

Lesson 06 Demo 03

Converting Collections to Array in Java

Objective: To convert collections to arrays in Java 11 using efficient methods and proper data structures

Tools required: Eclipse IDE

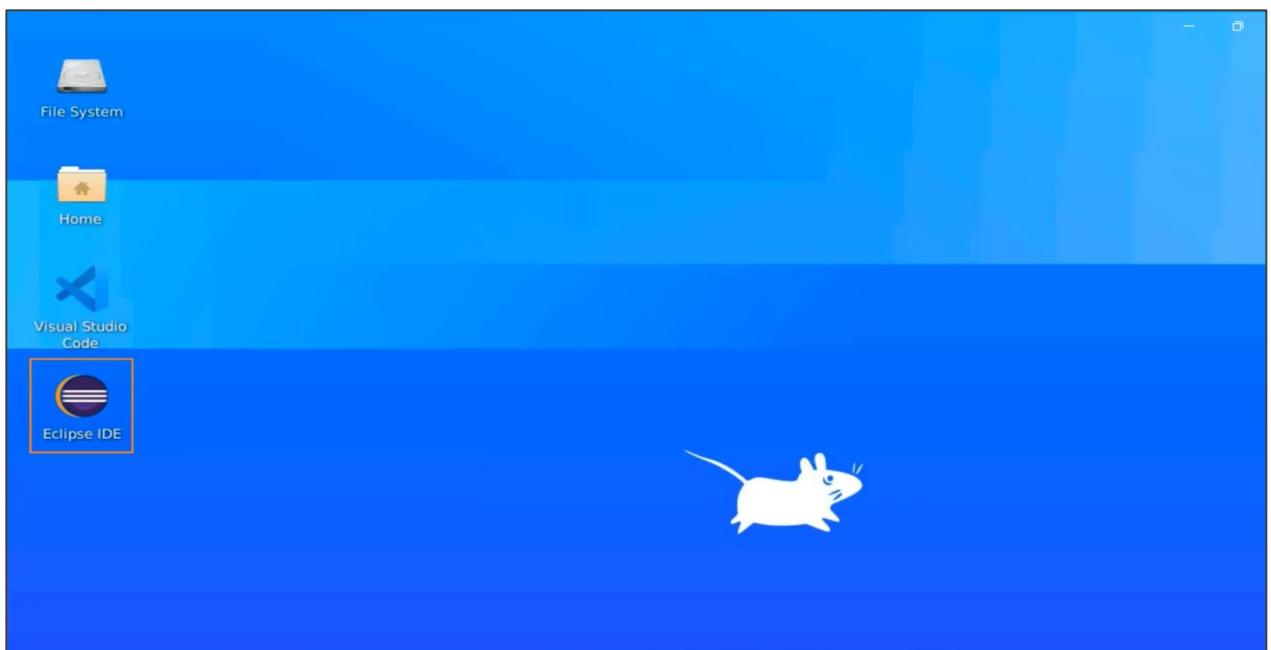
Prerequisites: None

Steps to be followed:

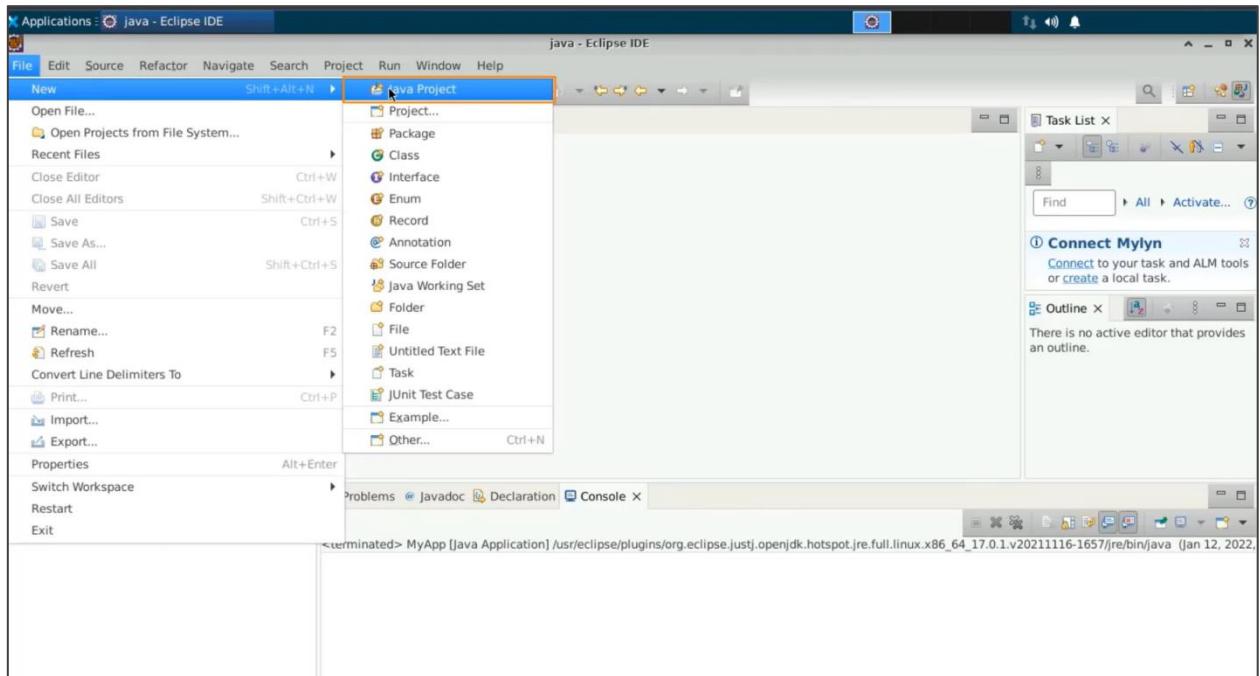
1. Open the Eclipse IDE and create a new Java project
2. Write few emails as a new array list
3. Convert the array list back to an array
4. Print emails and the data coming in
5. Create an array of objects

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

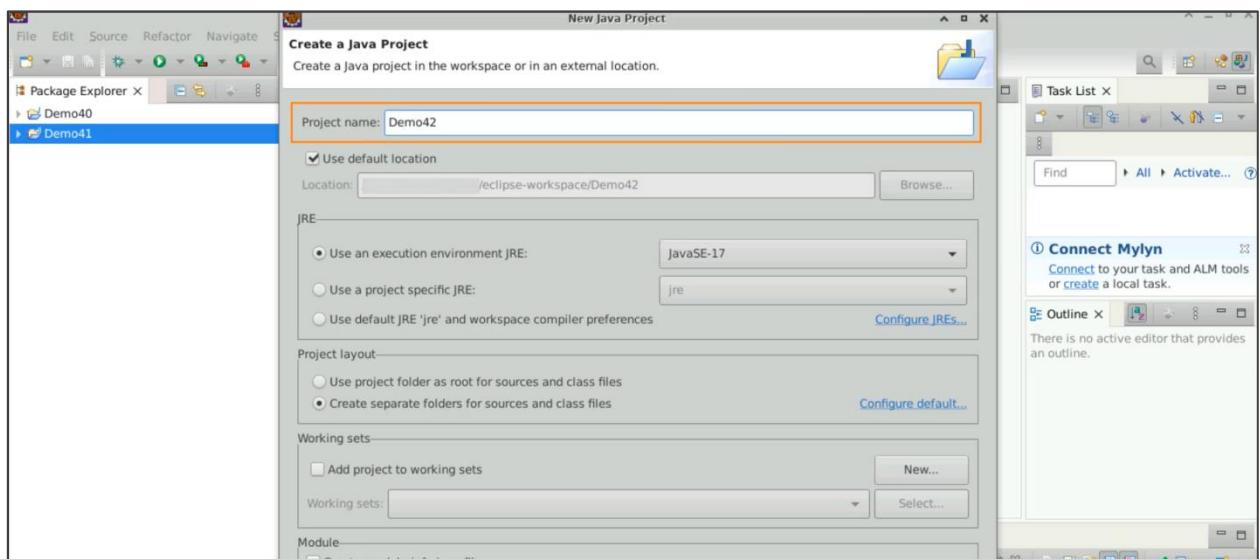
1.1 Open the Eclipse IDE



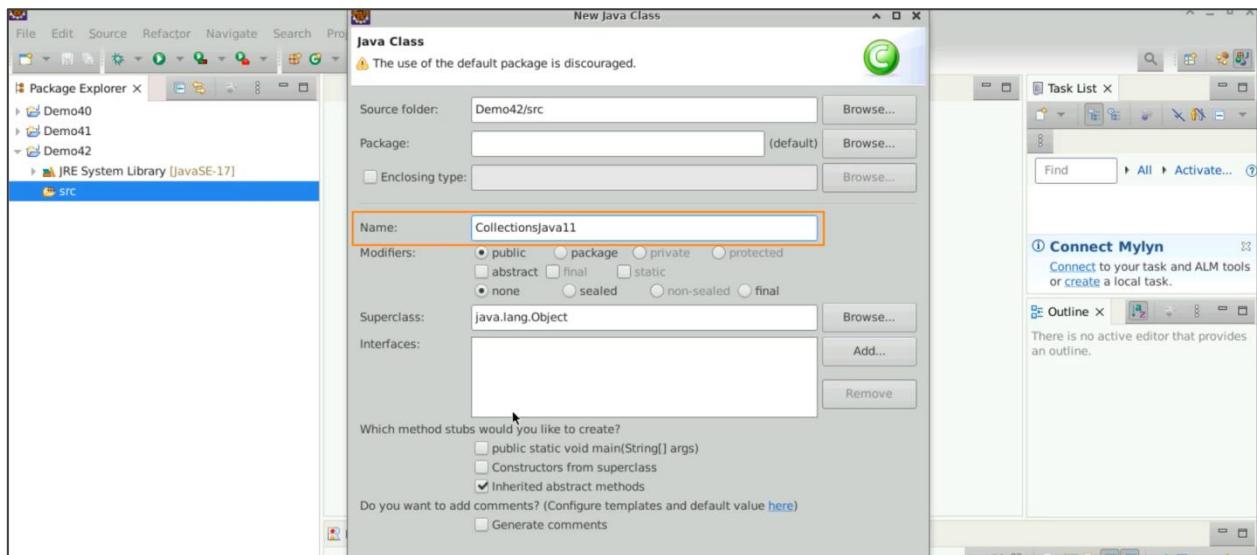
1.2 Select File, then New, and then Java project



1.3 Name the project **Demo42**, uncheck **Create a module-info.java file**, and press **Finish**



1.4 With **Demo42** selected in the **src** folder, right-click and create a new class. Name this class **CollectionsJava11**, then select the main method, and then select Finish



Step 2: Write few emails as a new array list

2.1 Let us write a type String and add a few emails as a new **ArrayList**

```

eclipse-workspace - Demo42/src/CollectionsJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CollectionsJava11.java X
1 import java.util.ArrayList;
2
3 public class CollectionsJava11 {
4
5     public static void main(String[] args) {
6
7         ArrayList<String> emails = new ArrayList<String>();
8         emails.add("john@example.com");
9         emails.add("fionna@example.com");
10        emails.add("mike@example.com");
11
12    }
13
14 }
15
16 }
17

```

2.2 To simplify, you can create an array of emails and convert it into a list. This allows you to have a simple representation of the data using a homogeneous container

```

eclipse-workspace - Demo42/src/CollectionsJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*CollectionsJava11.java
1 import java.util.ArrayList;
2
3 public class CollectionsJava11 {
4
5     public static void main(String[] args) {
6
7         ArrayList<String> emails = new ArrayList<String>();
8         emails.add("john@example.com");
9         emails.add("fionna@example.com");
10        emails.add("mike@example.com");
11
12        String[] emailsArray = {"john@example.com", "fionna@example.com", "mike@example.com"};
13
14    }
15
16 }
17

```

2.3 Next, you can use an email array and convert it to a list using **Arrays.asList()**. This will give you a list of emails which you can refer to as "**emails list**" from the **java.util** package

```

eclipse-workspace - Demo42/src/CollectionsJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*CollectionsJava11.java
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4
5 public class CollectionsJava11 {
6
7     public static void main(String[] args) {
8
9         ArrayList<String> emails = new ArrayList<String>();
10        emails.add("john@example.com");
11        emails.add("fionna@example.com");
12        emails.add("mike@example.com");
13
14        String[] emailsArray = {"john@example.com", "fionna@example.com", "mike@example.com"};
15        List<String> emailsList = Arrays.asList(emailsArray);
16        System.out.println(emailsList);
17
18    }
19
20 }
21

```

2.4 By converting the array to a list, you can process the emails list faster and more efficiently than adding emails one by one. It provides a quicker alternative to create and represent the list

The screenshot shows the Eclipse IDE interface with the following code in the CollectionsJava1.java file:

```

eclipse-workspace - Demo42/src/CollectionsJava1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CollectionsJava1.java
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4
5 public class CollectionsJava1 {
6
7     public static void main(String[] args) {
8
9         ArrayList<String> emails = new ArrayList<String>();
10        emails.add("john@example.com");
11        emails.add("fionna@example.com");
12        emails.add("mike@example.com");
13
14        String[] emailsArray = {"john@example.com", "fionna@example.com", "mike@example.com"};
15        List<String> emailsList = Arrays.asList(emailsArray);
16        System.out.println(emailsList);
17
18    }
19
20 }
21

```

The Console view shows the output: <terminated> CollectionsJava1 [java Application] /usr/eclipse/plugins/org.eclipse.just_ops/[john@example.com, fionna@example.com, mike@example.com]

Step 3: Convert the array list back to an array

3.1 With the Java 11 feature, you can convert a list or ArrayList back to an array. For instance, use the **toArray** method on your **emails list** and specify the data type, such as creating a new string array using “**Array myEmails is : +myEmails**

The screenshot shows the Eclipse IDE interface with the following code in the CollectionsJava1.java file:

```

eclipse-workspace - Demo42/src/CollectionsJava1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CollectionsJava1.java
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4
5 public class CollectionsJava1 {
6
7     public static void main(String[] args) {
8
9         ArrayList<String> emails = new ArrayList<String>();
10        emails.add("john@example.com");
11        emails.add("fionna@example.com");
12        emails.add("mike@example.com");
13
14        String[] emailsArray = {"john@example.com", "fionna@example.com", "mike@example.com"};
15        List<String> emailsList = Arrays.asList(emailsArray);
16        System.out.println(emailsList);
17
18        String[] myEmails = emailsList.toArray(String[]::new);
19
20        System.out.println("Array myEmails is: "+myEmails);
21
22    }
23
24 }
25

```

3.2 As you know, whenever you print the array, it will always give the hash code. This is how you can see the hash code coming in

The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - Demo42/src/CollectionsJava11.java - Eclipse IDE". The code editor displays the following Java code:

```

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
public class CollectionsJava11 {
    public static void main(String[] args) {
        ArrayList<String> emails = new ArrayList<String>();
        emails.add("john@example.com");
        emails.add("fionna@example.com");
        emails.add("mike@example.com");
        String[] emailsArray = {"john@example.com", "fionna@example.com", "mike@example.com"};
        List<String> emailsList = Arrays.asList(emailsArray);
        System.out.println(emailsList);
        String[] myEmails = emailsList.toArray(String[]::new);
        System.out.println("Array myEmails is: "+myEmails);
    }
}

```

The "Console" tab in the bottom right shows the output of the program:

```

<terminated> CollectionsJava11 [java Application] /usr/eclipse/plugins/org.eclipse.justj.opc/john@example.com, fionna@example.com, mike@example.com
Array myEmails is: [Ljava.lang.String;@71bc1ae4

```

3.3 Now, let us write a loop to print each email in the **myEmails** array

The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - Demo42/src/CollectionsJava11.java - Eclipse IDE". The code editor displays the same Java code as the previous screenshot, but with a for-loop added to print each email in the **myEmails** array:

```

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
public class CollectionsJava11 {
    public static void main(String[] args) {
        ArrayList<String> emails = new ArrayList<String>();
        emails.add("john@example.com");
        emails.add("fionna@example.com");
        emails.add("mike@example.com");
        String[] emailsArray = {"john@example.com", "fionna@example.com", "mike@example.com"};
        List<String> emailsList = Arrays.asList(emailsArray);
        System.out.println(emailsList);
        String[] myEmails = emailsList.toArray(String[]::new);
        System.out.println("Array myEmails is: "+myEmails);
        for(String email : myEmails) {
            System.out.println(email);
        }
    }
}

```

A red circle with a question mark icon is shown over the first line of the for-loop, likely indicating a warning or error.

Step 4: Print emails and the data coming in

4.1 Let us print email one by one and here is the data coming in

The screenshot shows the Eclipse IDE interface. The left pane displays the Java code for `CollectionsJava1.java`. The right pane shows the `Console` tab with the following output:

```
<terminated> CollectionsJava1 [Java Application] /usr/eclipse/plugins/org.eclipse.justj.opc/john@example.com, fionna@example.com, mike@example.com]
Array myEmails is: [Ljava.lang.String;@6ed3ef1
john@example.com
fionna@example.com
mike@example.com
```

Step 5: Create an array of objects

5.1 If you have heterogeneous list, you can create an array of objects as per your requirement. Currently, you have string array coming up

The screenshot shows the Eclipse IDE interface. The left pane displays the Java code for `CollectionsJava1.java`. The right pane shows the `Console` tab with the following output:

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4
5 public class CollectionsJava1 {
6
7     public static void main(String[] args) {
8
9         ArrayList<String> emails = new ArrayList<String>();
10        emails.add("john@example.com");
11        emails.add("fionna@example.com");
12        emails.add("mike@example.com");
13
14        String[] emailsArray = {"john@example.com", "fionna@example.com", "mike@example.com"};
15        List<String> emailsList = Arrays.asList(emailsArray);
16        System.out.println(emailsList);
17
18        String[] myEmails = emailsList.toArray(Object[]::new);
19
20        System.out.println("Array myEmails is: "+myEmails);
21
22        for(String email : myEmails) {
23            System.out.println(email);
24        }
25    }
}
```

By following these steps, you have successfully converted collections to arrays in Java 11 using efficient methods and proper data structures.