasks / Task1 /

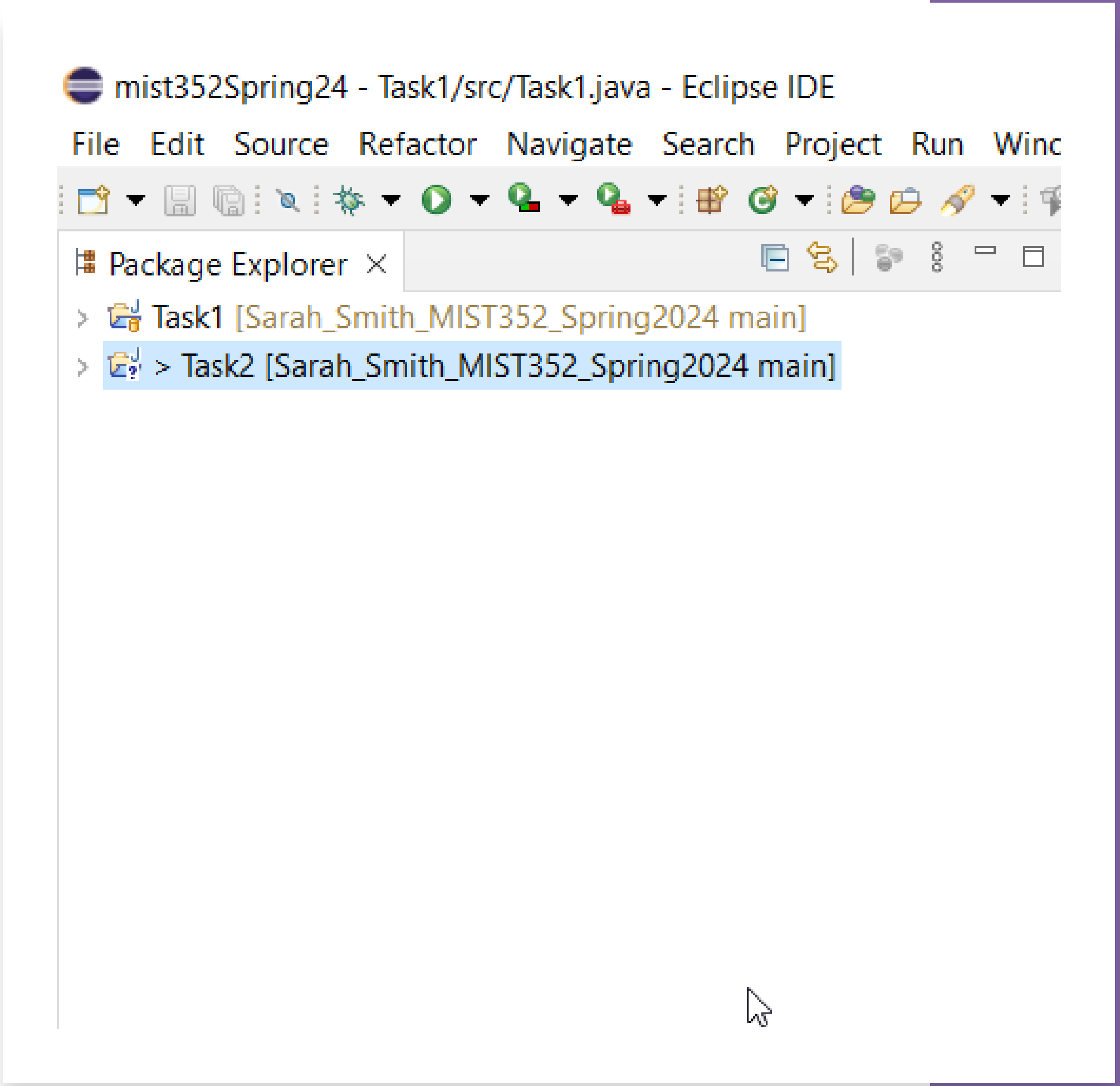
mjahmad added Task1

Name	Last commit message
..	
src	added Task1
.classpath	added Task1
.gitignore	added Task1
.project	added Task1

How do you verify and ensure your code is pushed correctly?

2. From Eclipse:

From the package explorer, If you see a ? Mark on the project’s name, then it has not been pushed correctly. If you see the other mark (as shown in Task1 of the screenshot), then your project has been pushed correctly.


mist352Spring24 - Task1/src/Task1.java - Eclipse IDE

Task1 [Sarah_Smith_MIST352_Spring2024 main]

> Task2 [Sarah_Smith_MIST352_Spring2024 main]

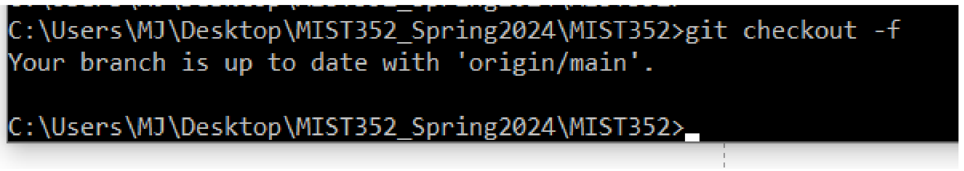
Obtaining the latest code from the MIST352 repo

- Although all codes will be released at once
  - Dr. Ahmad might upload code during classes or for homeworks
- You will need to update the MIST352 by “pulling” the latest copy of the code and obtaining any new codes
- See next slide

Obtaining the latest code from the MIST352 repo

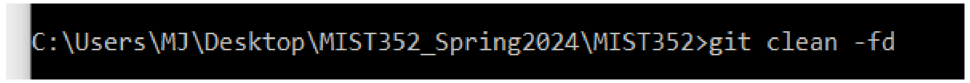
run these commands in this order:

1. Git checkout -f



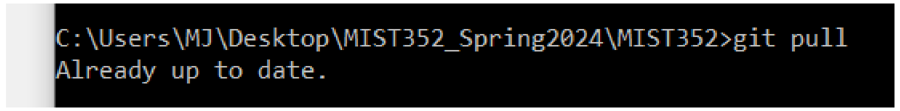
```
C:\Users\MJ\Desktop\MIST352_Spring2024\MIST352>git checkout -f
Your branch is up to date with 'origin/main'.

C:\Users\MJ\Desktop\MIST352_Spring2024\MIST352>
```

2. Git clean –fd



```
C:\Users\MJ\Desktop\MIST352_Spring2024\MIST352>git clean -fd
```

3. Git pull



```
C:\Users\MJ\Desktop\MIST352_Spring2024\MIST352>git pull
Already up to date.
```

Last notes

- Every time you create a new project or modify an existing one, you have to run the four past commands to sync everything t GitHub
- Always go to GitHub and verify that you can see your project there
- You may still commit code from Eclipse however, this has shown to be problematic.