Lesson 06 Demo 06

Implementing Method References in Java

Objective: To implement method references in Java, including creating functional interfaces and executing code with example data.

Tools Required: Eclipse IDE

Prerequisites: None

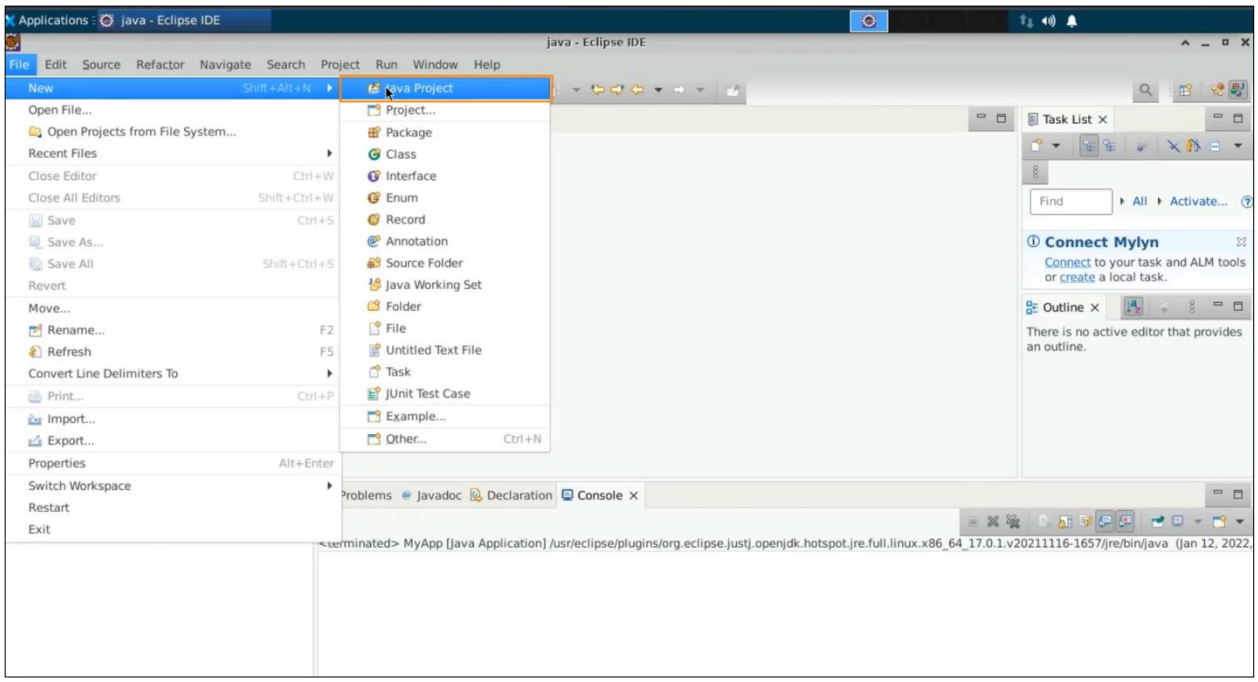
Steps to be followed:

1. Create a Java project
2. Create a functional interface and a class for a static void registered user
3. Create Lambda expressions
4. Create a reference to the interface and execute the code
5. Execute the log in reference
6. Create methods that can do a return, and execute the code with example data
7. Write the reference variable notification and execute the code

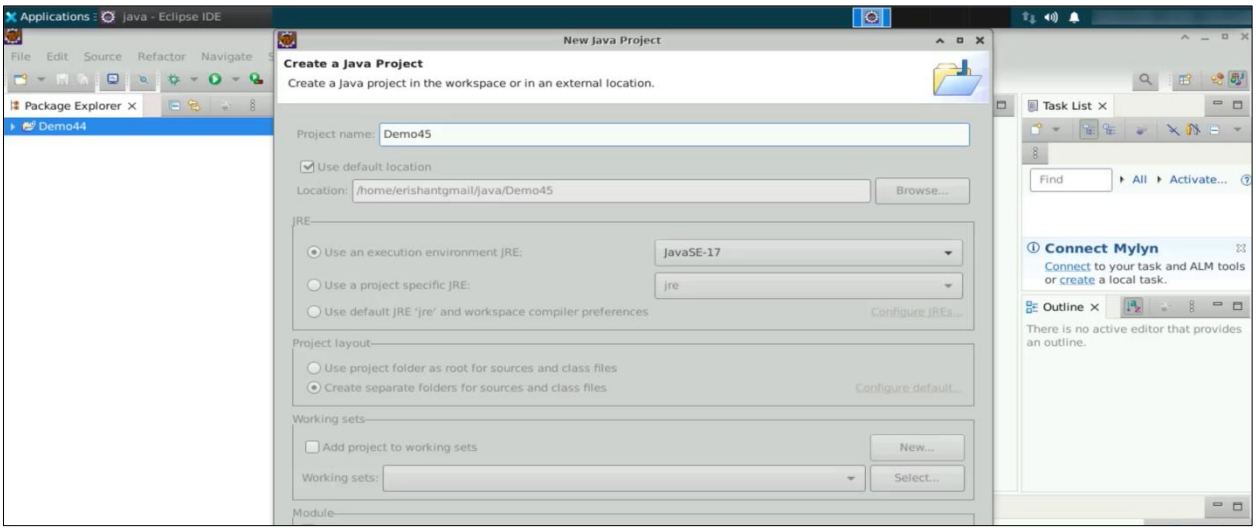
Step 1: Create a Java project

1.1 Open the **Eclipse IDE**

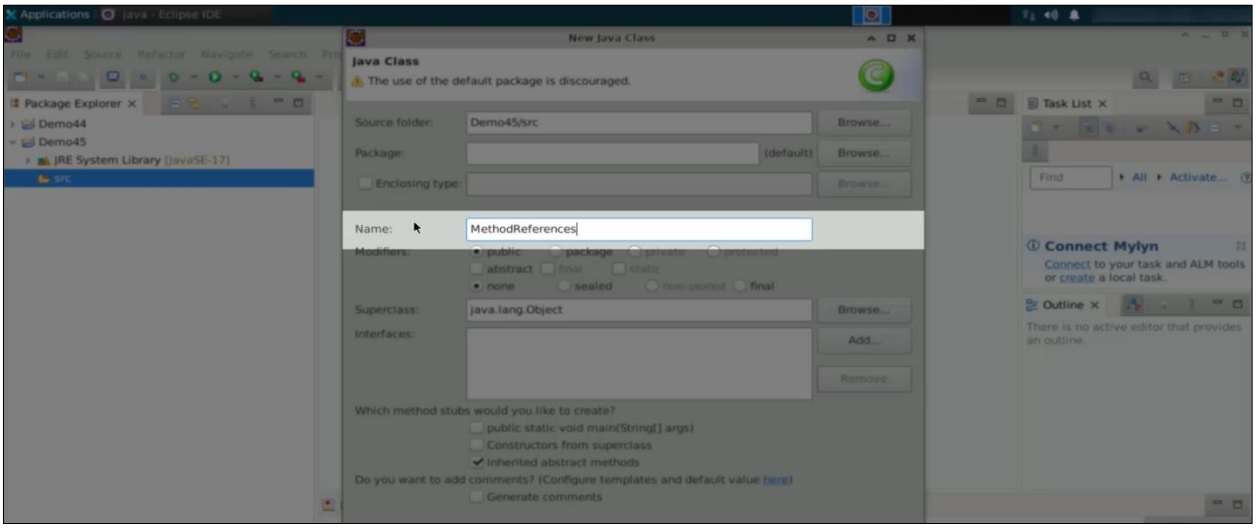
1.2 Select **File**, then **New,** and click **Java project**



1.3 Enter the name of the project as "**Demo45**", uncheck "**Create a module-info.java file**", and click Finish

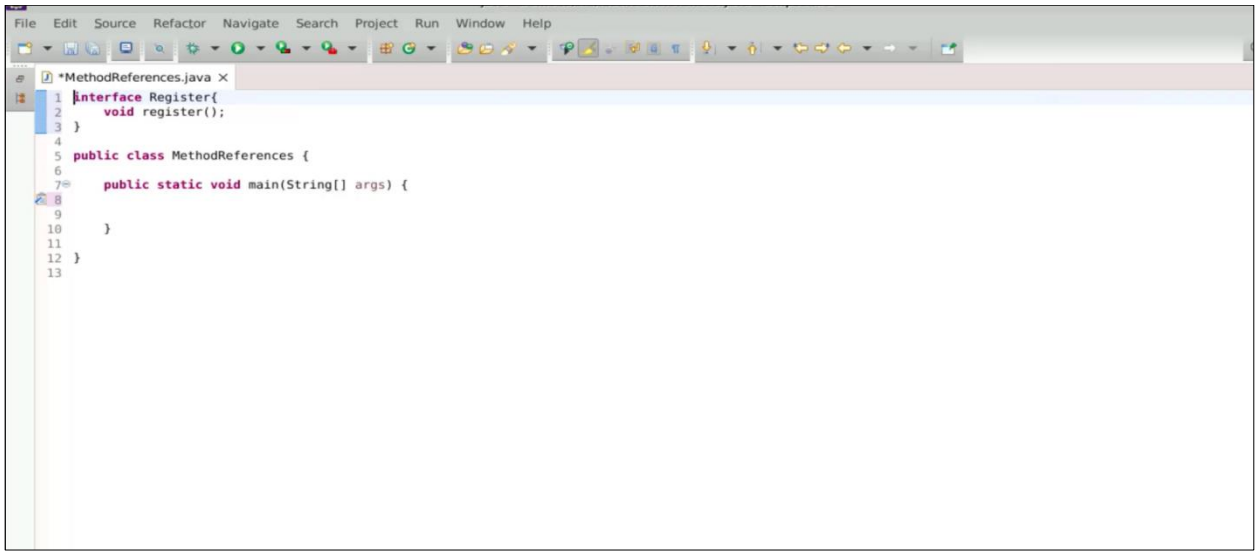


1.4 With **Demo45** selected in the **src** folder, right-click and create a new class. Name this class **MethodReferences**, then select the main method, and click Finish

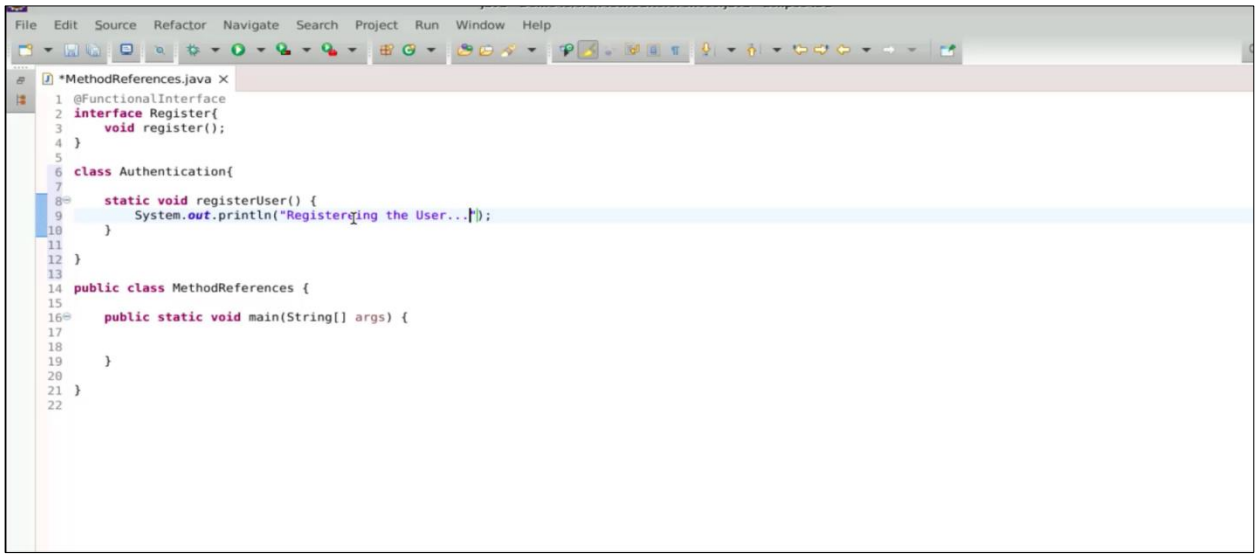


Step 2: Create a functional interface and a class for a static void registered user

2.1 Create a functional interface called **RegisteredInterface**. Annotate it with the **@FunctionalInterface** annotation. Add a method with the name register

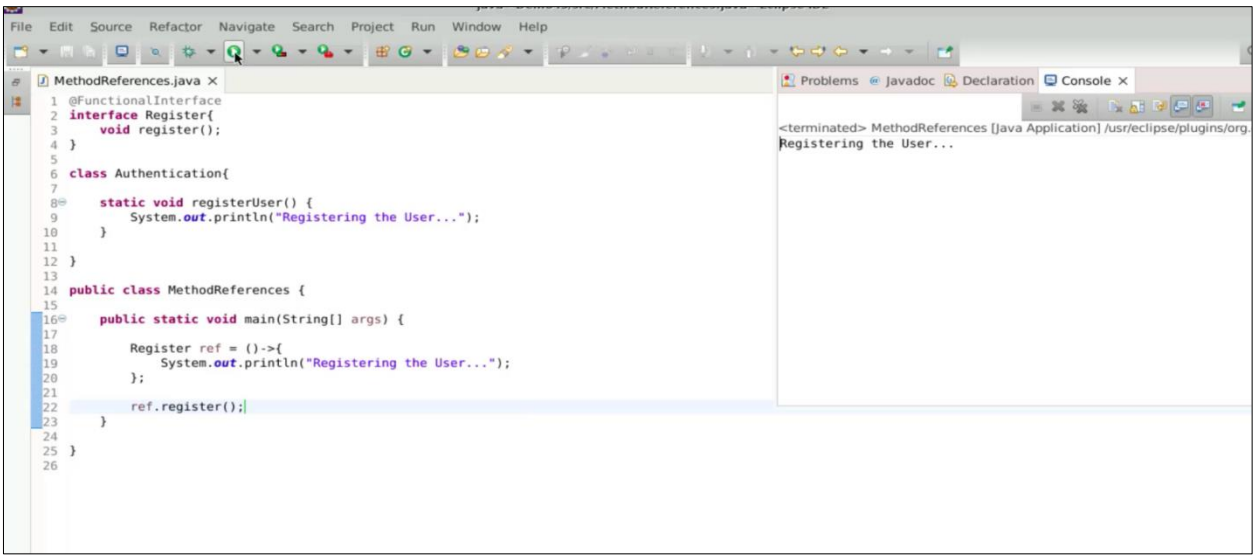


2.2 Create a class and name it Authentication Class. Implement a static void method and name it **registeredUser**, which prints Registering the user



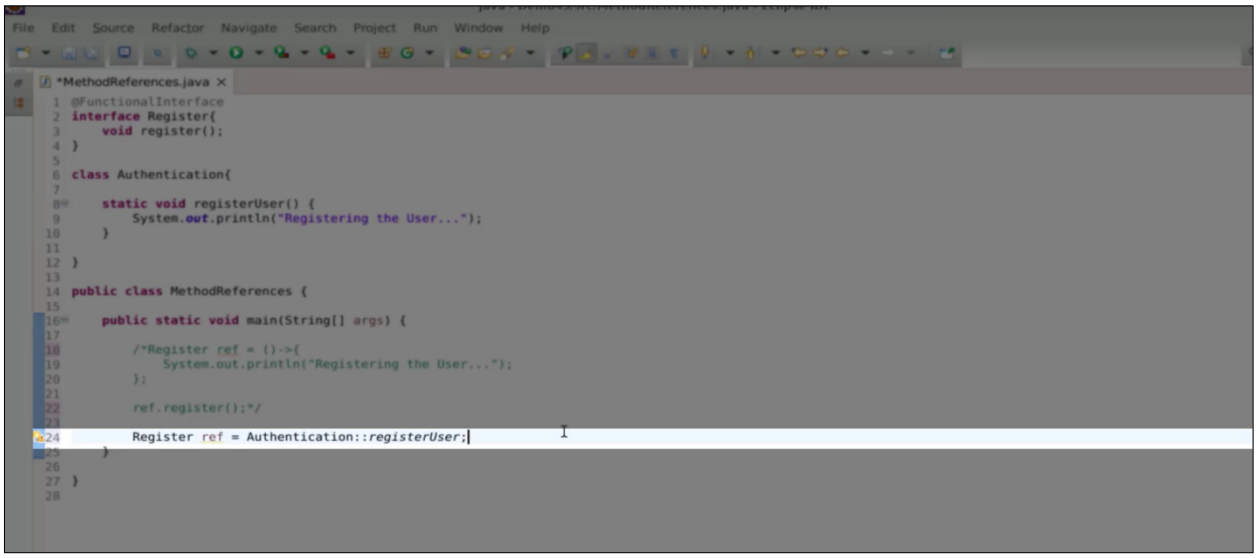
Step 3: Create Lambda expressions

3.1 In the main method, for the **register** method, you will create lambda expressions. This means that you can have the method definition **registerUser**, and then execute this method with the name **register** on the object. This is how lambda expressions are used, as you have done earlier

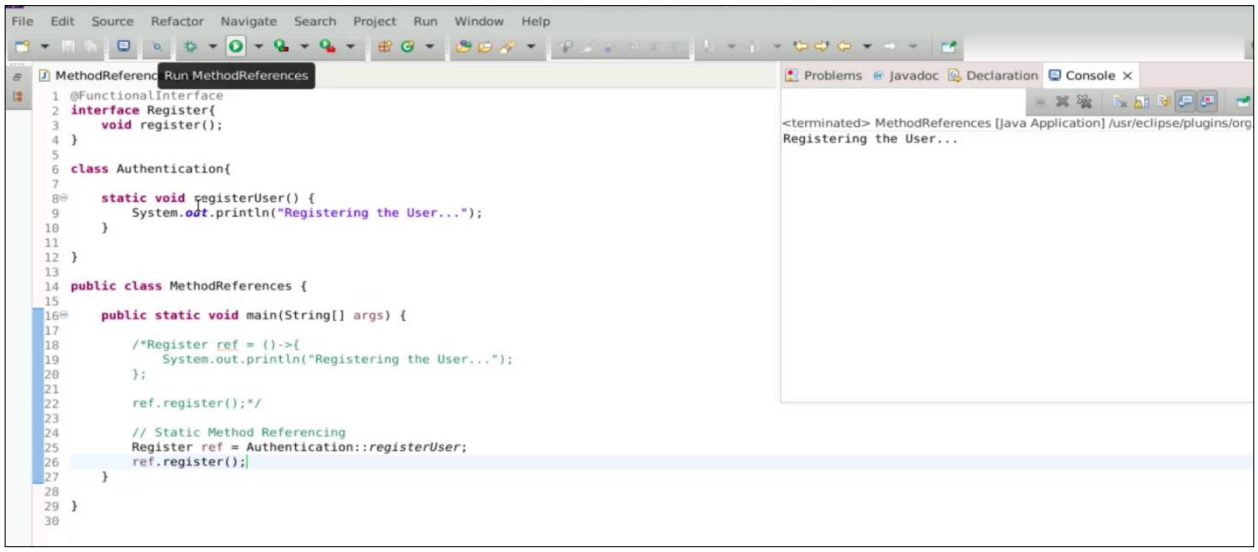


Step 4: Create a reference to the interface and execute the code

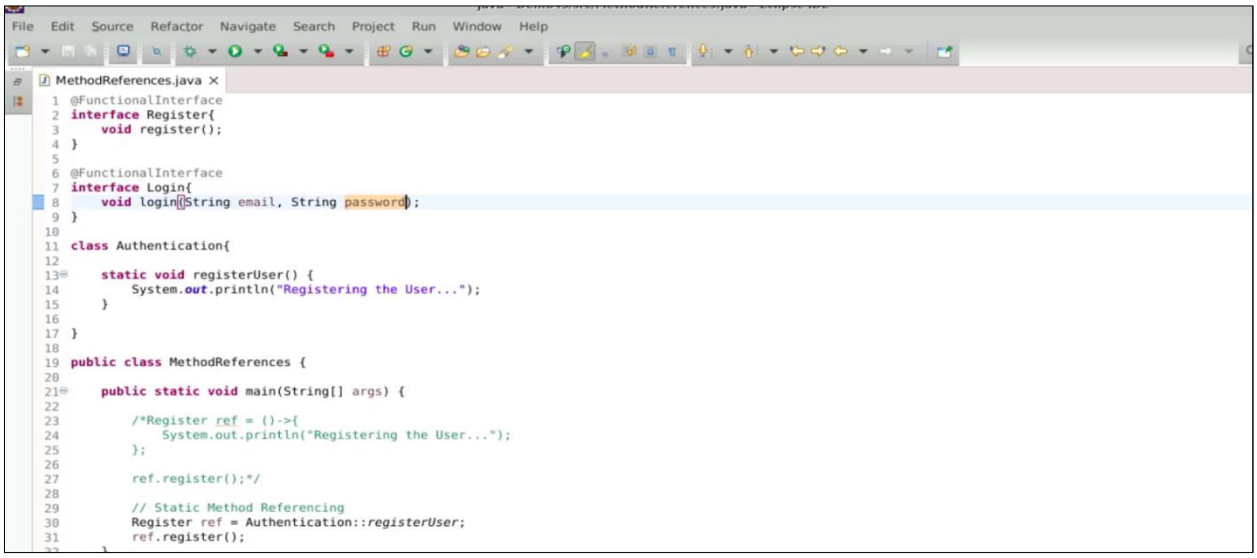
4.1 Next, replace the lambda expression with a static method reference. Create the reference to the interface and then with the class **AuthenticationClass** using the method reference operator (**::**). Assign the reference of the **registerUser** method. This is static method referencing.



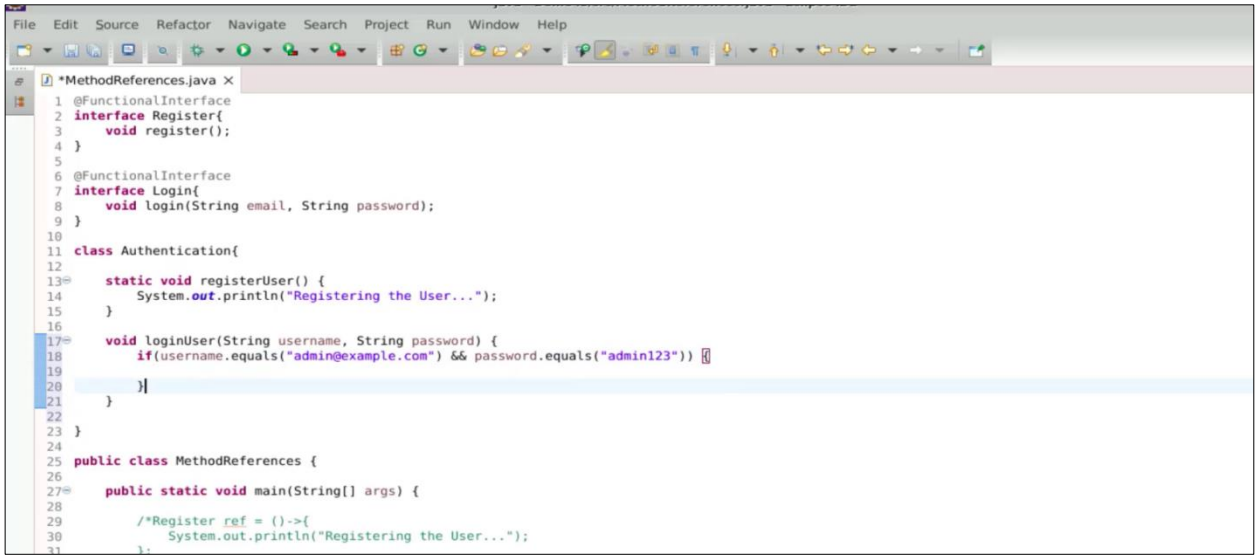
4.2 Execute the **register** method



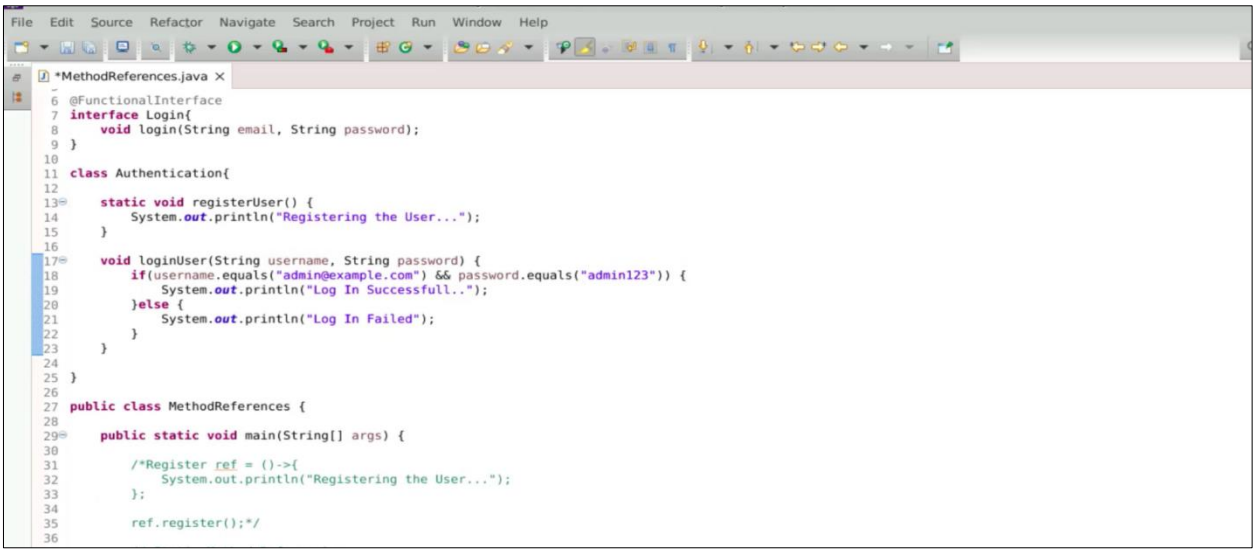
4.3 If you want an interface with input parameters, create an interface and name it **Login** with a **logIn** method that takes two string parameters: email and password



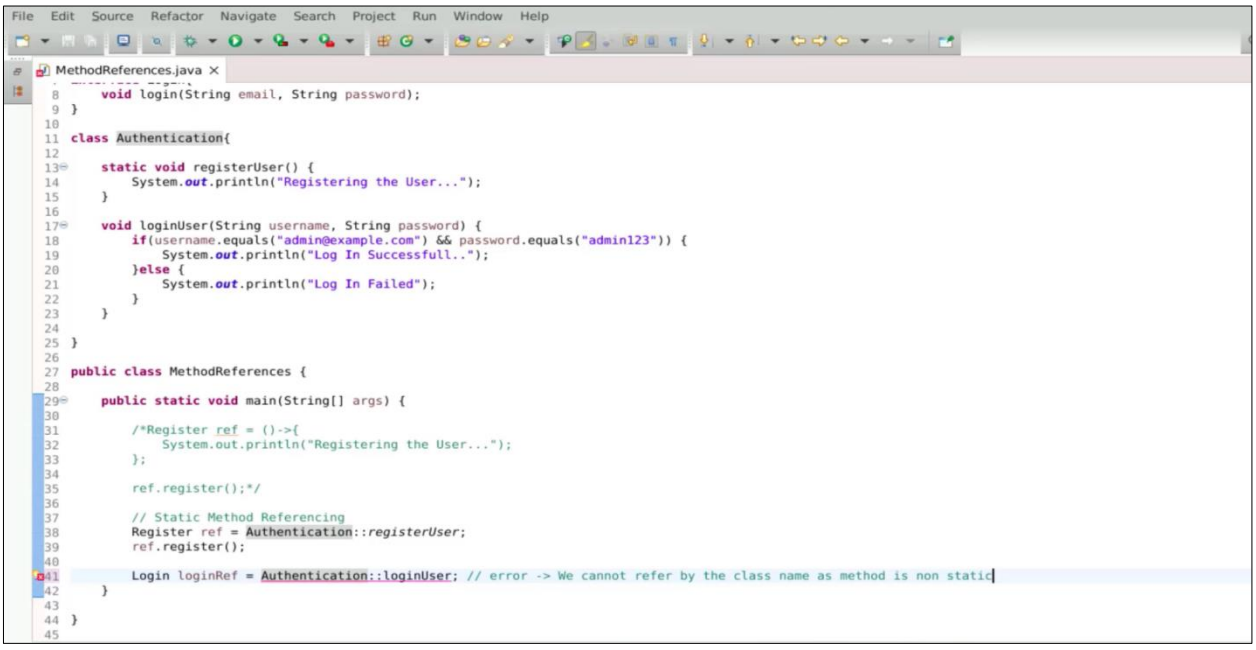
4.4 In the **AuthenticationClass,** create a non-static method called **logInUser** that takes a username and password as inputs



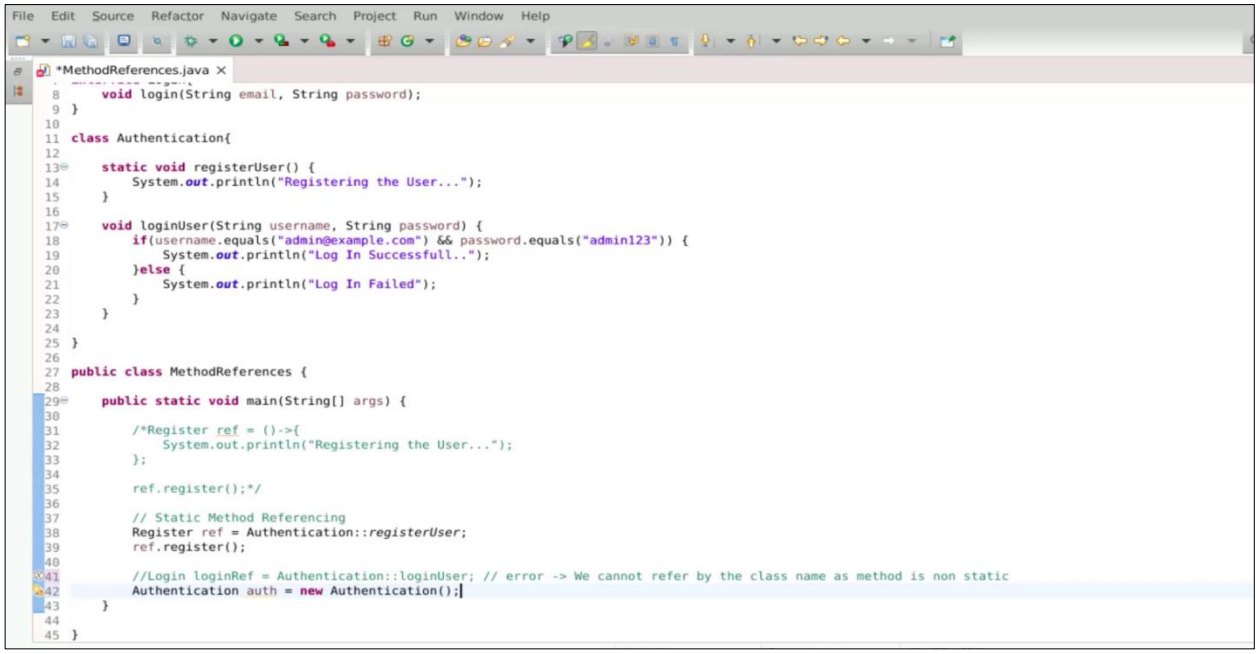
4.5 In the else block, you can print **login is successful** or **login failed**. This is the method definition you created for the **logInUser** method in the **AuthenticationClass**



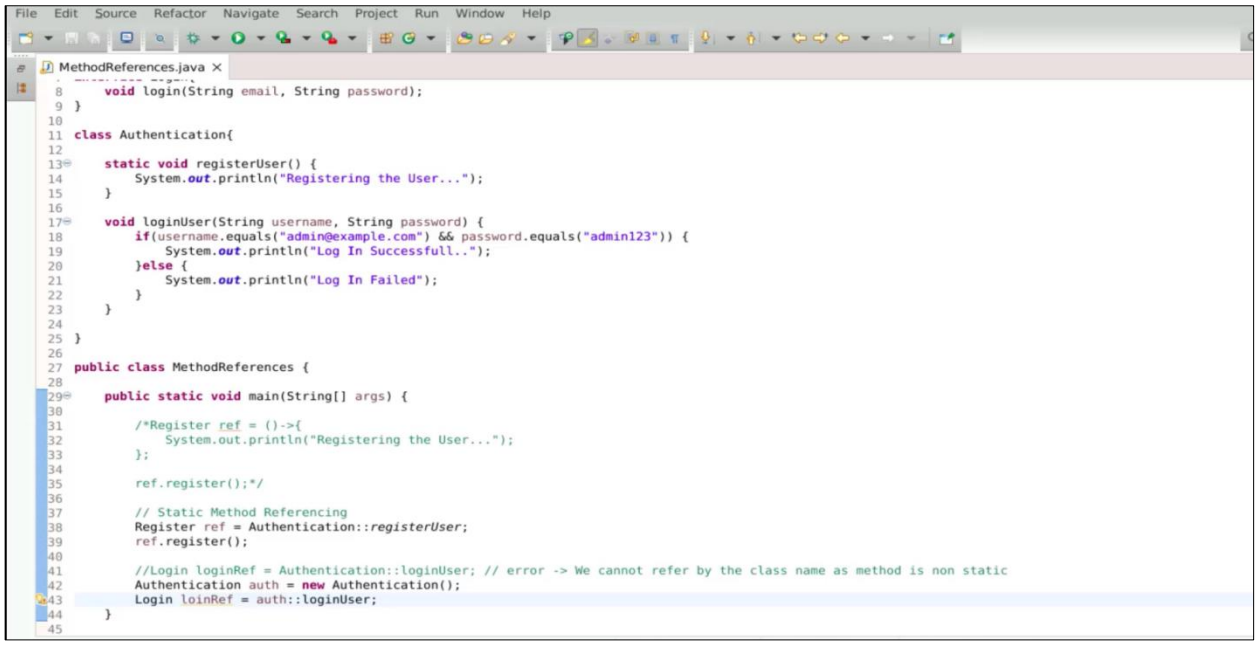
4.6 Create a reference variable **loginRef** to refer to the **loginUser** method of the **Authentication** class. You need to use the object reference to access non-static methods



4.7 Create an object of the **Authentication** class, as shown:
Authentication auth = new Authentication()

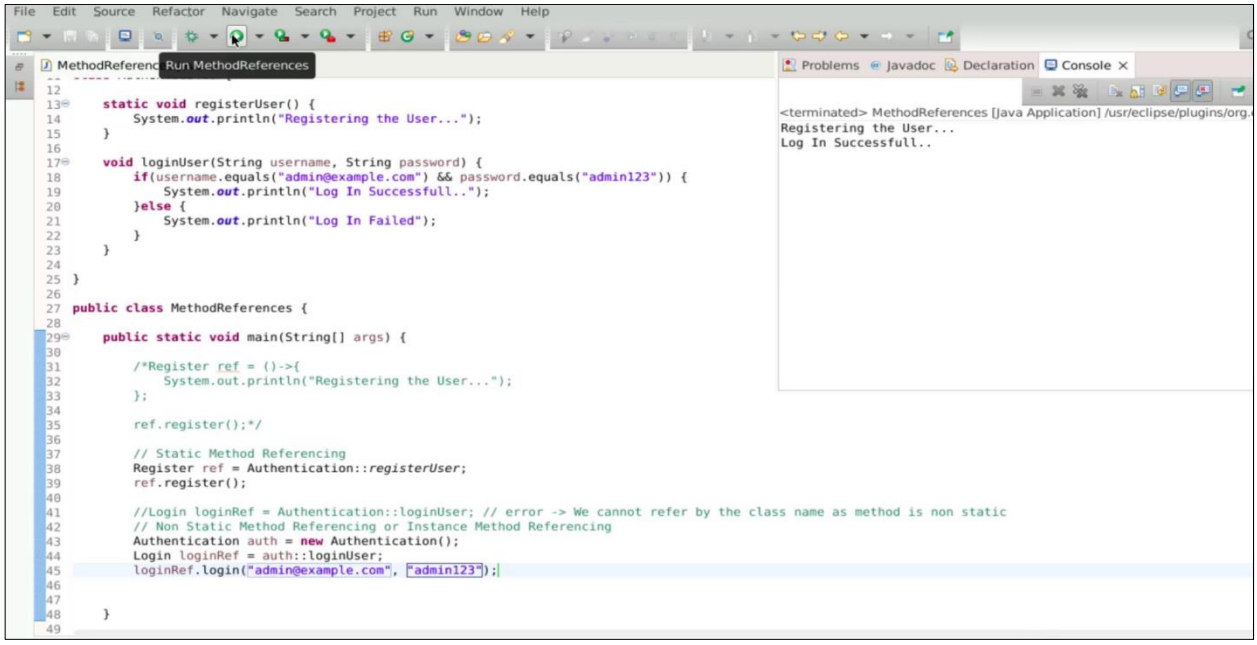


4.8 Now that you have created the object of the **AuthenticationClass**, you can proceed with method referencing



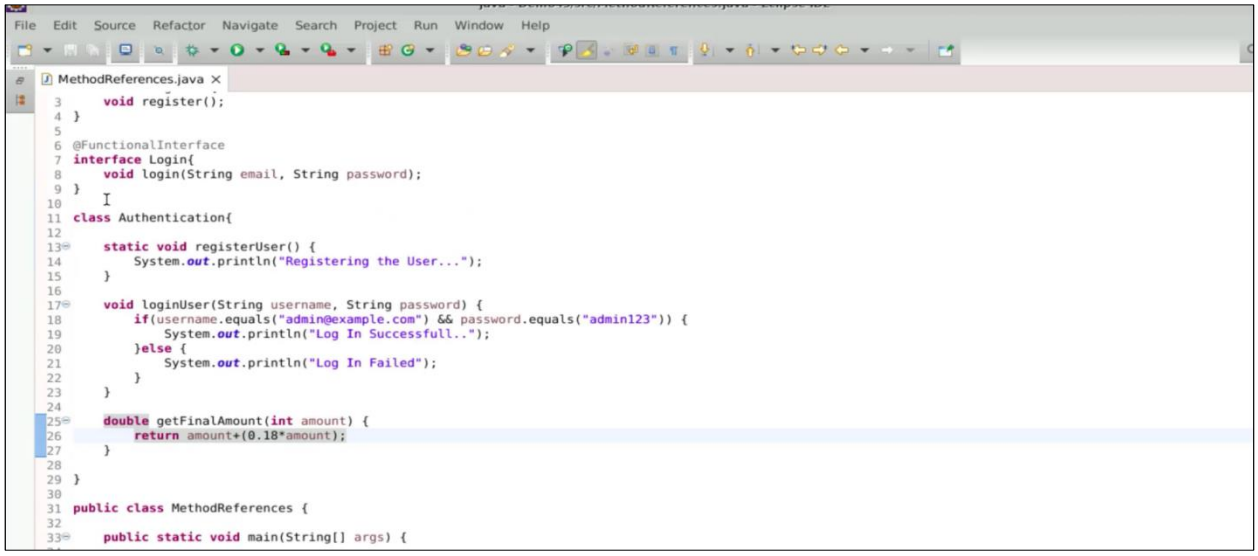
Step 5: Execute the log in reference

5.1 Execute **loginRef**. On the login reference, execute the method **login**. Pass the email **admin@example.com** and the password **admin123**. You will observe that the login is successful

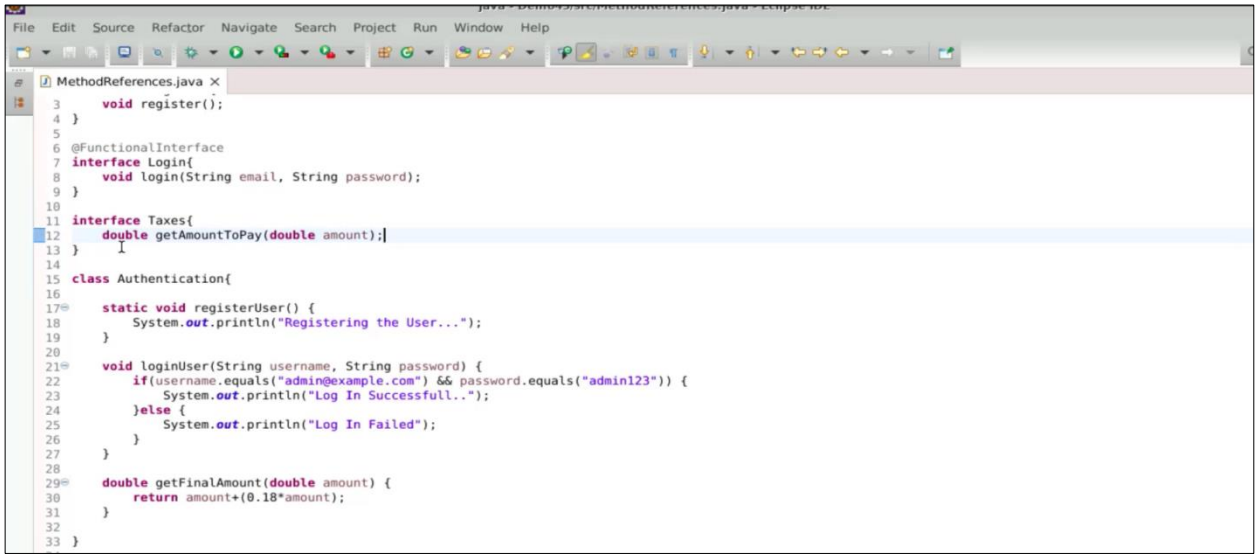


Step 6: Create methods that can return values, and execute the code with example data

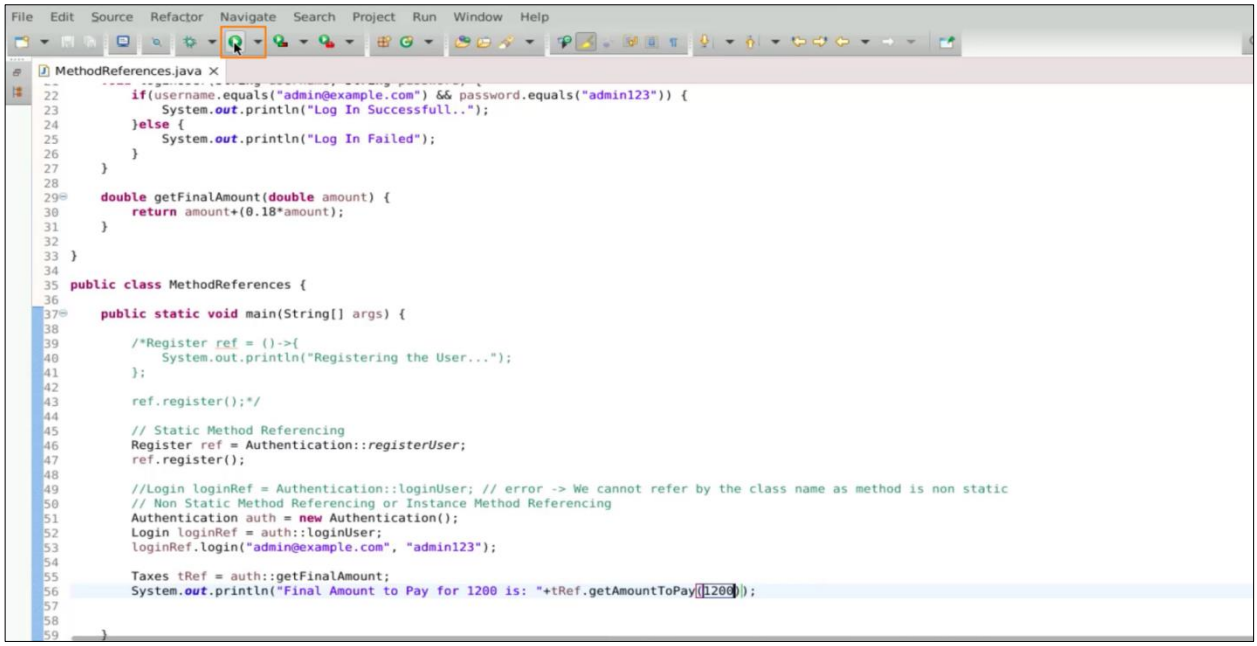
6.1 Create a functional interface called Taxes. Add a method with the name **getFinalAmount** that takes an amount as input. Implement a method with the name **getAmountToPay** in the Taxes interface, which returns the amount plus 18% of the amount



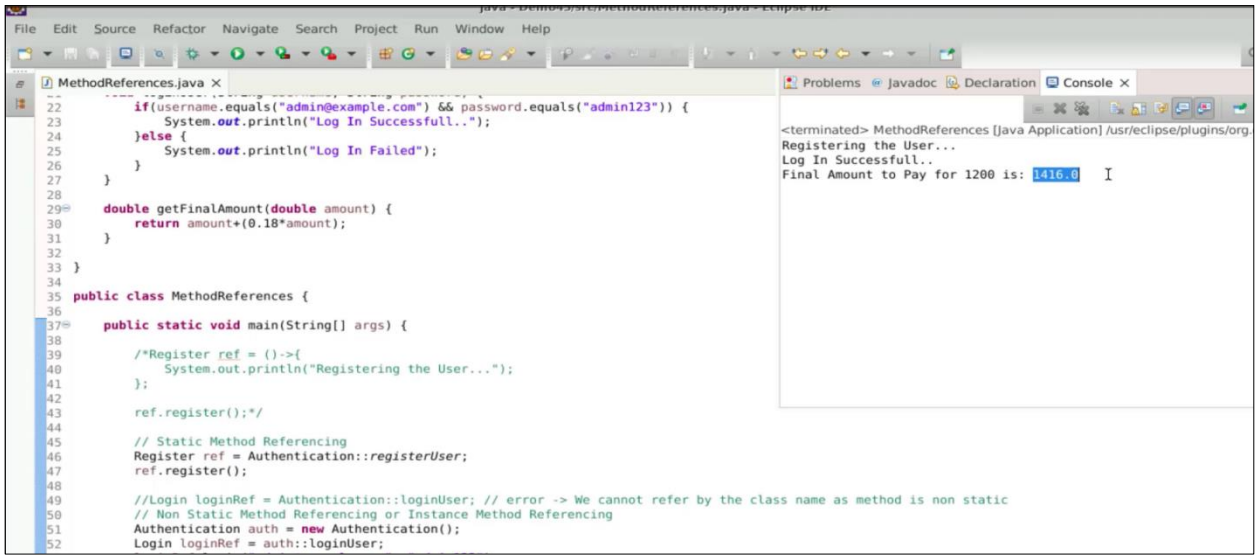
6.2 Modify the **getAmountToPay** method to take a double as the input instead of an integer



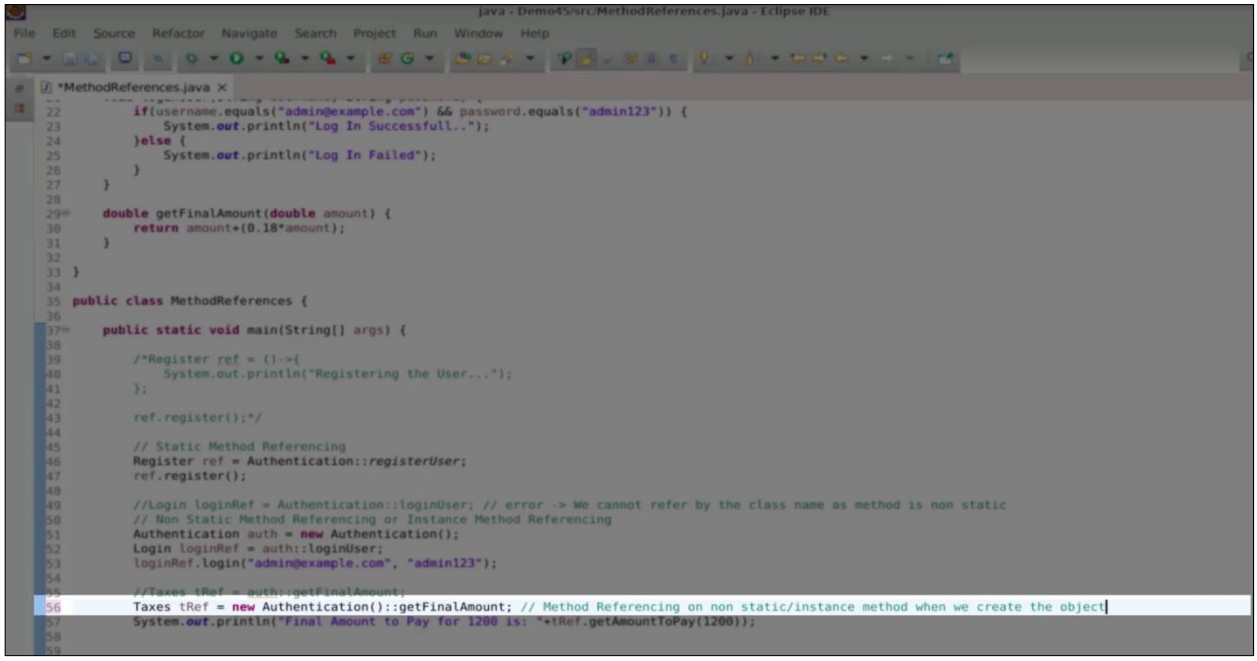
6.3 Create a reference variable of type Taxes. Assign the reference of the **getFinalAmount** method to the reference variable. Use **tRef.getAmountToPay(1200)** to calculate the final amount to pay after applying 18% taxes to 1200



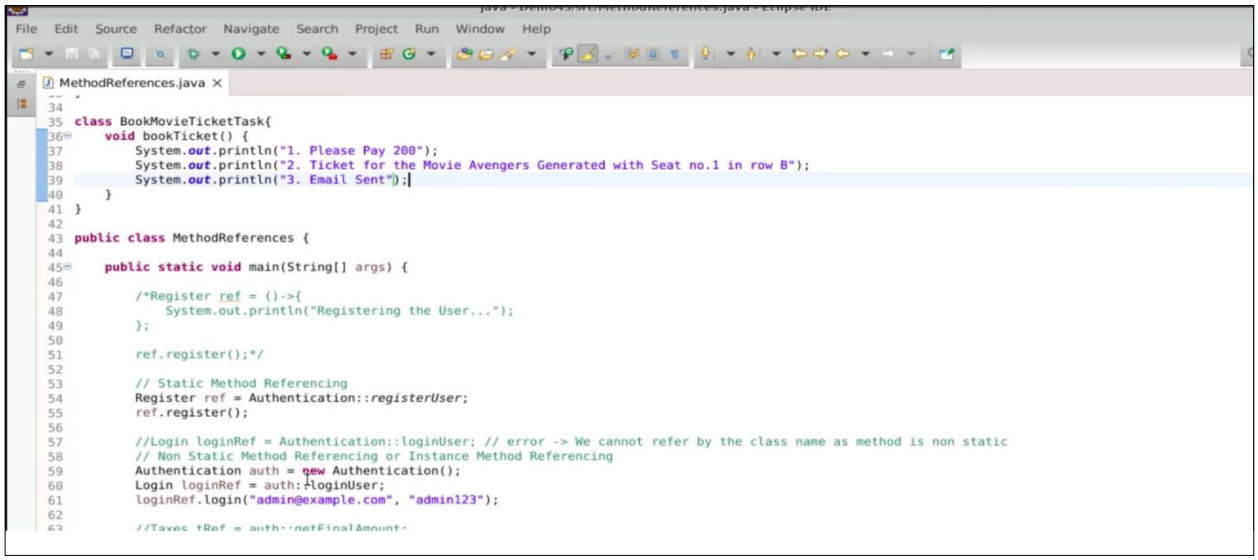
6.4 When you run the code, it should display **1416** as the final amount after applying taxes



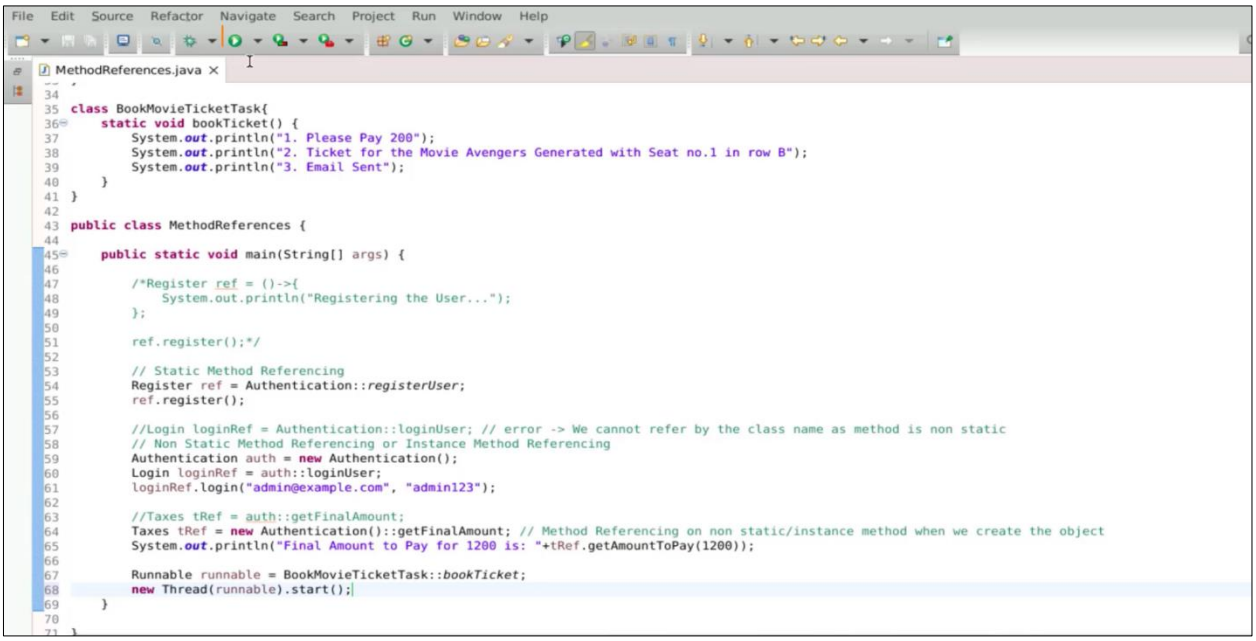
6.5 Another way to do method referencing is by creating an object of the **Authentication** class and immediately referencing the **getFinalAmount** method of the Taxes interface



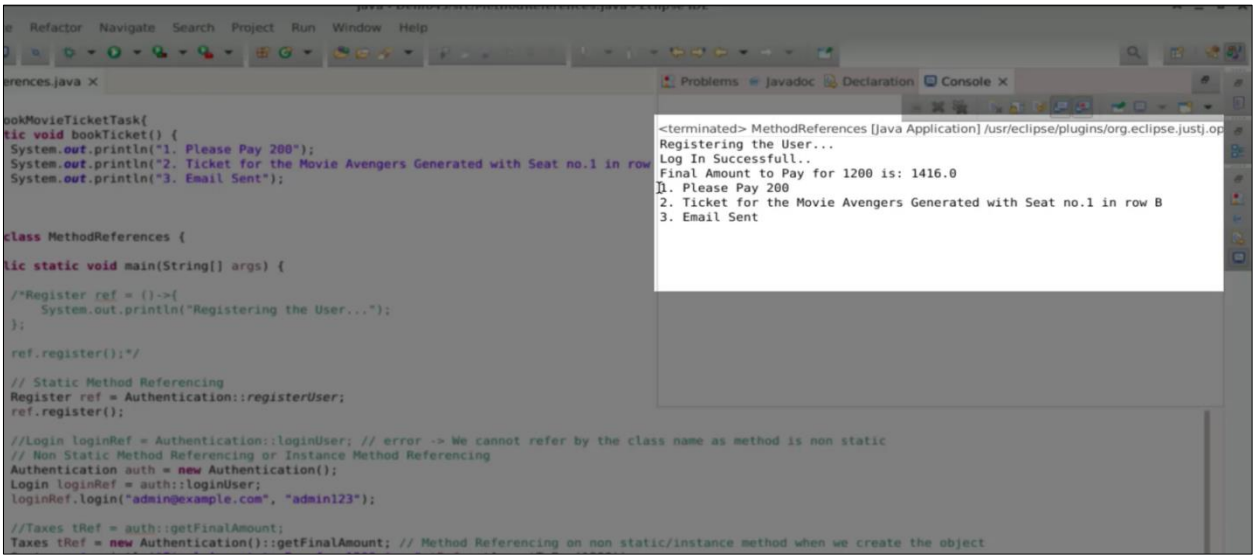
6.6 The **BookMovieTicketTask** includes a static method with the name **bookTicket**. The method prints messages related to movie ticket booking



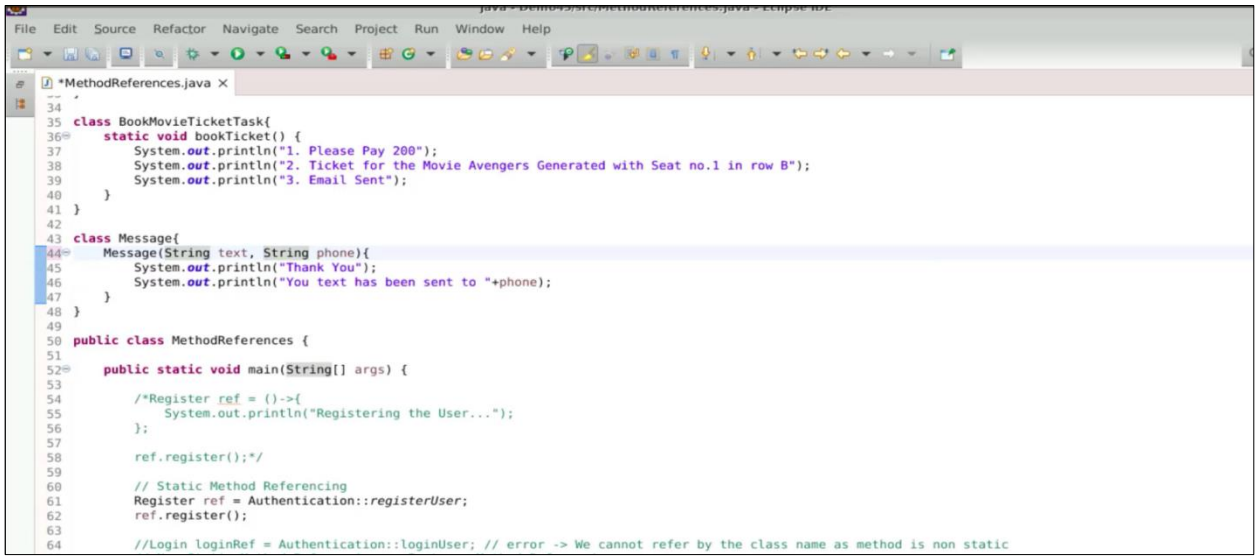
6.7 Create a Runnable object and assign the **bookTicket** method to it. Create a new thread object, pass the Runnable object as a parameter, and start the thread



6.8 When you run the program, it will asynchronously execute the **bookTicket** method and display the corresponding messages

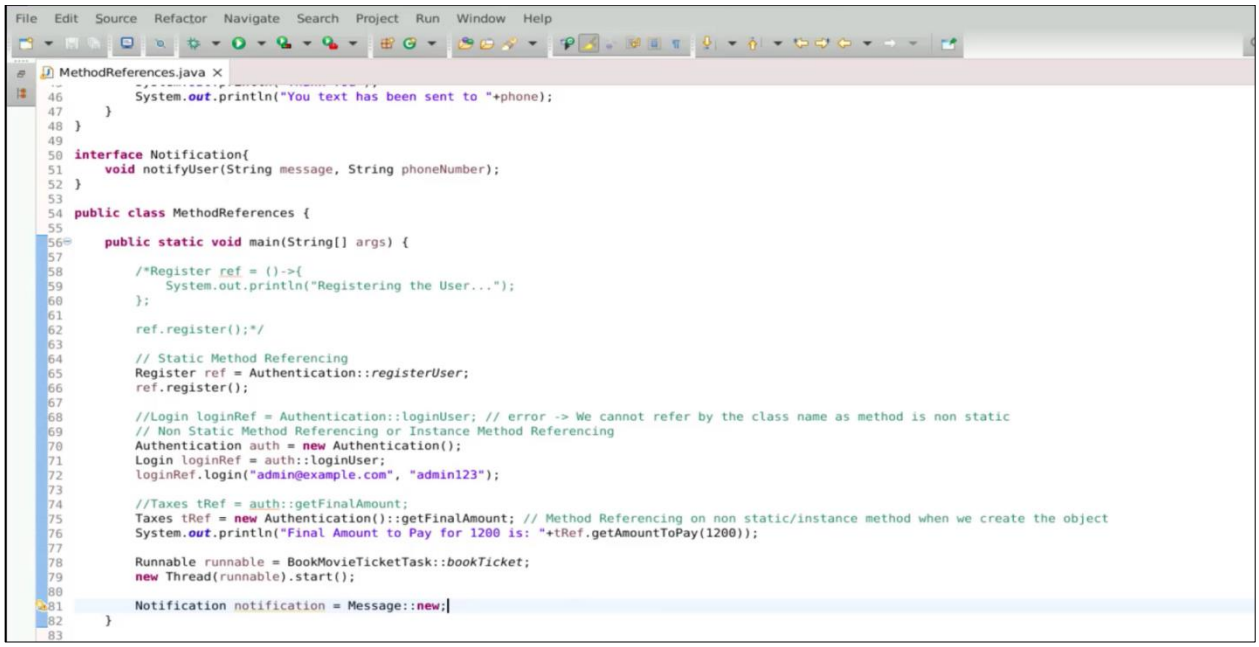


6.9 The **Message** class includes a constructor that takes a string parameter

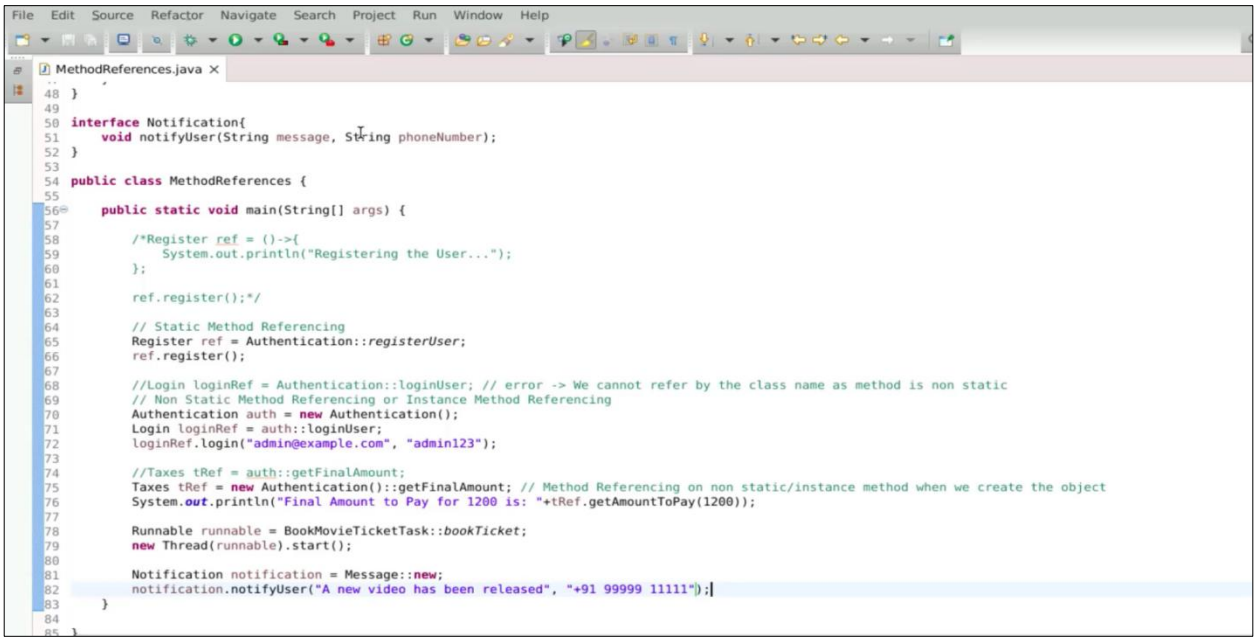


Step 7: Write the reference variable notification and execute the code

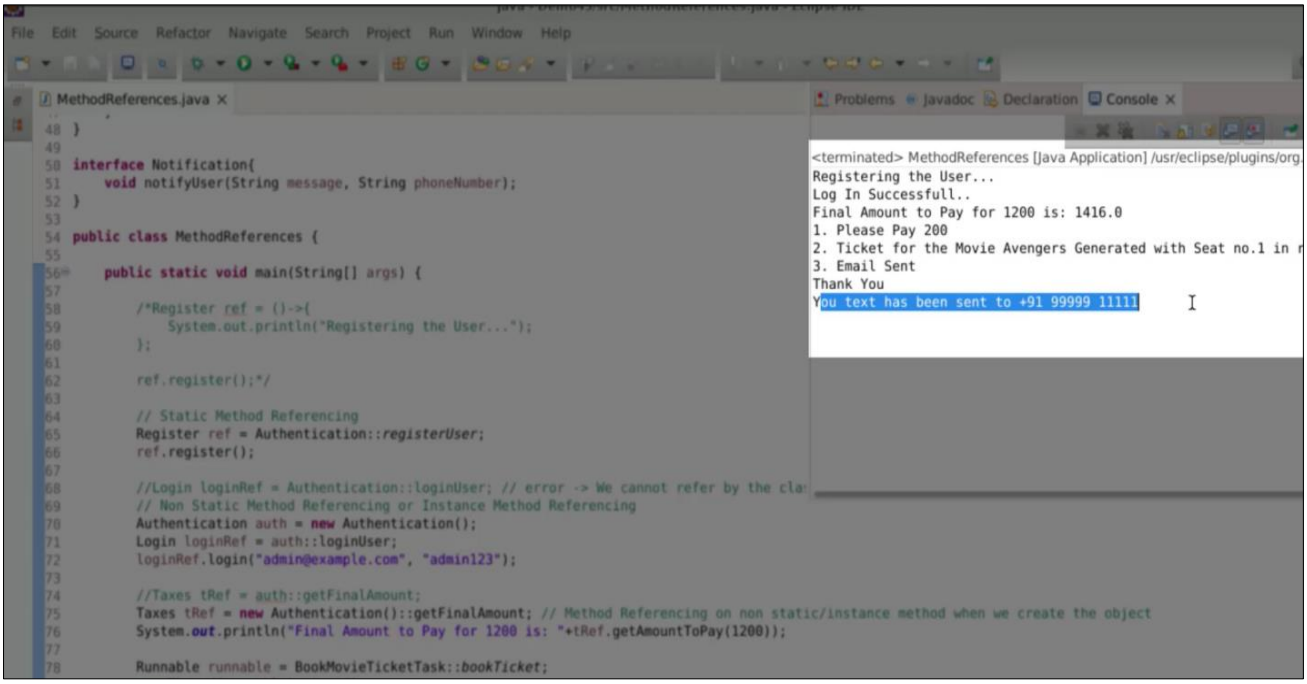
7.1 Create an interface called **Notification** with the **notifyUser** method, which takes a Message object and a phone number as input.



7.2 Create a reference variable notification of type **Notification** and assign it the reference of the **Message** constructor using the **new** keyword. Use the **notifyUser** method to send a notification with the message **A new video has been released** and a phone number



7.3 When you run the code, it will display the message as shown:



By following these steps, you have successfully implemented method references in Java, including creating functional interfaces and executing code with example data.