

**1. Create a presentation to highlight five or more Views that may be of interest to a programmer using a Java IDE. Use the help system to learn about the Views available in a Java IDE. Work in teams of two to create and deliver the presentation. The presentation should include the following: a. A presentation introduction defining the presentation purpose and the team members. b. A list of five or more Views in a Java IDE that will be highlighted. c. The reason your team selected the five Views to demonstrate. d. The process your team went through to choose the five Views. e. The actual demonstration and description of the components. f. The presentation summary.**

A. Creating a presentation that highlights five or more views of interest to a programmer using a Java IDE can be both informative and engaging. Here's a structured approach to creating this presentation:

#### **Slide 1: Title Slide**

- **Title:** Views of Interest in a Java IDE
- **Subtitle:** An In-depth Look at Key IDE Views for Java Programmers
- **Team Members:** [Name 1] & [Name 2]

#### **Slide 2: Introduction**

- **Purpose:** To introduce and explain five key views in a Java IDE that can significantly enhance a programmer's productivity and efficiency.
- **Team Members:** [Name 1] and [Name 2]

#### **Slide 3: List of Views**

- **Views to be Highlighted:**
  1. **Project Explorer**
  2. **Package Explorer**
  3. **Editor View**
  4. **Console View**
  5. **Debug View**
  6. **Outline View**

## Slide 4: Selection Rationale

- **Why These Views?**
  - **Project Explorer:** Central hub for navigating projects.
  - **Package Explorer:** Detailed view of package hierarchy.
  - **Editor View:** Main area for code writing and editing.
  - **Console View:** Displays output and error messages.
  - **Debug View:** Crucial for debugging and stepping through code.
  - **Outline View:** Provides an overview of class structure and methods.

## Slide 5: Selection Process

- **Process Description:**
  1. **Initial Research:** Using the IDE's help system to list all available views.
  2. **Narrowing Down:** Based on common tasks and usage frequency in Java development.
  3. **Team Discussion:** Prioritizing views that enhance productivity and ease of use.
  4. **Final Selection:** Choosing views that offer diverse functionalities and cover different aspects of development.

## Slide 6: Demonstration - Project Explorer

- **Description:** Central navigation pane for managing projects.
- **Features:**
  - Organize multiple projects.
  - Easy access to files and resources.
- **Demonstration:**
  - Show how to create a new project.
  - Navigate through an existing project structure.

## Slide 7: Demonstration - Package Explorer

- **Description:** Displays the hierarchical structure of packages.

- **Features:**
  - Detailed view of packages and classes.
  - Facilitates navigation and organization of code.
- **Demonstration:**
  - Show the structure of a sample Java project.
  - Highlight navigation between packages and classes.

### **Slide 8: Demonstration - Editor View**

- **Description:** The main workspace for writing and editing code.
- **Features:**
  - Syntax highlighting.
  - Code completion and suggestions.
- **Demonstration:**
  - Write a simple Java program.
  - Highlight code completion and syntax checking.

### **Slide 9: Demonstration - Console View**

- **Description:** Displays program output and error messages.
- **Features:**
  - Real-time feedback during program execution.
  - Error logs for debugging.
- **Demonstration:**
  - Run a Java program and show output.
  - Demonstrate error messages and logs.

### **Slide 10: Demonstration - Debug View**

- **Description:** Tools for debugging and stepping through code.
- **Features:**
  - Breakpoints and step execution.
  - Variable inspection and watch.

- **Demonstration:**
  - Set breakpoints in a sample program.
  - Step through code and inspect variables.

### **Slide 11: Demonstration - Outline View**

- **Description:** Overview of class structure and methods.
- **Features:**
  - Quick navigation to methods and fields.
  - Visual representation of code structure.
- **Demonstration:**
  - Show the outline of a complex class.
  - Navigate to specific methods and fields.

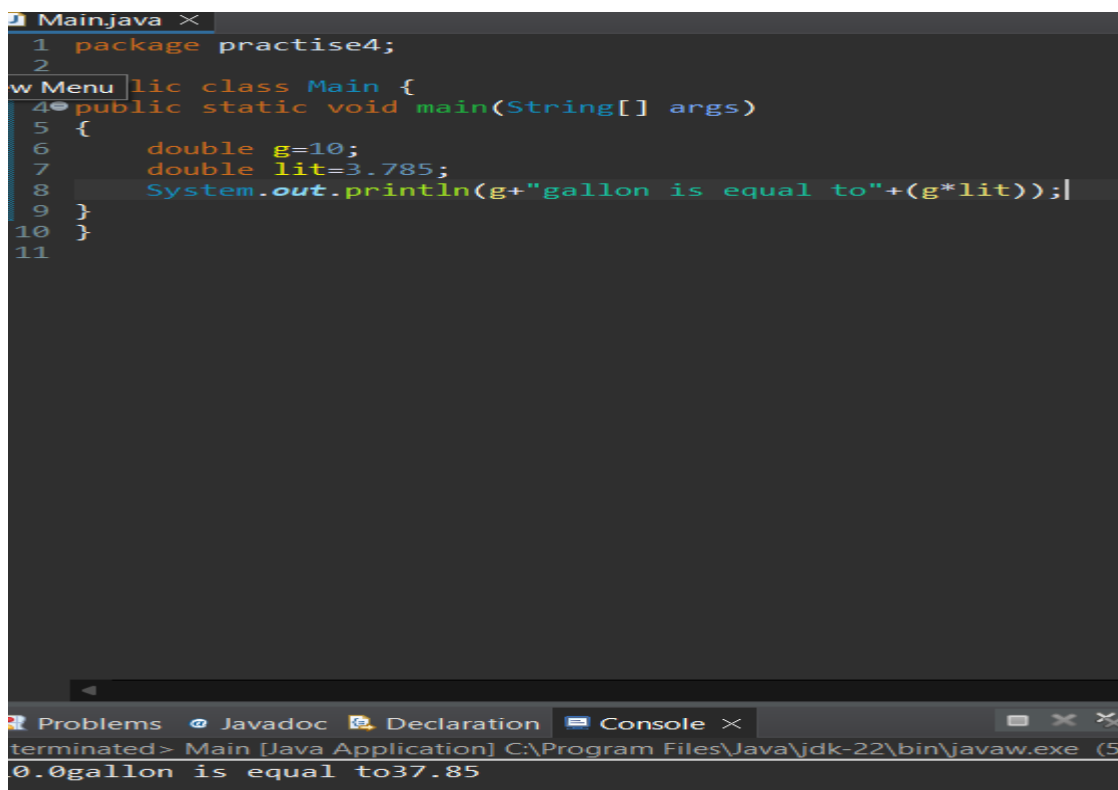
### **Slide 12: Summary**

- **Recap of Views:**
  - Project Explorer
  - Package Explorer
  - Editor View
  - Console View
  - Debug View
  - Outline View
- **Benefits:**
  - Improved navigation and organization.
  - Enhanced coding and debugging experience.
  - Streamlined development process.

### **Slide 13: Q&A**

- **Invite Questions:** Encourage the audience to ask questions for further clarification or demonstration.
- **Closing Remarks:** Thank the audience for their attention and participation.

2. The formula for converting gallons to liters is: 1 US gallon = 3.785 liters. This program will convert a specific number of gallons (10) to liters and then display the output. The concepts in this practice will be explored in more detail throughout the course. Create a new project, package, and java class with a main method. Use the code below as a starting point and complete the code for the program. (Name your package galToLit and class GalToLit).

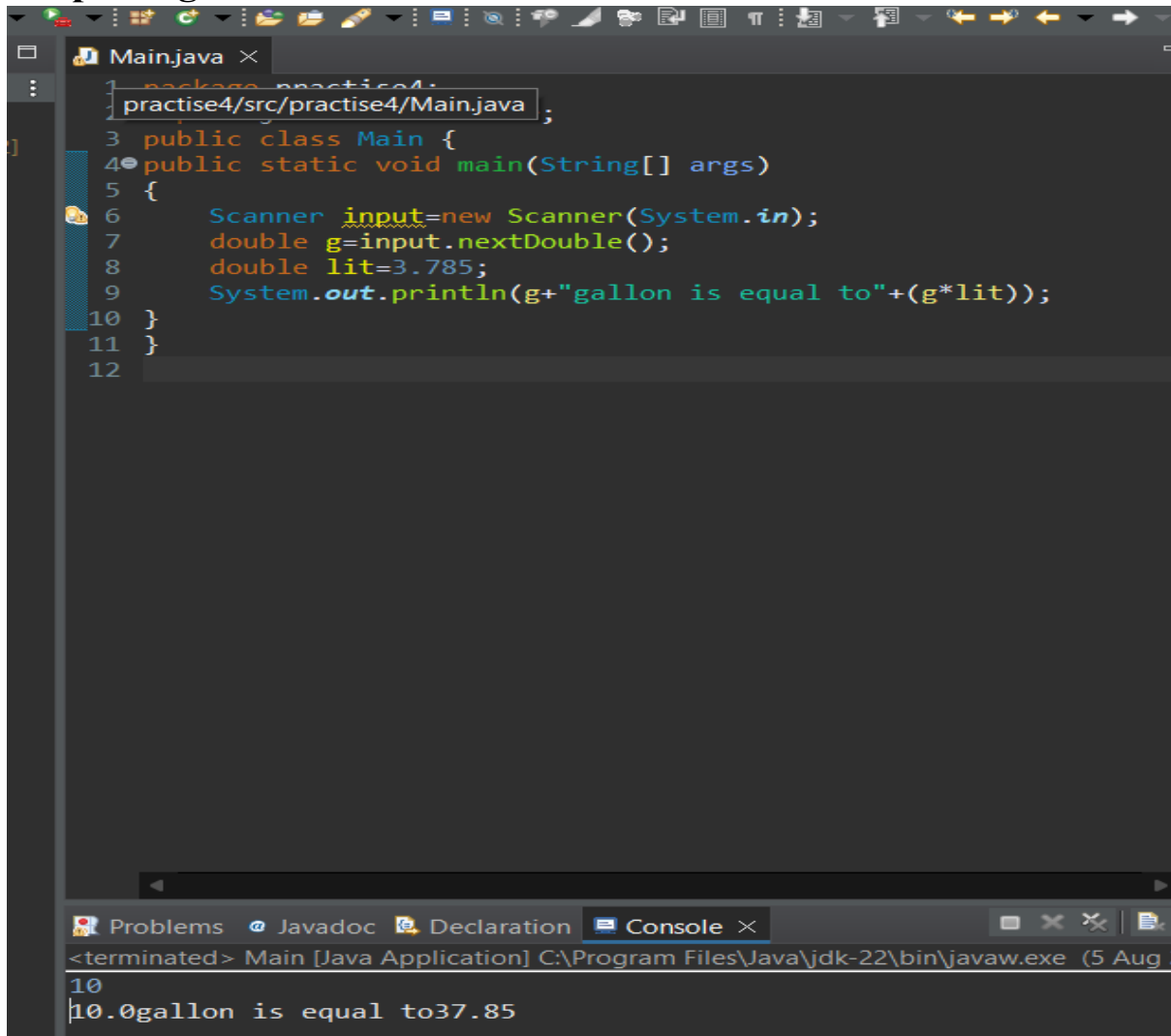


```
1 package practise4;
2
3 public class Main {
4     public static void main(String[] args)
5     {
6         double g=10;
7         double lit=3.785;
8         System.out.println(g+"gallon is equal to"+(g*lit));
9     }
10 }
11
```

terminated> Main [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (5  
0.0gallon is equal to37.85

3. The Scanner class can be used to accept input from the user. Modify the code written in step 2 to prompt a user for the number of gallons to compute. To declare an instance of the Scanner class, use the code below: Scanner in = new Scanner(System.in); Your Java IDE may prompt you to import the java.util.Scanner package, or you can manually enter the import statement between

the package name and the class declaration as shown below



The screenshot shows an IDE with a file named `Main.java` open. The code in the file is as follows:

```
1 package practise4;  
2 practise4/src/practise4/Main.java;  
3 public class Main {  
4     public static void main(String[] args)  
5     {  
6         Scanner input=new Scanner(System.in);  
7         double g=input.nextDouble();  
8         double lit=3.785;  
9         System.out.println(g+"gallon is equal to"+(g*lit));  
10    }  
11 }  
12
```

Below the code editor, the `Console` tab is active, showing the output of the program:

```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (5 Aug  
10  
10.0gallon is equal to37.85
```

#### 4. Describe three ways you can test the program that converts gallons to liters.

A.1. `import static org.junit.Assert.assertEquals;`

`import org.junit.Test;`

`public class MainTest {`

`@Test`

`public void testPositiveNumber() {`

`assertEquals(3.785, Main.convertGallonsToLiters(1), 0.001);`

```

        assertEquals(37.85, Main.convertGallonsToLiters(10), 0.001);
    }

    @Test
    public void testZero() {
        assertEquals(0.0, Main.convertGallonsToLiters(0), 0.001);
    }

    @Test
    public void testNegativeNumber() {
        assertEquals(-3.785, Main.convertGallonsToLiters(-1), 0.001);
    }

    @Test
    public void testLargeNumber() {
        assertEquals(3785000.0, Main.convertGallonsToLiters(1000000), 0.001);
    }
}

2. public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number of gallons: ");
        double g = input.nextDouble();
        System.out.println(g + " gallon(s) is equal to " + convertGallonsToLiters(g)
+ " liters.");
    }

    public static double convertGallonsToLiters(double gallons) {
        double lit = 3.785;

```

```
        return gallons * lit;
    }
}

3. package practise4;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number of gallons: ");
        double g = input.nextDouble();
        double lit = 3.785;
        System.out.println(g + " gallon(s) is equal to " + (g * lit) + " liters.");
    }
}
```