

Execute, Verify & Submit – Java Programs



| | |
|------------------------|----------------|
| Author(s) | Seetha Lakshmi |
| Authorized by | Head, ETA |
| Creation/Revision Date | Mar 2021 |
| Version | 1.1 |

Usage Guidelines

Do not forward this document to any non-Infosys mail ID. Forwarding this document to a non-Infosys mail ID may lead to disciplinary action against you, including termination of employment.

Contents of this material cannot be used in any other internal or external document without explicit permission from ETA@infosys.com.

COPYRIGHT NOTICE

© 2020 Infosys Limited, Bangalore, India. All Rights Reserved.

Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.

Education, Training and Assessment Department
Infosys Limited
Electronics City
Hosur Road
Bangalore – 561 229, India.

Tel: 91 80 852 0261-270

Fax: 91 80 852 0362

www.infosys.com

<mailto:ETA@infosys.com>

Document Revision History

| Version | Date | Author(s) | Reviewer(s) | Description |
|---------|----------|----------------|-------------------------|---|
| 1.0 | Mar 2021 | Seetha Lakshmi | Malathi_M,Komal_Papdeja | First draft of document |
| 1.1 | Mar 2021 | Seetha Lakshmi | Malathi_M,Komal_Papdeja | Added steps related to verify, submit and last submission |

CONTENTS

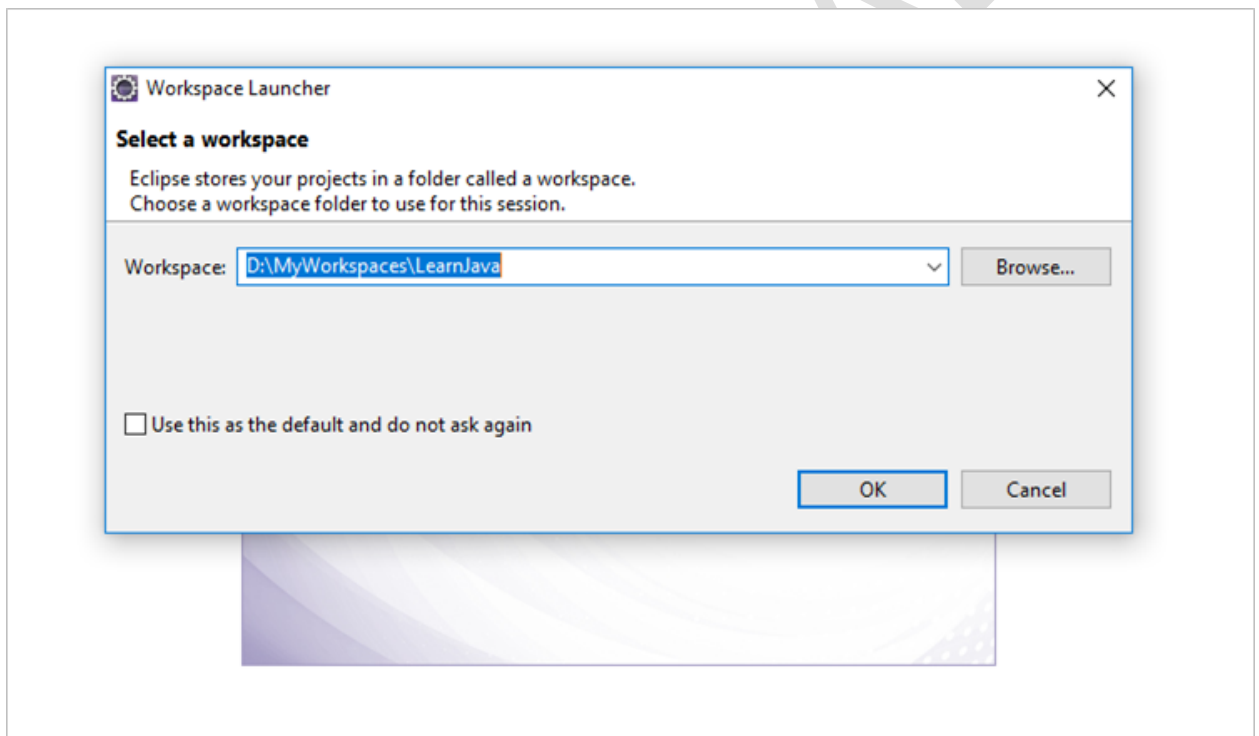
| | |
|--|---|
| COPYRIGHT NOTICE..... | 2 |
| Document Revision History..... | 3 |
| CONTENTS..... | 4 |
| Follow the below Steps to run the Java programs..... | 5 |

Follow the below Steps to run the Java programs

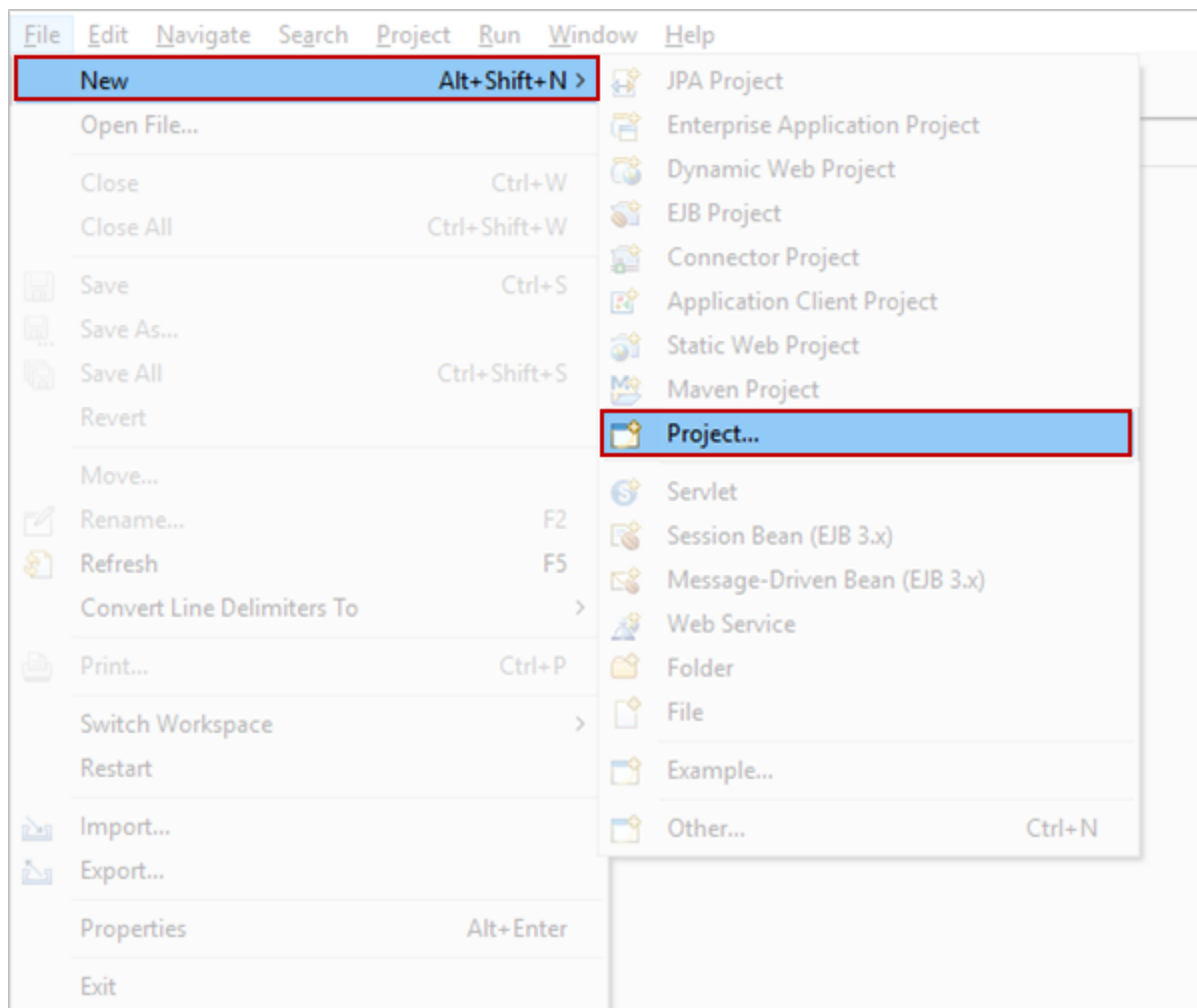
Step 1: For installing Java, click [here](#) (Topics Name - Adopt OpenJDK8)

Step 2: To configure Eclipse IDE, click [here](#) (Topic Name : Eclipse IDE)

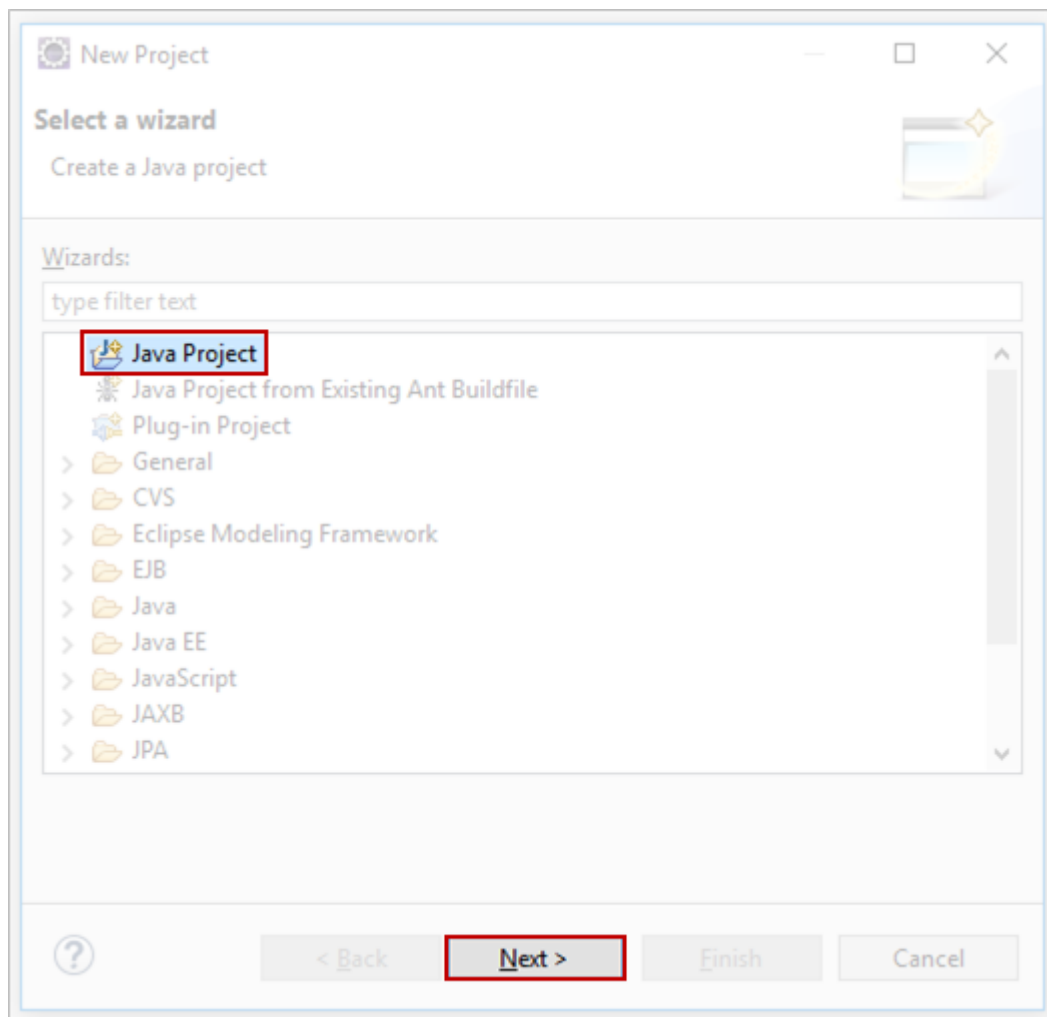
Step 3: Once eclipse is configured, launch Eclipse by clicking on “eclipse.exe” and create a workspace.



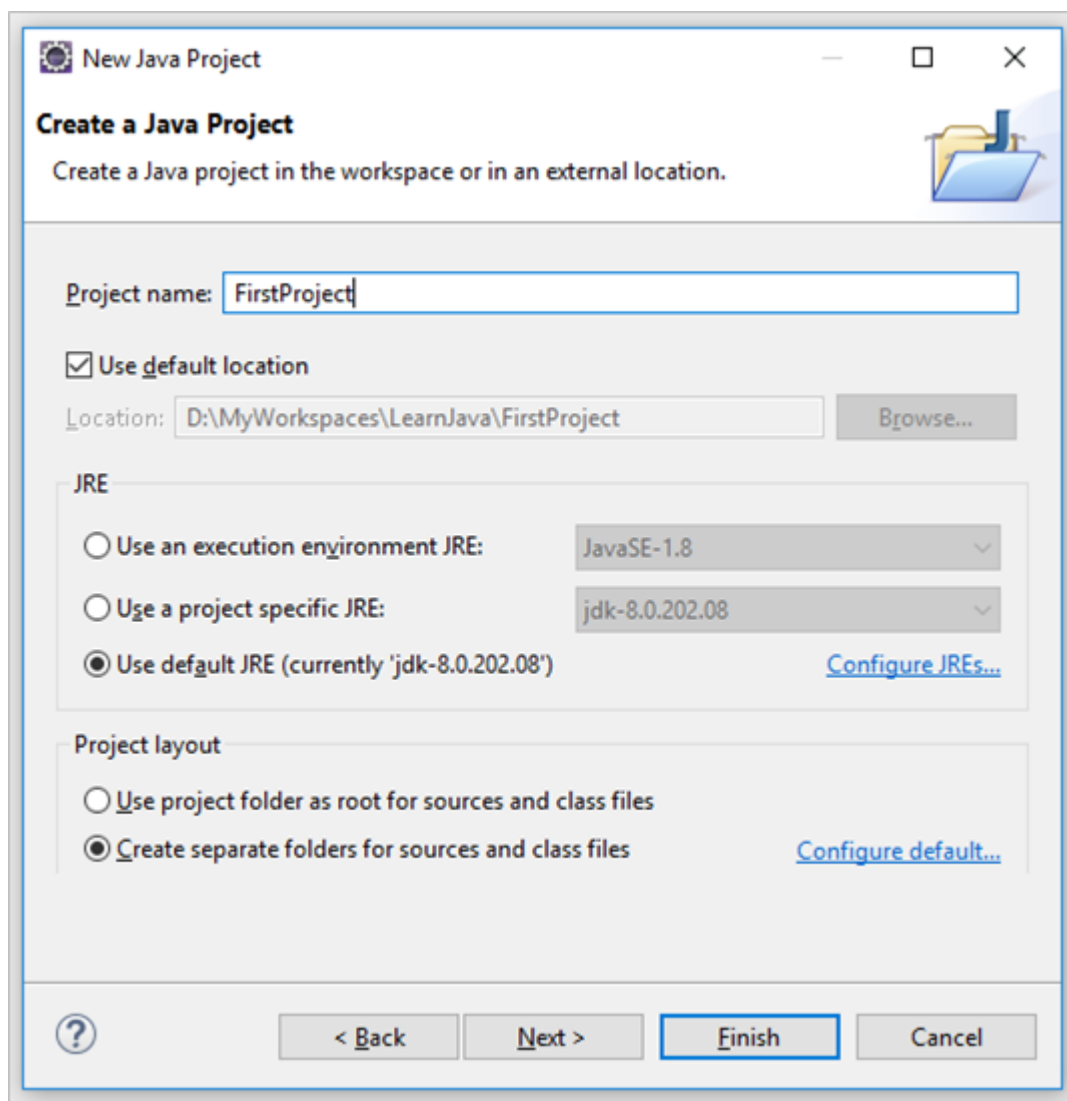
Step 4: Create a new Java project in Eclipse by clicking on menu options – **File → New → Project**



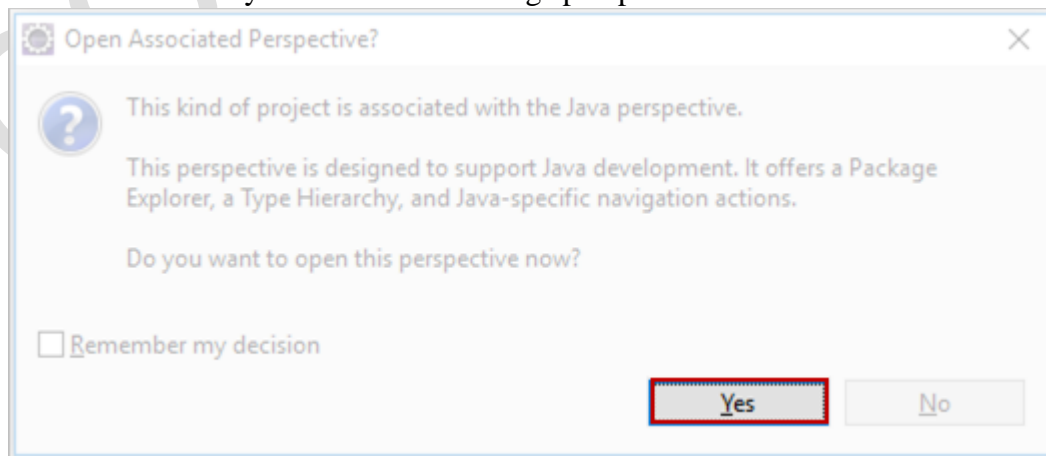
Step 5: Select “Java Project” and click on “Next”



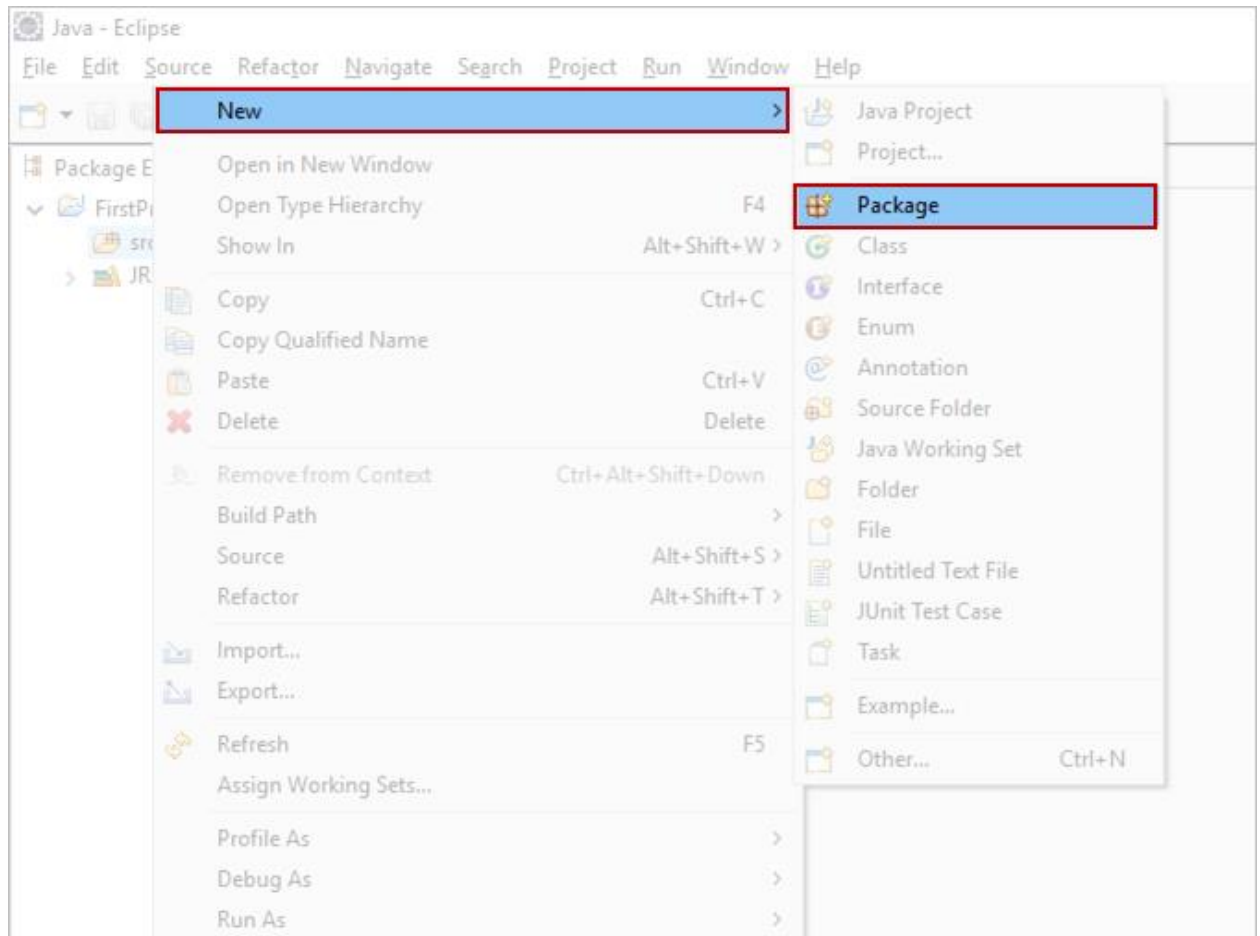
Step 6: Enter the “**Project name**”, use the default options for JRE and click on “**Finish**”.



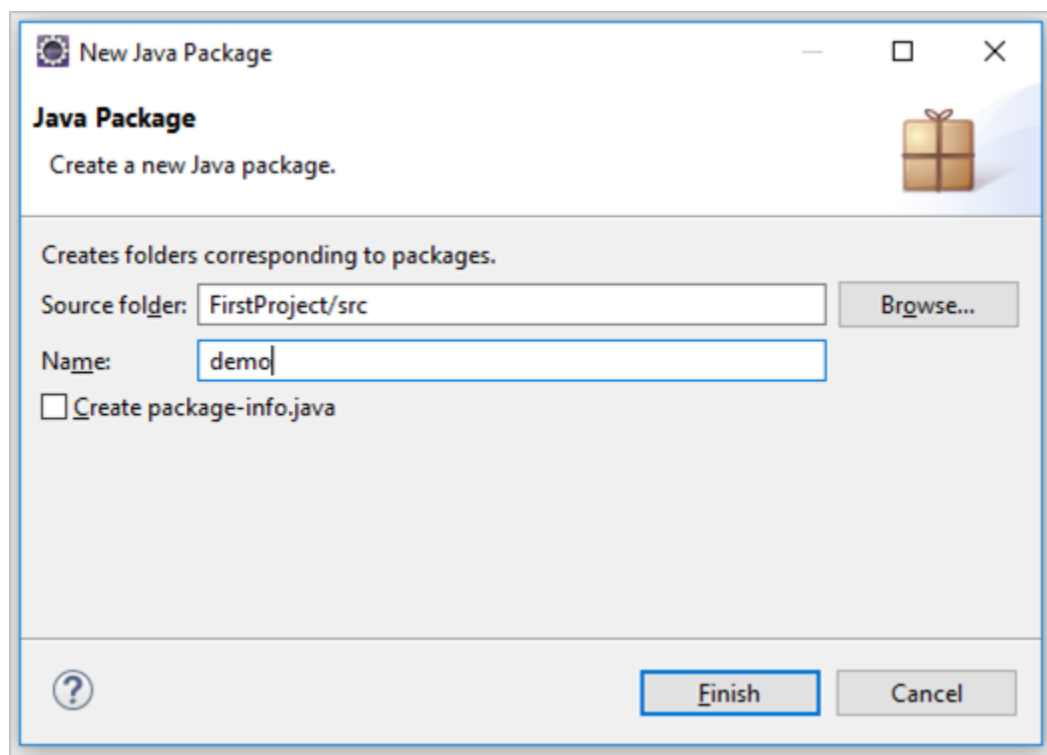
Click on “Yes” if you are asked to change perspective.



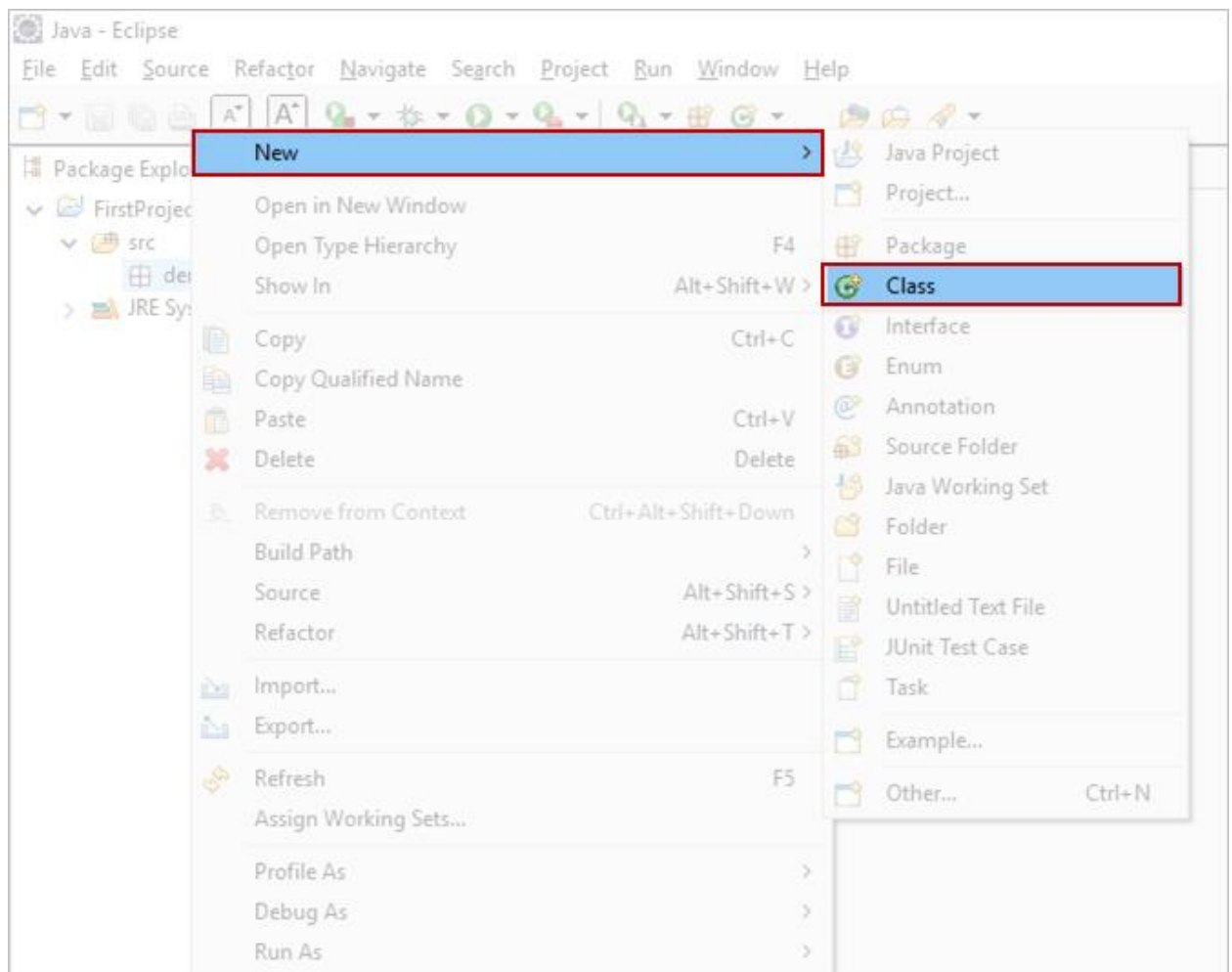
Step 7: Right-click on “src” and select **New -> Package**



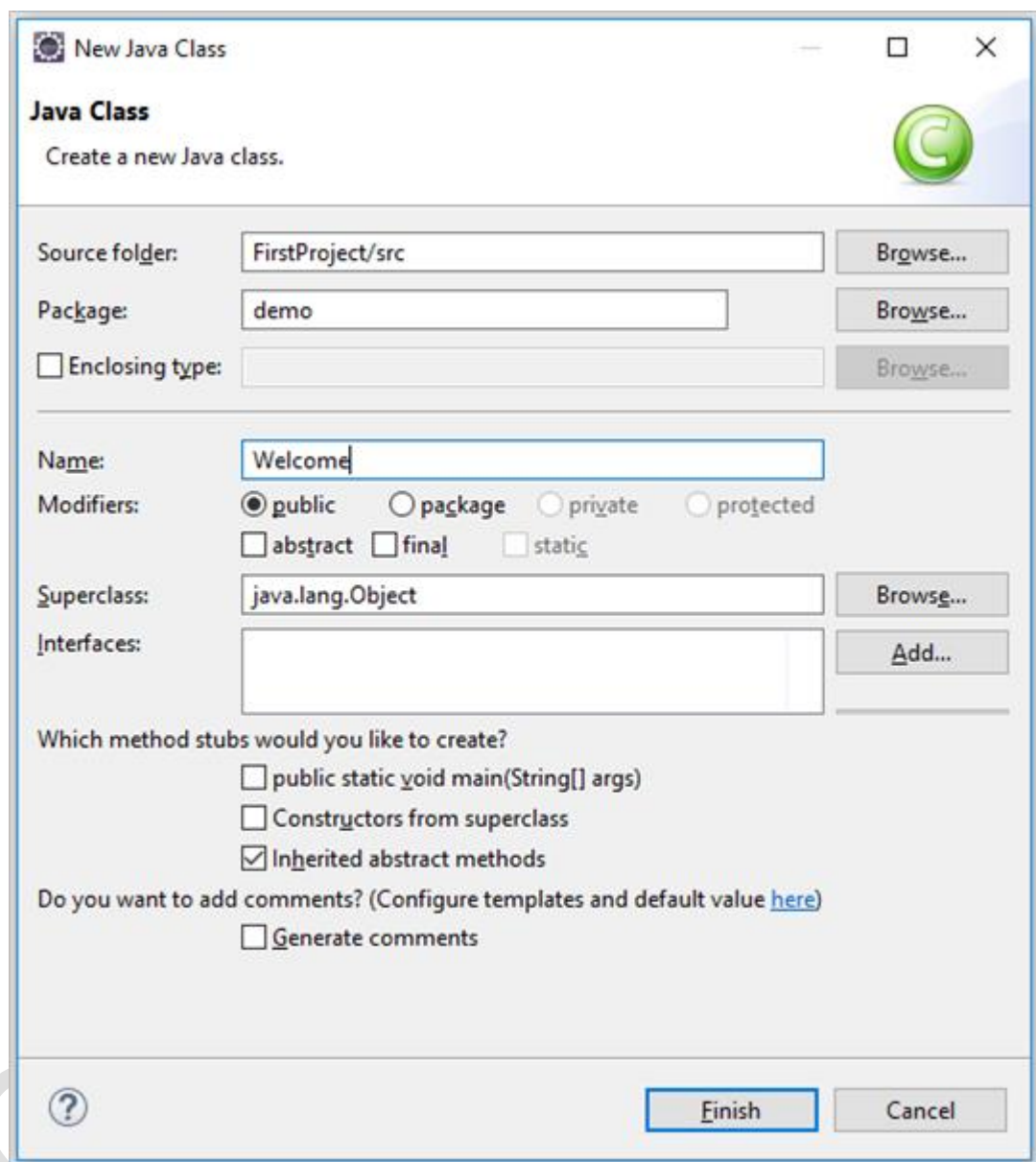
Step 8: Enter the “Name” of the package and click on “Finish”.



Step 9: Right-click on the package("demo"), select **New -> Class**. This will create a new .java file where you can write your program.



Step 10: Enter the name of the class as “Welcome” and click on “Finish”.



The image shows a 'New Java Class' dialog box in an IDE. The title bar says 'New Java Class'. Below the title bar, it says 'Java Class' and 'Create a new Java class.' There is a green circular icon with a 'C' on the right. The dialog has several fields and buttons:

- Source folder:** A text box containing 'FirstProject/src' and a 'Browse...' button.
- Package:** A text box containing 'demo' and a 'Browse...' button.
- Enclosing type:** A checkbox labeled 'Enclosing type:' followed by an empty text box and a 'Browse...' button.
- Name:** A text box containing 'Welcome'.
- Modifiers:** A group of radio buttons: 'public' (selected), 'package', 'private', and 'protected'. Below them are checkboxes for 'abstract', 'final', and 'static'.
- Superclass:** A text box containing 'java.lang.Object' and a 'Browse...' button.
- Interfaces:** An empty text box and an 'Add...' button.
- Which method stubs would you like to create?** A group of checkboxes: 'public static void main(String[] args)' (unchecked), 'Constructors from superclass' (unchecked), and 'Inherited abstract methods' (checked).
- Do you want to add comments? (Configure templates and default value [here](#))** A checkbox labeled 'Generate comments' (unchecked).
- Buttons:** A 'Finish' button and a 'Cancel' button at the bottom right.

Step 11: The .java file will open in the editor.


Step 12: Copy the code from [here](#)

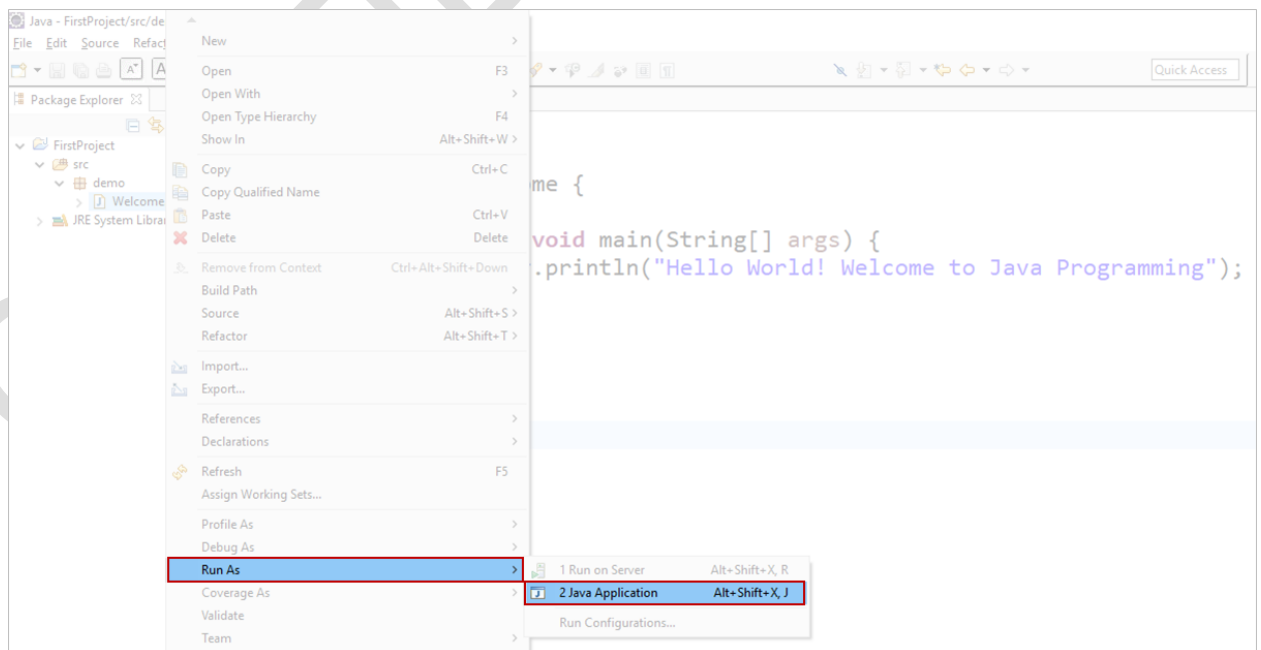
Code in Java

```
1 package demo;
2 public class Welcome
3 {
4     public static void main(String[] args)
5     {
6         System.out.println("Hello World! Welcome to Java Programming");
7     }
8 }
9
10
```

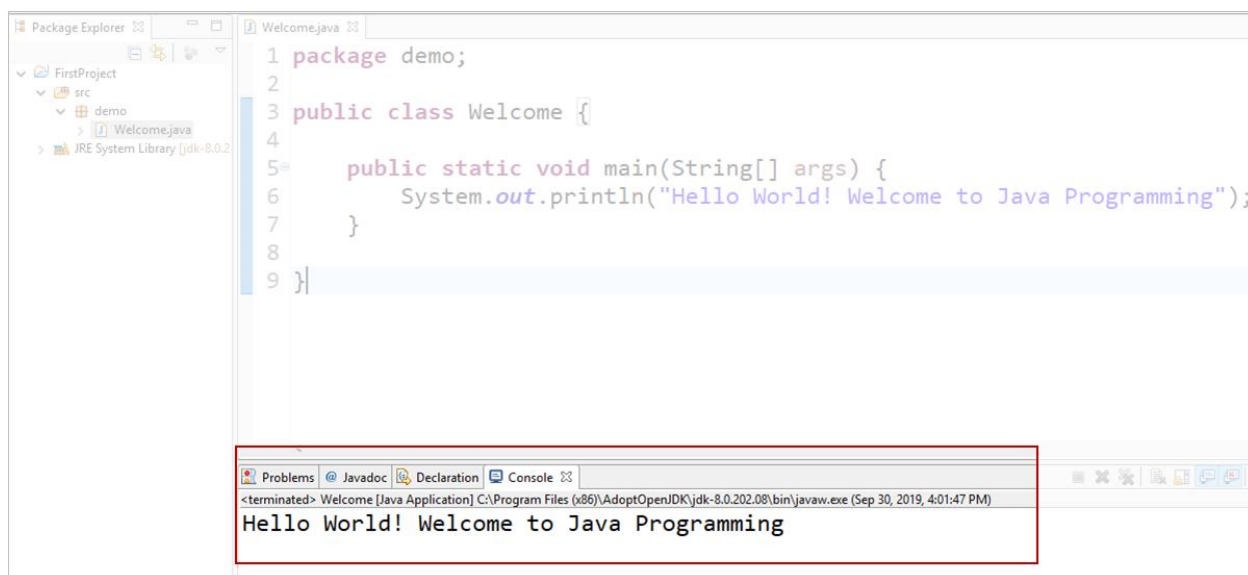
Reset Execute **Copy Code**

NOTE: By default, Eclipse will automatically compile the .java file into a .class file (byte code) when the file is saved.

Step 13: To execute the program, right-click on the .java file, select **Run As -> Java Application** or click  button in the tool bar



The output will be displayed in the Console window as shown below.



Let us now understand the steps to be followed to solve the assignments in Eclipse, execute and verify in Lex

Step 14: Download the Java project artifacts for any of the assignments (Eg: [Encapsulation – Assignment 2](#))

Encapsulation - Assignment 2

Sample Input and Output

Input

| Instance variables | Values |
|--------------------|--------|
| movieId | 112 |
| noOfSeats | 3 |

Output

Total amount for booking : \$24.48

Input

| Instance variables | Values |
|--------------------|--------|
| movieId | 114 |
| noOfSeats | 3 |

Output

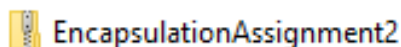
Sorry! Please enter valid movie Id and number of seats

Download the Java project from [here](#) to solve this assignment in Eclipse.

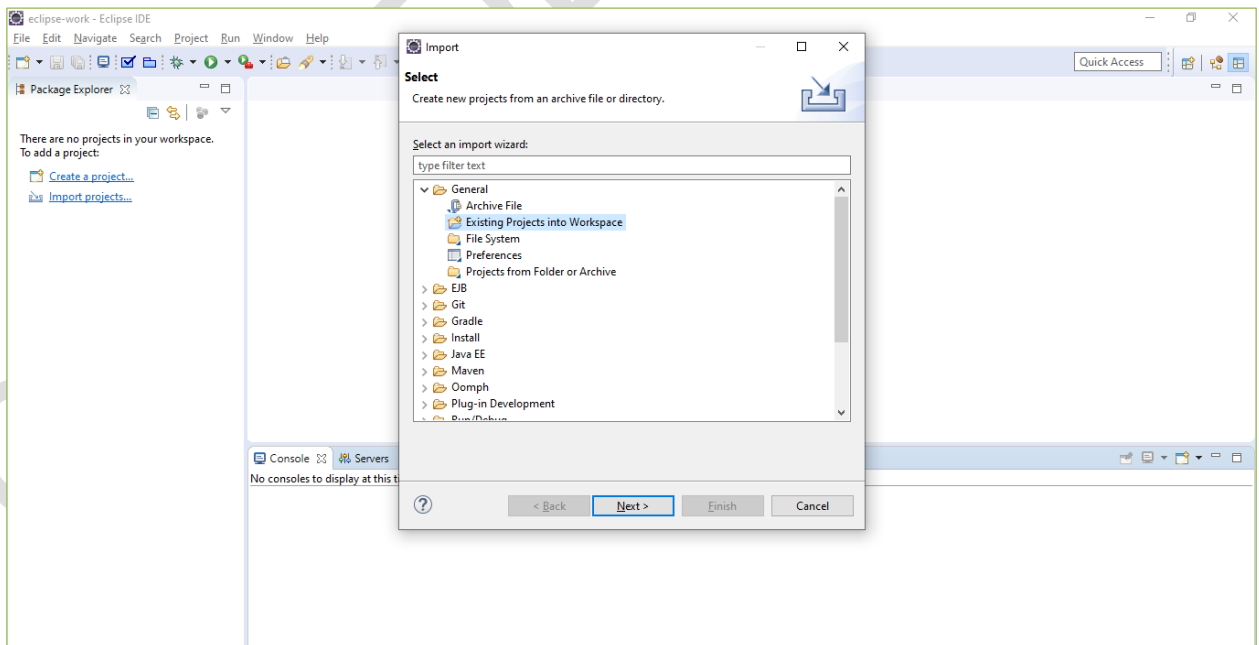
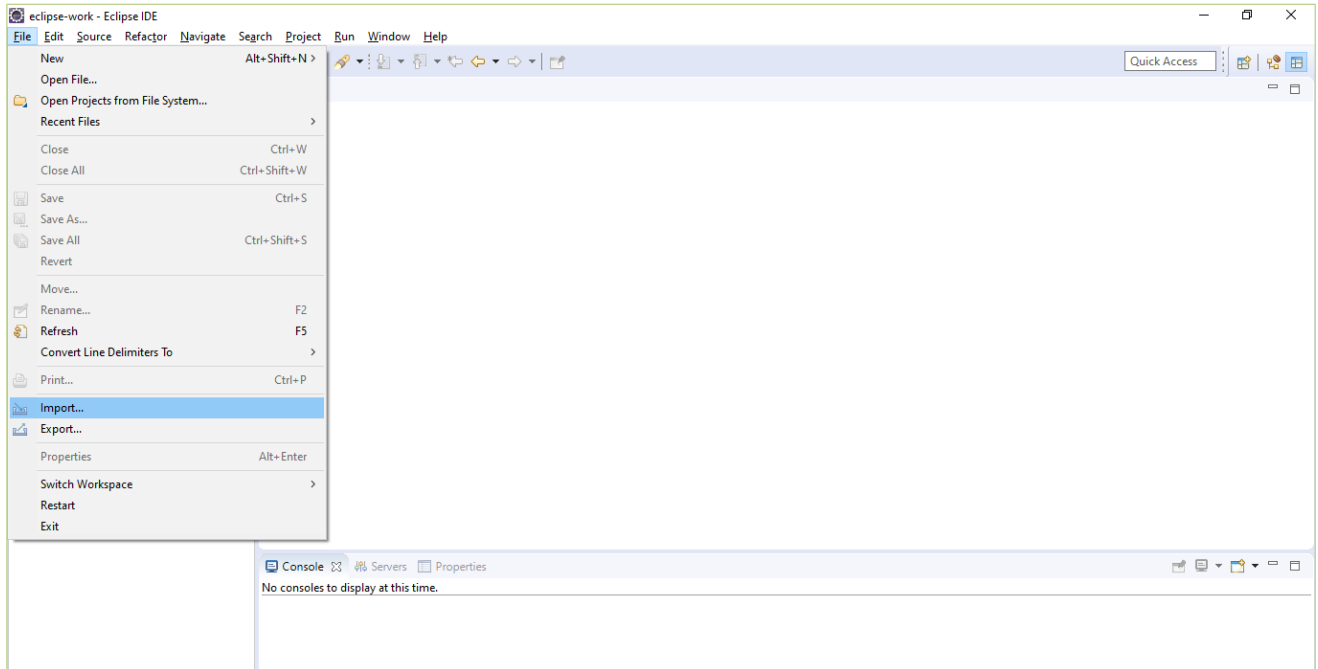
Queue

- Getting Started 2h 1m ✓
- Introduction to Java 3h 31m ✓
- Keywords, Variables, Identifiers an... 43m ✓
- Operators 2h ✓
- Type Conversion 1h 28m ✓
- Control Structures 16h 20m ✓
- Introduction to Object Oriented... 2h 20m ✓
- Methods 2h 10m
- Constructors 1h 25m ✓

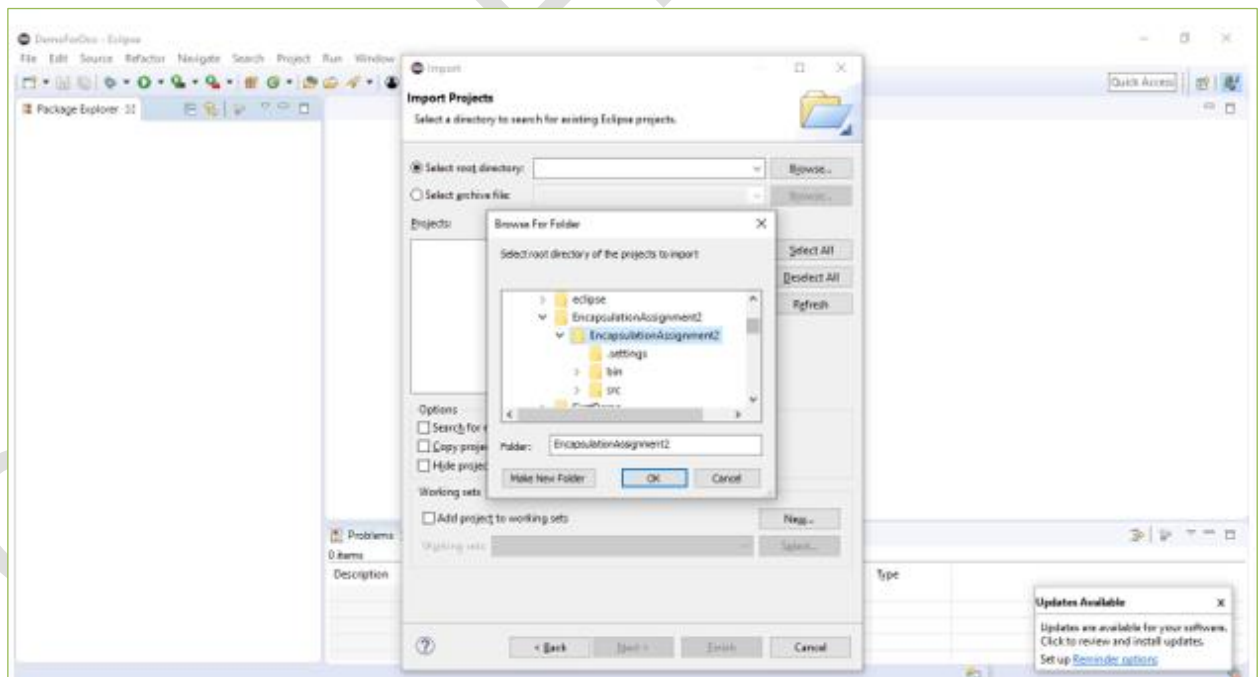
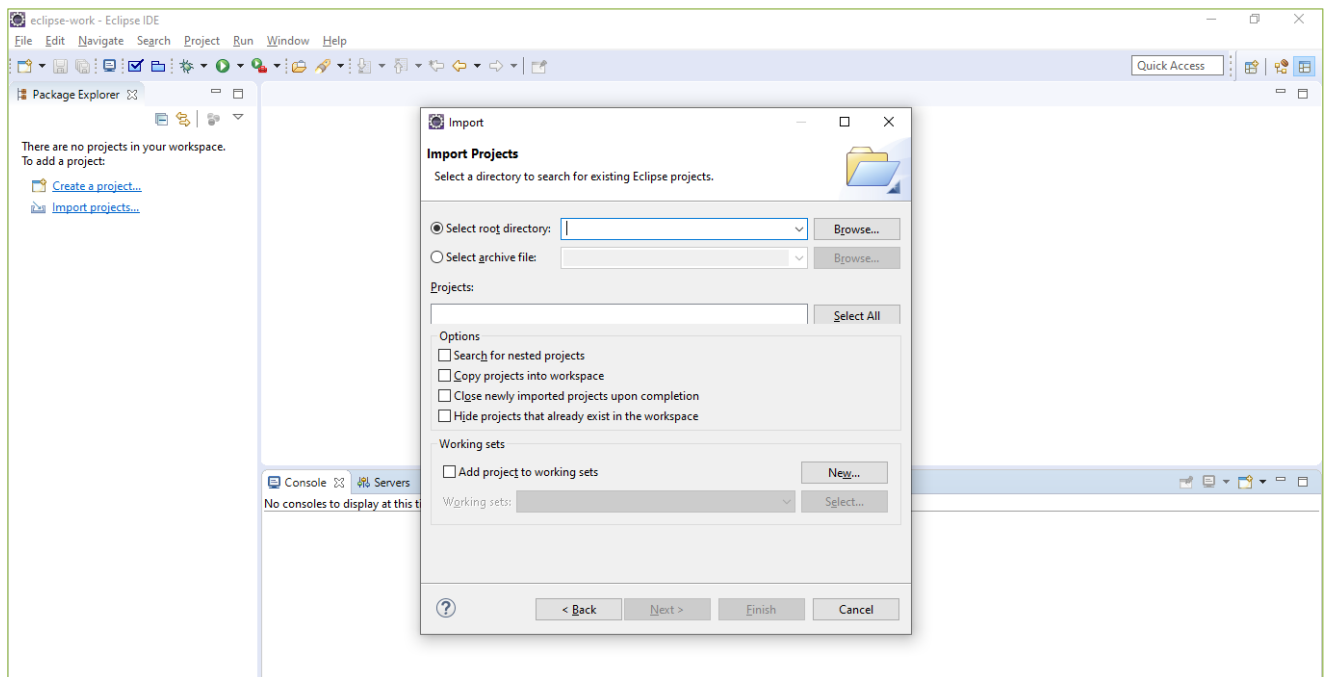
Step 15: Project artifacts as a zipped file will be downloaded under “Downloads” as below. Unzip the file.



Step 16: In eclipse, Go to **File → Import**. Expand “**General**” and select “**Existing Projects into Workspace**”

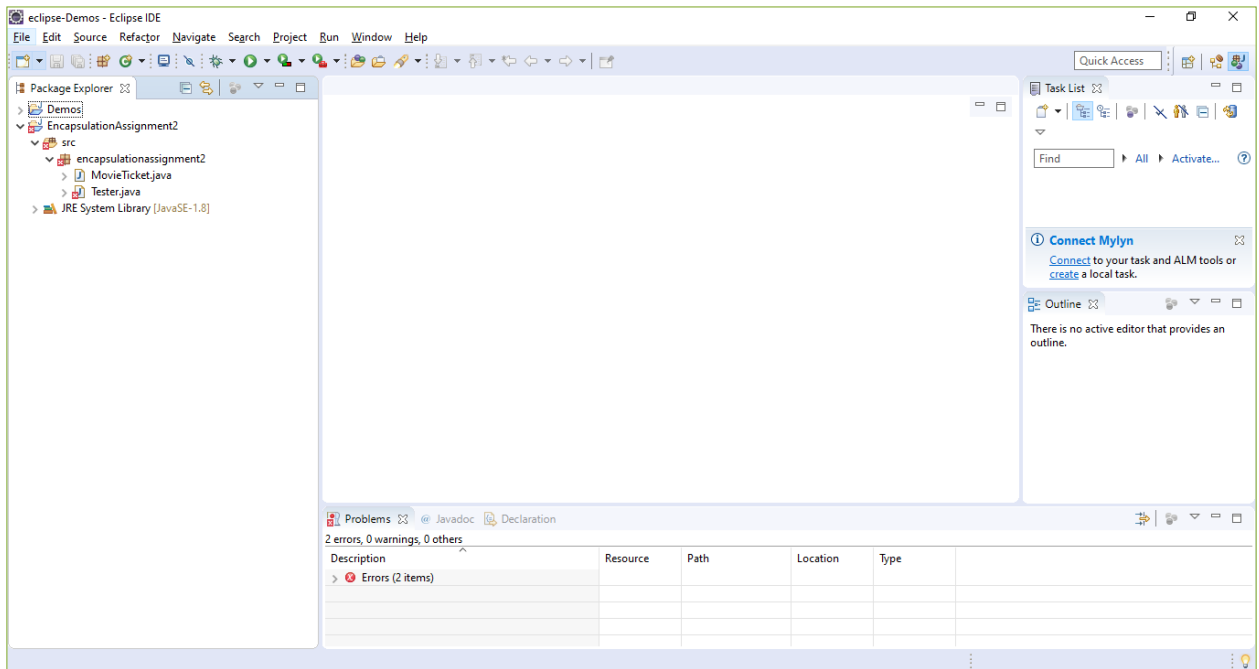


Step 17: Click on “**Browse**” and select the unzipped folder containing “**.settings**” folder **ONLY**

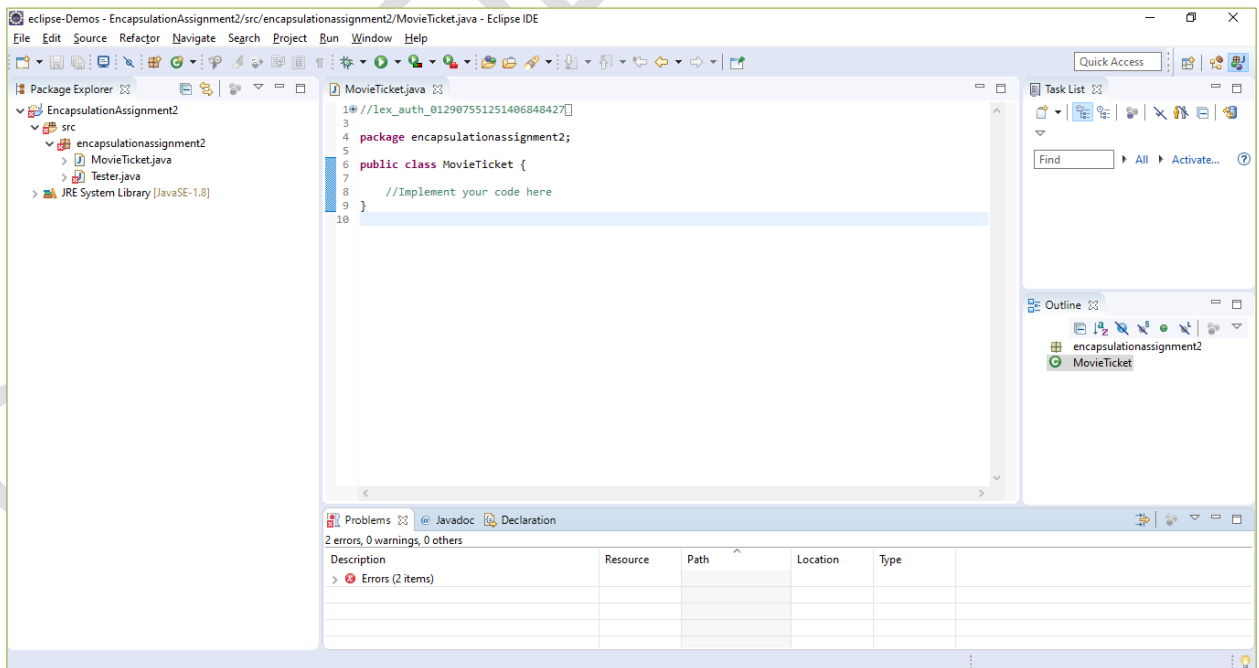


Step 18: Click on “OK” and “Finish”

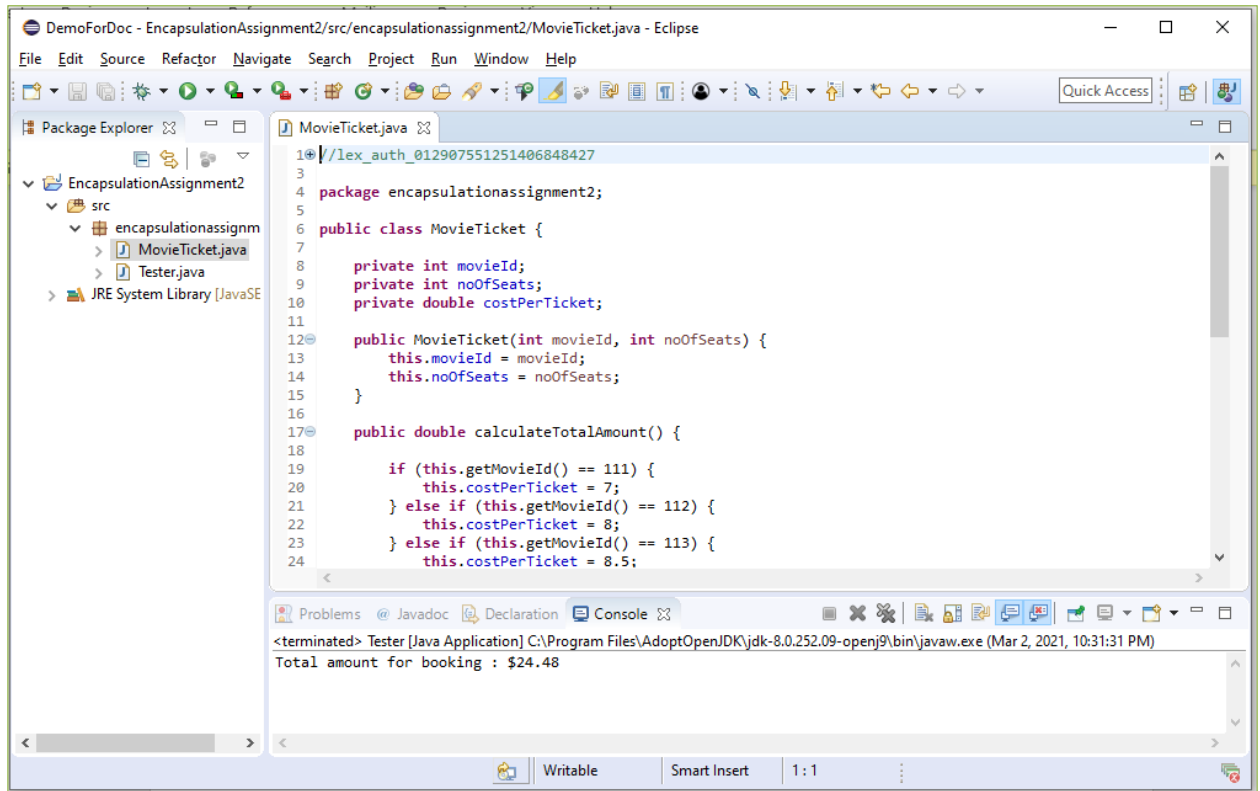
Step 19: Expand the imported project
(Tester.java has errors as the code is not completely implemented)



Step 20: Double click on “**MovieTicket.java**”. The file opens towards the right.



Step 21: Implement the code for “**MovieTicket.java**” and save the file. You would now observe that there are no errors in “**Tester.java**”
Run **Tester.java** and observe the output in the “**Console**”



To Execute, Verify and Submit the implemented code on Lex, follow the below steps

Step 22: Go to [Encapsulation – Assignment 2](#) page in Lex. In the code pane, replace the line “//Implement your code here” with the code written in eclipse.

NOTE: In case you are copying the entire class definition, ensure that only “Tester class is public”

Code in Java

```
1 class MovieTicket {
2     private int movieId;
3     private int noOfSeats;
4     private double costPerTicket;
5
6     public MovieTicket(int movieId, int noOfSeats) {
7         this.movieId = movieId;
8         this.noOfSeats = noOfSeats;
9     }
10
11     public double calculateTotalAmount() {
12
13         if (this.getMovieId() == 111) {
14             this.costPerTicket = 7;
15         } else if (this.getMovieId() == 112) {
16             this.costPerTicket = 8;
17         } else if (this.getMovieId() == 113) {
18             this.costPerTicket = 8.5;
19         } else {
20             this.costPerTicket = 0;
21         }
22     }
23 }
```

Reset Execute Copy Code Verify Submit Last Submission

Step 23: Click on “Execute” and observe the results

Encapsulation - Assignment 2 - \ x +

lex.infosysapps.com/en/viewer/hands-on/lex_auth_012907551251406848427?collectionId=lex_auth_012880464547618816347&collectionType=Course

Encapsulation - Assignment 2

```
1 class MovieTicket {
2     //Implement your code here
3     private int movieId;
4     private int noOfSeats;
5     private double costPerTicket;
6
7     public MovieTicket(int movieId, int noOfSeats) {
8         this.movieId = movieId;
9         this.noOfSeats = noOfSeats;
10    }
11
12
13    public double calculateTotalAmount() {
14
15        if (true) {
16            this.costPerTicket = 7;
17        } else if (this.getMovieId() == 112) {
18            this.costPerTicket = 8;
19        } else if (this.getMovieId() == 113) {
20            this.costPerTicket = 8.5;
21        } else {
22            this.costPerTicket = 0;
23        }
24    }
25 }
```

Reset Execute Copy Code Verify Submit Last Submission

Execution Result

Output:

Total amount for booking : \$21.42

Contents Details

Queue

- Getting Started 2h 1m ✓
- Introduction to Java 3h 31m ✓
- Keywords, Variables, Identifiers an... 43m ✓
- Operators 2h ✓
- Type Conversion 1h 28m ✓
- Control Structures 16h 20m ✓
- Introduction to Object Oriented... 2h 20m ✓
- Methods 2h 10m

Step 24: Click on “Verify” and observe the test cases. Few of the test cases are failing because the code implemented is partially correct

Encapsulation - Assignment 2

Verification Result

Test Cases Passed

17 / 23 Total 13 / 13 Structural 4 / 5 Sample 0 / 5 Actual

| Sl. No. | Type | S/A | Target | Input | Expected Output | Actual Output | Result |
|---------|------|-----|---|-------|-----------------|---------------|--------|
| 1 | | N/A | Class MovieTicket | N/A | N/A | N/A | ✓ |
| 2 | | N/A | Class MovieTicket Method: public MovieTicket(int moviend, int noOfSeats) | N/A | N/A | N/A | ✓ |
| 3 | | N/A | Class MovieTicket Method: private int moviend Target: Class Property | N/A | N/A | N/A | ✓ |
| 4 | | N/A | Class MovieTicket Method: private double costPerTicket Target: Class Property | N/A | N/A | N/A | ✓ |
| | | | Class MovieTicket | | | | |

Contents Details

Programming using Ja...
Course | Beginner
113h 2m

Queue

- Getting Started 2h 1m ✓
- Introduction to Java 3h 31m ✓
- Keywords, Variables, Identifiers an... 43m ✓
- Operators 2h ✓
- Type Conversion 1h 28m ✓
- Control Structures 16h 20m ✓
- Introduction to Object Oriented... 2h 20m ✓
- Methods 2h 10m
- Constructors 1h 25m ✓

Encapsulation - Assignment 2

| | | | | | | | |
|----|------------|--|--|------------|--------------------|--------------------|---|
| 17 | | | Class MovieTicket Method: calculateTotalAmount | [[111, 7]] | 49.980000000000004 | 49.980000000000004 | ✓ |
| 18 | | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [112] | 8.0 | 7.0 | ✗ |
| 19 | Procedural | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [113] | N/A | 7.0 | ✗ |
| 20 | | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [114] | N/A | 7.0 | ✗ |
| 21 | Actual | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [115] | N/A | 7.0 | ✗ |
| 22 | | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [110] | N/A | 7.0 | ✗ |
| 23 | | | Class MovieTicket Method: calculateTotalAmount | [[112, 2]] | N/A | 14.280000000000001 | ✗ |

Code Analysis

No Quality Violations have been found in the code

Contents Details

Programming using Ja...
Course | Beginner
113h 2m

Queue

- Getting Started 2h 1m ✓
- Introduction to Java 3h 31m ✓
- Keywords, Variables, Identifiers an... 43m ✓
- Operators 2h ✓
- Type Conversion 1h 28m ✓
- Control Structures 16h 20m ✓
- Introduction to Object Oriented... 2h 20m ✓
- Methods 2h 10m
- Constructors 1h 25m ✓

Step 25: Verify the code again with the correct solution. Observe the result. All the test cases are now passing

Encapsulation - Assignment 2

Verification Result

Test Cases Passed

23 / 23 Total 13 / 13 Structural 5 / 5 Sample 5 / 5 Actual

| Sl. No. | Type | S/A | Target | Input | Expected Output | Actual Output | Result |
|---------|------|-----|---|-------|-----------------|---------------|--------|
| 1 | | N/A | Class MovieTicket | N/A | N/A | N/A | ✓ |
| 2 | | N/A | Class MovieTicket Method: public MovieTicket(int movioid, int noOfSeats) | N/A | N/A | N/A | ✓ |
| 3 | | N/A | Class MovieTicket Method: private int movioid Target: Class Property | N/A | N/A | N/A | ✓ |
| 4 | | N/A | Class MovieTicket Method: private double costPerTicket Target: Class Property | N/A | N/A | N/A | ✓ |
| | | | Class MovieTicket | | | | ✓ |

Contents Details

Programming using Ja...
Course | Beginner
113h 2m

Queue

- Getting Started 2h 1m ✓
- Introduction to Java 3h 31m ✓
- Keywords, Variables, Identifiers an... 43m ✓
- Operators 2h ✓
- Type Conversion 1h 28m ✓
- Control Structures 16h 20m ✓
- Introduction to Object Oriented... 2h 20m ✓
- Methods 2h 10m ...
- Constructors 1h 25m ✓

Encapsulation - Assignment 2

| | | | | | | | |
|----|------------|--------|--|------------|-----|--------------------|---|
| 15 | | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [112] | 8.0 | 8.0 | ✓ |
| 16 | | Sample | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [113] | 8.5 | 8.5 | ✓ |
| 17 | | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [114] | 0.0 | 0.0 | ✓ |
| 18 | | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [115] | 0.0 | 0.0 | ✓ |
| 19 | Procedural | | Class MovieTicket Method: calculateTotalAmountCostPerTicket | [110] | N/A | 0.0 | ✓ |
| 20 | | | Class MovieTicket Method: calculateTotalAmount | [[111, 1]] | N/A | 7.140000000000001 | ✓ |
| 21 | | Actual | Class MovieTicket Method: calculateTotalAmount | [[111, 4]] | N/A | 28.560000000000002 | ✓ |
| 22 | | | Class MovieTicket Method: calculateTotalAmount | [[111, 7]] | N/A | 49.980000000000004 | ✓ |
| 23 | | | Class MovieTicket Method: calculateTotalAmount | [[112, 2]] | N/A | 16.32 | ✓ |

Code Analysis

No Quality Violations have been found in the code

Contents Details

Programming using Ja...
Course | Beginner
113h 2m

Queue

- Getting Started 2h 1m ✓
- Introduction to Java 3h 31m ✓
- Keywords, Variables, Identifiers an... 43m ✓
- Operators 2h ✓
- Type Conversion 1h 28m ✓
- Control Structures 16h 20m ✓
- Introduction to Object Oriented... 2h 20m ✓
- Methods 2h 10m ...
- Constructors 1h 25m ✓

Step 26: Once the verification is successful (All test cases passing) the code can be submitted on the server using “Submit”

The screenshot displays a web-based coding environment for "Encapsulation - Assignment 2". The main area shows a Java code editor with the following code:

```
1 class MovieTicket {
2
3     //Implement your code here
4     private int movieId;
5     private int noOfSeats;
6     private double costPerTicket;
7
8     public MovieTicket(int movieId, int noOfSeats) {
9         this.movieId = movieId;
10        this.noOfSeats = noOfSeats;
11    }
12
13    public double calculateTotalAmount() {
14
15        if (this.getMovieId() == 111) {
16            this.costPerTicket = 7;
17        } else if (this.getMovieId() == 112) {
18            this.costPerTicket = 8;
19        } else if (this.getMovieId() == 113) {
20            this.costPerTicket = 8.5;
21        } else {
22            this.costPerTicket = 0;
23        }
24    }
25 }
```

Below the code editor are buttons for "Reset", "Execute", "Copy Code", "Verify", "Submit", and "Last Submission". A "Submission Result" box at the bottom states: "Your solution is submitted!!".

On the right side, there is a "Contents" tab and a "Queue" section. The "Queue" section lists various topics with their durations and completion status:

- Getting Started (2h 1m) ✓
- Introduction to Java (3h 31m) ✓
- Keywords, Variables, Identifiers an... (43m) ✓
- Operators (2h) ✓
- Type Conversion (1h 28m) ✓
- Control Structures (16h 20m) ✓
- Introduction to Object Oriented... (2h 20m) ✓
- Methods (2h 10m) ⋮
- Constructors (1h 25m) ✓

Step 27: At any given point, the last submitted code can be retrieved by clicking on “**Last Submission**” option