

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

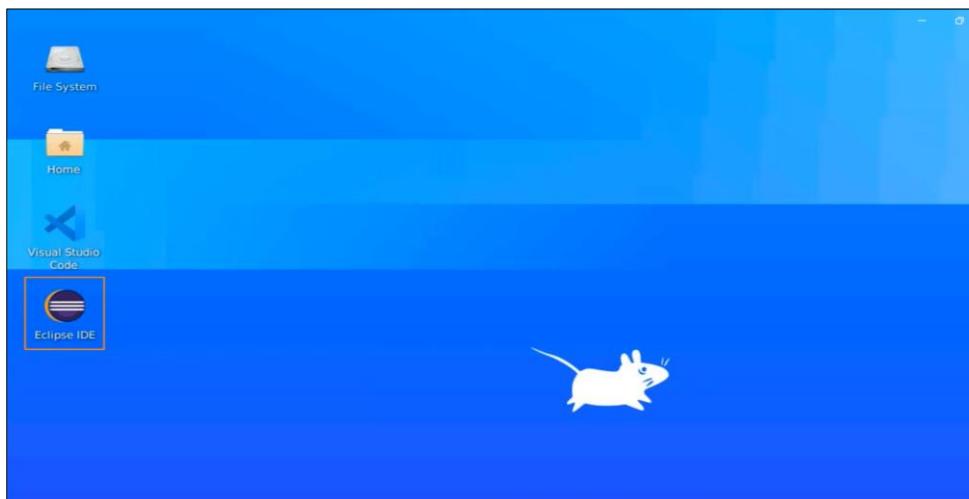
Prerequisites: None

Steps to be followed:

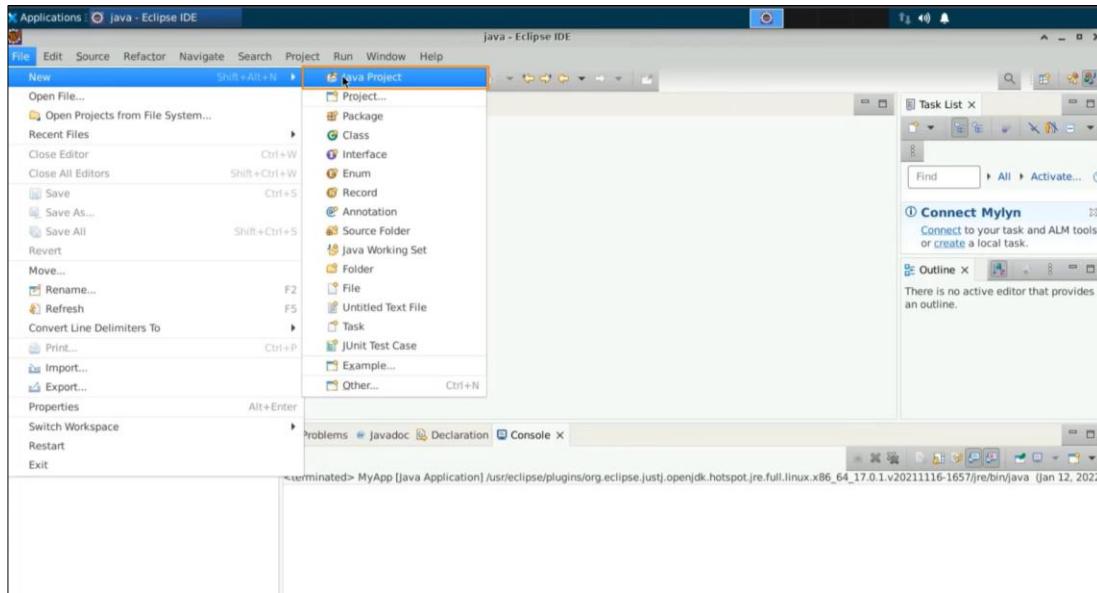
1. Open Eclipse IDE, and open a new Java project and a class
2. Create a default constructor
3. Write a function to update data in the object
4. Create real objects in memory and execute the code
5. Create an attribute and differentiate static and non-static attributes
6. Create more objects, use a reference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

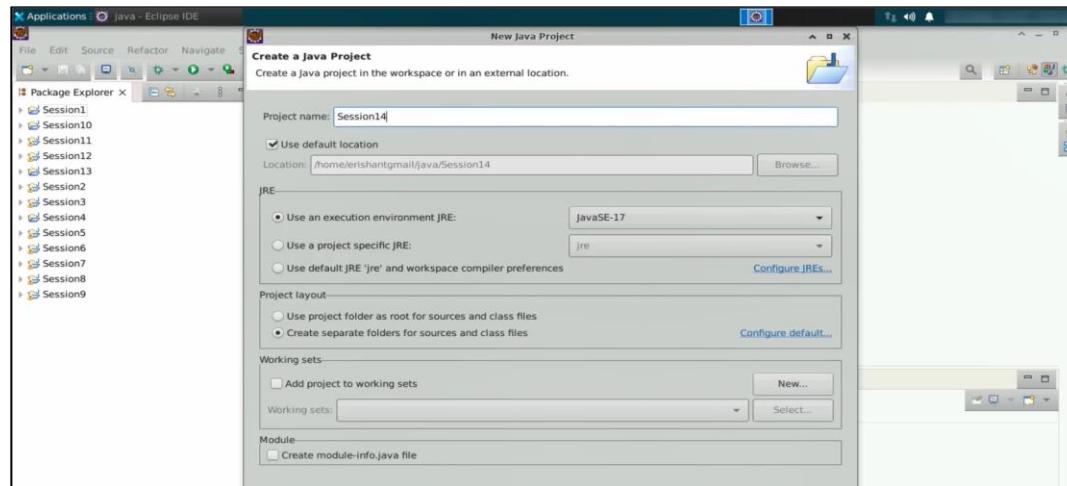
1.1 Open the Eclipse IDE



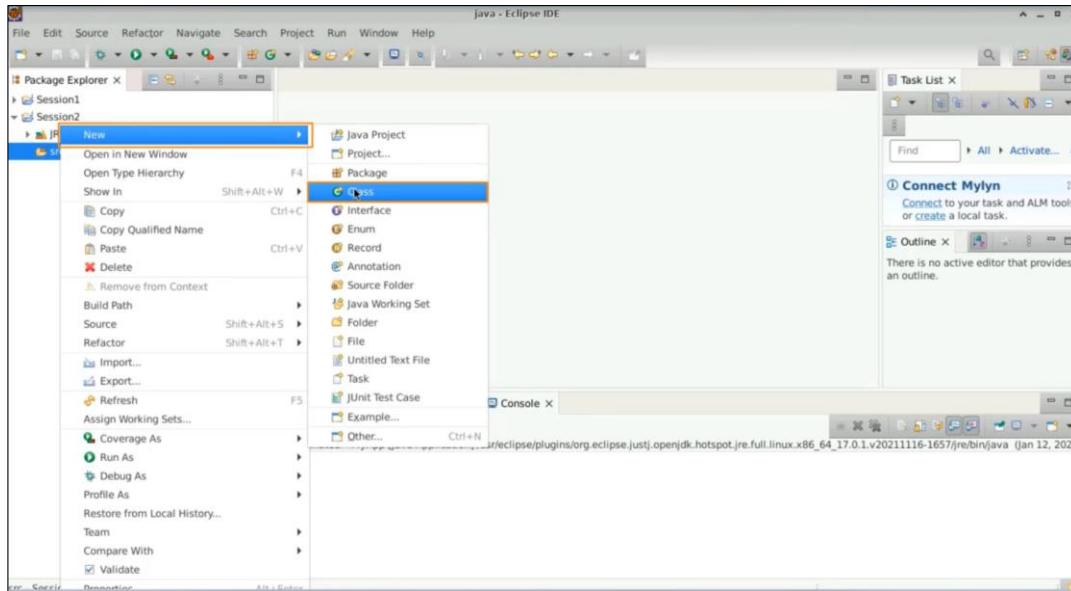
1.2. Select File, then New, and then Java project



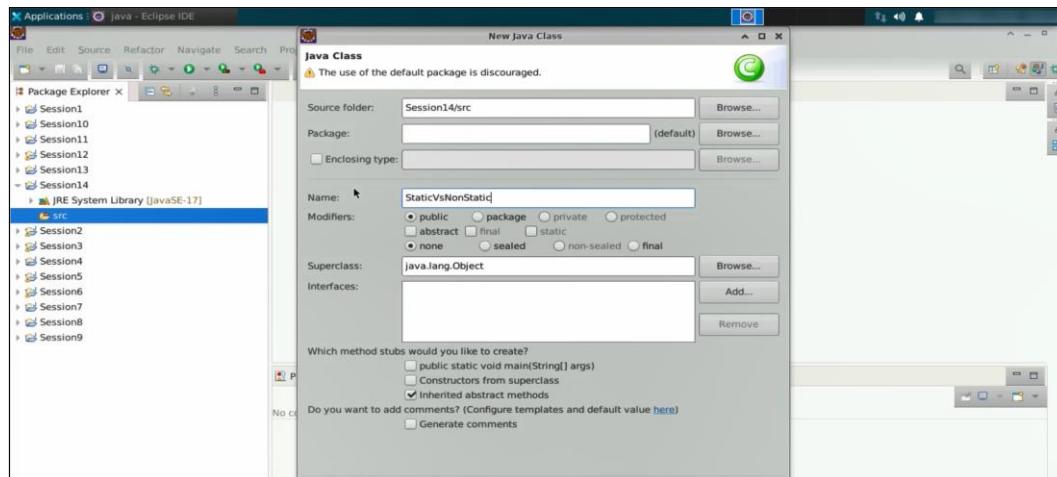
1.3 Name the project “Session14”, uncheck “Create a module info dot Java file”, and press Finish



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**

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

* * * * * StaticVsNonStatic.java X
1 /**
2  * 1. Dish: name, price, quantity
3 */
4
5 // 2. Create class
6 class Dish{
7
8     String name;
9     int price;
10    int quantity;
11
12}
13
14
15
16
17 public class StaticVsNonStatic {
18
19    public static void main(String[] args) {
20
21
22    }
23
24}
25

```

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

```

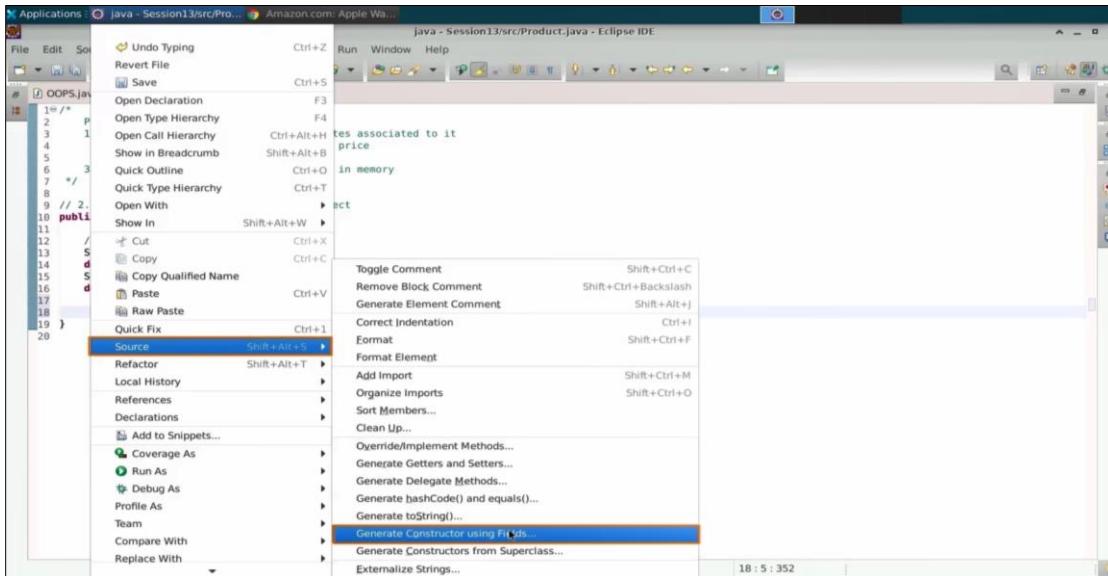
java - Session14/src/StaticVsNonStatic.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

* * * * * StaticVsNonStatic.java X
1 /**
2  * 1. Dish: name, price, quantity
3 */
4
5 // 2. Create class
6 class Dish{
7
8     String name;
9     int price;
10    int quantity;
11
12    Dish(){
13        name = "Salad";
14        price = 100;
15        quantity = 1;
16    }
17
18}
19
20
21 public class StaticVsNonStatic {
22
23    public static void main(String[] args) {
24
25
26    }
27
28}
29

```

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

```

1 /**
2  * 1. Dish: name, price, quantity
3 */
4
5 // 2. Create class
6 class Dish{
7
8     String name;
9     int price;
10    int quantity;
11
12    Dish(){
13        name = "Salad";
14        price = 100;
15        quantity = 1;
16    }
17
18    Dish(String name, int price, int quantity) {
19        this.name = name;
20        this.price = price;
21        this.quantity = quantity;
22    }
23
24    void showDish() {
25        System.out.println("Dish Details: "+name+" "+price+" "+quantity);
26    }
27
28 }
29
30
31
32 public class StaticVsNonStatic {
33

```

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

```

Java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
# StaticVsNonStatic.java X
#   class Dish{
#     String name;
#     int price;
#     int quantity;
#   }
#   Dish(){
#     name = "Salad";
#     price = 100;
#     quantity = 1;
#   }
#   Dish(String name, int price, int quantity) {
#     this.name = name;
#     this.price = price;
#     this.quantity = quantity;
#   }
#   void showDish() {
#     System.out.println("Dish Details: "+name+" "+price+" "+quantity);
#   }
# }
# public class StaticVsNonStatic {
#   public static void main(String[] args) {
#     // 3. Create real object in memory
#     Dish dish1 = new Dish();
#     Dish dish2 = new Dish("Pizza", 300, 2);
#   }
# }

```

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

```

NonStatic.java X
Dish{
String name;
int price;
int quantity;
}
Dish(){
  name = "Salad";
  price = 100;
  quantity = 1;
}
Dish(String name, int price, int quantity) {
  this.name = name;
  this.price = price;
  this.quantity = quantity;
}
void showDish() {
  System.out.println("Dish Details: "+name+" "+price+" "+quantity);
}

public class StaticVsNonStatic {
  public static void main(String[] args) {
    // 3. Create real object in memory
    Dish dish1 = new Dish();
    Dish dish2 = new Dish("Pizza", 300, 2);
  }
}

```

Console X

```

<terminated> StaticVsNonStatic [Java Application] /usr/eclipse/plugins/org.eclipse.justj.opc.vfs/StaticVsNonStatic.jar
Dish Details: Salad 100 1
Dish Details: 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

```

Java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
1 package com.simplilearn;
2
3 public class StaticVsNonStatic {
4     public static void main(String[] args) {
5         // 3. Create real object in memory
6     }
7 }
8
9 class Dish {
10     String name;
11     int price;
12     int quantity;
13
14     Dish() {
15         name = "Salad";
16         price = 100;
17         quantity = 1;
18     }
19
20     Dish(String name, int price, int quantity) {
21         this.name = name;
22         this.price = price;
23         this.quantity = quantity;
24     }
25
26     void showDish() {
27         System.out.println("Dish Details: "+name+" "+price+" "+quantity);
28     }
29
30 }
31
32 }
33
34
35 public class StaticVsNonStatic {
36
37     public static void main(String[] args) {
38         // 3. Create real object in memory
39     }
40 }

```

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

```

Java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
1 package com.simplilearn;
2
3 public class StaticVsNonStatic {
4     public static void main(String[] args) {
5         // 3. Create real object in memory
6     }
7 }
8
9 class Dish {
10     // non static attributes: belong to object
11     String name; // Property of Object
12     int price;
13     int quantity;
14
15     // static attributes: belong to Class
16     static int numberOfDishes; // Property of Class
17     // Property of class i.e. static attributes is accessible in constructors as well as methods
18
19     Dish() {
20         name = "Salad";
21         price = 100;
22         quantity = 1;
23     }
24
25     Dish(String name, int price, int quantity) {
26         this.name = name;
27         this.price = price;
28         this.quantity = quantity;
29     }
30
31     void showDish() {
32         System.out.println("Dish Details: "+name+" "+price+" "+quantity);
33     }
34
35 }
36
37 public class StaticVsNonStatic {
38
39     public static void main(String[] args) {
40         // 3. Create real object in memory
41     }
42 }

```

5.3 Now increment the number of dishes with ++

```

11     int quantity;
12
13     // static attributes: belong to Class
14     static int numDishes=0; // Property of Class
15     // Property of class i.e. static attributes is accessible in constructors as well as methods
16
17     Dish(){
18         name = "Salad";
19         price = 100;
20         quantity = 1;
21         numDishes++;
22     }
23
24     Dish(String name, int price, int quantity) {
25         this.name = name;
26         this.price = price;
27         this.quantity = quantity;
28         numDishes++;
29     }
30
31     void showDish() {
32         System.out.println("Dish Details: "+name+" "+price+" "+quantity);
33     }
34
35
36
37 }
38
39
40
41 public class StaticVsNonStatic {
42
43     public static void main(String[] args) {
44
45         // 3. Create real object in memory
46     }
47 }
```

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**

```

11     int quantity;
12
13     // static attributes: belong to Class
14     static int numDishes=0; // Property of Class
15     // Property of class i.e. static attributes is accessible in constructors as well as methods
16
17     Dish(){
18         name = "Salad";
19         price = 100;
20         quantity = 1;
21         numDishes++;
22     }
23
24     Dish(String name, int price, int quantity) {
25         this.name = name;
26         this.price = price;
27         this.quantity = quantity;
28         numDishes++;
29     }
30
31     void showDish() {
32         System.out.println("Dish Details: "+name+" "+price+" "+quantity);
33     }
34
35     // This is property of class and can be accessed by Object's reference or the Class Name
36     // inside the static methods, object's attributes are not accessible.
37     static void showNumberOfDishes() {
38         System.out.println("Number of Dishes are: "+numDishes);
39         //System.out.println(quantity); // this is an error
40     }
41
42
43 }
44
45
46 public class StaticVsNonStatic {
47
48     public static void main(String[] args) {
49
50         // 3. Create real object in memory
51         Dish dish1 = new Dish();
52         Dish dish2 = new Dish("Pizza", 300, 2);
53
54         dish1.showDish();
55         dish2.showDish();
56
57         Dish.showNumberOfDishes();
58     }
59 }
60 
```

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

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java
23
24    Dish(String name, int price, int quantity) {
25        this.name = name;
26        this.price = price;
27        this.quantity = quantity;
28        numOfDishes++;
29    }
30
31    void showDish() {
32        System.out.println("Dish Details: "+name+" "+price+" "+quantity);
33    }
34
35    // This is property of class and can be accessed by Object's reference or the Class Name
36    // inside the static methods, object's attributes are not accessible.
37    static void showNumberOfDishes() {
38        System.out.println("Number of Dishes are: "+numOfDishes);
39        //System.out.println(quantity); // this is an error
40    }
41
42
43 }
44
45 public class StaticVsNonStatic {
46
47     public static void main(String[] args) {
48         // 3. Create real object in memory
49         Dish dish1 = new Dish();
50         Dish dish2 = new Dish("Pizza", 300, 2);
51
52         dish1.showDish();
53         dish2.showDish();
54     }
55
56 }

```

<terminated> StaticVsNonStatic [Java Application] /usr/eclipse/plugins/org.eclipsesource.jdt.ls.core/lib/jdtls.jar
Dish Details: Salad 100 1
Dish Details: Pizza 300 2
Number of Dishes are: 2

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

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java
23
24    Dish(String name, int price, int quantity) {
25        this.name = name;
26        this.price = price;
27        this.quantity = quantity;
28        numOfDishes++;
29    }
30
31    void showDish() {
32        System.out.println("Dish Details: "+name+" "+price+" "+quantity);
33    }
34
35    // This is property of class and can be accessed by Object's reference or the Class Name
36    // inside the static methods, object's attributes are not accessible.
37    static void showNumberOfDishes() {
38        System.out.println("Number of Dishes are: "+numOfDishes);
39        //System.out.println(quantity); // this is an error
40    }
41
42
43 }
44
45 public class StaticVsNonStatic {
46
47     public static void main(String[] args) {
48         // 3. Create real object in memory
49         Dish dish1 = new Dish();
50         Dish dish2 = new Dish("Pizza", 300, 2);
51         Dish dish3 = new Dish();
52         Dish dish4 = dish1; // reference copy operation
53
54         dish1.showDish();
55
56     }
57 }

```

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

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

Dish(String name, int price, int quantity) {
    this.name = name;
    this.price = price;
    this.quantity = quantity;
    numDishes++;
}

void showDish() {
    System.out.println("Dish Details: "+name+" "+price+" "+quantity);
}

// This is property of class and can be accessed by Object's reference or the Class Name
// inside the static methods, object's attributes are not accessible.
static void showNumberDishes() {
    System.out.println("Number of Dishes are: "+numDishes);
    //System.out.println(quantity); // this is an error
}

public class StaticVsNonStatic {

    public static void main(String[] args) {
        // 3. Create real object in memory
        Dish dish1 = new Dish();
        Dish dish2 = new Dish("Pizza", 300, 2);
        Dish dish3 = new Dish();
        Dish dish4 = dish1; // reference copy operation
        dish1.showDish();
    }
}

```

<terminated> StaticVsNonStatic [Java Application] /usr/eclipse/plugins/org.eclipsesource.jdt.ls.core/lib/jdtls.jar

Dish Details: Salad 100 1
Dish Details: Pizza 300 2
Number of Dishes are: 3

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

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

// non static attributes: belong to object
String name;           // Property of Object
int price;
int quantity;

// static attributes: belong to Class
static int numDishes=0; // Property of Class
// Property of class i.e. static attributes is accessible in constructors as well as methods

Dish(){
    name = "Salad";
    price = 100;
    quantity = 1;
    numDishes++;
}

Dish(String name, int price, int quantity) {
    this.name = name;
    this.price = price;
    this.quantity = quantity;
    numDishes++;
}

void incrementQuantity() {
    quantity++;
}

void decrementQuantity() {
    quantity--;
}

```

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
45  static void showNumberOfDishes() {
46      System.out.println("Number of Dishes are: "+numOfDishes);
47      //System.out.println(quantity); // this is an error
48  }
49
50
51  }
52
53
54  public class StaticVsNonStatic {
55
56      public static void main(String[] args) {
57
58          // 3. Create real object in memory
59          Dish dish1 = new Dish();           // 1
60          Dish dish2 = new Dish("Pizza", 300, 2);    // 2
61          //Dish dish3 = new Dish();
62
63          //Dish dish4 = dish1; // reference copy operation
64
65          dish1.incrementQuantity();
66          dish1.incrementQuantity();
67          dish2.incrementQuantity();
68
69          dish2.incrementQuantity();
70          dish2.incrementQuantity();
71
72          dish1.decrementQuantity();
73
74          dish1.showDish();    // 2
75          dish2.showDish();    // 5
76
77          Dish.showNumberOfDishes();
78

```

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

```

Java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
45  static void showNumberOfDishes() {
46      System.out.println("Number of Dishes are: "+numOfDishes);
47      //System.out.println(quantity); // this is an error
48  }
49
50
51
52
53
54  public class StaticVsNonStatic {
55
56      public static void main(String[] args) {
57
58          // 3. Create real object in memory
59          Dish dish1 = new Dish();           // 1
60          Dish dish2 = new Dish("Pizza", 300, 2);    // 2
61          //Dish dish3 = new Dish();
62
63          //Dish dish4 = dish1; // reference copy operation
64
65          dish1.incrementQuantity();
66          dish1.incrementQuantity();
67          dish2.incrementQuantity();
68
69          dish2.incrementQuantity();
70          dish2.incrementQuantity();
71
72          dish1.decrementQuantity();
73
74          dish1.showDish();    // 2
75          dish2.showDish();    // 5
76
77          Dish.showNumberOfDishes();
78

```

Console

```

<terminated> StaticVsNonStatic [java Application] /usr/eclipse/plugins/org.eclipse.justj.opc.Dish Details: Salad 100 2
Dish Details: Pizza 300 5
Number of Dishes are: 2

```

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

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
# StaticVsNonStatic.java X
1/*
2 * 1. Dish: name, price, quantity
3 */
4
5 // 2. Create class
6 class Dish{
7
8     // non static attributes: belong to object
9     String name;           // Property of Object
10    int price;
11    int quantity;
12
13    // static attributes: belong to Class
14    static int numDishes=0; // Property of Class
15    // Property of class i.e. static attributes is accessible in constructors as well as methods
16
17    static int totalQuantity;
18
19    Dish(){
20        name = "Salad";
21        price = 100;
22        quantity = 1;
23
24        totalQuantity += quantity;
25        numDishes++;
26    }
27
28    Dish(String name, int price, int quantity) {
29        this.name = name;
30        this.price = price;
31        this.quantity = quantity;
32
33        totalQuantity += quantity;
34        numDishes++;
35    }
36
37    void incrementQuantity() {
38        totalQuantity++;
39        quantity++;
40    }
41
42    void decrementQuantity() {
43        totalQuantity--;
44        quantity--;
45    }
46}

```

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

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
# StaticVsNonStatic.java X
1
2
3    int quantity;
4
5    // static attributes: belong to Class
6    static int numDishes=0; // Property of Class
7    // Property of class i.e. static attributes is accessible in constructors as well as methods
8
9    static int totalQuantity;
10
11    Dish(){
12        name = "Salad";
13        price = 100;
14        quantity = 1;
15
16        totalQuantity += quantity;
17        numDishes++;
18    }
19
20    Dish(String name, int price, int quantity) {
21        this.name = name;
22        this.price = price;
23        this.quantity = quantity;
24
25        totalQuantity += quantity;
26        numDishes++;
27    }
28
29    void incrementQuantity() {
30        totalQuantity++;
31        quantity++;
32    }
33
34    void decrementQuantity() {
35        totalQuantity--;
36        quantity--;
37    }
38}

```

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

The screenshot shows the Eclipse IDE interface with the code editor and console view. The code in the editor is:

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
53  static void showNumberOfDishes() {
54      System.out.println("Number of Dishes are: "+numOfDishes);
55      System.out.println("Total Quantity: "+totalQuantity);
56      //System.out.println(quantity); // this is an error
57  }
58
59
60 }
61
62
63 public class StaticVsNonStatic {
64     public static void main(String[] args) {
65         // 3. Create real object in memory
66         Dish dish1 = new Dish();           // 1
67         Dish dish2 = new Dish("Pizza", 300, 2); // 2
68         //Dish dish3 = new Dish();
69
70         //Dish dish4 = dish1; // reference copy operation
71
72         dish1.incrementQuantity();
73         dish1.incrementQuantity();
74         dish2.incrementQuantity();
75         dish2.incrementQuantity();
76
77         dish2.incrementQuantity();
78         dish2.incrementQuantity();
79
80         dish1.decrementQuantity();
81
82         dish1.showDish(); // 2
83         dish2.showDish(); // 5
84
85         Dish.showNumberOfDishes();
86
87     }
88
89 }
90
91
92
93
94
95
96
97
98
99
99 }

```

The console output shows:

```

<terminated> StaticVsNonStatic [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core.prefs
Dish Details: Salad 100 2
Dish Details: Pizza 300 5
Number of Dishes are: 2
Total Quantity: 7

```

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

The screenshot shows the Eclipse IDE interface with the code editor and console view. The code in the editor is identical to the previous one, except for the addition of a warning message in the console:

```

java - Session14/src/StaticVsNonStatic.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StaticVsNonStatic.java X
53  static void showNumberOfDishes() {
54      System.out.println("Number of Dishes are: "+numOfDishes);
55      System.out.println("Total Quantity: "+totalQuantity);
56      //System.out.println(quantity); // this is an error
57  }
58
59
60 }
61
62
63 public class StaticVsNonStatic {
64     public static void main(String[] args) {
65         // 3. Create real object in memory
66         Dish dish1 = new Dish();           // 1
67         Dish dish2 = new Dish("Pizza", 300, 2); // 2
68         //Dish dish3 = new Dish();
69
70         //Dish dish4 = dish1; // reference copy operation
71
72         dish1.incrementQuantity();
73         dish1.incrementQuantity();
74         dish2.incrementQuantity();
75         dish2.incrementQuantity();
76
77         dish2.incrementQuantity();
78         dish2.incrementQuantity();
79
80         dish1.decrementQuantity();
81
82         dish1.showDish(); // 2
83         dish2.showDish(); // 5
84
85         Dish.showNumberOfDishes();
86         dish1.showNumberOfDishes(); // warning
87
88     }
89
90 }
91
92
93
94
95
96
97
98
99
99 }

```

The console output shows:

```

<terminated> StaticVsNonStatic [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core.prefs
Dish Details: Salad 100 2
Dish Details: Pizza 300 5
Number of Dishes are: 2
Total Quantity: 7

```

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