### 4-1: Getting Started with Java Practice Activities

### Vocabulary

**1.A naming convention to eliminate spaces in a name, but to ease readability with capitalization**: Camel Case

**2.To change the different physical location onto which you will store and save your files**: Change Directory

**3.Stored inside a project, a mechanism for organizing Java classes into namespaces, or containers**: Package

**4.The method inside a class that runs when the class is compiled and ran**: main Method

**5.A construct that is used as a blueprint to create objects. Also a construct in which objects are created**: Class

**6.An option to choose a combination of views and editors**: Perspective

**7.Areas within the Java IDE that provide a way to navigate a hierarchy of information and allow modifications to elements**: Views

### 1.Try It/Solve It

#### 1. Presentation on Views in a Java IDE

To create a presentation highlighting five or more views in a Java IDE, consider the following structure:

**a. Presentation Introduction**

* **Purpose**: Explain the purpose of the presentation, which is to introduce and demonstrate useful views in a Java IDE for programmers.
* **Team Members**: Introduce the team members who created and are presenting the information.

**b. List of Views**

1. **Project Explorer**: Displays the project structure and files, allowing easy navigation and management of project resources.
2. **Editor**: Where you write and edit code. It provides syntax highlighting and code completion.
3. **Console**: Shows the output of the program and logs messages, helpful for debugging and runtime information.
4. **Debug**: Allows you to set breakpoints, step through code, and inspect variables during debugging sessions.
5. **Problems**: Lists compilation errors and warnings in the code, providing feedback to correct issues.

**c. Reason for Selection**

* **Project Explorer**: Essential for navigating and managing project files.
* **Editor**: Central to writing and editing code.
* **Console**: Crucial for viewing program output and debugging.
* **Debug**: Important for diagnosing and fixing runtime issues.
* **Problems**: Helps identify and resolve compilation issues.

**d. Process to Choose Views**

* Discussed the functionality each view offers and its importance in the development process.
* Selected views based on their impact on productivity and debugging efficiency.

**e. Demonstration and Description**

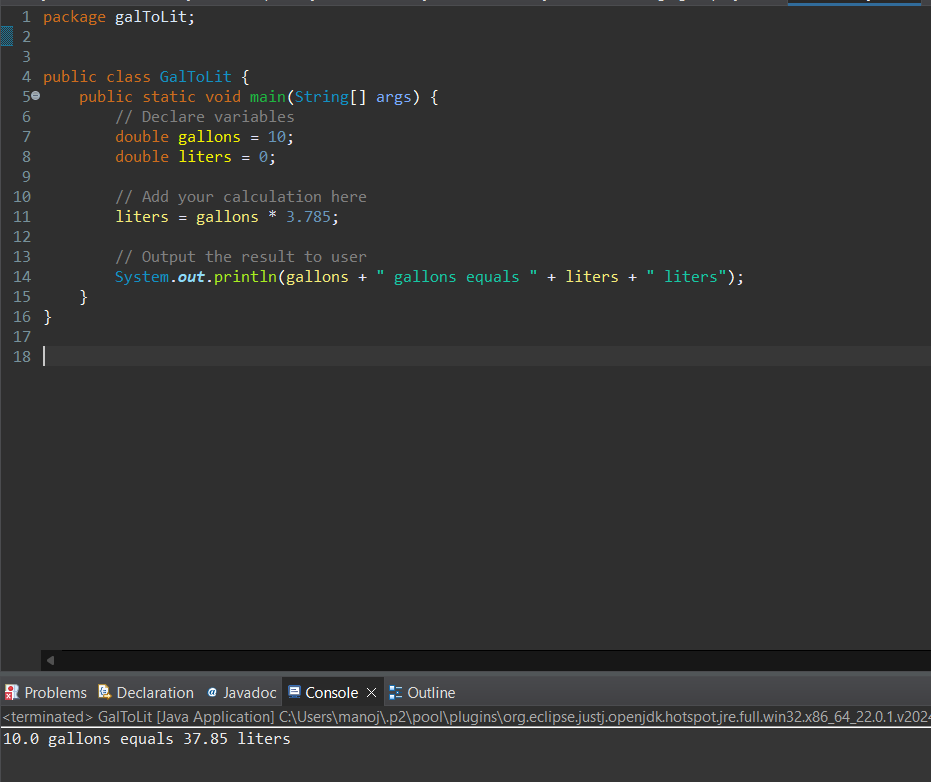
* **Project Explorer**: Show how to navigate the project structure and open files.
* **Editor**: Demonstrate code writing, editing, and the use of code completion features.
* **Console**: Run a sample program and show output in the console.
* **Debug**: Set a breakpoint, start a debugging session, and inspect variables.
* **Problems**: Introduce common issues and how they appear in the Problems view.

**f. Presentation Summary**

* Recap the importance of each view and how they contribute to effective programming.
* Emphasize how these views help streamline development and debugging.

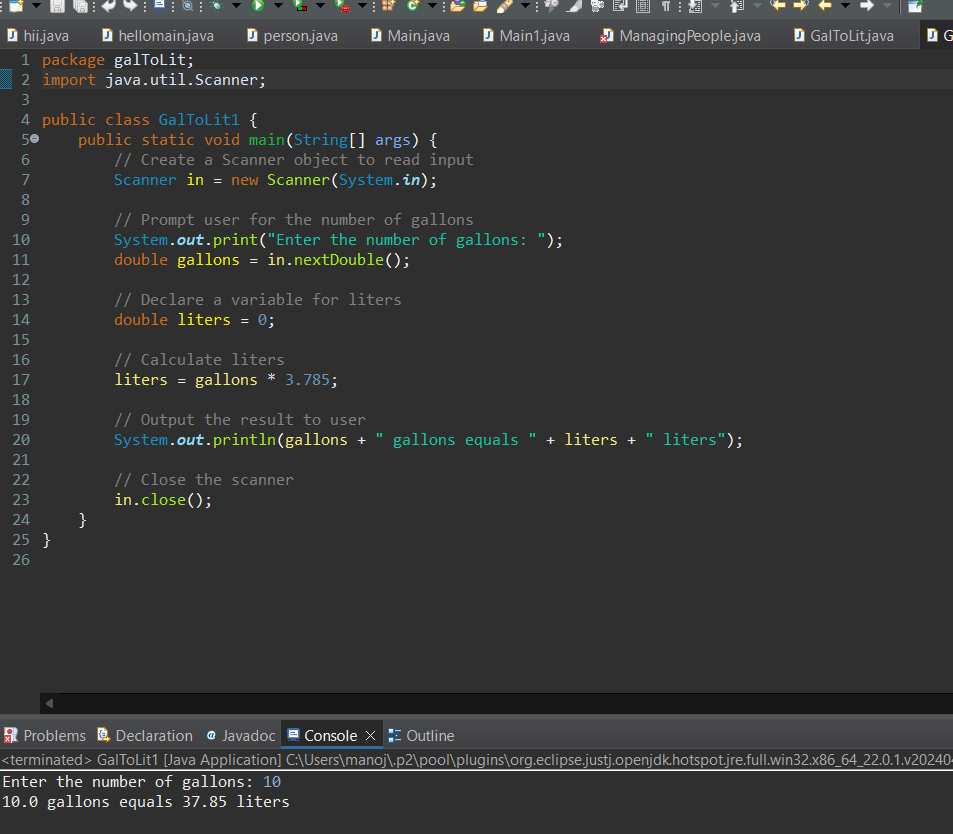
#### 2. Code for GalToLit.java

Here’s the completed GalToLit class to convert gallons to liters:



#### 3. Modifying Code to Use Scanner

Here’s the modified code using the Scanner class to accept user input:



#### 4. Ways to Test the Program

**1.Manual Testing**: Run the program and enter various values for gallons to check if the conversion is correct and the output is accurate. For example, input 10, 5.5, and 0 gallons and verify the results.

**2.Boundary Testing**: Test edge cases such as zero gallons (0), a very large number of gallons, and negative numbers (if valid) to ensure the program handles different ranges of input properly.

**3.Automated Testing**: Write unit tests using a framework like JUnit to automate the testing of the conversion logic. For example, create a test class to verify that the conversion from gallons to liters is computed correctly for different inputs.