

Lesson 06 Demo 04

Implementing Lambda and Local Var in Java

Objective: To implement lambda expressions and the var keyword in Java 11

Tools required: Eclipse IDE

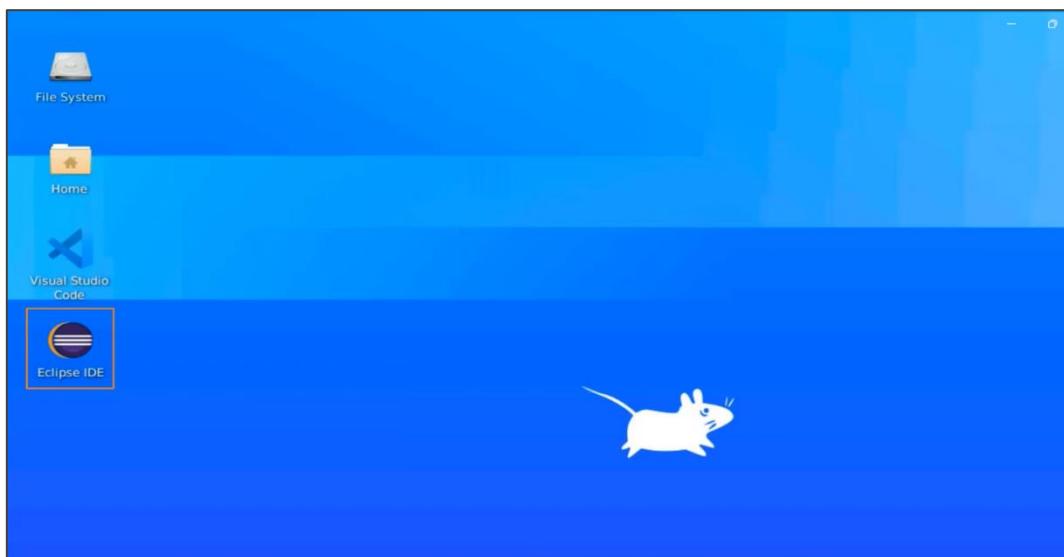
Prerequisites: None

Steps to be followed:

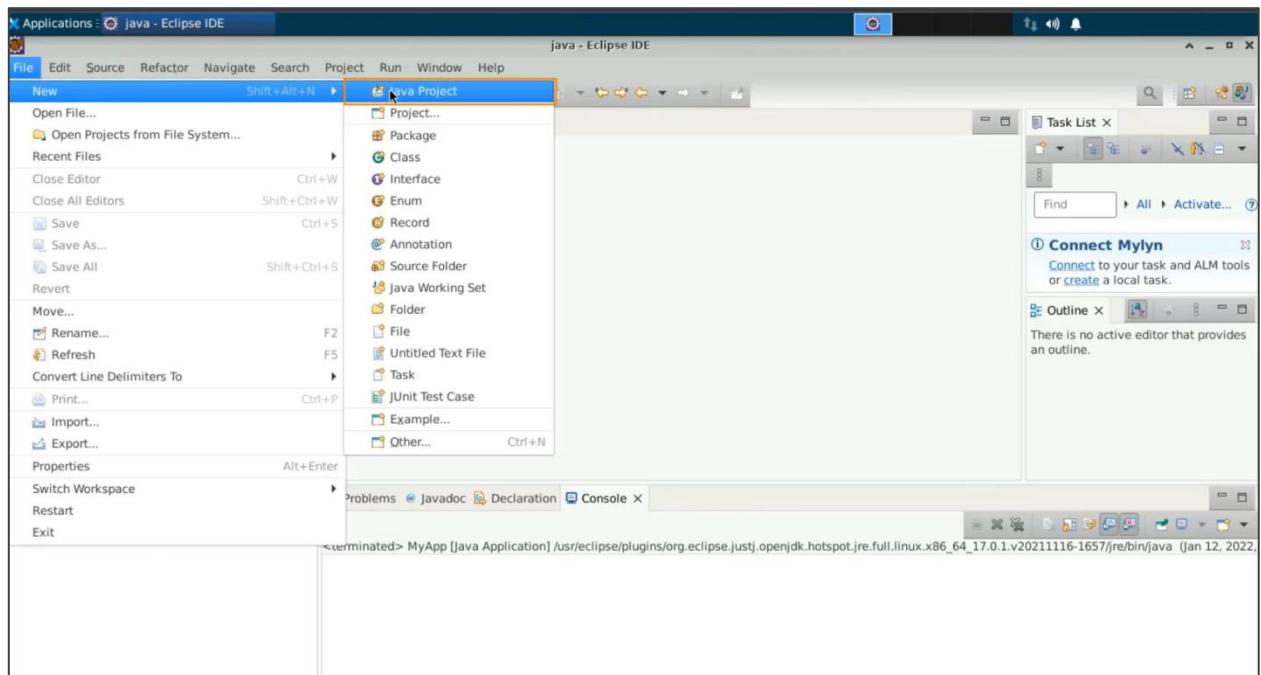
1. Open the Eclipse IDE and create a new Java project
2. Create a list which goes as the list of type string and then the emails as arrays dot as a list and executing the code
3. Create the data as comma separated values and execute the code
4. Write the variable or the var keywords, inside the lambdas

Step 1: Open the Eclipse IDE and create a new Java project

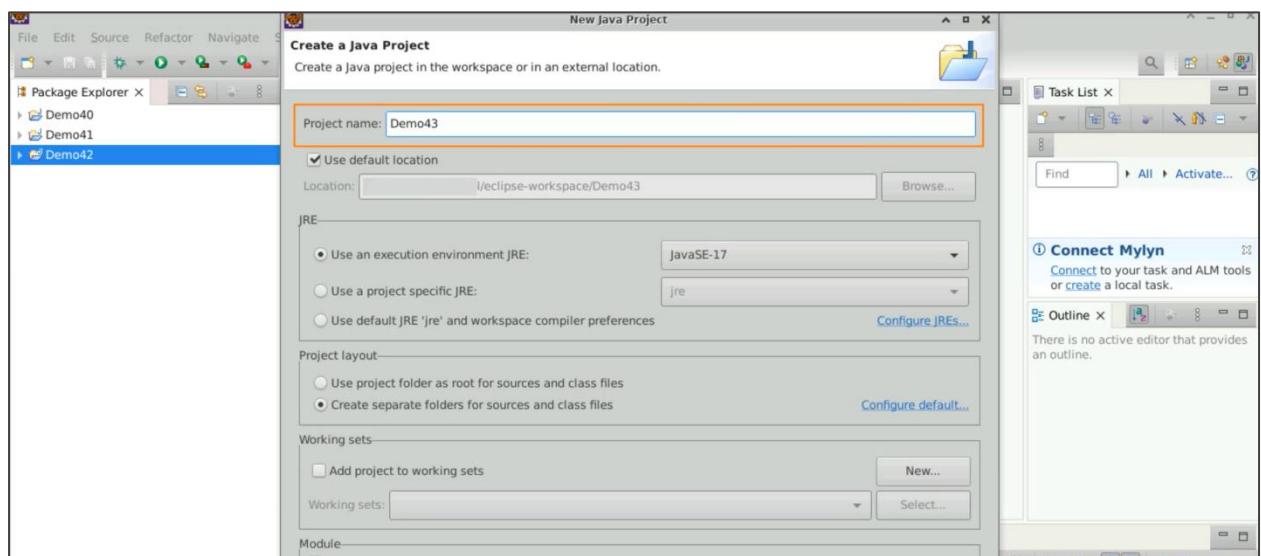
1.1 Open the Eclipse IDE



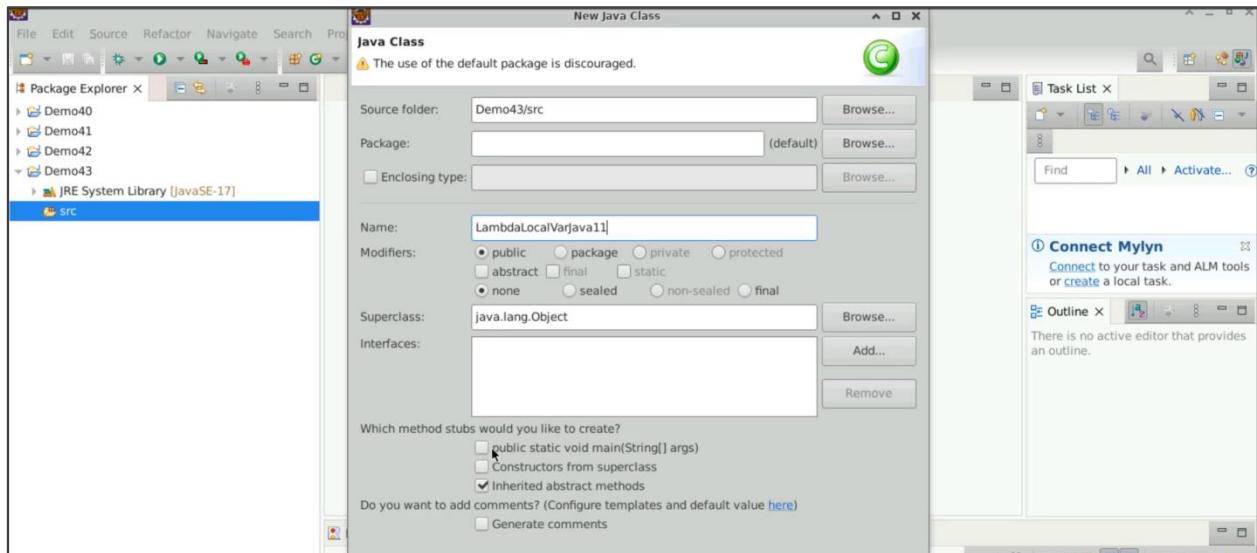
1.2 Select File, then New, and then Java project



1.3 Name the project **Demo43**, uncheck **Create a module-info.java file**, and press **Finish**



- 1.4 With **Demo43** selected in the **src** folder, right-click and create a new class. Name this class **LambdaLocalVarJava11**, then select the **main** method, and then select **Finish**



Step 2: Create a list of type String and add emails using Arrays.asList(), then execute the code

- 2.1 First, create a list of type String using **Arrays.asList()**. Then, add a few elements such as **john@example.com**, **fionna@example.com**, and **mike@example.com**. This will give us a list of emails. Finally, print the emails

```

eclipse-workspace - Demo43/src/LambdaLocalVarJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*LambdaLocalVarJava11.java X
1 import java.util.Arrays;
2 import java.util.List;
3
4 public class LambdaLocalVarJava11 {
5
6     public static void main(String[] args) {
7
8         List<String> emails = Arrays.asList("john@example.com", "fionna@example.com", "mike@example.com");
9         System.out.println(emails);
10    }
11
12
13 }
14

```

2.2 When you run the code, it will show the basic list of emails

The screenshot shows the Eclipse IDE interface. On the left, the code editor displays a file named `LambdaLocalVarJava11.java` with the following content:

```

1 import java.util.Arrays;
2 import java.util.List;
3
4 public class LambdaLocalVarJava11 {
5
6     public static void main(String[] args) {
7
8         List<String> emails = Arrays.asList("john@example.com", "fionna@example.com", "mike@example.com");
9         System.out.println(emails);
10    }
11
12 }
13
14

```

On the right, the `Console` view shows the output of the program's execution:

```

<terminated> LambdaLocalVarJava11 [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/bin [john@example.com, fionna@example.com, mike@example.com]

```

Step 3: Create the data as comma-separated values and execute the code

3.1 First, create the data as comma-separated values. Write it `String csvData = emails.stream()`

The screenshot shows the Eclipse IDE interface. On the left, the code editor displays a file named `*LambdaLocalVarJava11.java` with the following content:

```

1 import java.util.Arrays;
2 import java.util.List;
3
4 public class LambdaLocalVarJava11 {
5
6     public static void main(String[] args) {
7
8         List<String> emails = Arrays.asList("john@example.com", "fionna@example.com", "mike@example.com");
9         System.out.println(emails);
10
11     String csvData = emails.stream()
12         .map(
13             (String email) -> email.replace("example.com", "abc.com")
14         )
15
16     }
17
18 }
19

```

A code completion tooltip is visible over the line `(String email) ->`, showing the suggestion `email.replace("example.com", "abc.com")`.

3.2 Use `.collect(Collectors.joining(","))` and add a semicolon at the end. Then, print the CSV data, which is a new joining created from a domain name

```

eclipse-workspace - Demo43/src/LambdaLocalVarJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
LambdaLocalVarJava11.java X
1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.stream.Collectors;
4
5 public class LambdaLocalVarJava11 {
6
7     public static void main(String[] args) {
8
9         List<String> emails = Arrays.asList("john@example.com", "fionna@example.com", "mike@example.com");
10        System.out.println(emails);
11
12        String csvData = emails.stream()
13            .map(
14                (String email) -> email.replace("example.com", "abc.com")
15            )
16            .collect(Collectors.joining(","));
17
18        System.out.println("CSV Data");
19        System.out.println(csvData);
20
21    }
22
23 }
24

```

3.3 Run the code, and you will notice that the emails are separated by commas. You can even use a comma and a space to introduce a space in between

```

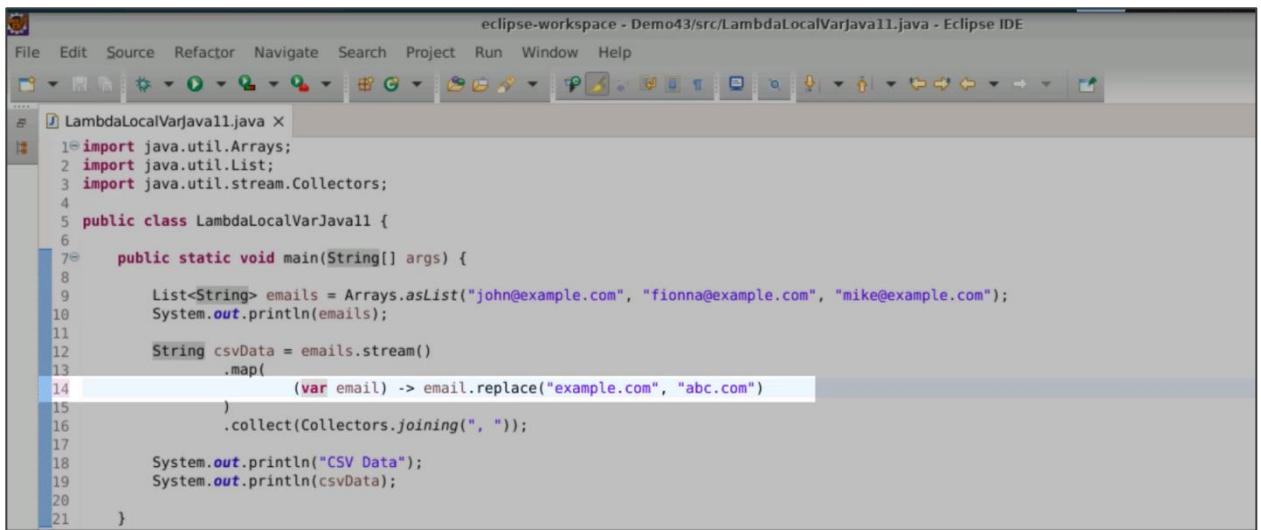
eclipse-workspace - Demo43/src/LambdaLocalVarJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
LambdaLocalVarJava11.java X
1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.stream.Collectors;
4
5 public class LambdaLocalVarJava11 {
6
7     public static void main(String[] args) {
8
9         List<String> emails = Arrays.asList("john@example.com", "fionna@example.com", "mike@example.com");
10        System.out.println(emails);
11
12        String csvData = emails.stream()
13            .map(
14                (String email) -> email.replace("example.com", "abc.com")
15            )
16            .collect(Collectors.joining(","));
17
18        System.out.println("CSV Data");
19        System.out.println(csvData);
20
21    }
22
23 }
24

```

Problems Javadoc Declaration Console X

<terminated> LambdaLocalVarJava11 [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/john@example.com, fionna@example.com, mike@example.com
CSV Data
john@abc.com, fionna@abc.com, mike@abc.com

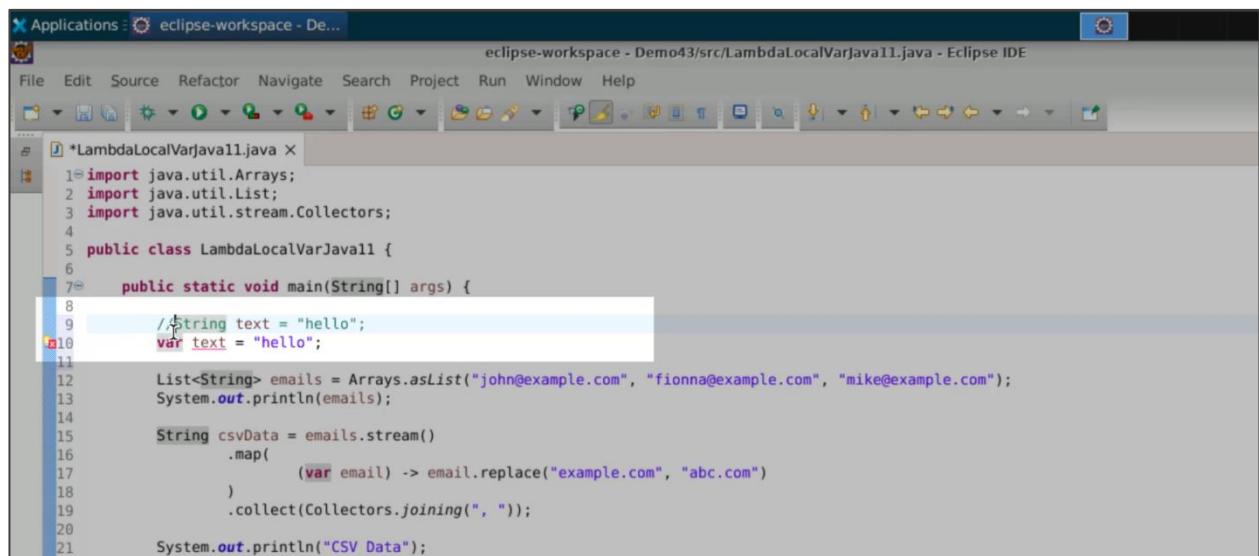
3.4 Instead of using **String** here, you can just write **var email**



```
eclipse-workspace - Demo43/src/LambdaLocalVarJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
* LambdaLocalVarJava11.java X
1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.stream.Collectors;
4
5 public class LambdaLocalVarJava11 {
6
7     public static void main(String[] args) {
8
9         List<String> emails = Arrays.asList("john@example.com", "fionna@example.com", "mike@example.com");
10        System.out.println(emails);
11
12        String csvData = emails.stream()
13            .map(
14                (var email) -> email.replace("example.com", "abc.com")
15            )
16            .collect(Collectors.joining(", "));
17
18        System.out.println("CSV Data");
19        System.out.println(csvData);
20    }
21 }
```

Step 4: Write the variable or the var keywords, inside the lambdas

4.1 Use the variable **String text** as **hello**. You can even write **var text = "hello"**. The same variable or the **var** keyword can be used inside your lambdas as well. This is one of the new features in Java 11



```
Applications : eclipse-workspace - De...
eclipse-workspace - Demo43/src/LambdaLocalVarJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
* * LambdaLocalVarJava11.java X
1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.stream.Collectors;
4
5 public class LambdaLocalVarJava11 {
6
7     public static void main(String[] args) {
8
9         //String text = "hello";
10        var text = "hello";
11
12        List<String> emails = Arrays.asList("john@example.com", "fionna@example.com", "mike@example.com");
13        System.out.println(emails);
14
15        String csvData = emails.stream()
16            .map(
17                (var email) -> email.replace("example.com", "abc.com")
18            )
19            .collect(Collectors.joining(", "));
20
21        System.out.println("CSV Data");
22 }
```

By following these steps, you have successfully implemented lambda expressions and the **var** keyword in Java 11.