CS489:
Applied Software Development

Lesson 1a: Development Environment and Tools

Wholeness

- In this lesson 1a, we will focus on setting up our software development environment, with the essential tools.
- Having a well set-up development environment, with the tools correctly configured and ready for use, is important for effective software development process.
- Science of Consciousness: *The field of all possibilities is the source of all solutions.*

Outline

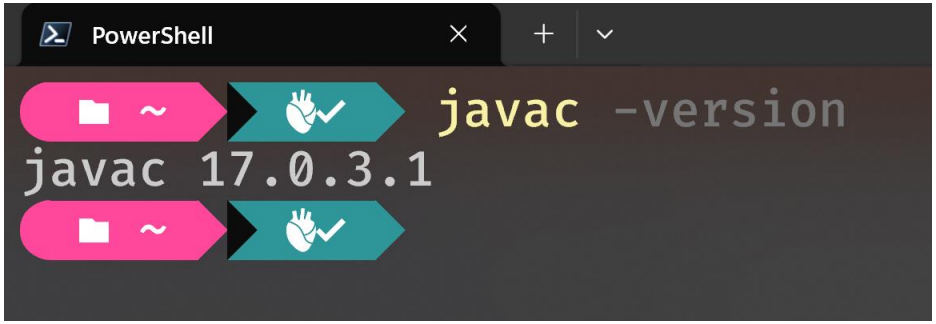
- Java Platform, Standard Edition (Java SE):
- Integrated Development Environment (IDE):
  - Jetbrains IntelliJ IDEA
  - Spring Tools Suite (Eclipse-based IDE for Spring)
- Version Control System: Git and Github
- DevOps and Build Automation Tools:
  - Apache Maven
  - Gradle

Java SE Development Kit (JDK)

- Obtain, install and configure the Java JDK for your OS. Minimum required version is JDK17
  - Oracle JDK: https://www.oracle.com/java/technologies/downloads/
  - Eclipse Temurin JDK: https://adoptium.net/temurin/releases/
- If required, consider using SDKMAN (https://sdkman.io/) for managing multiple versions of software development kits (SDKs).

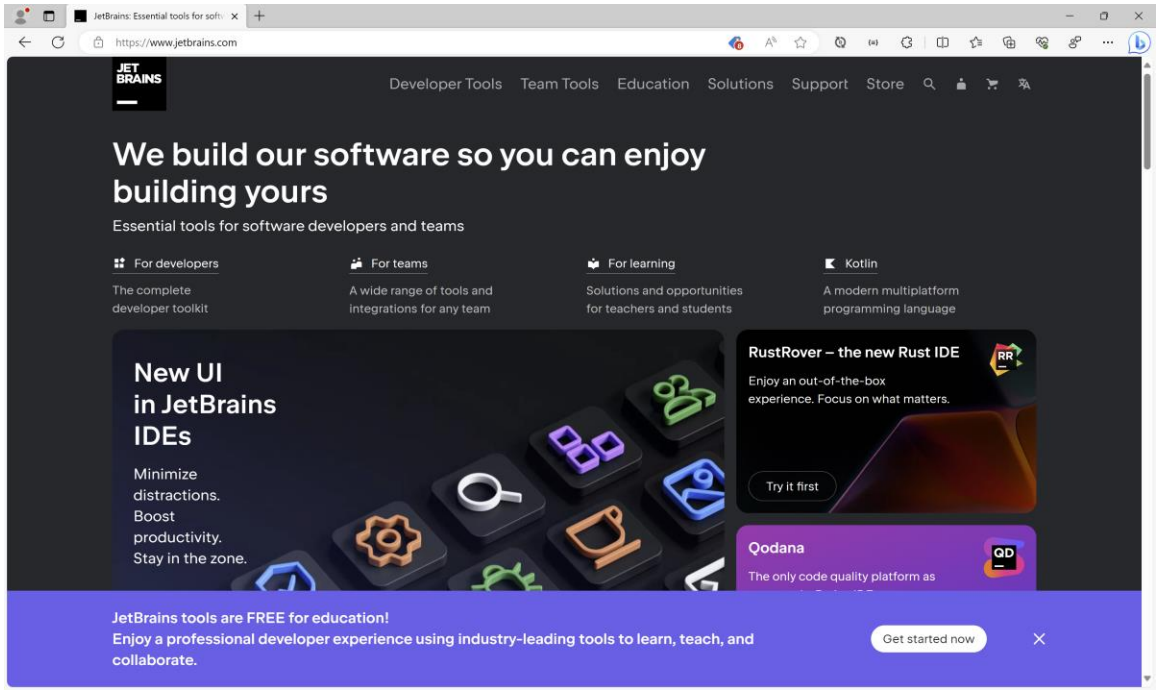
Exercise

- Open a Command terminal/shell
- Execute the command: *javac –version*
- Verify that the command executes successfully and displays the Java Compiler info with a version of 17 or higher. If not, please obtain and install JDK version 17 or higher.



Integrated Development Environment (IDE)

- Jetbrains IntelliJ IDEA (Ultimate Edition):



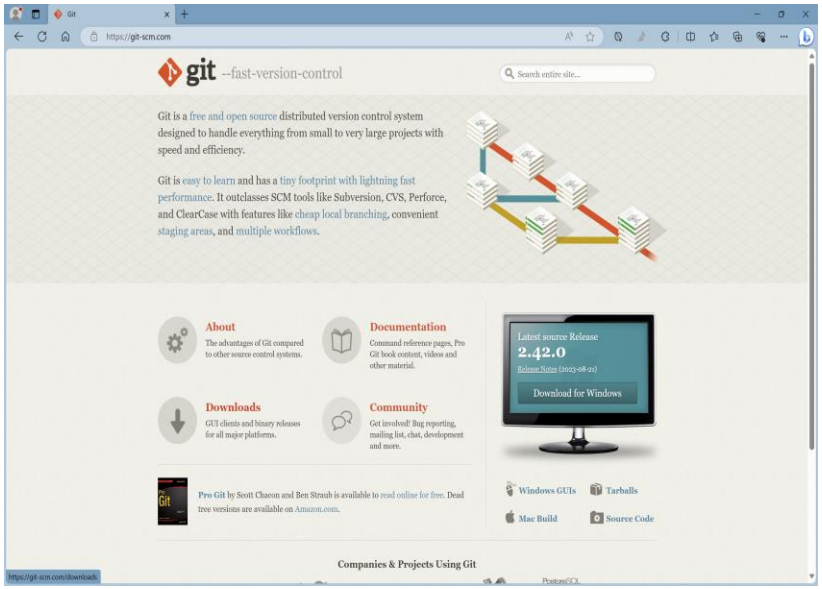
- Spring Tools 4 for Eclipse (a.k.a Spring Tools Suite)

https://spring.io/tools

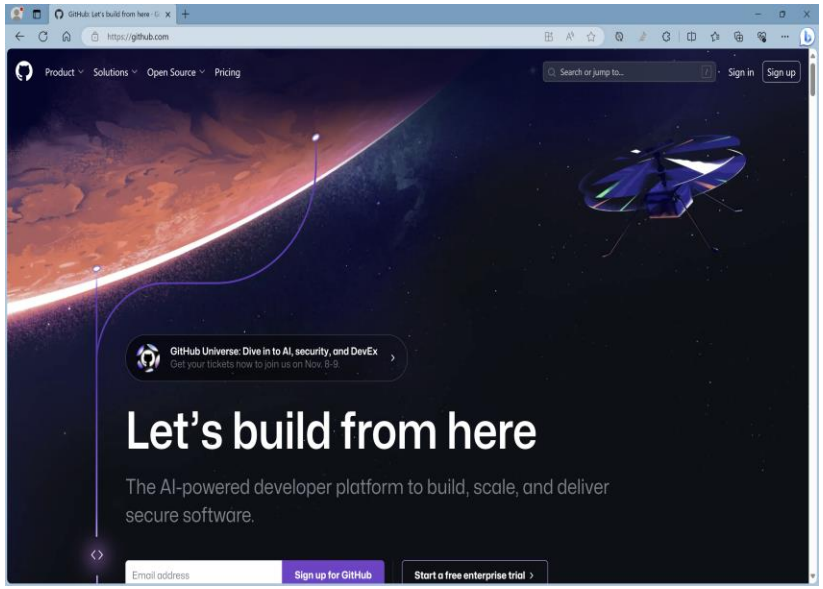
- Visual Studio Code

Version Control System (VCS)

Git



Github



Exercise

- Open a Command terminal/shell
- Execute the command: *git --version*
- Verify that the command executes successfully and displays the version info for your 'git' CLI tool, with a version of 2.40.x or higher. If not, please obtain and install/upgrade git and configure it properly, as directed in the next slides.

Git

- Git is a distributed Version Control System
- It is a tool for tracking changes in computer files
- It is primarily used for source code management in Software development projects
- It helps in coordinating the work of multiple developers in a project team
- It was created by Linus Torvalds for managing development of the Linux kernel source code

Obtain, Install and Configure Git

- For Windows:
  - Go to https://git-scm.com/download/win and download the Git for Windows installer
- For Mac OS:
  - Go to https://git-scm.com/download/mac and install Git for macOS using Homebrew
- For Linux:
  - Use the package management tool for your distro

Git is a CLI tool

- To see the general usage guide for Git commands, open a terminal/shell and execute: git --help

```
PowerShell
workspace-gitdemo  git --help
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
```

Configurations for Git

- Git can store/use 3 types of configuration data located in 3 different places, as follows:

  – **System config**:

  These are configuration settings that applies to all users on the system:

  – On MacOS and Linux: */etc/gitconfig*

  – For Windows: *C:\Program Files\Git\etc\gitconfig*

Configurations for Git

- **Exercise:**
  - To display the System config settings for your Git, open a Command terminal/shell and execute:

    git config --list --system

```
PowerShell
workspace-gitdemo  git config --list --system
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
core.editor="C:\\Program Files\\Microsoft VS Code\\bin\\code" --wait
core.fsmonitor=true
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
```

Configurations for Git

- Git can store/use 3 types of configuration data located in 3 different places, as follows:

  – **Global config**:

  These are custom configuration settings that applies to the user who is currently logged into the system:

  – On MacOS and Linux: *~/.gitconfig*

  – For Windows: *C:\Users\{username}\.gitconfig*

Configurations for Git

- **Exercise:**
  - To display the Global config settings for your Git, open a Command terminal/shell and execute:

    git config --list --global

```
PowerShell
workspace-gitdemo  git config --list --global
user.name=Prof OKalu
user.email=okalu1@outlook.com
```

If the two global config settings are not displayed by your Git, then add them by executing the following commands:

git config --global user.name “your_name"

git config --global user.email “your_email“

Also, to set/change the default name of the initial branch created when a new repository is initialized, execute the command:

git config --global init.defaultbranch “main"

Configurations for Git

- Git can store/use 3 types of configuration data located in 3 different places, as follows:

  – **Local config**:

  These are custom configuration settings that applies to a specific project repository:

  – These will be located in a file named, config, stored inside a hidden directory named, .git, found inside the project repository's directory

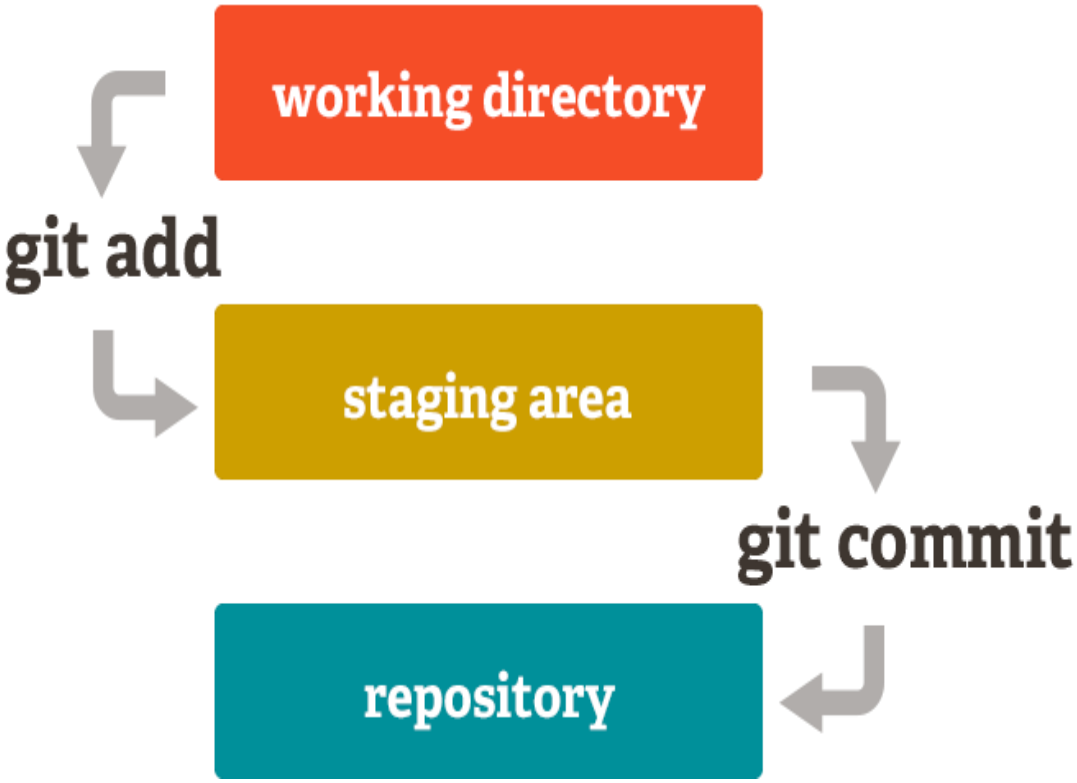
Create a new Git repository

- **Exercise**:
  - Open a Command terminal/shell
  - Create and change into a new directory (choose a name)
  - To make the directory a new Project/Git Repository, simply execute the command, *git init*

```
PowerShell
workspace-gitdemo  mkdir cs489-appsd-202310 && chdir cs489-appsd-202310

    Directory: C:\oak\mymiu\teaching\courses\cs489-applied-software-dev\_course-materials\worksp
aces\workspace-gitdemo


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          9/28/2023   3:09 PM                cs489-appsd-202310

cs489-appsd-202310  git init
Initialized empty Git repository in C:/oak/mymiu/teaching/courses/cs489-applied-software-dev/_course-
materials/workspaces/workspace-gitdemo/cs489-appsd-202310/.git/
cs489-appsd-202310  main ≠
```

Git Components

- Working Tree
  - Where you're currently working, where your files live
  - "untracked" area of git
  - Git is not aware of the files or changes in the working tree until you tell it to pay attention to them.
- Staging Area
  - Git starts tracking and saving changes that occurs in files
  - The saved changes reflect in the `.git` directory
- Local Repo
  - Everything in your `.git` directory
  - All of your checkpoints or commits are in Local Repository
  - Don't delete it.



Common Git commands for working with a local repository

- git status
- git add hello.txt (or git add .)
- git commit -m "First cut"
- git log

Common Git commands for working with a local repository

- git config --global alias.st status (creates an alias)
- git diff hello.txt (see diff btw changes in working area and committed changes)
- git branch
- git branch <new-branch-name> (creates a new branch)
- git branch -d <branch-name> (deletes an existing branch)
- git merge <branch-name> (Merges changes from <branch-name> into current active branch

Github

- Github is an online service (website) for hosting remote Git repositories
- It typically serves as a team collaboration site, where a software project team can host/share remote copy of the project repository among members
- Github can be said to be equivalent to Git as a service (web-hosted)
- Other alternatives are: gitlab and bitbucket

Common Git commands for working with a remote repository

- git clone https://github.com/spring-project/petclinic-repo.git
- git remote add origin https://github.com/okalu/git-repo1.git
- git remote (or git remote -v)
- git push -u origin main
- git pull origin main

Exercise 1

- Create a new project repository named, cs489-appsd-202310, and publish it to your github account:
  - Create and Initialize a new git repository for the project, on your local machine.
  - Create a directory named, "docs", and in it add a file named, ProjectSpecification.txt and add the following content into the file: Welcome to my project
  - Commit your change(s) and publish the repository to your github account

Exercise 2

- Contribute to 3rd party github repository:
  - You have been tasked to work on this repository: https://github.com/okalu/cs489-apsd-gitdemo1
  - Fork the repository from my account to your github account.
  - Clone your forked copy from your github account to your local machine
  - Add your new folder(s)/file(s)
  - Commit your change(s)/addition(s)
  - Push your commit(s) to the remote repository named origin, which publishes it to your github account
  - Create/send a Pull Request from your Github account to the upstream remote repository

CS489:
Applied Software Development