SOFTWARE ENGINEERING LAB

EXERCISE – 5

TOPIC – 2

CREATING MAVEN JAVA PROJECT USING ECLIPSE AND PUSH INTO TO GITHUB

Note: At every step take screenshots and save in a document

1. Introduction to Maven

What is Maven? Maven is a tool that automates tasks like managing dependencies, building, testing, and packaging Java projects. Imagine Maven as an assistant that keeps everything organized, making the project easier to work with and maintain.

Why Use Maven?

- **Handles Dependencies**: Automatically adds libraries your project needs.
- **Standard Structure**: Follows a consistent project format.
- **Automates Processes**: Streamlines tasks like compiling and testing.
- **Integration**: Works smoothly with tools like GitHub (for version control) and Jenkins (for continuous integration).

2. Setting Up a Maven Project in Eclipse

1. **Creating a New Maven Project**:
   - In Eclipse, go to **File > New > Maven Project**.
   - Choose the `quickstart` archetype under `org.apache.maven.archetypes`.
   - Set the **Group ID** and **Artifact ID** (the unique identifier for your project).
   - Click **Finish** to create the project.

2. **Understanding Reverse Domain Naming**:
    - Reverse domain naming is a way of creating unique identifiers for projects by using your website address backward.
    - **How It Works**: If your website is `mywebsite.com`, you'd write it backward as `com.mywebsite` for your **Group ID**. This helps avoid naming conflicts by making your project's name globally unique.
    - **Example**: For a project created by `example.com`, you'd use `com.example.projectname` as the **Group ID**.
    - **Why Maven Uses It**: This naming convention is widely recognized, ensuring that your project doesn't accidentally share the same name with others.
3. **Why Use the `org.apache.maven` Archetype and Quickstart?**
    - **What's an Archetype?** An archetype is a template that provides a ready-made project structure.
    - **Why `org.apache.maven.archetypes`?**: This is a standard, reliable setup provided by Maven.
    - **Why Quickstart?** The `quickstart` archetype is perfect for simple Java applications, generating:
        - A basic directory structure,
        - A sample Java file (`App.java`), and
        - Essential configuration files.
    - This setup is great for starting a new project quickly and efficiently.
4. **Understanding the Generated Files**:
    - **`pom.xml`**: The Project Object Model file, `pom.xml`, is the core of your Maven project. It defines the project's dependencies, settings, and configurations.
    - **`App.java`**: Located in `src/main/java`, this file contains a basic "Hello World" program to verify your setup.

3. Running Essential Maven Goals

Maven goals are tasks that Maven performs on your project, such as cleaning, compiling, and testing.

1. **Clean**:
    - **Purpose**: Removes any old compiled files.
    - **Why Clean?** Ensures you start fresh each time, avoiding old data that could cause errors.
    - **How to Run**: Right-click the project, choose **Run As > Maven Clean**, and check the console for "Build Success."
2. **Install**:
    - **Purpose**: Compiles and packages your code, then saves it in your local repository.
    - **Why Install?** Puts the project into a distributable format (like a `.jar` file) for use elsewhere.
    - **How to Run**: Right-click the project, select **Run As > Maven Install**, and confirm "Build Success" in the console.
3. **Test**:
    - **Purpose**: Runs tests to ensure your code behaves as expected.
    - **Why Test?** Helps catch errors early by checking that everything works correctly.
    - **How to Run**: Right-click on the project and select **Run As > Maven Test**.
4. **Running All Goals Together**:
    - Enter `Clean Install Test` in the Goals field to run all three tasks sequentially.
    - **How to Run**: Click **Apply** and **Run**. This runs clean, compile, and test in one step.

4. Running the Project to View Output

- Right-click the project, select **Run As > Java Application**.
- Choose the project and click **OK**.
- The console should display the output, such as "Hello World," if everything is set up correctly.

5. Pushing the Project to GitHub

Why Use GitHub? GitHub is a cloud-based platform where you can store, manage, and collaborate on your projects. It acts as a backup and allows others to work on the project with you.

Steps for Pushing the Project to GitHub:

1. **Create a Repository on GitHub**:
   - Go to GitHub and create a new repository.
   - Select `.gitignore` for Maven.
2. **Why Use `.gitignore`?**
   - **Purpose**: `.gitignore` is a file that tells Git which files or directories to ignore.
   - **Why for Maven Projects?** Maven creates many temporary files and folders, like `/target/`, which don't need to be stored in GitHub. `.gitignore` helps keep your repository clean by ignoring these unnecessary files.
3. **Copy the Repository URL**:
   - Copy the HTTPS link from GitHub to link your local project.
4. **Push the Project Using Git Bash**:
   - **Clone** the GitHub repository to your local system, creating a folder connected to your GitHub repository.
   - Copy files from your Eclipse project folder into this cloned repository.
   - Open Git Bash in the project directory and use the following commands:

```
git add .                # Adds all files
git commit -m "Initial commit"   # Commits changes with a message
git push origin main     # Pushes your project to GitHub
```

   - Refresh GitHub to verify the project has been successfully uploaded.