

Lesson 03 Demo 01

Designing the User and Product Objects in OOPs

Objective: Implementation of designing the user and product object in object-oriented programming

Tools required: Eclipse IDE

Prerequisites: None

Steps to be followed:

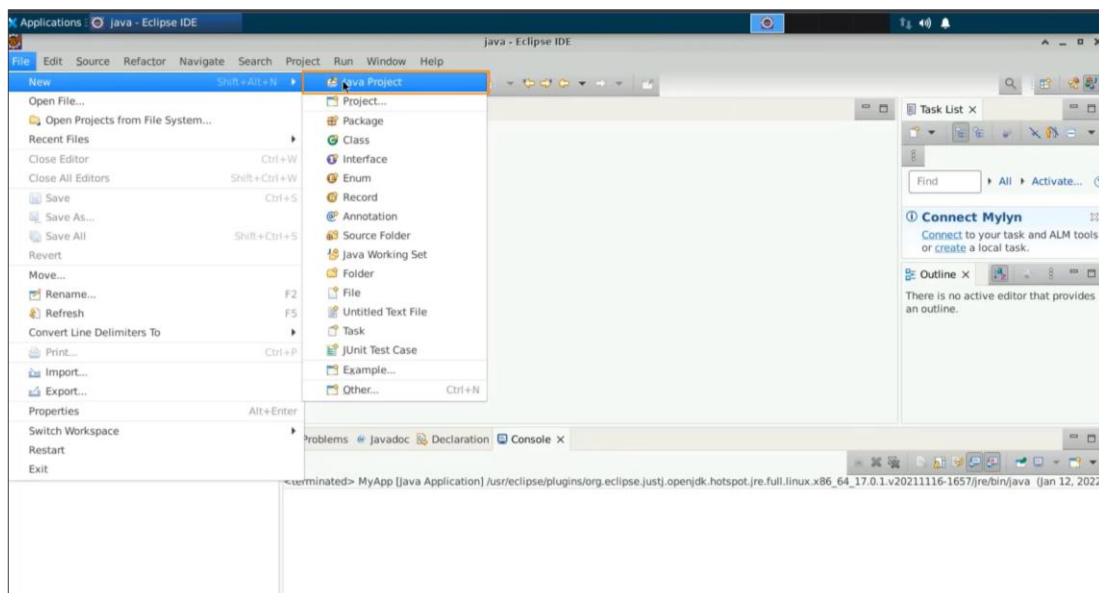
1. Open the IDE and Use a one-dimensional array with suitable examples
2. Create a new class called OOPS
3. Use the sample e-store application to add objects
4. Identify the data in the attributes
5. Create a real object in the memory
6. Assign hash codes to the objects
7. Implement a read operation on the user object
8. Perform operations on the object
9. Access the attribute and print the details
10. Run the code and populate the data
11. Select the default constructor from the source, then generate the constructor with fields
12. Create an array with a specific size
13. Implement a two-dimensional array with suitable examples

Step 1: Open the IDE and use a one-dimensional array with suitable examples

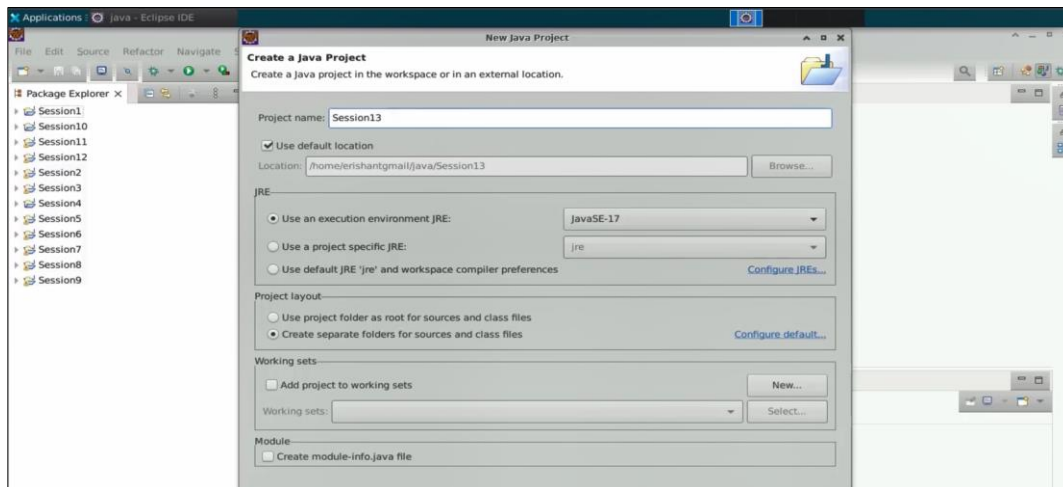
1.1 Open the Eclipse IDE



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

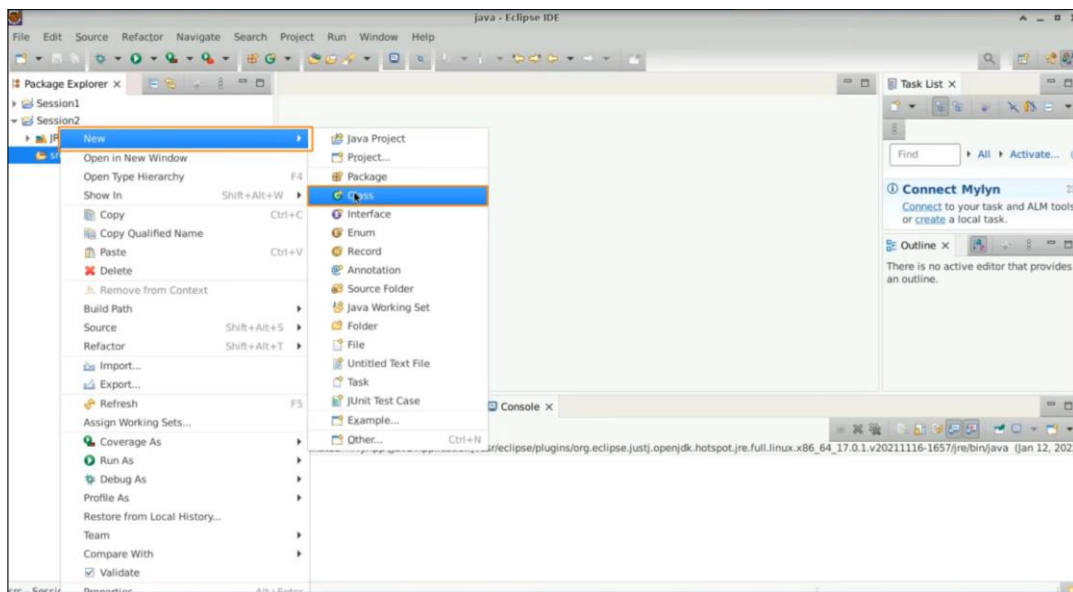


1.3 Name the project **“Session14”**, uncheck **“Create a module info dot Java file”**, and press **Finish**

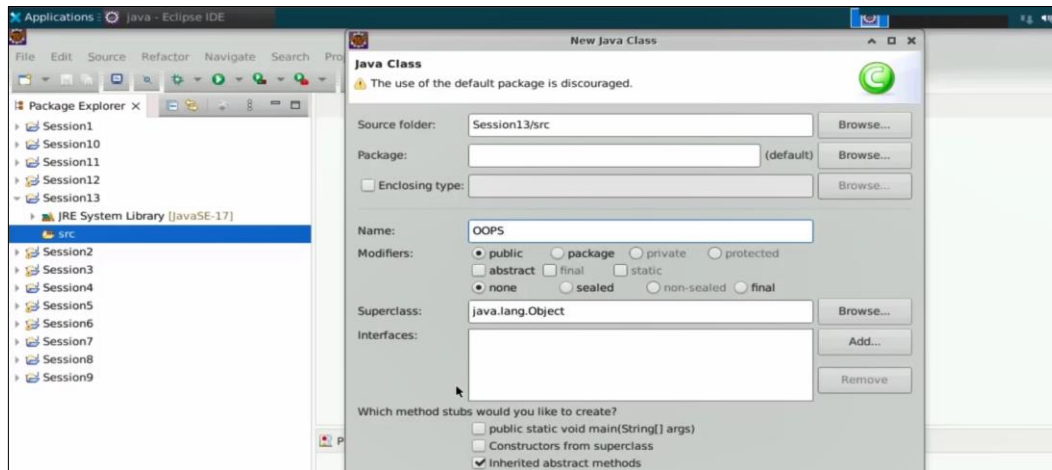


Step 2: Create a new class called OOPS

2.1 With a **Session13** on the src, do a right-click and create a **new class**

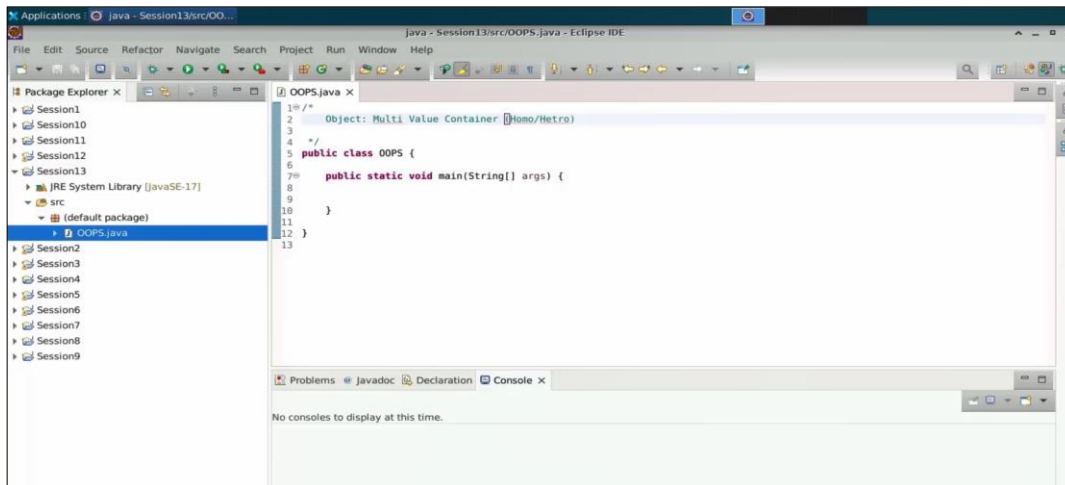


2.2 Name this class as an **OOPS**, then select the **main method**, and then select **finish**

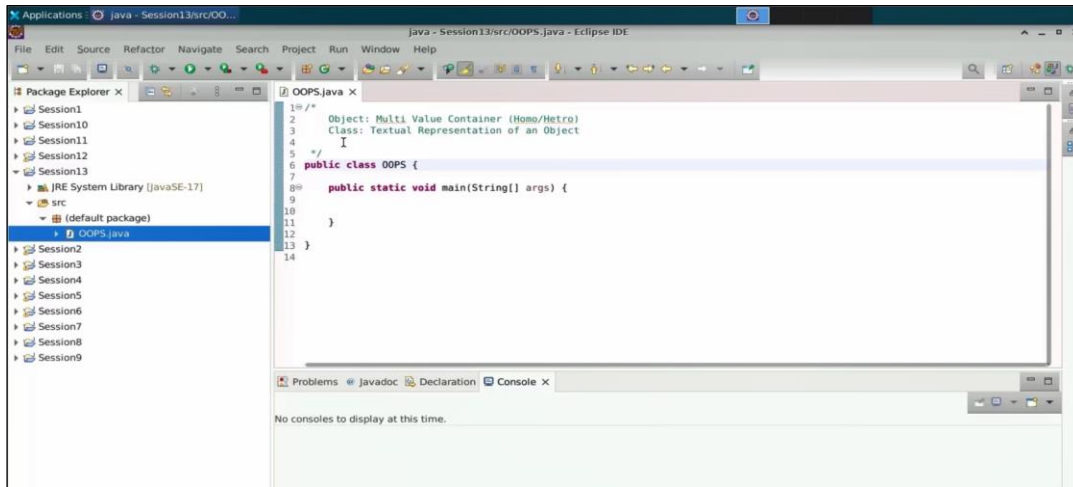


Step 3: Use the sample e-store application to add objects

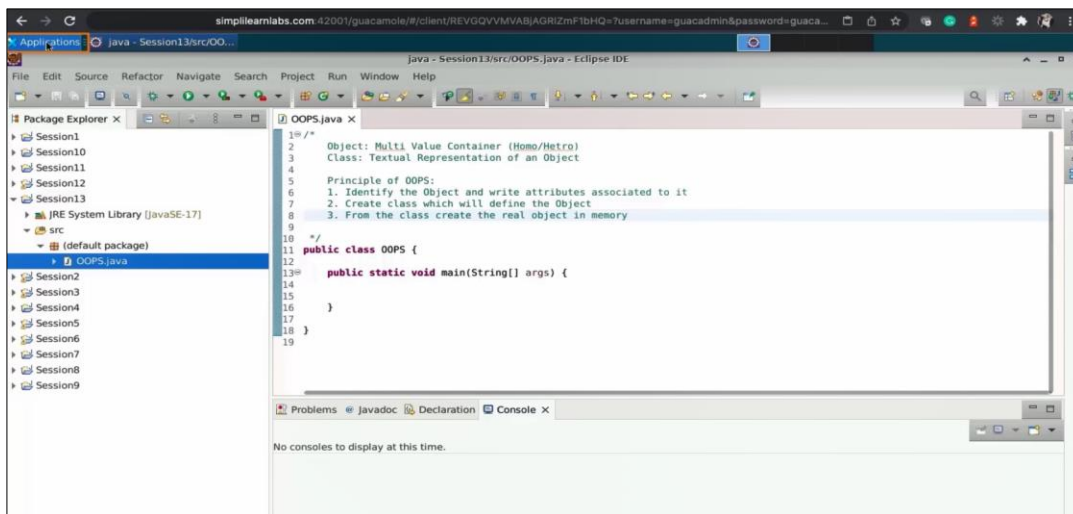
3.1 In order to understand object-oriented programming structure, there are two important key concepts linked to it. The fundamental thing here is an object, which is a multi-value container which can hold a lot of data. It can be homogeneous or it can be heterogeneous, which means it can store the data of the same type or of different type. Hence, the object is a user defined data type



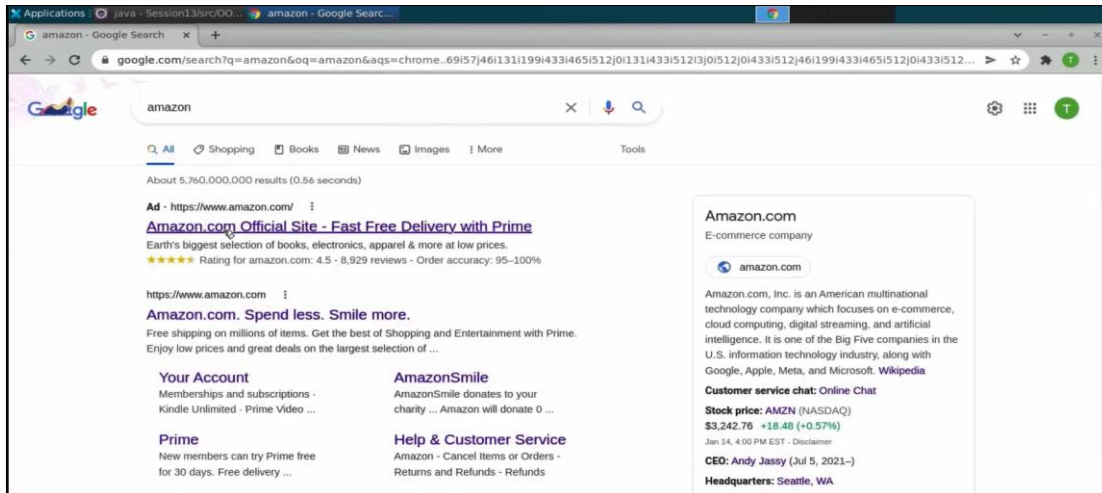
3.2 As a developer, you can write a class that basically is a textual representation of an object. It can also be defined as a blueprint that defines the structure of an object



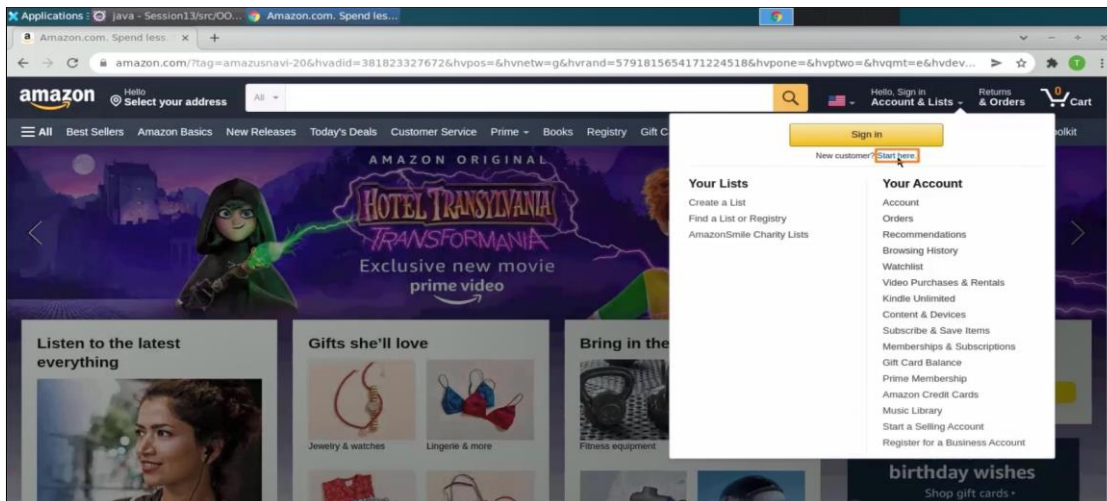
3.3 For the object, let us understand the principle of OOPS. First thing is to identify the object and write the attributes associated with it. Number two, create a class that will define the object, and number three from the class create the real object in the memory



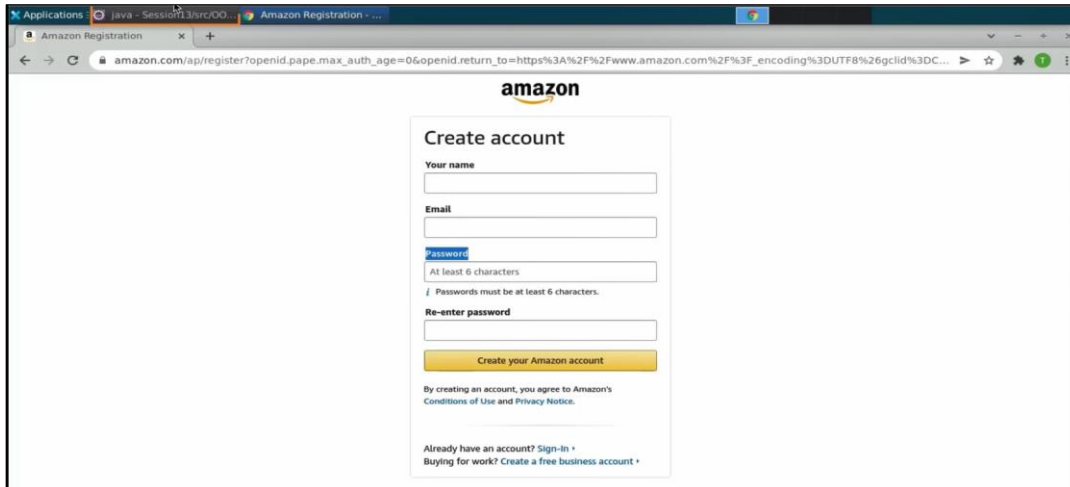
3.4 Let us open the sample e-store application and open the browser window and search for Amazon and open Amazon.com



3.5 Whenever you open up any e-commerce platform, there is an option called to sign in or an option for new customers, start here

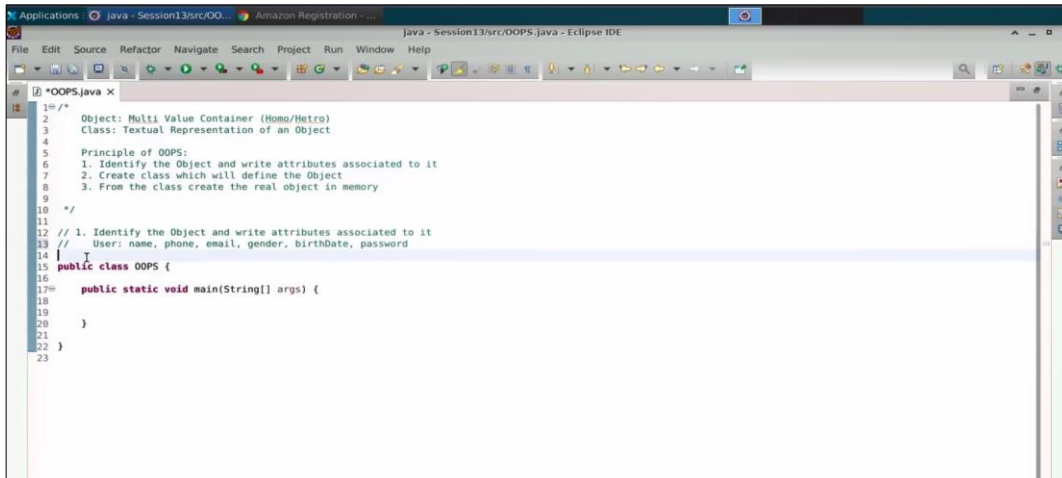


3.6 When you click on start here, what you observe is that in order to create a user account, the user has to have a name, email, and password associated with the account. Hence, the user is an object, where the user would have a name, email, and password and that is the identification for your object



The screenshot shows the Amazon website's registration page. The title is 'Create account'. It contains four input fields: 'Your name', 'Email', 'Password', and 'Re-enter password'. The 'Password' field has a hint 'At least 6 characters' and a note 'Passwords must be at least 6 characters.' Below the fields is a yellow button labeled 'Create your Amazon account'. At the bottom, there are links for 'Already have an account? Sign-in' and 'Buying for work? Create a free business account'.

3.7 Let us use the same use case of Amazon or any other E-commerce platform where you can have the user object. Your first goal is to identify the object. Here the object is identified as a user, hence the user has a name and then adds a few more details like the Phone, email, gender, birth date, and password associated with the user



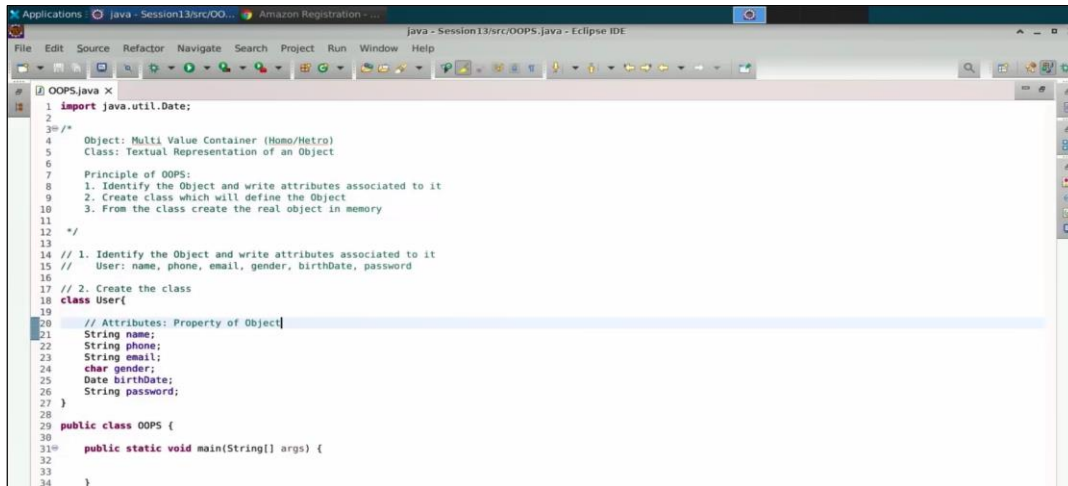
```

1  /**
2   * Object: Multi Value Container (Homo/Hetro)
3   * Class: Textual Representation of an Object
4   *
5   * Principle of OOPS:
6   * 1. Identify the Object and write attributes associated to it
7   * 2. Create class which will define the Object
8   * 3. From the class create the real object in memory
9   *
10  */
11
12  // 1. Identify the Object and write attributes associated to it
13  // User: name, phone, email, gender, birthDate, password
14  public class OOPS {
15      public static void main(String[] args) {
16
17      }
18  }
19
20
21
22
23

```


Step 4: Identify the data in the attributes

4.1 The second step is to create the class. Let us write class user and then start the bracket and close the bracket. Now you need to add the data inside as your attributes. Next, identify the type of these attributes, where the name will be string type, so is your phone number, the email also goes as a string, and for the birthdate, you can use **Java.util.date**. Then for the password, you can take the data type as a string

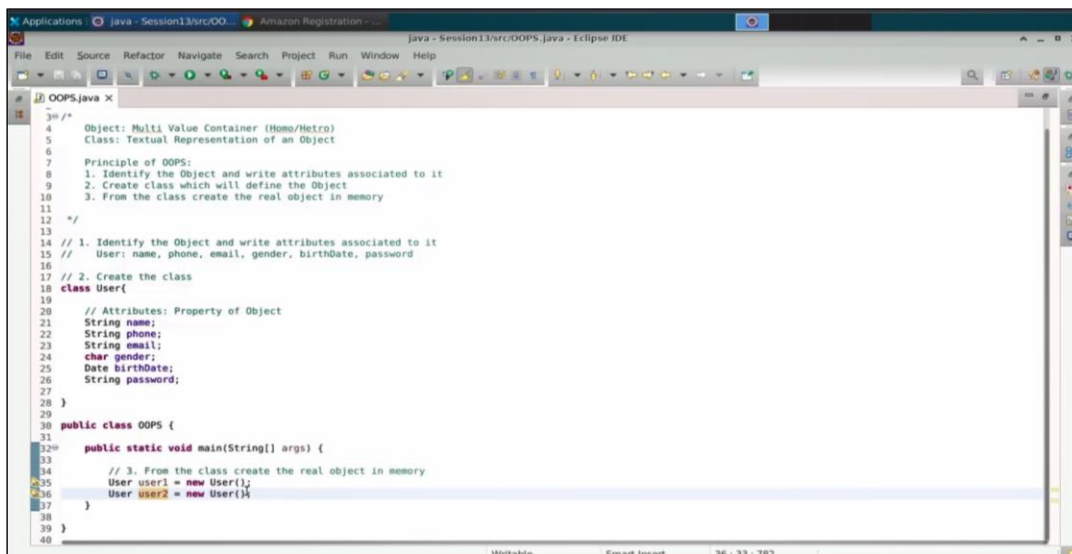


```

1  import java.util.Date;
2
3  /**
4   * Object: Multi Value Container (Homo/Hetro)
5   * Class: Textual Representation of an Object
6   *
7   * Principle of OOPS:
8   * 1. Identify the Object and write attributes associated to it
9   * 2. Create class which will define the Object
10  * 3. From the class create the real object in memory
11  */
12
13
14  // 1. Identify the Object and write attributes associated to it
15  // User: name, phone, email, gender, birthDate, password
16
17  // 2. Create the class
18  class User{
19
20      // Attributes: Property of Object
21      String name;
22      String phone;
23      String email;
24      char gender;
25      Date birthDate;
26      String password;
27  }
28
29  public class OOPS {
30
31      public static void main(String[] args) {
32
33      }
34  }

```

4.2 If you want to associate some values with these attributes. Hence, step number three, where you will create the real object in the memory. Let us use the class name, and then create one of the reference variables. Assign a new operator, then again the class name, and add a parenthesis and a semi-colon. Next, let us write **user 2** as a new user and whenever you will create another object, then **user 2** is the second reference variable.



```

1  import java.util.Date;
2
3  /**
4   * Object: Multi Value Container (Homo/Hetro)
5   * Class: Textual Representation of an Object
6   *
7   * Principle of OOPS:
8   * 1. Identify the Object and write attributes associated to it
9   * 2. Create class which will define the Object
10  * 3. From the class create the real object in memory
11  */
12
13
14  // 1. Identify the Object and write attributes associated to it
15  // User: name, phone, email, gender, birthDate, password
16
17  // 2. Create the class
18  class User{
19
20      // Attributes: Property of Object
21      String name;
22      String phone;
23      String email;
24      char gender;
25      Date birthDate;
26      String password;
27  }
28
29  public class OOPS {
30
31      public static void main(String[] args) {
32
33          // 3. From the class create the real object in memory
34          User user1 = new User();
35          User user2 = new User();
36      }
37  }
38
39
40

```


4.3 Let us print out both these references. Write **System.out.println("User 1 is " + user1);**. In the same way, print out **user2** by writing **System.out.println("User 2 is " + user2);**

```

6
7 Principle of OOPS:
8 1. Identify the Object and write attributes associated to it
9 2. Create class which will define the Object
10 3. From the class create the real object in memory
11
12 */
13
14 // 1. Identify the Object and write attributes associated to it
15 // User: name, phone, email, gender, birthDate, password
16
17 // 2. Create the class
18 class User{
19
20 // Attributes: Property of Object
21 String name;
22 String phone;
23 String email;
24 char gender;
25 Date birthDate;
26 String password;
27
28 }
29
30 public class OOPS {
31
32 public static void main(String[] args) {
33
34 // 3. From the class create the real object in memory
35 User user1 = new User();
36 User user2 = new User();
37
38 System.out.println("user1 is: " + user1);
39 System.out.println("user2 is: " + user2);
40
41 }
42
43 }

```

4.4 Run this program and you will observe that user one is one of the user objects at some hash codes. And user 2 is referring to the other user at a different hash code. Hence, there are two of these objects available in the memory

```

6
7 Principle of OOPS:
8 1. Identify the Object and write attributes associated to it
9 2. Create class which will define the Object
10 3. From the class create the real object in memory
11
12 */
13
14 // 1. Identify the Object and write attributes associated to it
15 // User: name, phone, email, gender, birthDate, password
16
17 // 2. Create the class
18 class User{
19
20 // Attributes: Property of Object
21 String name;
22 String phone;
23 String email;
24 char gender;
25 Date birthDate;
26 String password;
27
28 }
29
30 public class OOPS {
31
32 public static void main(String[] args) {
33
34 // 3. From the class create the real object in memory
35 User user1 = new User();
36 User user2 = new User();
37
38 System.out.println("user1 is: " + user1);
39 System.out.println("user2 is: " + user2);
40
41 }
42
43 }

```

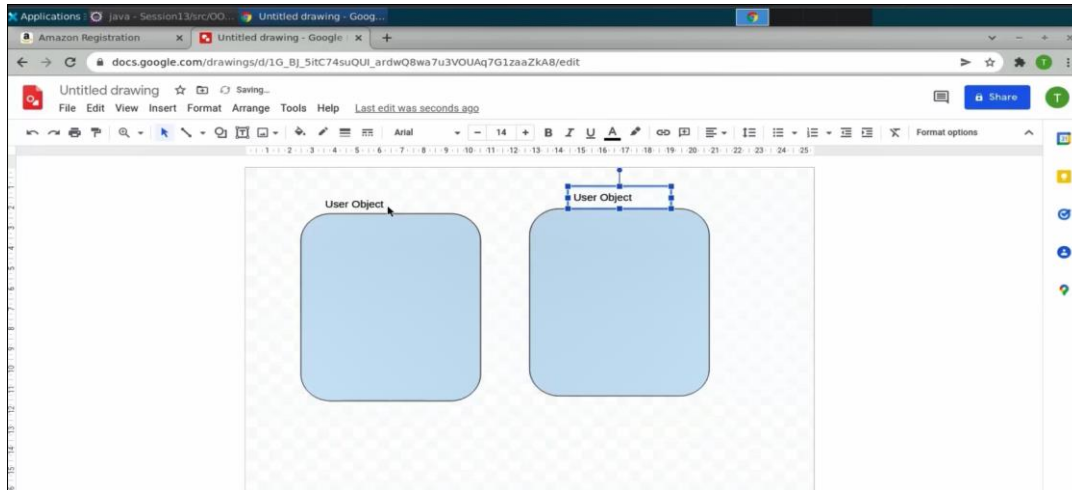
```

<terminated> OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot
user1 is: User@5cfe4b2
user2 is: User@ea30797

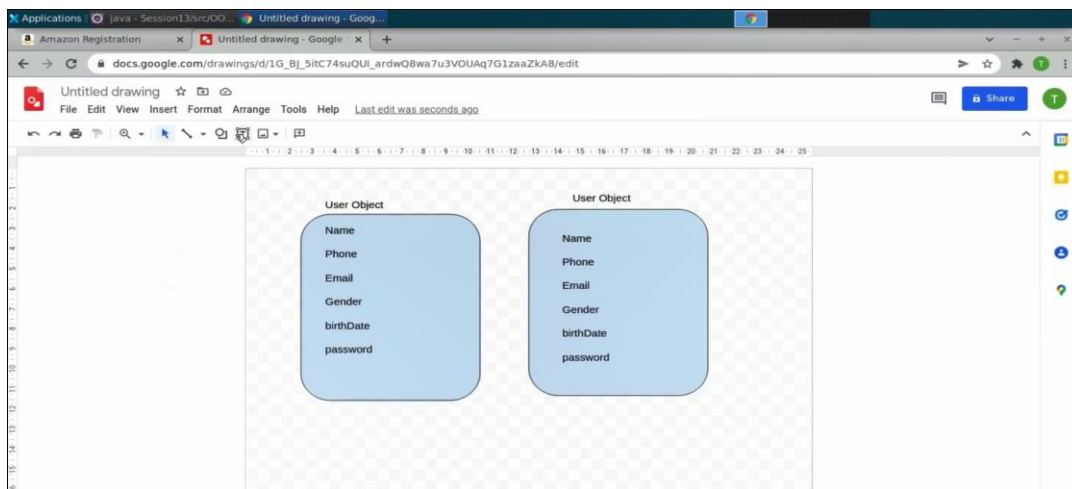
```

Step 5: Create a real object in the memory

5.1 Open Google Draw and draw these two objects. Both objects are named user objects

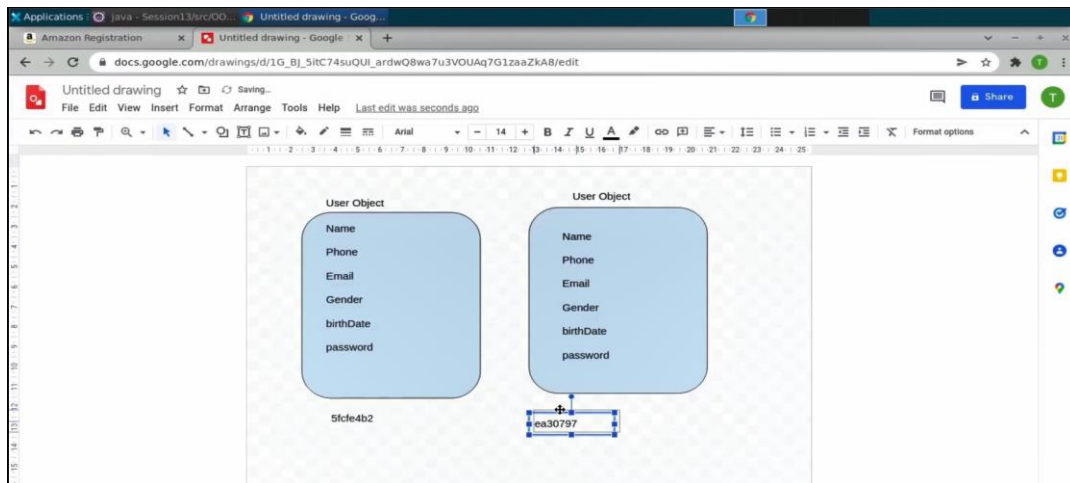


5.2 Inside the object, you will have the attributes which are the name, the phone, email, gender, birth date, and then the password

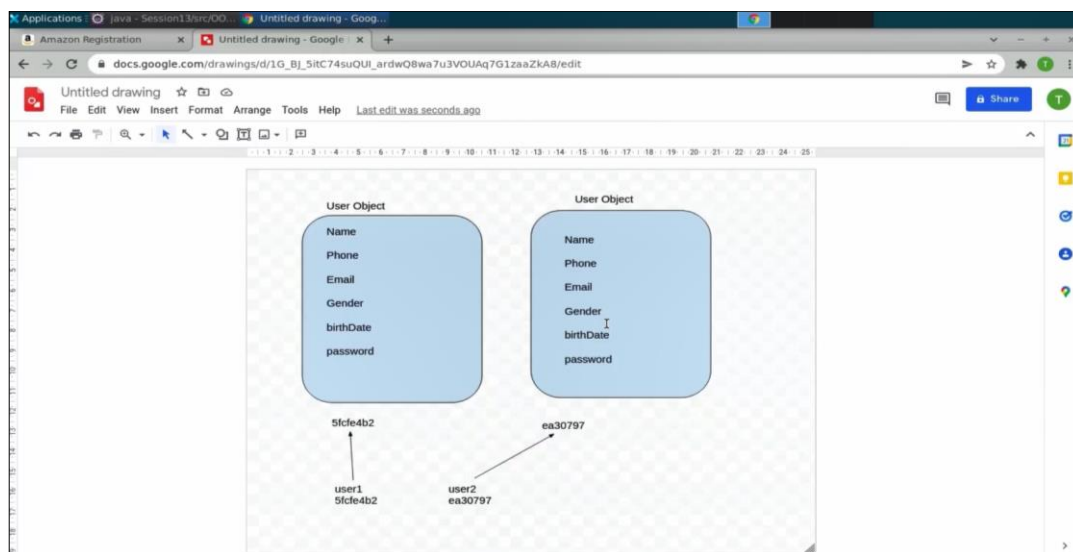


Step 6: Assign hash codes to the objects

- 6.1 As you have created these two objects, copy this first hash code associated with the first object and paste it here and the same way for the second hash code for the second object



- 6.2 User one and user two are the reference variables, and these users as reference variable has these hash codes. Thus, the reference variables point out as these, which means, these reference variables refer to these two objects

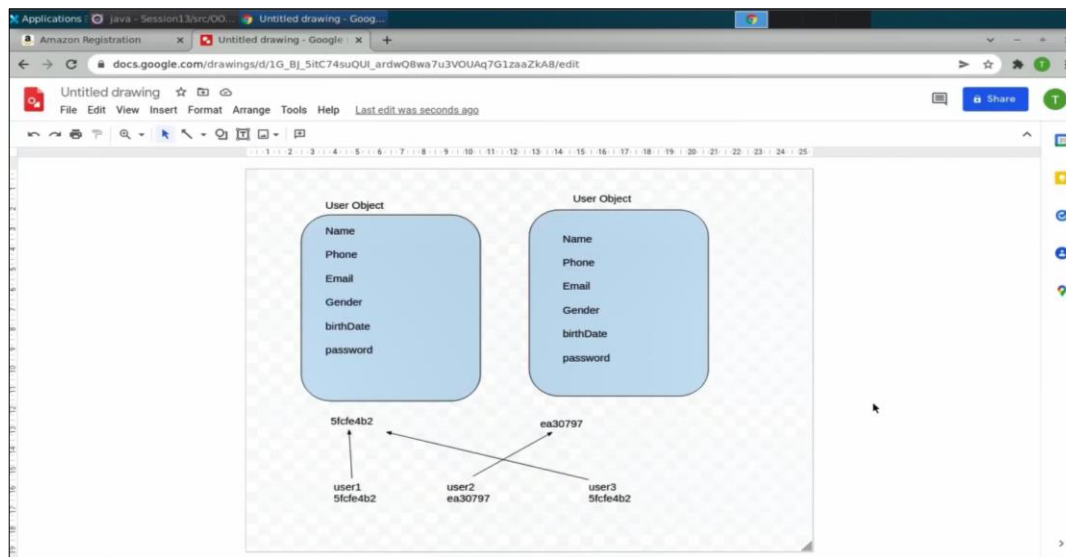


Step 7: Implement a read operation on the user object

7.1 In the next approach, if you want to give a user, user three is using one. What you did here is a copy operation and this is known as a reference copy operation. It means, there are only two objects, **User 3** is not an object. **User 1**, **user 2**, and **user 3** are the reference variables

7.2 Now if you print **user 3**, you will notice that user one and user three have the same hash codes. Every time you run the program, hash codes may be different, so do not get confused on the part that why the hash codes are coming differently now

7.3 **User 1** has the same hash code as **user 3**, hence **user 1** and **user 3** are kind of different references, but pointing to the same object



7.4 You have the object constructed, but the objects do not have data. Hence, let us first try to perform a read operation on the user object. Write `print("Reading data from user1")`, and you can read that with the help of your reference variable. Add a dot operator and access your attributes. For example, **user1.name** can be concatenated with **user1.phone** and **user1.email**. Do the same with the other objects

```

18 class User {
19
20     // Attributes: Property of Object
21     String name;
22     String phone;
23     String email;
24     char gender;
25     Date birthDate;
26     String password;
27 }
28
29
30 public class OOPS {
31
32     public static void main(String[] args) {
33
34         // 3. From the class create the real object in memory
35         User user1 = new User();
36         User user2 = new User();
37
38         User user3 = user1; // Reference Copy Operation
39
40         System.out.println("user1 is: " + user1);
41         System.out.println("user2 is: " + user2);
42         System.out.println("user3 is: " + user3);
43
44         System.out.println("Reading Data from user1");
45         System.out.println(user1.name + " can be called at " + user1.phone + " and can be emailed at " + user1.email);
46
47         System.out.println("Reading Data from user2");
48         System.out.println(user2.name + " can be called at " + user2.phone + " and can be emailed at " + user2.email);
49
50         System.out.println("Reading Data from user3");
51         System.out.println(user3.name + " can be called at " + user3.phone + " and can be emailed at " + user3.email);
52     }
53 }

```

7.5 whenever you execute this program, the data inside the object is coming as null. Hence, It is all empty, which means null can be called at null and can be emailed at null

```

18 class User{
19
20 // Attributes: Property of Object
21 String name;
22 String phone;
23 String email;
24 char gender;
25 Date birthDate;
26 String password;
27
28 }
29
30 public class OOPS {
31
32 public static void main(String[] args) {
33
34 // 3. From the class create the real object in memory
35 User user1 = new User();
36 User user2 = new User();
37
38 User user3 = user1; // Reference Copy Operation
39
40 System.out.println("user1 is: "+user1);
41 System.out.println("user2 is: "+user2);
42 System.out.println("user3 is: "+user3);
43
44 System.out.println("Reading Data from user1");
45 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email);
46
47 System.out.println("Reading Data from user2");
48 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email);
49
50 System.out.println("Reading Data from user3");
51 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email);

```

```

<terminated> OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot
user1 is: User@4c98385c
user2 is: User@73a8dfcc
user3 is: User@4c98385c
Reading Data from user1
null can be called at null and can be emailed at null
Reading Data from user2
null can be called at null and can be emailed at null
Reading Data from user3
null can be called at null and can be emailed at null

```

Step 8: Perform operations on the object

8.1 Moving ahead, you need to perform operations on the object, which means writing or updating the data. The set operation is a combination of write and update, which means it will override the data once you have updated the value

```

18 class User{
19
20 // Attributes: Property of Object
21 String name;
22 String phone;
23 String email;
24 char gender;
25 Date birthDate;
26 String password;
27
28 }
29
30 public class OOPS {
31
32 public static void main(String[] args) {
33
34 // 3. From the class create the real object in memory
35 User user1 = new User();
36 User user2 = new User();
37
38 User user3 = user1; // Reference Copy Operation
39
40 System.out.println("user1 is: "+user1);
41 System.out.println("user2 is: "+user2);
42 System.out.println("user3 is: "+user3);
43
44 // Operations on Object
45 // 1. Write/Update i.e. Set the data
46
47 System.out.println("Reading Data from user1");
48 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email);
49
50 System.out.println("Reading Data from user2");
51 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email);

```

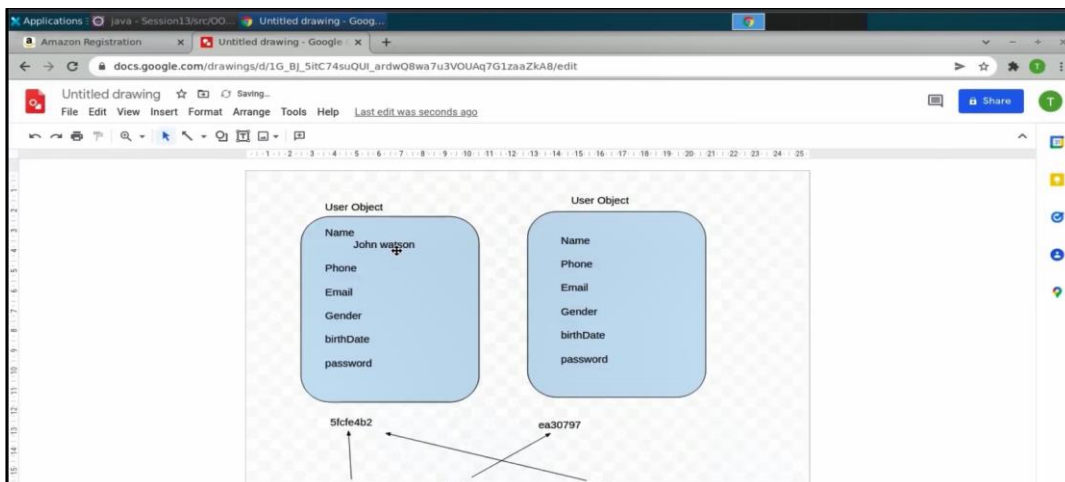

8.2 Let us write **user1.name**, which can be "John", and this can be either written or updated. In the next line, you can write **user3.name** and update the value to "John Watson"

```

18 class User {
19     // Attributes: Property of Object
20     String name;
21     String phone;
22     String email;
23     char gender;
24     Date birthDate;
25     String password;
26 }
27
28
29
30 public class OOPS {
31     public static void main(String[] args) {
32
33         // 3. From the class create the real object in memory
34         User user1 = new User();
35         User user2 = new User();
36
37         User user3 = user1; // Reference Copy Operation
38
39         System.out.println("user1 is: "+user1);
40         System.out.println("user2 is: "+user2);
41         System.out.println("user3 is: "+user3);
42
43         // Operations on Object
44         // 1. Write/Update i.e. Set the data
45         user1.name = "John";
46         user3.name = "John Watson";
47
48         System.out.println("Reading Data from user1");
49         System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email);
50
51     }
52 }

```

8.3 Hence, when it comes to the object here, the name is **John Watson**, and this is the record that would be added to the attribute name. Similarly, let us write **user2.name** as **Fionna**



8.4 As you have added the name, let us add the other details. Let us write **user3.phone** as +91 and a number. Then **user3.email** as **John.Watson@example.com** and **user1.birthdate** as a new Date object from **java.util**. Then **user1.password** goes like **john@123**. These are the values that will get populated for your first object

```

1  OOPS.java
2
3  Date birthDate;
4  String password;
5
6  }
7
8  public class OOPS {
9
10     public static void main(String[] args) {
11
12         // 3. From the class create the real object in memory
13         User user1 = new User();
14         User user2 = new User();
15
16         User user3 = user1; // Reference Copy Operation
17
18         System.out.println("user1 is: "+user1);
19         System.out.println("user2 is: "+user2);
20         System.out.println("user3 is: "+user3);
21
22         // Operations on Object
23         // 1. Write/Update i.e. Set the data
24         user1.name = "john";
25         user3.name = "john watson";
26
27         user3.phone = "+91 99999 11111";
28         user3.email = "john.watson@example.com";
29         user1.birthDate = new Date();
30         user1.password = "john@123";
31
32         user2.name = "fionna";
33
34         System.out.println("Reading Data from user1");
35         System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email);
36     }
37 }

```

8.5 Let us also feed similar details into **user2**. Let us give the phone number, then write as **fionna@example.com** and the password as **fionna@123**

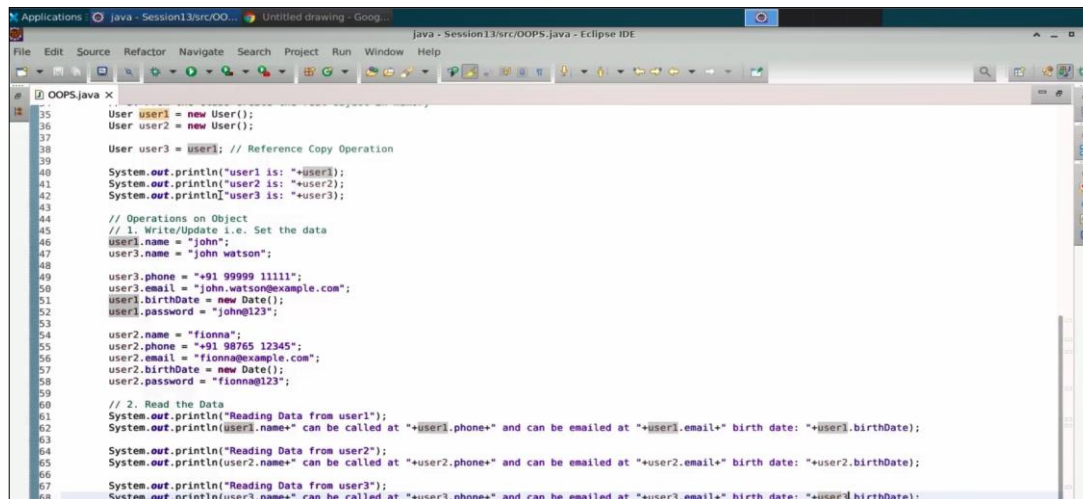
```

1  OOPS.java
2
3  Date birthDate;
4  String password;
5
6  }
7
8  public class OOPS {
9
10     public static void main(String[] args) {
11
12         // 3. From the class create the real object in memory
13         User user1 = new User();
14         User user2 = new User();
15
16         User user3 = user1; // Reference Copy Operation
17
18         System.out.println("user1 is: "+user1);
19         System.out.println("user2 is: "+user2);
20         System.out.println("user3 is: "+user3);
21
22         // Operations on Object
23         // 1. Write/Update i.e. Set the data
24         user1.name = "john";
25         user3.name = "john watson";
26
27         user3.phone = "+91 99999 11111";
28         user3.email = "john.watson@example.com";
29         user1.birthDate = new Date();
30         user1.password = "john@123";
31
32         user2.phone = "+91 98765 12345";
33         user2.email = "fionna@example.com";
34         user2.birthDate = new Date();
35         user2.password = "fionna@123";
36
37         System.out.println("Reading Data from user1");
38         System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email);
39     }
40 }

```

Step 9: Access the attribute and print the details

9.1 In The second operation called **Read the Data**, which means with the object's reference, you are accessing the attribute and it is printing the details

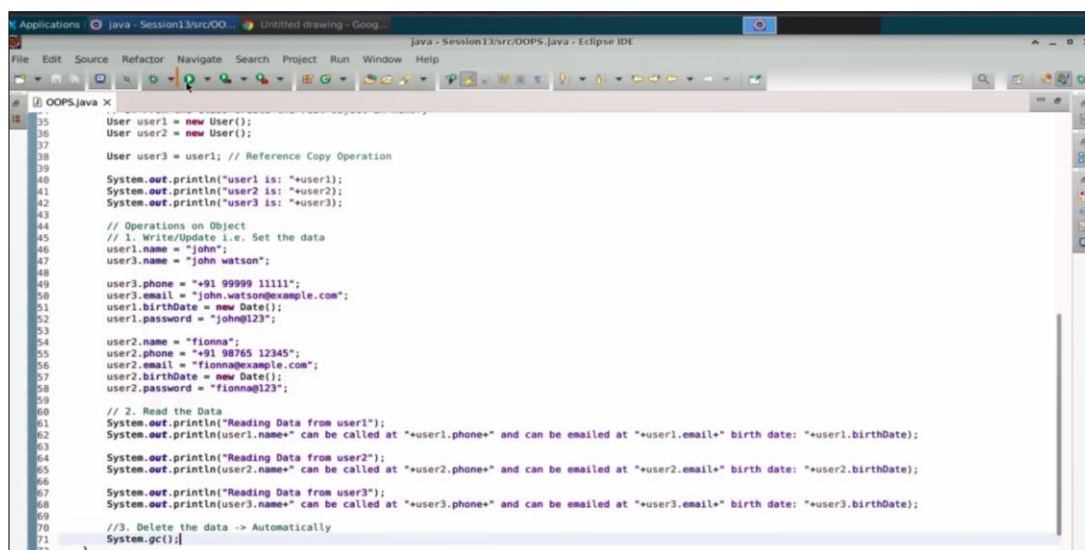


```

35 User user1 = new User();
36 User user2 = new User();
37
38 User user3 = user1; // Reference Copy Operation
39
40 System.out.println("user1 is: "+user1);
41 System.out.println("user2 is: "+user2);
42 System.out.println("user3 is: "+user3);
43
44 // Operations on Object
45 // 1. Write/Update i.e. Set the data
46 user1.name = "john";
47 user3.name = "john watson";
48
49 user3.phone = "+91 99999 11111";
50 user3.email = "john.watson@example.com";
51 user1.birthDate = new Date();
52 user1.password = "john@123";
53
54 user2.name = "fionna";
55 user2.phone = "+91 98765 12345";
56 user2.email = "fionna@example.com";
57 user2.birthDate = new Date();
58 user2.password = "fionna@123";
59
60 // 2. Read the Data
61 System.out.println("Reading Data from user1");
62 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
63
64 System.out.println("Reading Data from user2");
65 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
66
67 System.out.println("Reading Data from user3");
68 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);

```

9.2 The third operation is '**delete the data**,' which happens automatically. Java has a garbage collector that runs periodically, looks for unused objects, and deletes them from memory. However, you can also write it as **System.gc()**, which is an explicit call to the garbage collector that can eliminate or remove unused objects from memory with a single command



```

35 User user1 = new User();
36 User user2 = new User();
37
38 User user3 = user1; // Reference Copy Operation
39
40 System.out.println("user1 is: "+user1);
41 System.out.println("user2 is: "+user2);
42 System.out.println("user3 is: "+user3);
43
44 // Operations on Object
45 // 1. Write/Update i.e. Set the data
46 user1.name = "john";
47 user3.name = "john watson";
48
49 user3.phone = "+91 99999 11111";
50 user3.email = "john.watson@example.com";
51 user1.birthDate = new Date();
52 user1.password = "john@123";
53
54 user2.name = "fionna";
55 user2.phone = "+91 98765 12345";
56 user2.email = "fionna@example.com";
57 user2.birthDate = new Date();
58 user2.password = "fionna@123";
59
60 // 2. Read the Data
61 System.out.println("Reading Data from user1");
62 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
63
64 System.out.println("Reading Data from user2");
65 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
66
67 System.out.println("Reading Data from user3");
68 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
69
70 //3. Delete the data -> Automatically
71 System.gc();
72

```

Step 10: Run the code and populate the data

10.1 Let us run this code and what you will observe is while reading the data from user one, the data has been populated over here. And the rest of the details coming in, with the birthday coming as the entire data. One thing that you need to note is for user one and for user three all the details are going to be the same because they are not different objects

The screenshot shows the Eclipse IDE with a Java file named 'OOPS.java'. The code defines a 'User' class and creates three 'User' objects: 'user1', 'user2', and 'user3'. 'user1' and 'user3' are references to the same object, while 'user2' is a separate object. The code then prints details for each user. The console output shows that 'user1' and 'user3' have the same details (John Watson, +91 99999 11111, john.watson@example.com, 12345), while 'user2' has different details (Fionna, +91 98765 12345, fionna@example.com, 12345).

```

35 User user1 = new User();
36 User user2 = new User();
37
38 User user3 = user1; // Reference Copy Operation
39
40 System.out.println("user1 is: "+user1);
41 System.out.println("user2 is: "+user2);
42 System.out.println("user3 is: "+user3);
43
44 // Operations on Object
45 // 1. Write/Update i.e. Set the data
46 user1.name = "john";
47 user3.name = "john watson";
48
49 user3.phone = "+91 99999 11111";
50 user3.email = "john.watson@example.com";
51 user1.birthDate = new Date();
52 user1.password = "john@123";
53
54 user2.name = "fionna";
55 user2.phone = "+91 98765 12345";
56 user2.email = "fionna@example.com";
57 user2.birthDate = new Date();
58 user2.password = "fionna@123";
59
60 // 2. Read the Data
61 System.out.println("Reading Data from user1");
62 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
63
64 System.out.println("Reading Data from user2");
65 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
66
67 System.out.println("Reading Data from user3");
68 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);

```

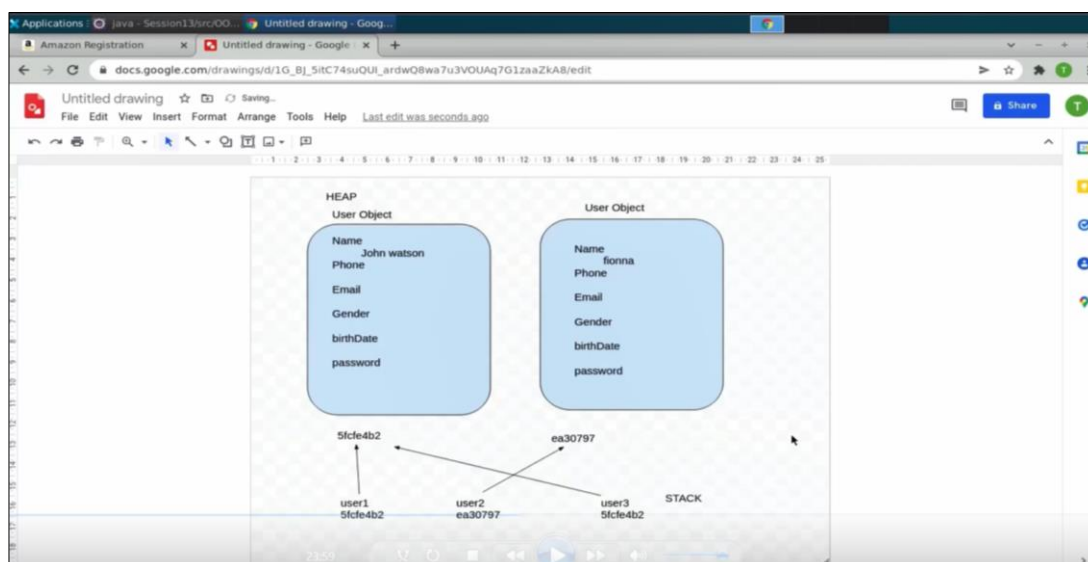
Console Output:

```

<terminated> OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux64/eclipse
user1 is: User@4c98385c
user2 is: User@73a8d4cc
user3 is: User@4c98385c
Reading Data from user1
John Watson can be called at +91 99999 11111 and can be emailed at john.watson@example.com birth date: Sun Aug 10 12:34:56 GMT+05:30 2020
Reading Data from user2
Fionna can be called at +91 98765 12345 and can be emailed at fionna@example.com birth date: Sun Aug 10 12:34:56 GMT+05:30 2020
Reading Data from user3
John Watson can be called at +91 99999 11111 and can be emailed at john.watson@example.com birth date: Sun Aug 10 12:34:56 GMT+05:30 2020

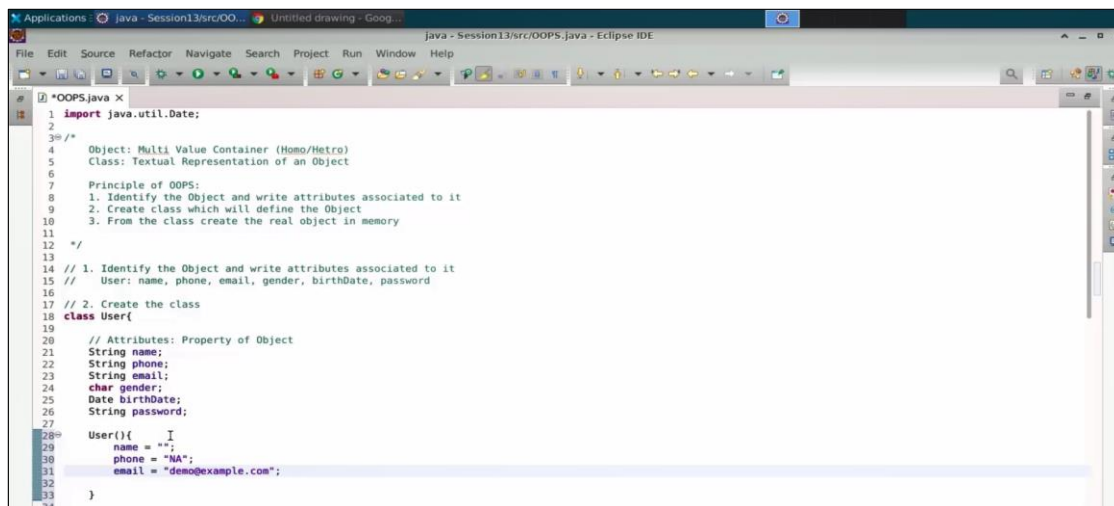
```

10.2 The objects get constructed in the area known as the heap of your ram area whereas these reference variables would be available in the stack region allocated to the main method



Step 11: Select the default constructor from the source, then generate the constructor with fields

11.1 Now, write a constructor which is a method inside your class, having the same name as that of the class name. You can give as the name is not available, which is the default value of the variable name. Phone is not available or you can even initialize it to empty, so rather than null, you can give it as not available. Then the email goes as **demo@example.com**.

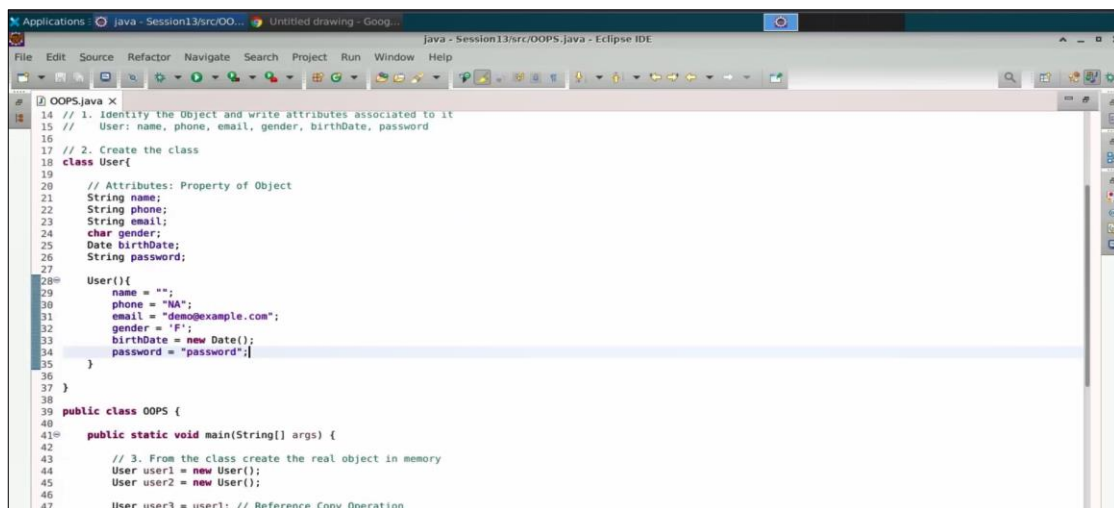


```

1  import java.util.Date;
2
3  /**
4   * Object: Multi Value Container (Homo/Metro)
5   * Class: Textual Representation of an Object
6   *
7   * Principle of OOPS:
8   * 1. Identify the Object and write attributes associated to it
9   * 2. Create class which will define the Object
10  * 3. From the class create the real object in memory
11  */
12
13
14 // 1. Identify the Object and write attributes associated to it
15 // User: name, phone, email, gender, birthDate, password
16
17 // 2. Create the class
18 class User{
19
20     // Attributes: Property of Object
21     String name;
22     String phone;
23     String email;
24     char gender;
25     Date birthDate;
26     String password;
27
28     User(){
29         name = "";
30         phone = "NA";
31         email = "demo@example.com";
32     }
33 }

```

11.2 Let us give the gender by default which is Female. Then the birthdate is a new date object from the java dot util. The password is by default password. These are some of the details, which will be automatically associated



```

14 // 1. Identify the Object and write attributes associated to it
15 // User: name, phone, email, gender, birthDate, password
16
17 // 2. Create the class
18 class User{
19
20     // Attributes: Property of Object
21     String name;
22     String phone;
23     String email;
24     char gender;
25     Date birthDate;
26     String password;
27
28     User(){
29         name = "";
30         phone = "NA";
31         email = "demo@example.com";
32         gender = 'F';
33         birthDate = new Date();
34         password = "password";
35     }
36 }
37
38 public class OOPS {
39
40     public static void main(String[] args) {
41
42         // 3. From the class create the real object in memory
43         User user1 = new User();
44         User user2 = new User();
45
46         User user3 = user1; // Reference Copy Operation
47     }
48 }

```

11.3 Whenever you create any user object, remember in order to create an object, you need to write new and then use your class name and add parenthesis. Note, that this is basically execution to the default constructor

```

17 // 2. Create the class
18 class User{
19
20     // Attributes: Property of Object
21     String name;
22     String phone;
23     String email;
24     char gender;
25     Date birthDate;
26     String password;
27
28     User(){
29         name = "";
30         phone = "NA";
31         email = "demo@example.com";
32         gender = 'F';
33         birthDate = new Date();
34         password = "password";
35     }
36
37 }
38
39 public class OOPS {
40
41     public static void main(String[] args) {
42
43         // 3. From the class create the real object in memory
44         User user1 = new User();
45         User user2 = new User();
46
47         User user3 = user1; // Reference Copy Operation
48
49         User user4 = new User();
50
51         System.out.println("user1 is: "+user1);
52         System.out.println("user2 is: "+user2);
53         System.out.println("user3 is: "+user3);
54     }
55 }

```

11.4 Let us create a **user4** and print it and the same way let us print the data from the **user4**. You will not be writing any data in **user4** as you have written the data in **user1, 2, and 3** as just a reference copy. Then, when you run your code, it will show empty can be called at NA, and can be emailed at **demo@example.com** and birthdate in the default manner

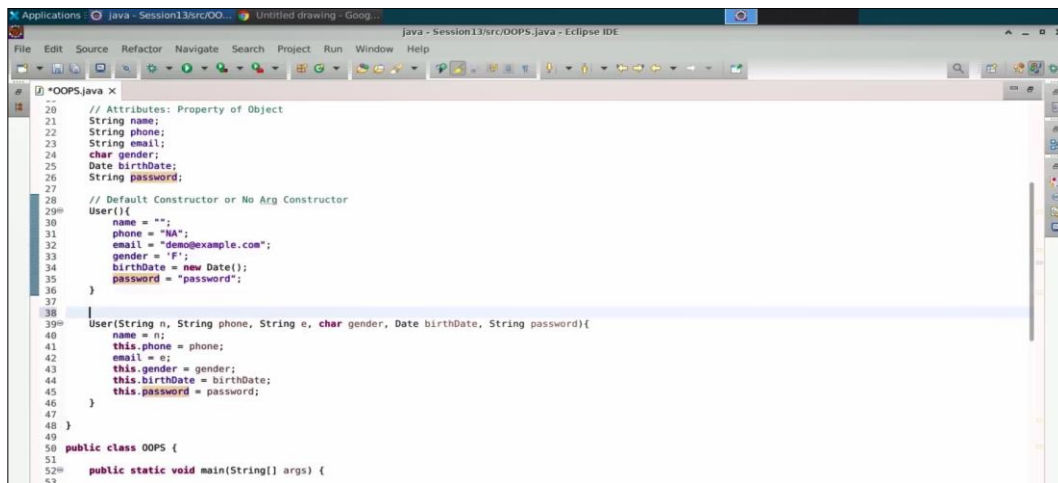
```

51
52 System.out.println("user1 is: "+user1);
53 System.out.println("user2 is: "+user2);
54 System.out.println("user3 is: "+user3);
55 System.out.println("user4 is: "+user4);
56
57 // Operations on Object
58 // 1. Write/Update i.e. Set the data
59 user1.name = "john";
60 user3.name = "john watson";
61
62 user3.phone = "+91 99999 11111";
63 user3.email = "john.watson@example.com";
64 user1.birthDate = new Date();
65 user1.password = "john@123";
66
67 user2.name = "fionna";
68 user2.phone = "+91 98765 12345";
69 user2.email = "fionna@example.com";
70 user2.birthDate = new Date();
71 user2.password = "fionna@123";
72
73 // 2. Read the Data
74 System.out.println("Reading Data from user1");
75 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
76
77 System.out.println("Reading Data from user2");
78 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
79
80 System.out.println("Reading Data from user3");
81 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
82
83 System.out.println("Reading Data from user4");
84 System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
85

```


Step 12: Create an array with a specific size

12.1 You can also write a constructor with inputs, and to do that, there are two ways. One way is to write **String n** and then copy this **n** into the name. You can then give the second input as **String phone**. Here, there is confusion: when you assign phone to **phone**, it creates ambiguity. To resolve this, you need to use a variable called **this**, which refers to the input to the constructor. Next, write **char gender**, then **Date birthDate**, and finally a String called **password**

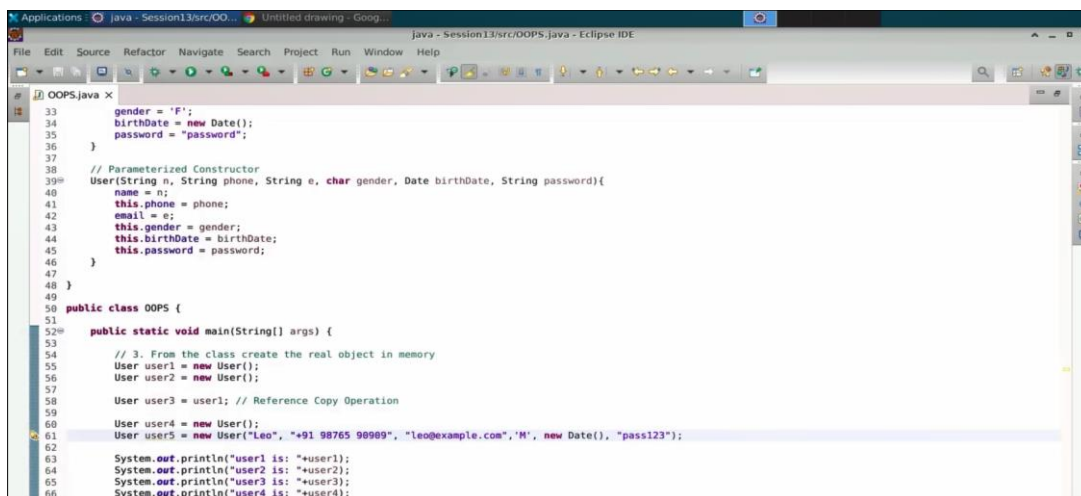


```

20 // Attributes: Property of Object
21 String name;
22 String phone;
23 String email;
24 char gender;
25 Date birthDate;
26 String password;
27
28 // Default Constructor or No Arg Constructor
29 User() {
30     name = "";
31     phone = "NA";
32     email = "demo@example.com";
33     gender = 'F';
34     birthDate = new Date();
35     password = "password";
36 }
37
38 // Parameterized Constructor
39 User(String n, String phone, String e, char gender, Date birthDate, String password) {
40     name = n;
41     this.phone = phone;
42     email = e;
43     this.gender = gender;
44     this.birthDate = birthDate;
45     this.password = password;
46 }
47
48 }
49
50 public class OOPS {
51
52     public static void main(String[] args) {
53

```

12.2 Next, let us create another user object called **user5**, which is a new user and this time you can pass the data. Let us write the name, **Leo**. A phone number is a phone number and then the gender. Then the birthdate, you can give a **new date** object here and then the **password**. Next, the email, which is like your third input, so let us give it as **Leo@example.com**.



```

33     gender = 'F';
34     birthDate = new Date();
35     password = "password";
36 }
37
38 // Parameterized Constructor
39 User(String n, String phone, String e, char gender, Date birthDate, String password) {
40     name = n;
41     this.phone = phone;
42     email = e;
43     this.gender = gender;
44     this.birthDate = birthDate;
45     this.password = password;
46 }
47
48 }
49
50 public class OOPS {
51
52     public static void main(String[] args) {
53
54         // 3. From the class create the real object in memory
55         User user1 = new User();
56         User user2 = new User();
57         User user3 = user1; // Reference Copy Operation
58
59         User user4 = new User();
60         User user5 = new User("Leo", "+91 98765 98989", "Leo@example.com", 'M', new Date(), "pass123");
61
62         System.out.println("user1 is: " + user1);
63         System.out.println("user2 is: " + user2);
64         System.out.println("user3 is: " + user3);
65         System.out.println("user4 is: " + user4);
66

```

12.3 Let us print the data for User5

```

65 System.out.println("user3 is: "+user3);
66 System.out.println("user4 is: "+user4);
67
68 // Operations on Object
69 // 1. Write/Update i.e. Set the data
70 user1.name = "john";
71 user3.name = "john watson";
72
73 user3.phone = "+91 99999 11111";
74 user3.email = "john.watson@example.com";
75 user1.birthDate = new Date();
76 user1.password = "john@123";
77
78 user2.name = "fionna";
79 user2.phone = "+91 98765 12345";
80 user2.email = "fionna@example.com";
81 user2.birthDate = new Date();
82 user2.password = "fionna@123";
83
84 // 2. Read the Data
85 System.out.println("Reading Data from user1");
86 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
87
88 System.out.println("Reading Data from user2");
89 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
90
91 System.out.println("Reading Data from user3");
92 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
93
94 System.out.println("Reading Data from user4");
95 System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
96
97 System.out.println("Reading Data from user5");
98 System.out.println(user5.name+" can be called at "+user5.phone+" and can be emailed at "+user5.email+" birth date: "+user5.birthDate);
99
100
101 //3. Delete the data -> Automatically
102 System.gc();

```

12.4 Run it and Here you are with the data in user five, which shows Leo can be called and email and accordingly the entire data coming in

```

65 System.out.println("user3 is: "+user3);
66 System.out.println("user4 is: "+user4);
67
68 // Operations on Object
69 // 1. Write/Update i.e. Set the data
70 user1.name = "john";
71 user3.name = "john watson";
72
73 user3.phone = "+91 99999 11111";
74 user3.email = "john.watson@example.com";
75 user1.birthDate = new Date();
76 user1.password = "john@123";
77
78 user2.name = "fionna";
79 user2.phone = "+91 98765 12345";
80 user2.email = "fionna@example.com";
81 user2.birthDate = new Date();
82 user2.password = "fionna@123";
83
84 // 2. Read the Data
85 System.out.println("Reading Data from user1");
86 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
87
88 System.out.println("Reading Data from user2");
89 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
90
91 System.out.println("Reading Data from user3");
92 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
93
94 System.out.println("Reading Data from user4");
95 System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
96
97 System.out.println("Reading Data from user5");

```

Console Output:

```

<terminated> OOPS [java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot
user1 is: User@ea30797
user2 is: User@5025a40
user3 is: User@ea30797
user4 is: User@1b701da1
Reading Data from user1
john watson can be called at +91 99999 11111 and can be emailed at john.wa
Reading Data from user2
fionna can be called at +91 98765 12345 and can be emailed at fionna@examp
Reading Data from user3
john watson can be called at +91 99999 11111 and can be emailed at john.wa
Reading Data from user4
 can be called at NA and can be emailed at demo@example.com birth date: Mo
Reading Data from user5
Leo can be called at +91 98765 90909 and can be emailed at leo@example.com

```


12.5 Let us write void and create a method called **setUserData**. For the user data, you can take all the inputs that are present in the constructor. To have a better approach, let us write this as **this.name = name** and then **this.email = email**

```

31 phone = "NA";
32 email = "demo@example.com";
33 gender = 'F';
34 birthDate = new Date();
35 password = "password";
36 }
37
38 // Parameterized Constructor
39 User(String n, String phone, String e, char gender, Date birthDate, String password){
40     name = n;
41     this.phone = phone;
42     email = e;
43     this.gender = gender;
44     this.birthDate = birthDate;
45     this.password = password;
46 }
47
48 void setUserData(String name, String phone, String email, char gender, Date birthDate, String password){
49     this.name = name;
50     this.phone = phone;
51     this.email = email;
52     this.gender = gender;
53     this.birthDate = birthDate;
54     this.password = password;
55 }
56
57 }
58
59 public class OOPS {
60
61     public static void main(String[] args) {
62         // 3. From the class create the real object in memory
63         User user1 = new User();
64     }

```

12.6 Instead of writing **user2.setAllDetails()**, you can just write **user2.setUserData()**, where the name is set to **Fionna**, then the **phone number**, and then the **email**. For a **new date** object, you can even manipulate the date objects and work on customized date objects

```

66
67 User user3 = user1; // Reference Copy Operation
68
69 User user4 = new User();
70 User user5 = new User("Leo", "+91 98765 98989", "leo@example.com", 'M', new Date(), "pass123");
71
72 System.out.println("user1 is: "+user1);
73 System.out.println("user2 is: "+user2);
74 System.out.println("user3 is: "+user3);
75 System.out.println("user4 is: "+user4);
76
77 // Operations on Object
78 // 1. Write/Update i.e. Set the data
79 user1.name = "john";
80 user3.name = "john watson";
81
82 user3.phone = "+91 99999 11111";
83 user3.email = "john.watson@example.com";
84 user1.birthDate = new Date();
85 user1.password = "john@123";
86
87 // user2.name = "fionna";
88 // user2.phone = "+91 98765 12345";
89 // user2.email = "fionna@example.com";
90 // user2.birthDate = new Date();
91 // user2.password = "fionna@123";
92
93 user2.setUserData("Fionna", "+91 98765 12345", "fionna@example.com", 'F', new Date(), "fionna@123");
94
95 // 2. Read the Data
96 System.out.println("Reading Data from user1");
97 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
98
99 System.out.println("Reading Data from user2");

```

Step 13: Implement a two-dimensional array with suitable examples

13.1 Run the code, and you will notice that wherever you are writing these five lines of code, you are done with a single line of code and then the data is available in the object. Let us try to find a solution where you need to read the data

The screenshot shows the Eclipse IDE with a Java file named 'OOPS.java'. The code defines a 'User' class and creates several user objects. It then prints out the details of each user. The console output shows the following:

```

user1 is: User@73a8dfcc
user2 is: User@1c655221
user3 is: User@73a8dfcc
user4 is: User@58d25a40
Reading Data from user1
john watson can be called at +91 99999 11111 and can be emailed at john.watson@example.com
Reading Data from user2
fionna can be called at +91 98765 12345 and can be emailed at fionna@example.com
Reading Data from user3
john watson can be called at +91 99999 11111 and can be emailed at john.watson@example.com
Reading Data from user4
can be called at NA and can be emailed at demo@example.com birth date: 4/1/2000
Reading Data from user5
Leo can be called at +91 98765 90909 and can be emailed at leo@example.com
  
```

13.2 For reading the data, you can write a method called **showUserData** and then print out the name, followed by the phone number. Then, you can print the email, followed by the name. Next, print the gender and the birth date. The password will not be shown. You can make this prettier by using this basic representation and adding an empty print line at the end

The screenshot shows the Eclipse IDE with the same 'OOPS.java' file. The code now includes a 'showUserData' method that prints the user details in a formatted way. The console output is as follows:

```

-----
john watson can be called at +91 99999 11111
john watson can be emailed at john.watson@example.com
john watson has a gender: M and was born on 4/1/2000
-----
fionna can be called at +91 98765 12345
fionna can be emailed at fionna@example.com
fionna has a gender: F and was born on 4/1/2000
-----
  
```

13.3 Moving ahead, rather than showing the data in this manner, you can just write **user1.showUserData()**, **user2.showUserData()**, **user3.showUserData()**, **user4.showUserData()**, and **user5.showUserData()**. This way, you have five objects

```

89 user3.name = "john watson";
90
91 user3.phone = "+91 99999 11111";
92 user3.email = "john.watson@example.com";
93 user1.birthDate = new Date();
94 user1.password = "john@123";
95
96 //
97 user2.name = "fionna";
98 user2.phone = "+91 98765 12345";
99 user2.email = "fionna@example.com";
100 user2.birthDate = new Date();
101 user2.password = "fionna@123";
102
103 user2.setUserData("Fionna", "+91 98765 12345", "fionna@example.com", 'F', new Date(), "fionna@123");
104
105 // 2. Read the Data
106 /*System.out.println("Reading Data from user1");
107 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
108
109 System.out.println("Reading Data from user2");
110 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
111
112 System.out.println("Reading Data from user3");
113 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
114
115 System.out.println("Reading Data from user4");
116 System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
117
118 System.out.println("Reading Data from user5");
119 System.out.println(user5.name+" can be called at "+user5.phone+" and can be emailed at "+user5.email+" birth date: "+user5.birthDate);*/
120
121 user1.showUserData();
122 user2.showUserData();
123 user3.showUserData();
124 user4.showUserData();
125 user5.showUserData();
126

```

13.4 Run the code, and here a beautiful representation of all five objects is shown. It has John Watson, then Fionna, and John Watson again, as user one and user three are different references but they are pointing to the same object

```

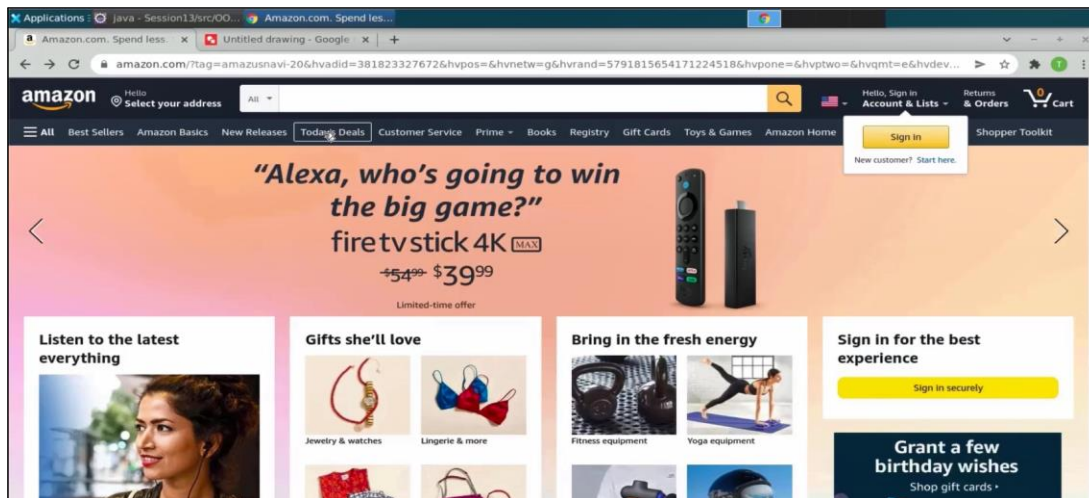
<terminated> OOPS [Java Application] Aus/eclipse/plugins/org.eclipse.justj.openjdk.hotspot
john watson can be called at +91 99999 11111
john watson can be emailed at john watson
john watson has a gender: F and was born on Mon Jan 17 14:18:30 UTC 2022

-----
can be called at NA
can be emailed at
has a gender: F and was born on Mon Jan 17 14:18:30 UTC 2022

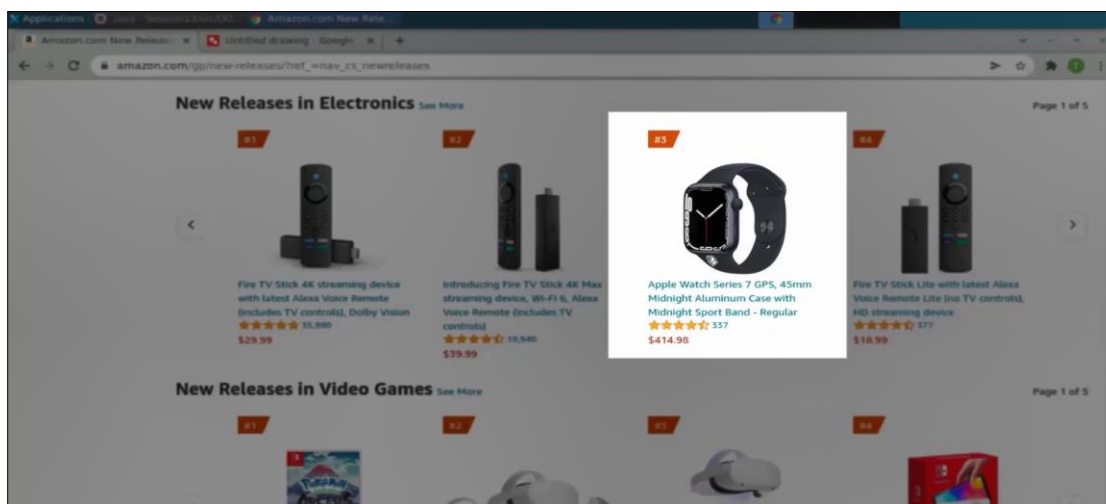
-----
Leo can be called at +91 98765 98989
Leo can be emailed at Leo
Leo has a gender: M and was born on Mon Jan 17 14:18:30 UTC 2022

```

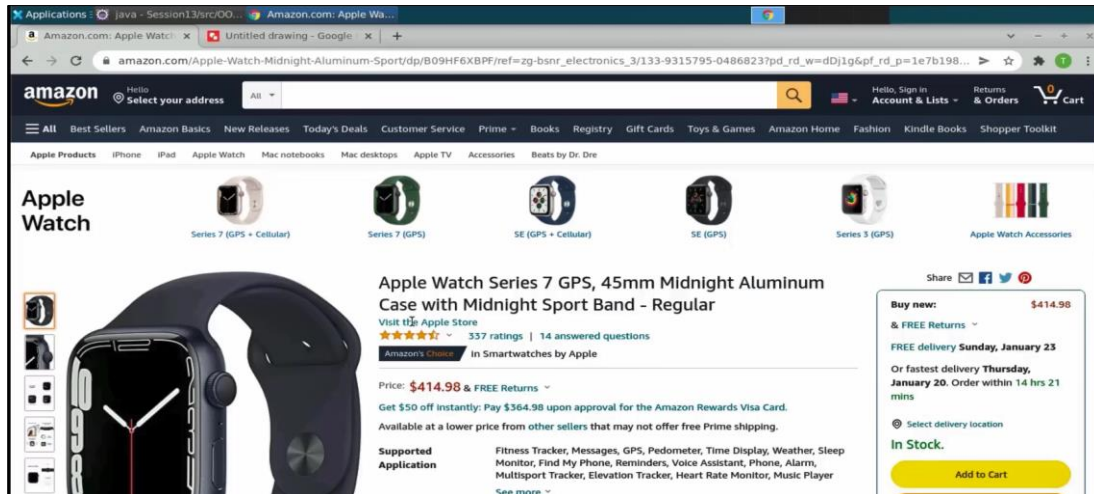
13.5 Moving ahead, you will now see how to design the product object with the same use case on the e-commerce platform. For that, let us go to the e-commerce web application on Amazon. Here, let us click on some new releases, and you can see some products listed



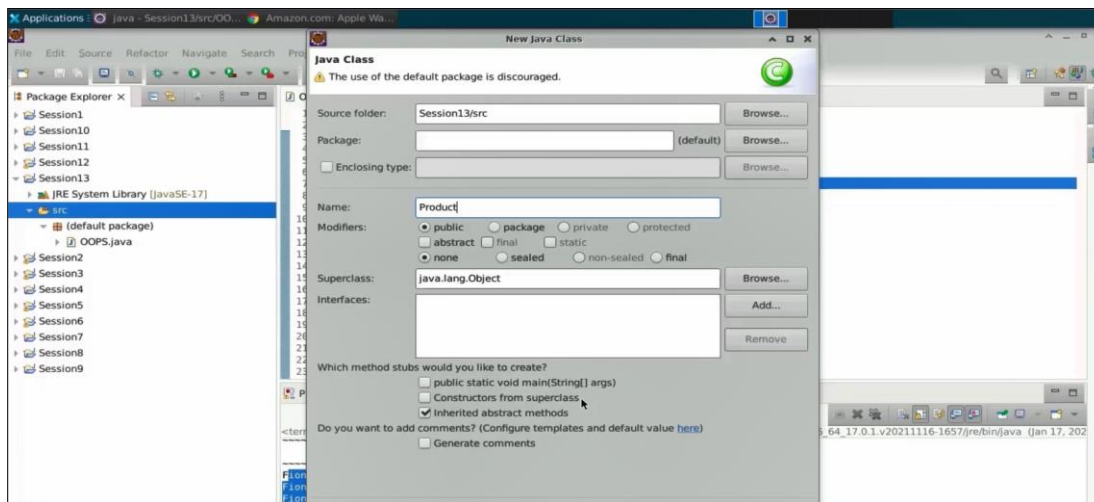
13.6 You can click on one of the products called the Apple Watch series.



13.7 This is how a product object is described, it has a name or a title, and it has some readings and moving ahead it has a description and then the price associated with the product object



13.8 Let us also follow the same philosophy, create a new class, and name the class itself as **product** and you will not be using the create method here, it is just one class by the name of the product which you will be creating separately as a separate program



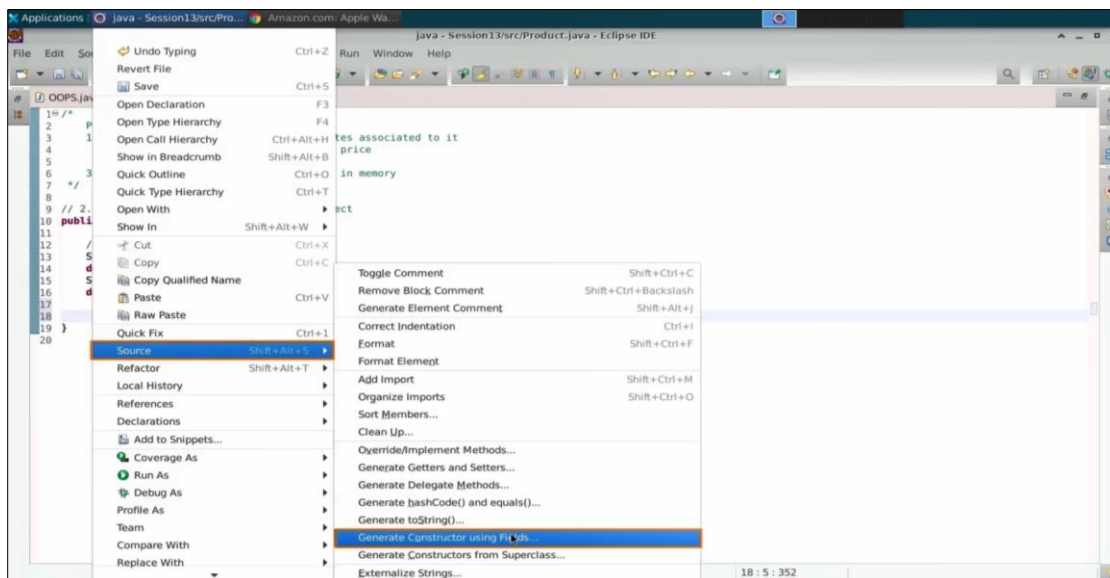
13.9 For the product object, the first key step is to identify the object and list the attributes associated with it. So, the object should be named **Product**, and it should have a name, ratings, description, and price. Let us now define these attributes. The name is going to be a string. The ratings can be a double or a floating-point number, the choice is yours. Then, you have the description as a string. And the price can again be doubled

```

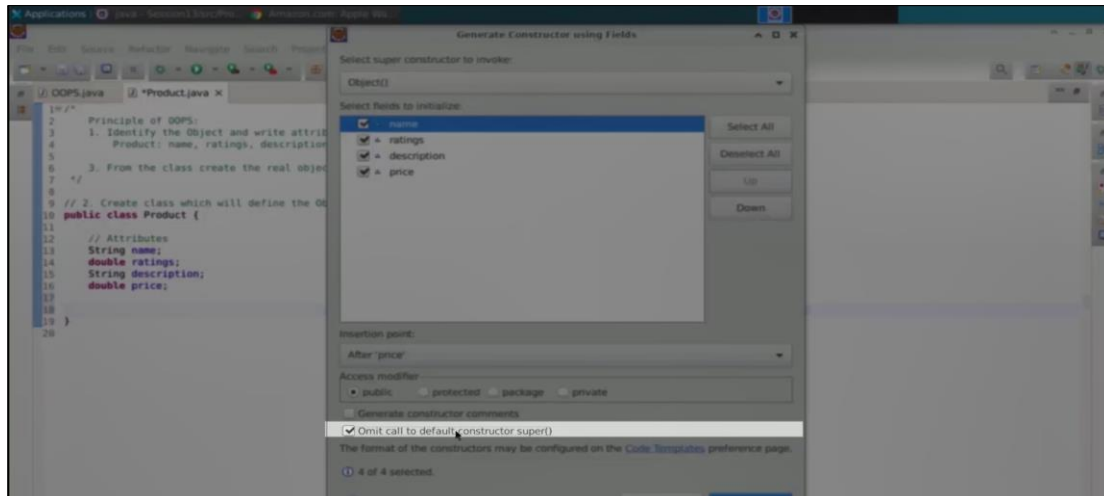
1  /*
2   Principle of OOPS:
3   1. Identify the Object and write attributes associated to it
4   Product: name, ratings, description, price
5
6   3. From the class create the real object in memory
7   */
8
9  // 2. Create class which will define the Object
10 public class Product {
11
12     // Attributes
13     String name;
14     double ratings;
15     String description;
16     double price;
17
18 }
19

```

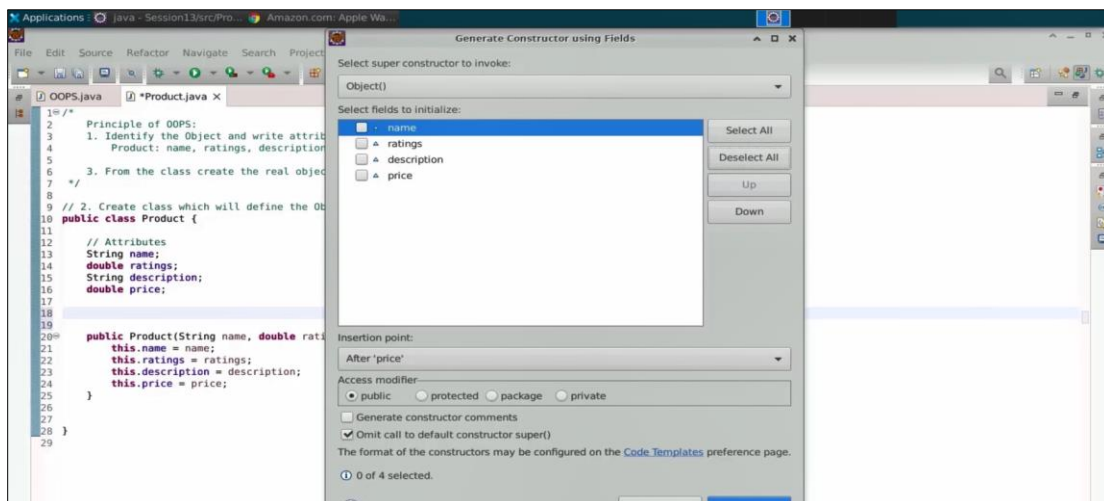
13.10 In order to make things a bit simpler, you can do right click and select **source > generate the constructor using the fields**



- 13.11 You will select, the **omit Call to the default constructor**, which is a super construction which is basically concerning inheritance. Select all these fields and click **generate** and you will notice that you are getting a parameterized constructor in action



- 13.12 Let us try to come here and select the **default constructor** which goes like source, then generate the constructor using the fields. You can **deselect all** and select **generate** and here you are with your default constructor



- 13.13 Next, let us create a method called **setProductData** (or any name of your choice). You can copy the entire definition of the parameterized constructor and use it within the setter method. Similarly, create a method called **showProductData**. In this method, write **Product** followed by the **name**, then "**has ratings**" and the **ratings**. Then write **Product** followed by the **name**, "**is priced at \$**" and the **price**. Finally, write **Product** followed by the **name**, "**description is**" and the **description**

```

1 // 2. Create class which will define the Object
2 class Product {
3     // Attributes
4     String name;
5     double ratings;
6     String description;
7     double price;
8
9     // Default Constructor
10    Product() {
11    }
12
13    // Parameterized Constructor
14    Product(String name, double ratings, String description, double price) {
15        this.name = name;
16        this.ratings = ratings;
17        this.description = description;
18        this.price = price;
19    }
20
21    void setProductData(String name, double ratings, String description, double price) {
22        this.name = name;
23        this.ratings = ratings;
24        this.description = description;
25        this.price = price;
26    }
27
28    void showProductData() {
29        System.out.println("Product "+name+" has "+ratings+" ratings");
30        System.out.println("Product "+name+" is priced at $"+price);
31        System.out.println("Product "+name+" description is: "+description);
32    }
33 }

```

- 13.14 Come back to **oops.java** and create the product objects. Let us write **Product P1** as a **new product** with the default configuration. Then, write **Product P2** as a **new product**, which will be created through the parameterized constructor. Next, create **P3** as a **new product** for which you can set the data later. To set the product data, you can write **ultraBoost Shoe**, rating as **5.0**, description as **'Adidas comfortable ultra-boost shoes, 21'**, and the price as **200**

```

102 user2.setUserData("Fionna", "+91 98765 12345", "fionna@example.com", 'F', new Date(), "fionna@123");
103
104 // 2. Read the Data
105 System.out.println("Reading Data from user1");
106 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
107
108 System.out.println("Reading Data from user2");
109 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
110
111 System.out.println("Reading Data from user3");
112 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
113
114 System.out.println("Reading Data from user4");
115 System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
116
117 System.out.println("Reading Data from user5");
118 System.out.println(user5.name+" can be called at "+user5.phone+" and can be emailed at "+user5.email+" birth date: "+user5.birthDate);
119
120 user1.showUserData();
121 user2.showUserData();
122 user3.showUserData();
123 user4.showUserData();
124 user5.showUserData();
125
126 Product p1 = new Product();
127
128 Product p2 = new Product("iphone 11 pro max", 4.7, "Apple iPhone with 3 cameras", 800);
129
130 Product p3 = new Product();
131 p3.setProductData("Ultraboot Shoe", 5.0, "Adidas comfortable Ultraboot Shoes 21", 200);
132
133
134

```

13.15 Let us understand how to see the data in the object. You can execute the method called **showProductData** on all three objects, where **P1**, **P2**, and **P3** are not objects themselves but references to the objects. In **Product.java**, you need to write an empty print line in the **showProductData** method to create a better-distinguished output

```

102 user2.setUserData("Fionna", "+91 98765 12345", "fionna@example.com", 'F', new Date(), "fionna@123");
103
104 // 2. Read the Data
105 //System.out.println("Reading Data from user1");
106 System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);
107
108 System.out.println("Reading Data from user2");
109 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
110
111 System.out.println("Reading Data from user3");
112 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
113
114 System.out.println("Reading Data from user4");
115 System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
116
117 System.out.println("Reading Data from user5");
118 System.out.println(user5.name+" can be called at "+user5.phone+" and can be emailed at "+user5.email+" birth date: "+user5.birthDate);
119
120 user1.showUserData();
121 user2.showUserData();
122 user3.showUserData();
123 user4.showUserData();
124 user5.showUserData();
125
126 Product p1 = new Product();
127
128 Product p2 = new Product("iphone 11 pro max", 4.7, "Apple iPhone with 3 cameras", 800);
129
130 Product p3 = new Product();
131 p3.setProductData("Ultraboost Shoe", 5.0, "Adidas comfortable Ultraboost Shoes 21", 200);
132
133 p1.showProductData();
134 p2.showProductData();
135 p3.showProductData();
136
137 //3. Delete the data -> Automatically
138 System.gc();
139

```

13.16 Run the code, and if you notice here, the default product is priced at zero, has zero ratings, and has a null name with a null description. When you create a parameterized constructor, these are the details that are included in the object. Using the **setData** method, you can also have the data populated

```

<terminated> OOPS [Java Application] [usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Jan 17, 2022, 2:49:23 PM - 2:49:23 PM)]
john watson can be emailed at john watson
john watson has a gender: F and was born on Mon Jan 17 14:49:23 UTC 2022

-----
can be called at NA
can be emailed at
has a gender: F and was born on Mon Jan 17 14:49:23 UTC 2022

-----
Leo can be called at +91 98765 90909
Leo can be emailed at Leo
Leo has a gender: M and was born on Mon Jan 17 14:49:23 UTC 2022

-----
Product null has 0.0 ratings
Product null is priced at $0.0
Product null description is: null

-----
Product iphone 11 pro max has 4.7 ratings
Product iphone 11 pro max is priced at $800.0
Product iphone 11 pro max description is: Apple iPhone with 3 cameras

-----
Product Ultraboost Shoe has 5.0 ratings
Product Ultraboost Shoe is priced at $200.0

```

By following the above steps, you have successfully Designed the User and Product Objects in Object-oriented programming.