Lesson 03 Demo 02

Comparing Static and Non-Static Methods in OOPs

Objective: Comparing the Static and Non-Static Methods in OOPs Tools: Eclipse IDE Prerequisites: None

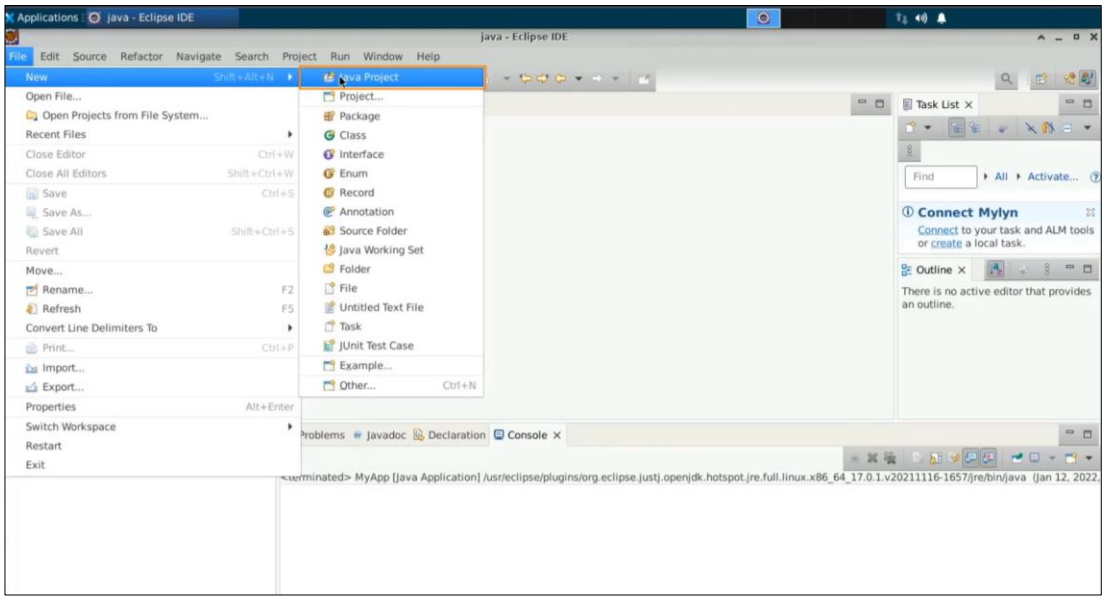
Steps to be followed:

1. Open Eclipse IDE, and open a new Java project and a class
2. Create a default constructor
3. Write a function to update data in the object
4. Create real objects in memory and execute the code
5. Create an attribute and differentiate static and non-static attributes
6. Create more objects, use a reference copy, and execute the code
7. Create methods for increment and decrement
8. Create a static variable and execute the code

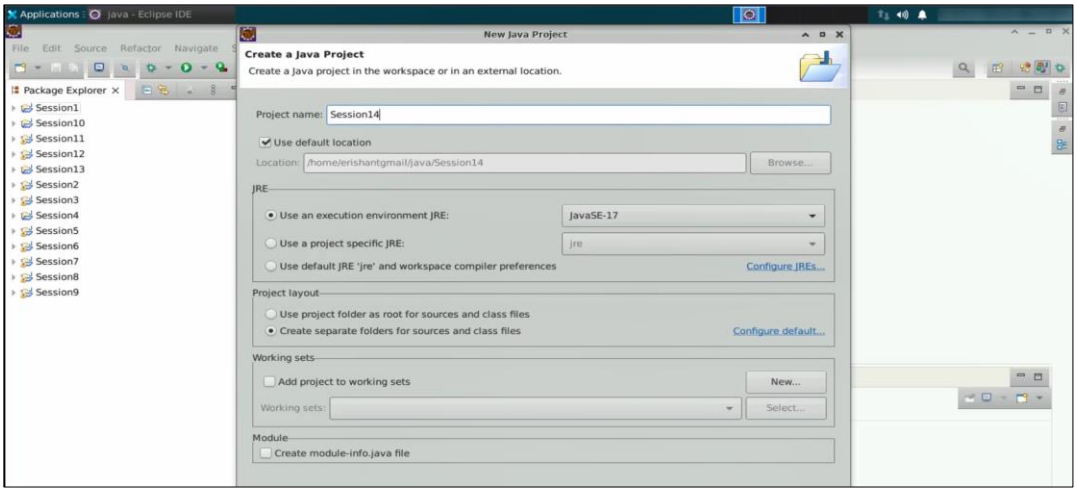
Step 1: Open Eclipse IDE, and open a new Java project and a class

1.1 Open the **Eclipse IDE**

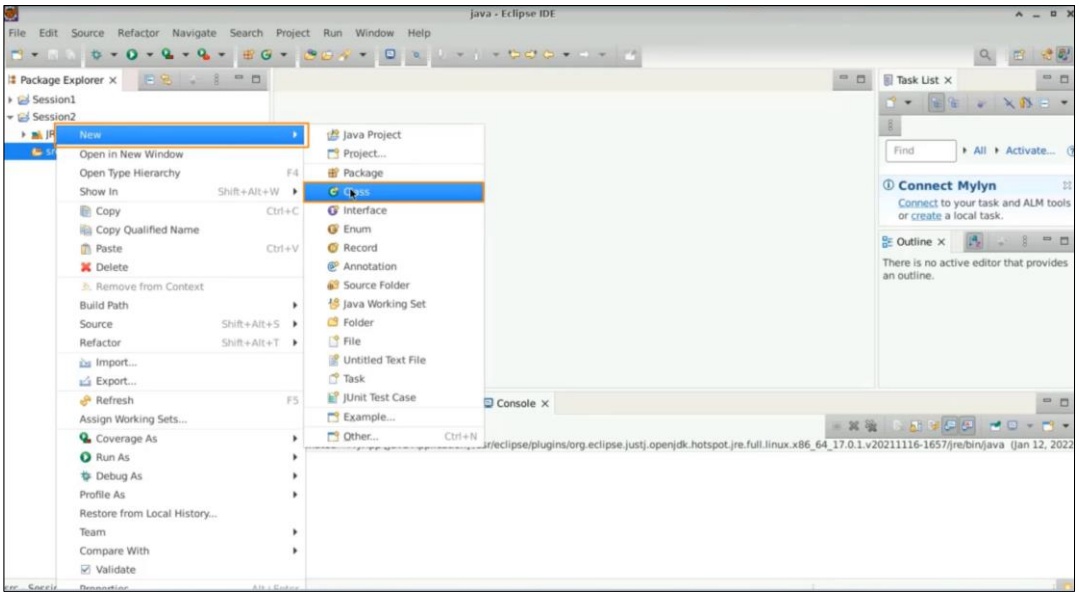
1.2. Select **File**, then **New,** and then **Java project**



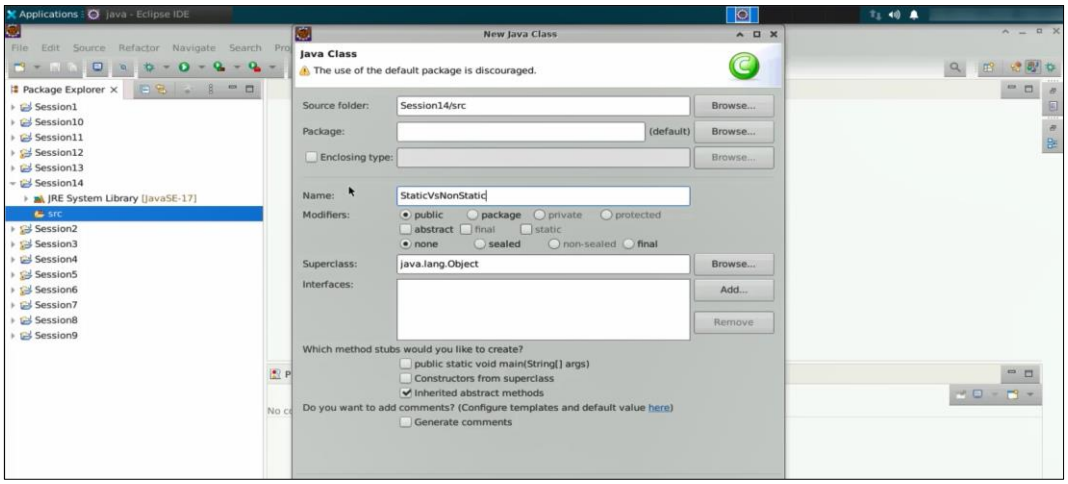
1.3 Name the project **“Session14”,** uncheck **“Create a module info dot Java file”**, and press **Finish**



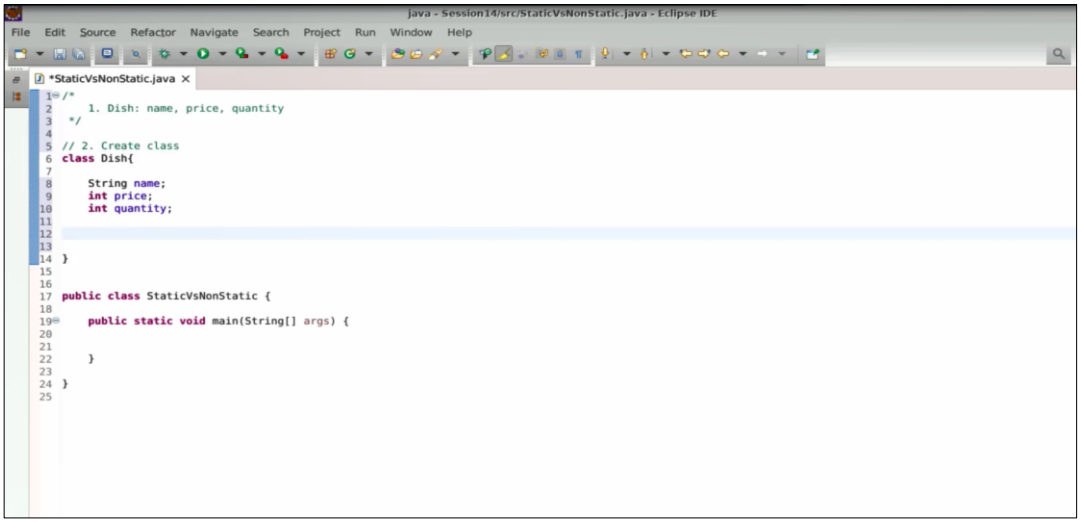
1.4 With a **Session14** on the src, do a right-click and create a **new class**



1.5 Name this class as a **StaticVsNonStatic**, then select the **main method,** and then select **finish**

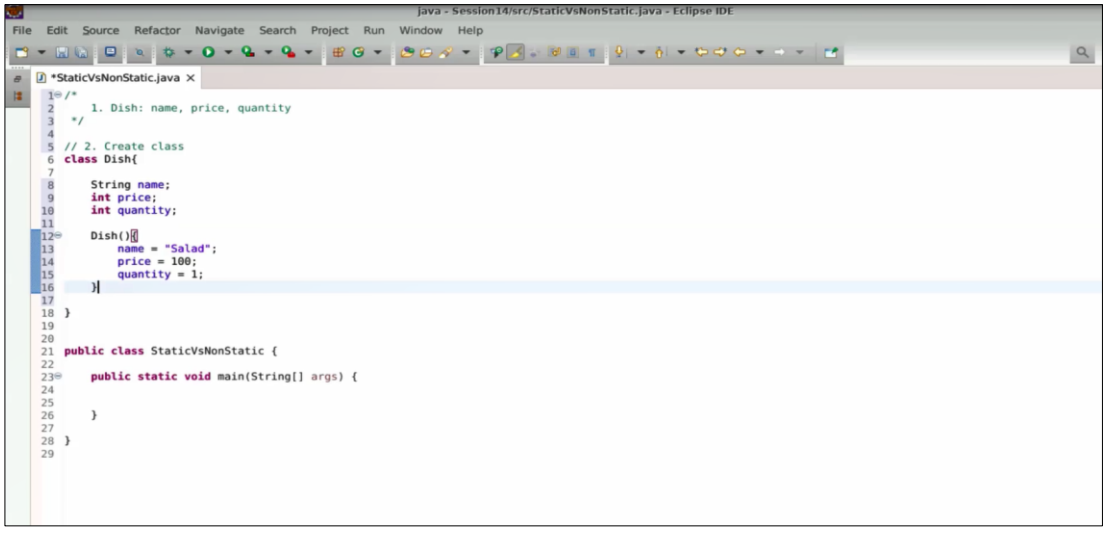


1.6 Create a class by the name **Dish**. As per the principle of OOPS, the dish is supposed to have some attributes. It will have a **name**, **price**, and **quantity**

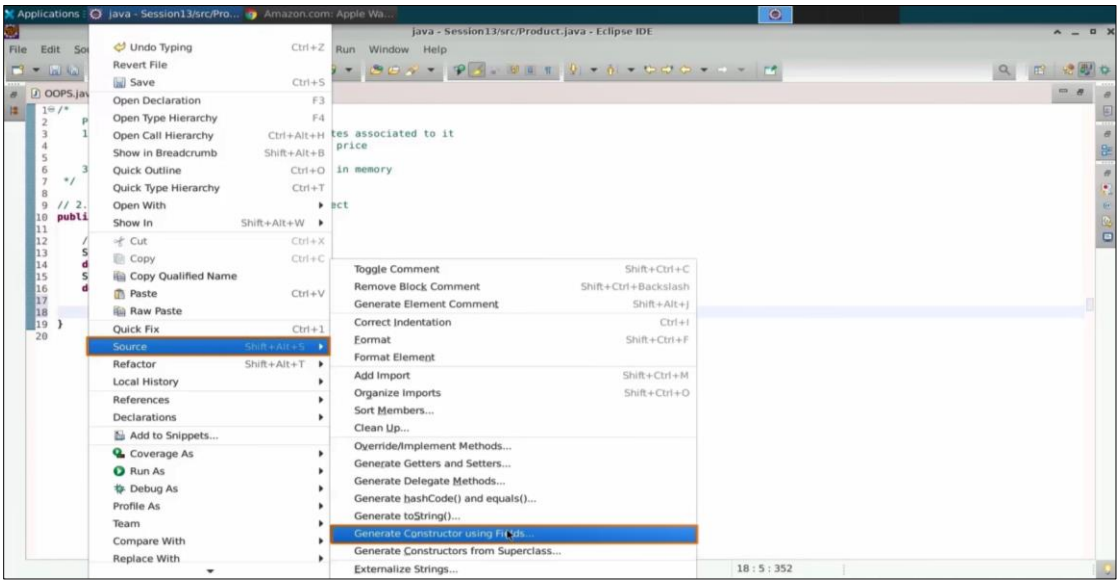


Step 2: Create a default constructor

2.1 Now for your object to be completed, you need to create a constructor that is the default, and if you want you can give some data. The price is maybe zero and the quantity is 1 by default. Or by default, the dish can be a salad whose price is some 100 and the quantity would be 1

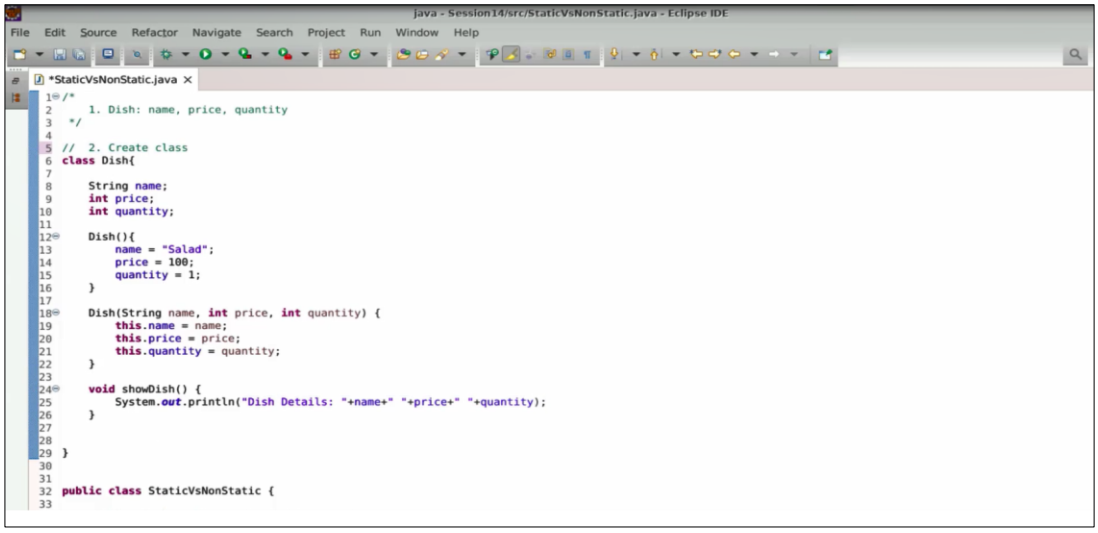


2.2 Right-click and select the source to generate the constructor using the fields



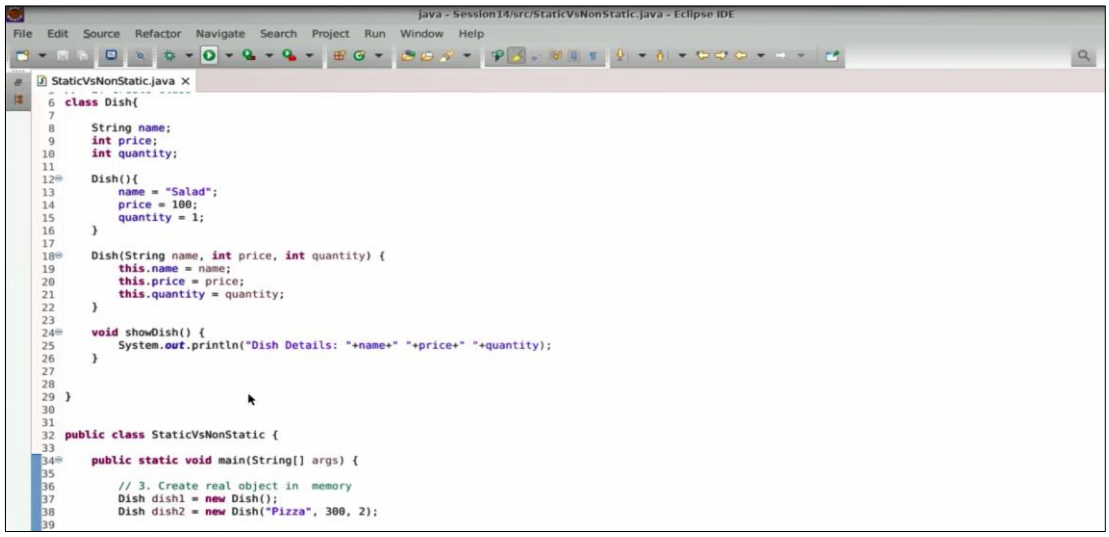
Step 3: Write a function to update data in the object

3.1 Let us create a function called **showDish**. If you want to update the data in the object later, you can create a **set** method. For the **showDish** method, type the **dish details**, including the **name**, a space, the **price**, and then the **quantity**. This is the second pattern to the principle of OOP

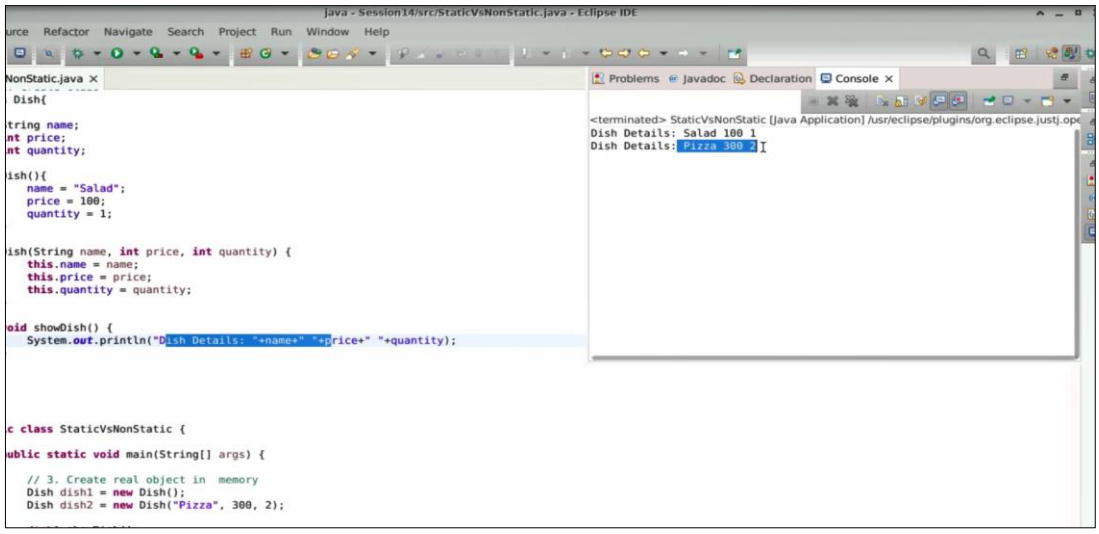


Step 4: Create real objects in memory and execute the code

4.1 The third step is to create real objects in memory from the class. Here, **dish1** is a new dish. You will have a dish named **'salad'**, with a price of 100 and a quantity of one. Create another object with a reference variable **dish2**. You are going to add **'pizza'**, then set the price to 300 and the quantity to two. Type **dish1.showDish()** and **dish2.showDish()**. It will print the data available in the dish

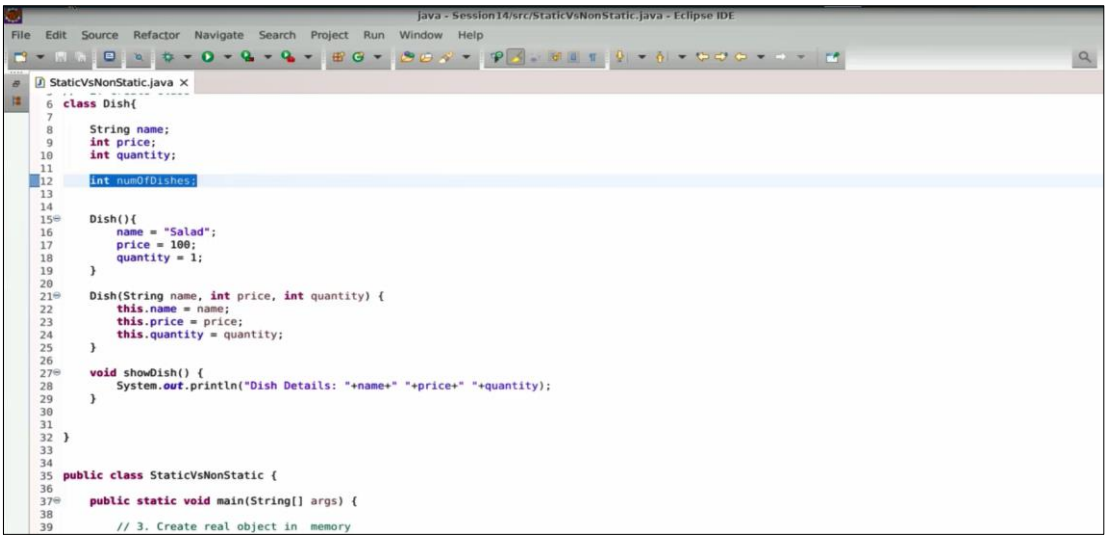


4.2 Run this code and it says that the dish details are **salad 100 1** and the other one has **pizza 300 2**

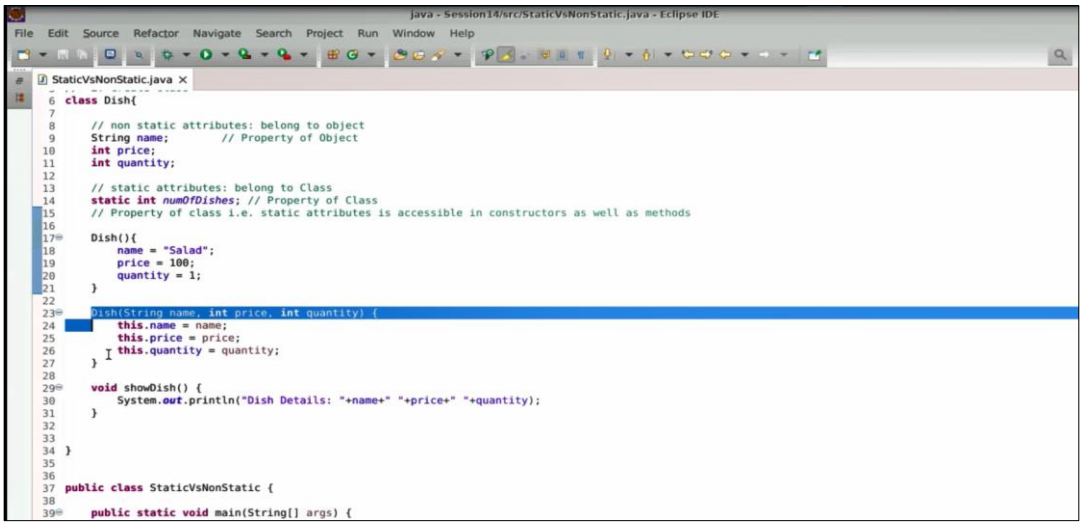


Step 5: Create an attribute and differentiate static and non-static attributes

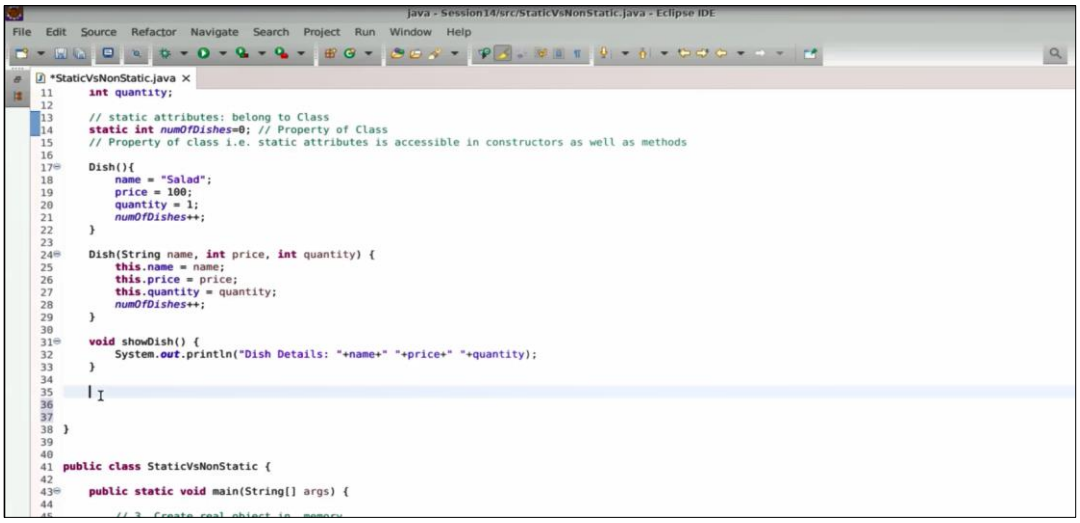
5.1 If you need to know how many dishes or dish objects are in memory, you need a certain mechanism. For that, you will create one more attribute called **int numberOfDishes**. Now, the moment you create an attribute, that attribute becomes the property of the object. It will be added to both objects, and every object will have its own copy. These are called instance variables for the same reason



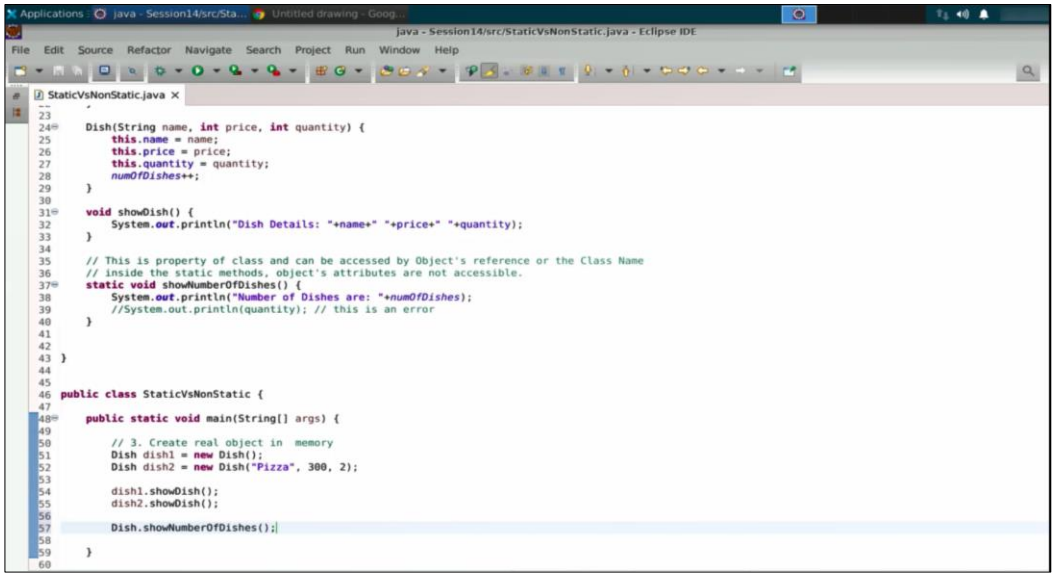
5.2 The moment you make this attribute static, it is known as a property of the class. In contrast, these three attributes are referred to as properties of the object. Non-static attributes belong to the object, whereas static attributes belong to the class. In the RAM area, a storage container is created and referred to as a class; this will hold the number of dishes



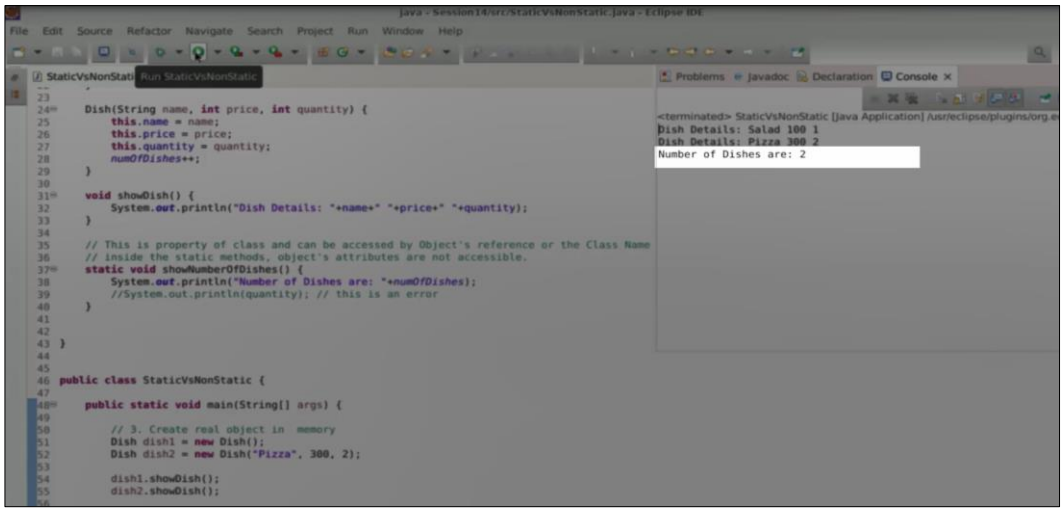
5.3 Now increment the number of dishes with **++**



5.4 You can even write a static method. Let us say **showNumberOfDishes** and add **numberOfDishes** is plus the **numberOfDishes**. This is a property of the class and can be accessed by object reference or the class name. Inside the static method, object attributes are not accessible. If you try to print the quantity here, it will result in an error. Now you can use the class name to execute this method called **showNumberOfDishes**

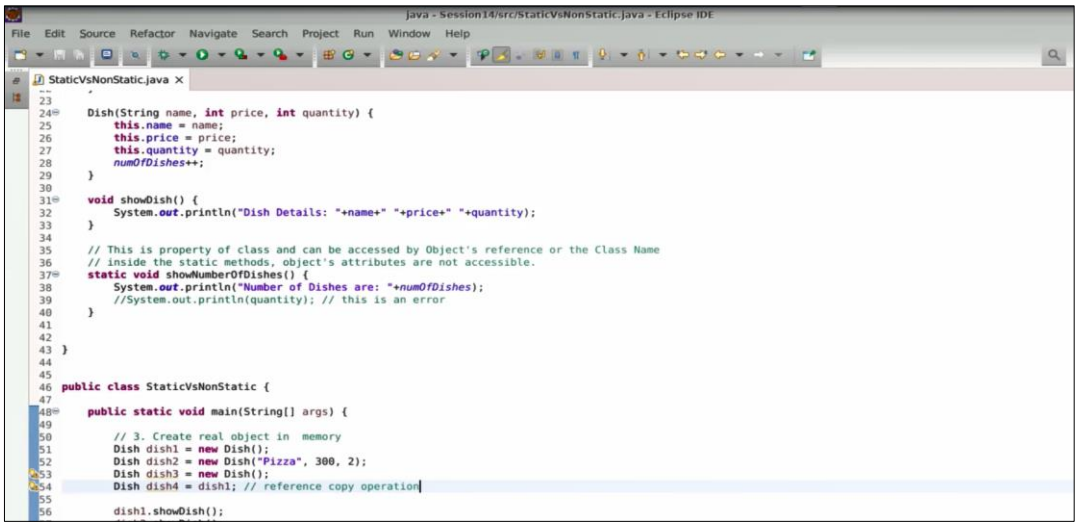


5.5 When you run the program it says that the number of dishes is two

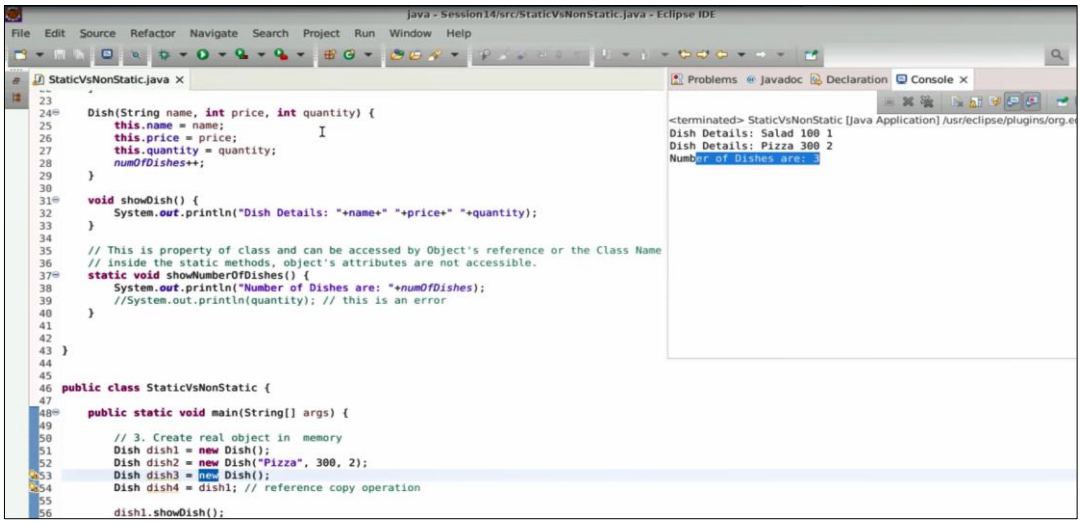


Step 6: Create more objects, use a reference copy, and execute the code

6.1 Create a few more dish objects. **dish3** is a new dish, and you will use a reference copy where you will type a dish. **dish4** is just the same as **dish1**. If you recall, this is known as the reference copy operation

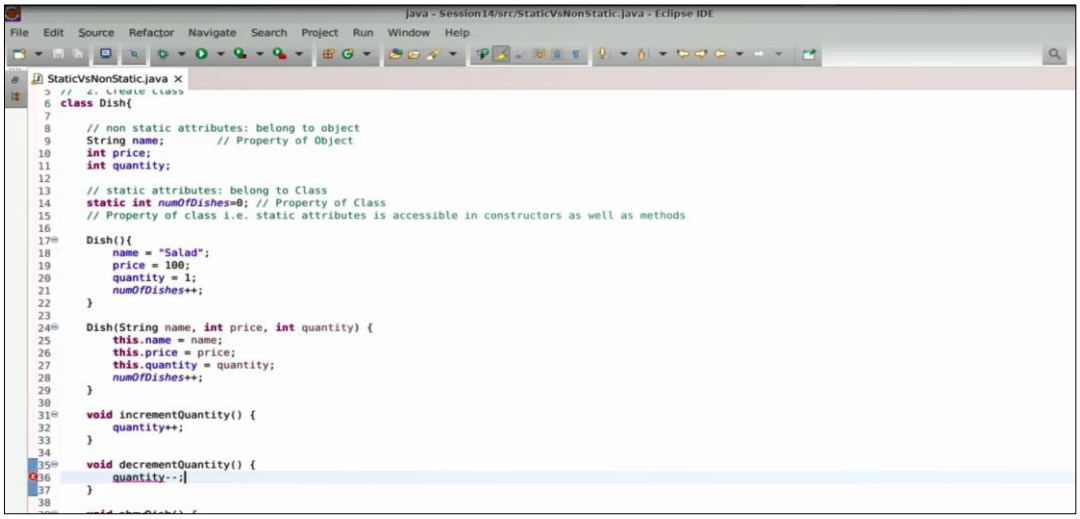


6.2 Run the program now, you get to see that there are three dishes

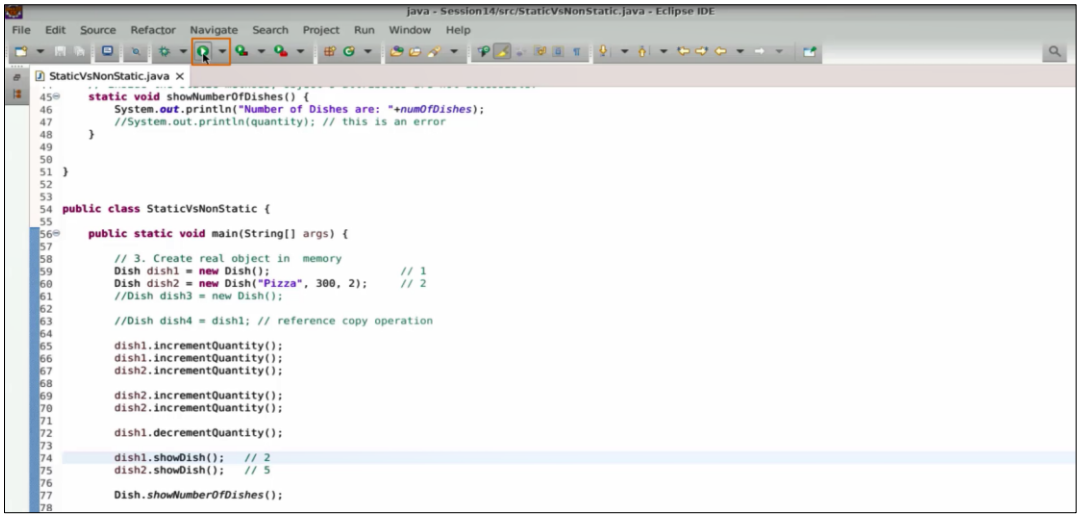


Step 7: Create methods for increment and decrement

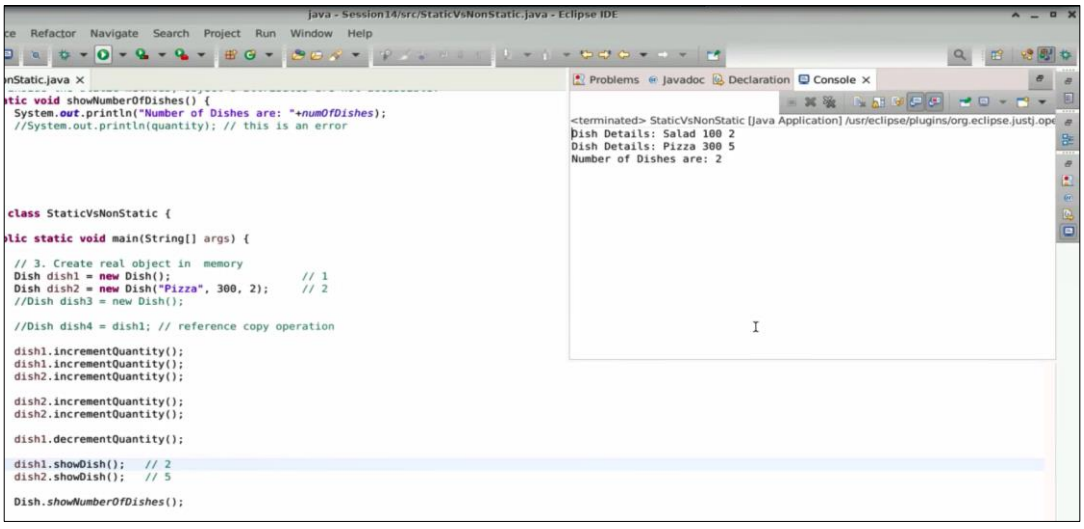
7.1 Let us create a method called increment quantity. It will do the job of incrementing the quantity by 1. In the same way, you have something known as decrement the quantity. The main role of this is to decrease the quantity by one



7.2 Let us first comment on these two. You are going to add **dish1.incrementQuantity()**, **dish1.incrementQuantity()**, and then you will add **dish2.incrementQuantity()**. Add **dish2.incrementQuantity()** again, **dish2.incrementQuantity()** once more, and then you will add **dish1.decrementQuantity()**. Finally, call **showDish()** to see the final output

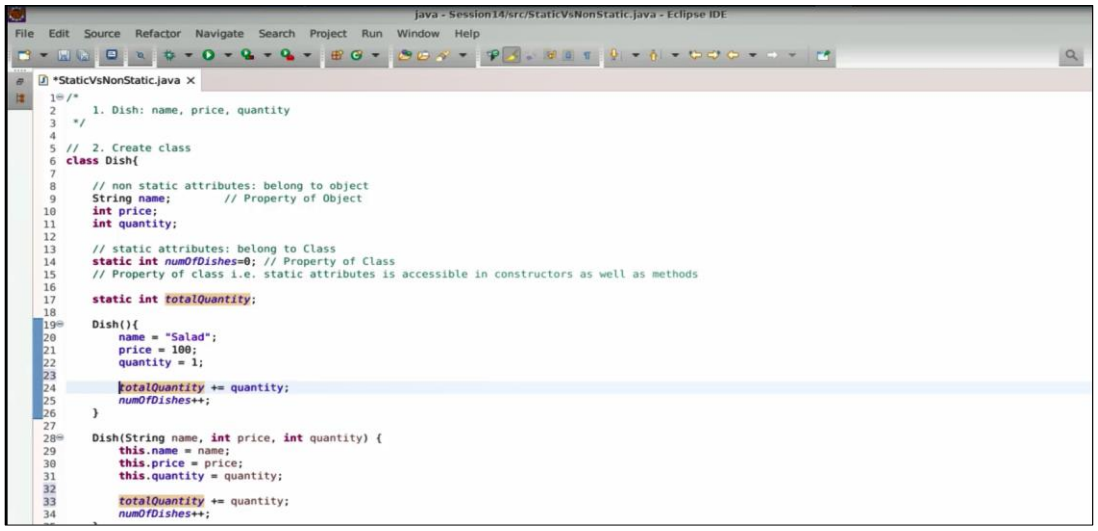


7.3 Now run this code. You can see the quantity for **dish1** is **2** and **dish2** is **5**

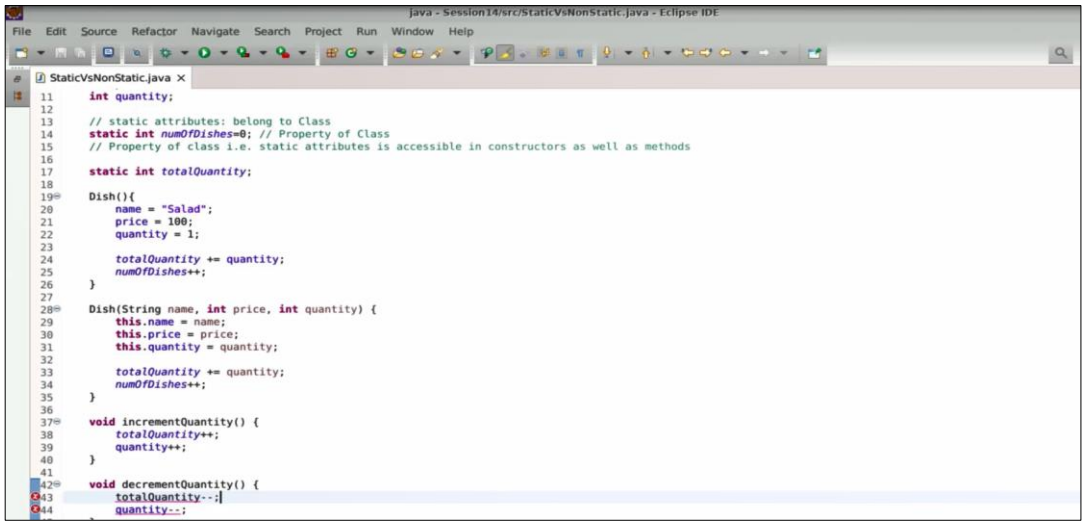


Step 8: Create a static variable and execute the code

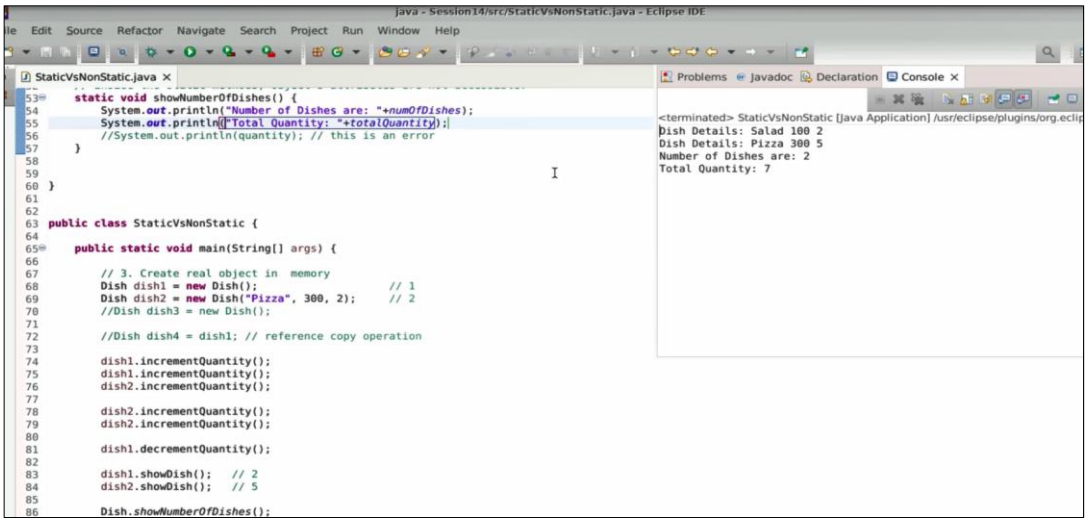
8.1 The other way is to create a static variable. Type **int totalQuantity** and initialize it to zero. Even if you do not initialize it, it is by default 0. Use this basic property where, when you set the quantity to 1, **totalQuantity** will be incremented by this value of quantity. Similarly, you will add to the quantity from the parameterized constructor as well



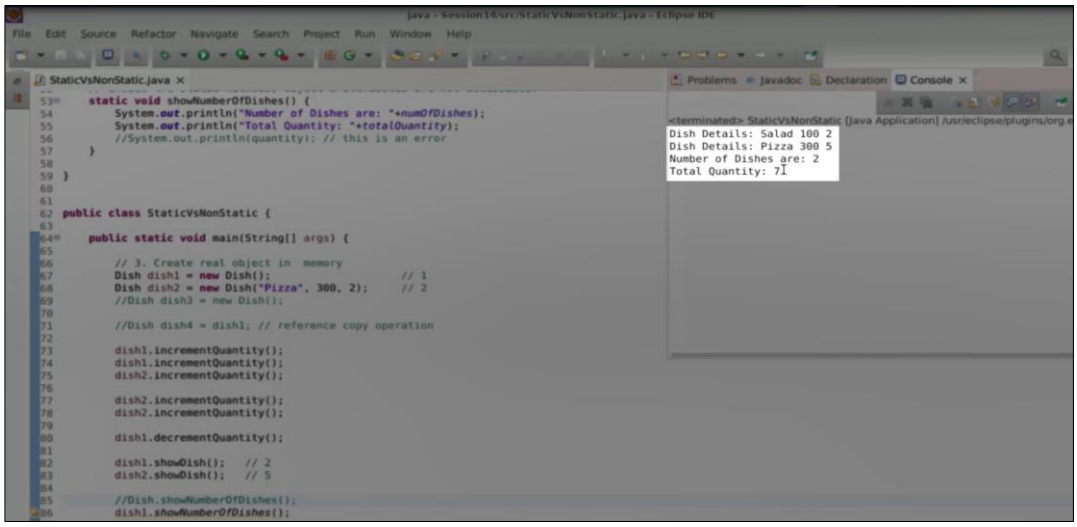
8.2 When you increment the quantity and when you decrement, the quantity will also increment and decrement our total quantity. Total quantity increment by one and here adds total quantity decrement by 1



8.3 When this program finishes, in the number of dishes, you are also going to display **totalQuantity**, and this will come up as **totalQuantity**. When you run the program, it shows the total quantity as 7, which is 5 plus 2. Hence, the use of static variables can be very meaningful, and you can use these variables as common variables



8.4 You can even add **dish1.showNumberOfDishes**, and it would not have any impact other than a warning. If you use it like this, it will give you the same output: "**Dishes are 2, quantities 7**". Even if you execute it with 2, it will give you the same output because this is not a method of the object; it's a property of the class



By following the above steps, you have successfully compared the creation and working of Static and Non-Static Methods in OOPs.