DEMO NOTES FOR CREATING JAVA PROJECT IN ECLIPSE

- How to Install JDK
    - Browse to the AdoptOpenJDK website.
    - Choose **OpenJDK 11 (LTS)** as the version.
    - Choose **OpenJ9** as the JVM.
    - Click the **Latest release** download button to download the package.
- How to Install and setup Eclipse/ Select Workspace
    - Browse to https://www.eclipse.org/downloads/packages/installer
    - Select installer bundle for your platform and download it. Run self-extracting program for the installer.
    - Choose "Eclipse IDE for Java Developers" and Click Install. (Note the installation folder and JDK version)
    - Exit Installer. Do not run the Launch button.
    - Go to the installation folder and launch eclipse. Choose your workspace. Workspace is the directory where Eclipse will place all of your projects.
    - Verify that the JDK version you chose is the one Eclipse is directed to use. Window-- Preferences-- Java-- Installed JREs.
- Create a project
    - Close welcome screen
    - File>New>Java Project. Name it *myFirstProject*. By default, Eclipse saves project in the default workspace. **Src** folder contains the packages you create. Under **Libraries**, if you want, you can add libraries, external JAR files. **Order and Export** – which part of the project or how project will be exported
    - You will see project in package explorer. With a src folder and a default JRE folders within.

- Perspectives/Views
    - Different perspectives to visually show the workspace you are dealing with. Visual container for a set of views and content editors
    - E.g. Java perspective: Package Explorer on the left pane. Package is where you organize classes and show the hierarchy
    - Source Code Editor in the middle
    - Outline on the right.
    - An eclipse window can have multiple perspectives open in it but only one perspective is active at any point of time. A user can switch between open perspectives or open a new perspective. Window > Perspective >Open perspective
    - View can be edited/added for each perspective

- Create a package and class and run 1st Java application
    - Inside src file , create a cs3300 package
    - Create a *PrintCheck* class. Options to consider it as main class so that this is the class that will run when your project is run.
    - In the editor, print a line.
    - Run as>Java application. Or green arrow button. Output in Console.

- Run Configuration
    - *Run configuration* to configure different parameters for your program. Define arguments, Dependencies, Class path and other environmental options.
    - New class called *AddNumbers*. Add 2 numbers.
    - Pass arguments and Run

- Debugging
    - Run, Result incorrect. Debug

- Debug as> Java application. Debug perspective. On right side, we see variables. On left, the debug pane.
- Breakpoint, Execute a line , step over, hover variables
- Red button to stop or complete debugging

DEMO NOTES FOR CREATING JUNIT TESTING IN ECLIPSE

- Create a new project m*yJUnitProject*, package *cs3300JUnit* and class *AddConcatenate*
- Library>Add Library>Junit (4 or 5)
- Add 2 methods, 1 to add 2 numbers and 1 to concatenate 2 strings
- Add 2 Junit test cases for both methods, Use Junit 4, add to build path when prompted.
- If not working, Right click on the Java project and select Build Path > Configure Build Path>Libraries>Add>Junit
- Run as Junit test

- Add a Junit test suite to run both simultaneously. Class>add>other>Junit Test Suite

---

DEMO NOTES FOR CREATING A MAVEN PROJECT IN ECLIPSE

- Eclipse comes with integrated Maven environment. No need to install any extra plugin to create a project.
- New>Project>Maven>Maven project
- Simple project.
- Groupid: uniquely identifies your project across all projects. like package. *cs3300*
- ArtifactId is the name of the project. *myMavenProject*
- version is SNAPSHOT if it is under construction
- Once the project is created. We can see it in package explorer. We have 2 src folders, 1 for packages and classes, another for

resources. 2 test folders. Unit test folders are always included. Test source files and resources for test.

- Pom.xml—open in editor. Contains project information.
- Create class in src/main/java. Package name will be same as group id. Class name, *MultiplyMaven*
- Create a package and a simple class that returns multiplication of 2 numbers
- Create a Junit test, but do not add Junit 4 to the build path as before
- So, we do not have Junit jar file in our dependencies, shows error. Normally we need to download jar file and add it to build path. But here we can add in pom.xml file- automatically downloaded and added to project from maven repository.
- Search maven junit dependency on google, copy in pom.xml, save.
- Automatically a folder called maven dependency appears- downloaded and ready.