

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:

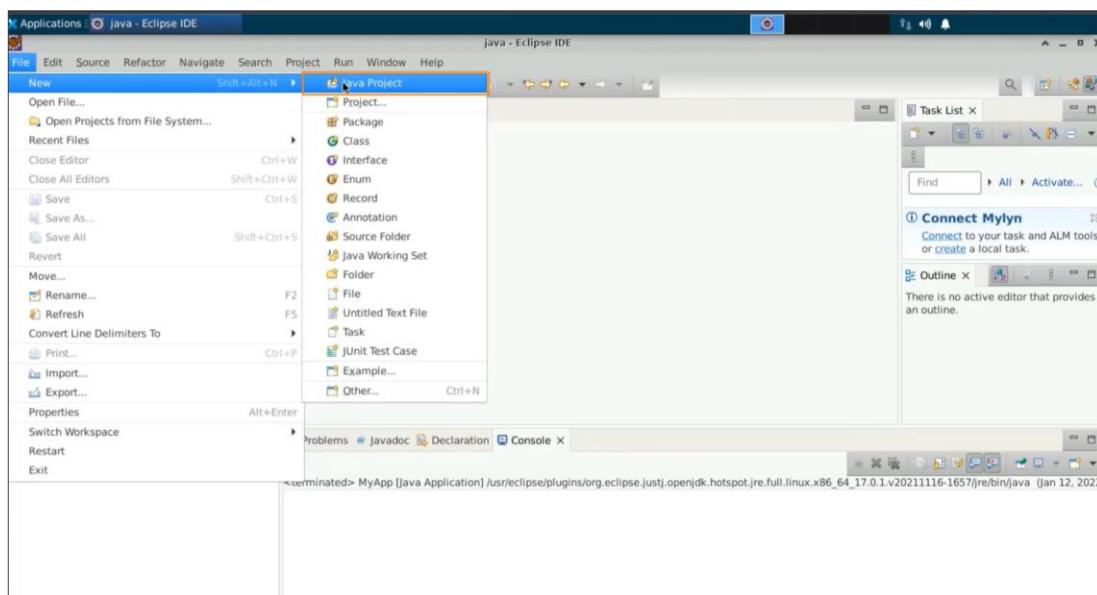
1. Open the IDE and Use a one-dimensional array with suitable examples
2. Create a new class called OOPS
3. Use the sample e-store application to add objects
4. Identify the data in the attributes
5. Create a real object in the memory
6. Assign hash codes to the objects
7. Implement a read operation 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

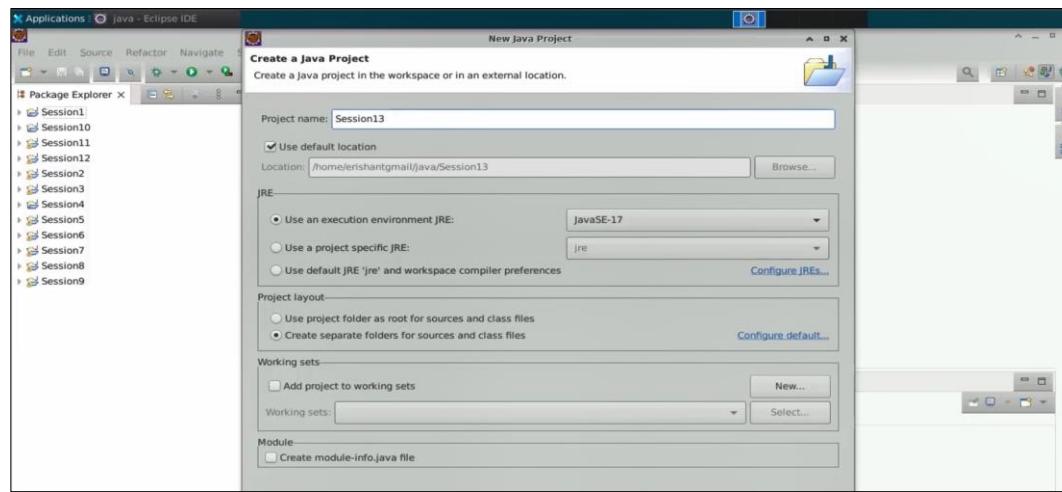
1.1 Open the Eclipse IDE



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

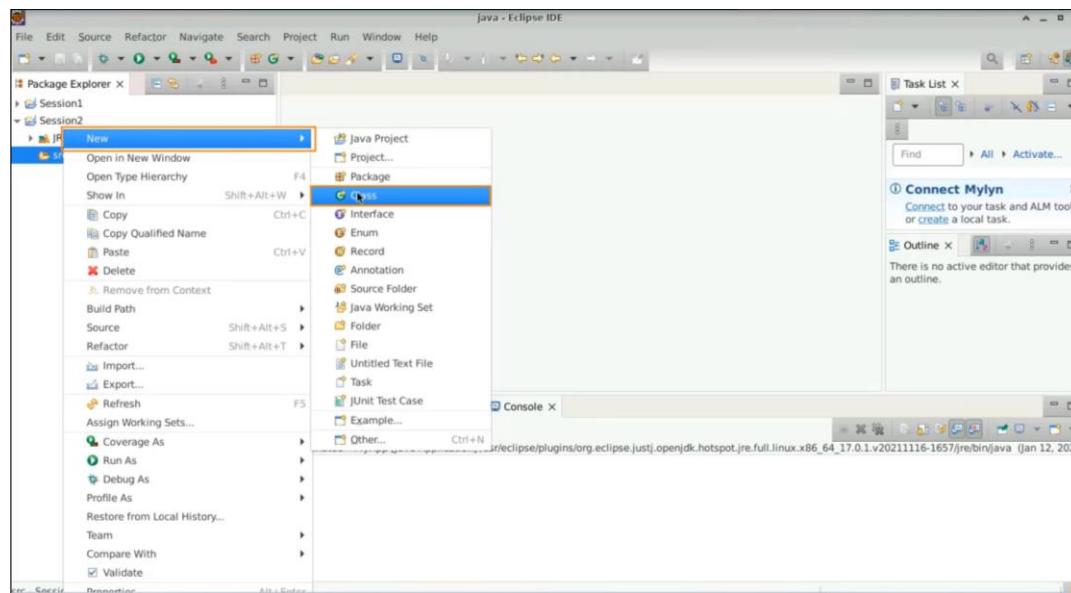


1.3 Name the project “**Session14**”, uncheck “**Create a module info dot Java file**”, and press **Finish**

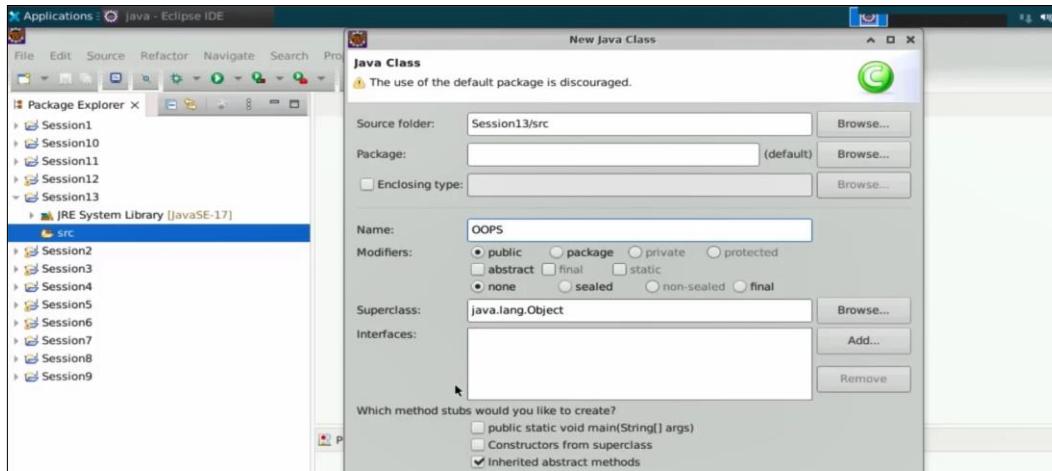


Step 2: Create a new class called OOPS

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

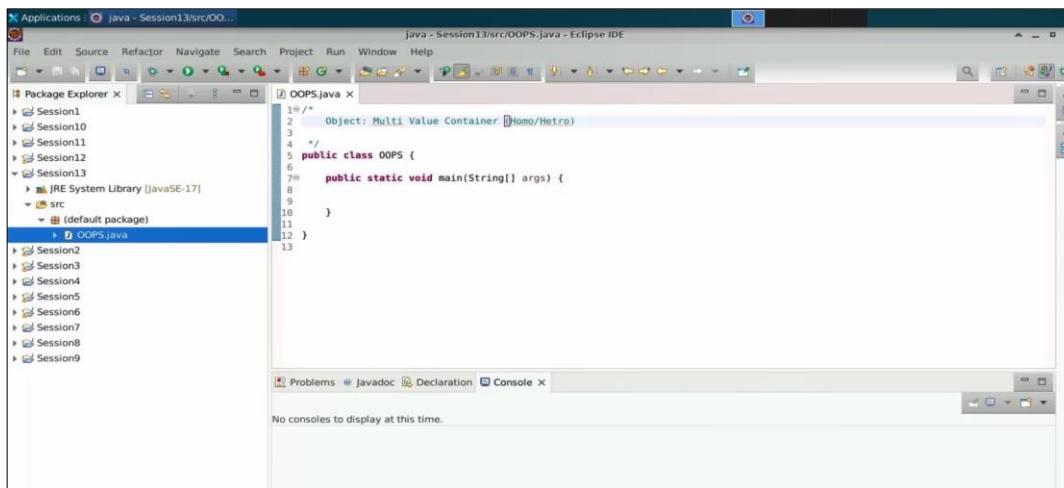


2.2 Name this class as an OOPS, then select the main method, and then select finish



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

```

10/*
2     Object: Multi Value Container (Homo/Hetro)
3     Class: Textual Representation of an Object
4
5 */
6 public class OOPS {
7
8     public static void main(String[] args) {
9
10    }
11
12 }
13
14

```

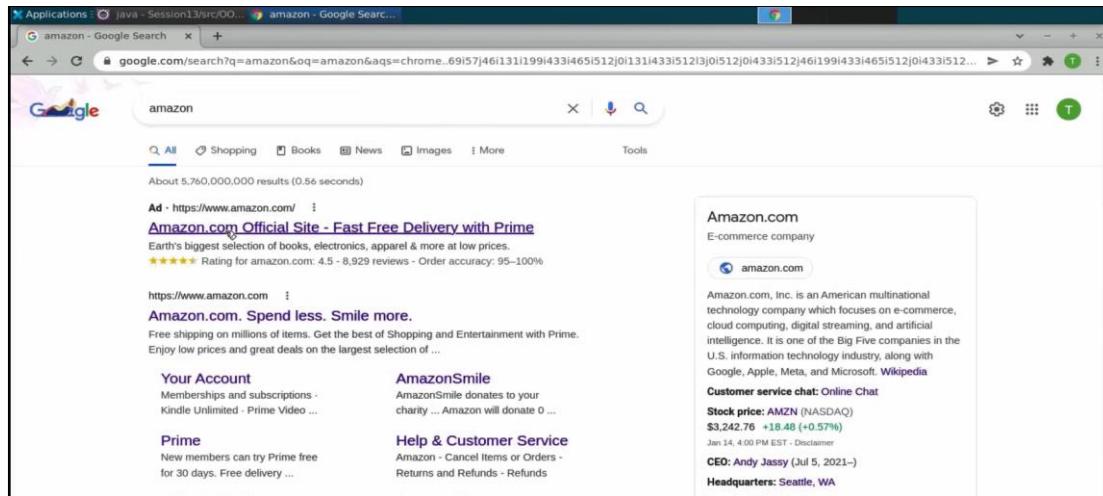
3.3 For the object, let us understand the principle of OOPS. First thing is to identify the object and write the attributes associated with it. Number two, create a class that will define the object, and number three from the class create the real object in the memory

```

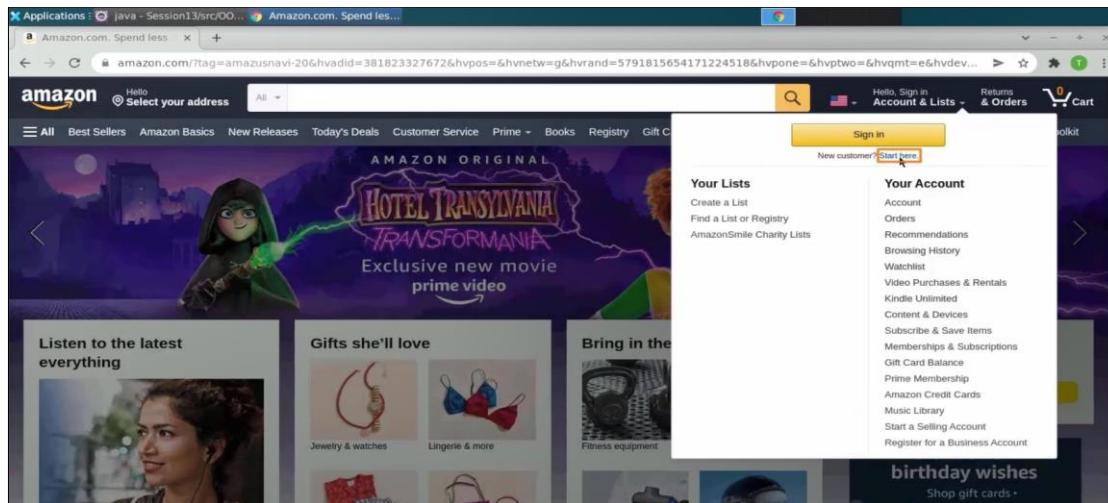
10/*
2     Object: Multi Value Container (Homo/Hetro)
3     Class: Textual Representation of an Object
4
5 Principle of OOPS:
6     1. Identify the Object and write attributes associated to it
7     2. Create class which will define the Object
8     3. From the class create the real object in memory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

```

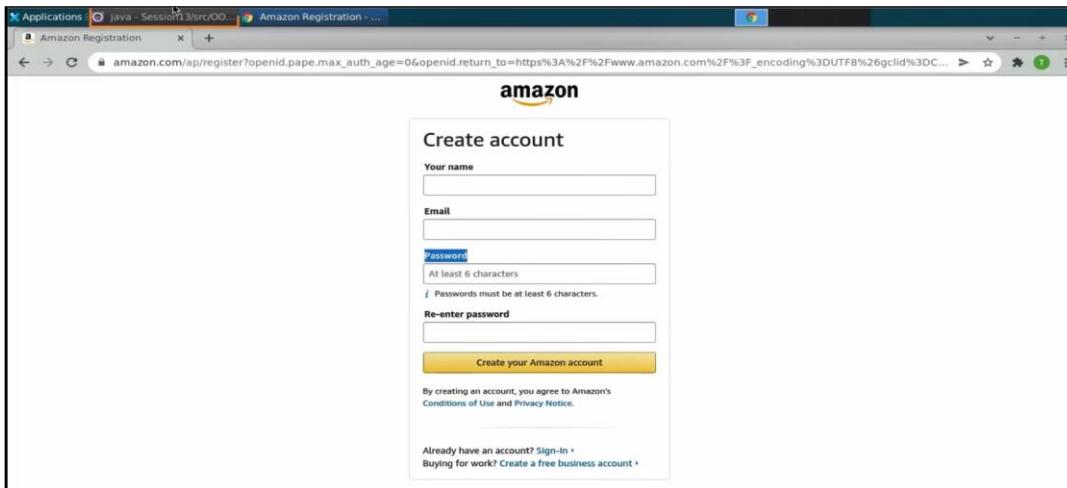
3.4 Let us open the sample e-store application and open the browser window and search for Amazon and open Amazon.com



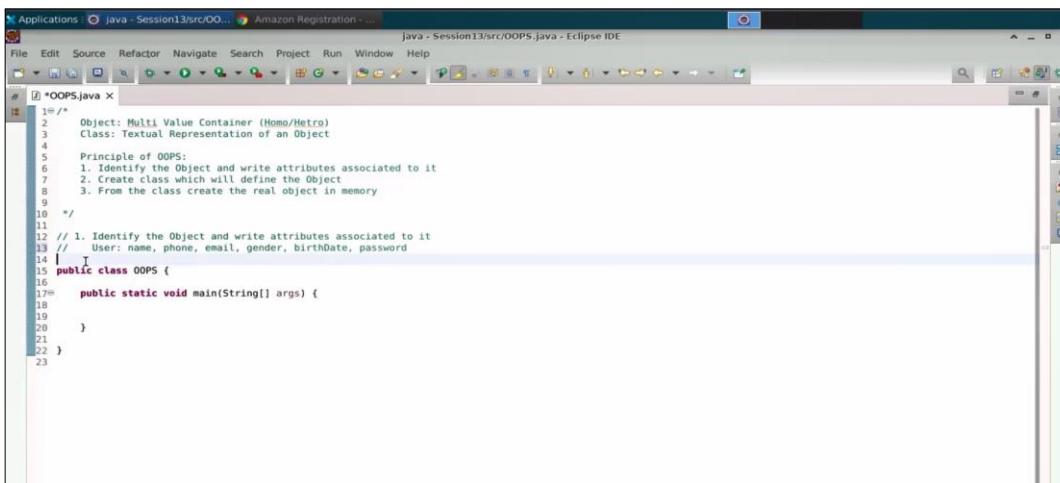
3.5 Whenever you open up any e-commerce platform, there is an option called to sign in or an option for new customers, start here



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



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



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         // 3. From the class create the real object in memory
34     }
35 }
36
37
38
39
40

```

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);**

```

1 // Principle of OOPS:
2 // 1. Identify the Object and write attributes associated to it
3 // 2. Create class which will define the Object
4 // 3. From the class create the real object in memory
5
6 /*
7
8
9
10
11 */
12
13 // 1. Identify the Object and write attributes associated to it
14 // User: name, phone, email, gender, birthdate, password
15
16 // 2. Create the class
17 class User{
18
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 public class OOPS {
30
31     public static void main(String[] args) {
32         // 3. From the class create the real object in memory
33         User user1 = new User();
34         User user2 = new User();
35
36         System.out.println("user1 is: "+user1);
37         System.out.println("user2 is: "+user2);
38     }
39 }
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

```

1 // Principle of OOPS:
2 // 1. Identify the Object and write attributes associated to it
3 // 2. Create class which will define the Object
4 // 3. From the class create the real object in memory
5
6 /*
7
8
9
10
11 */
12
13 // 1. Identify the Object and write attributes associated to it
14 // User: name, phone, email, gender, birthDate, password
15
16 // 2. Create the class
17 class User{
18
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 public class OOPS {
30
31     public static void main(String[] args) {
32         // 3. From the class create the real object in memory
33         User user1 = new User();
34         User user2 = new User();
35
36         System.out.println("user1 is: "+user1);
37         System.out.println("user2 is: "+user2);
38     }
39 }
40
41
42
43

```

Console Output:

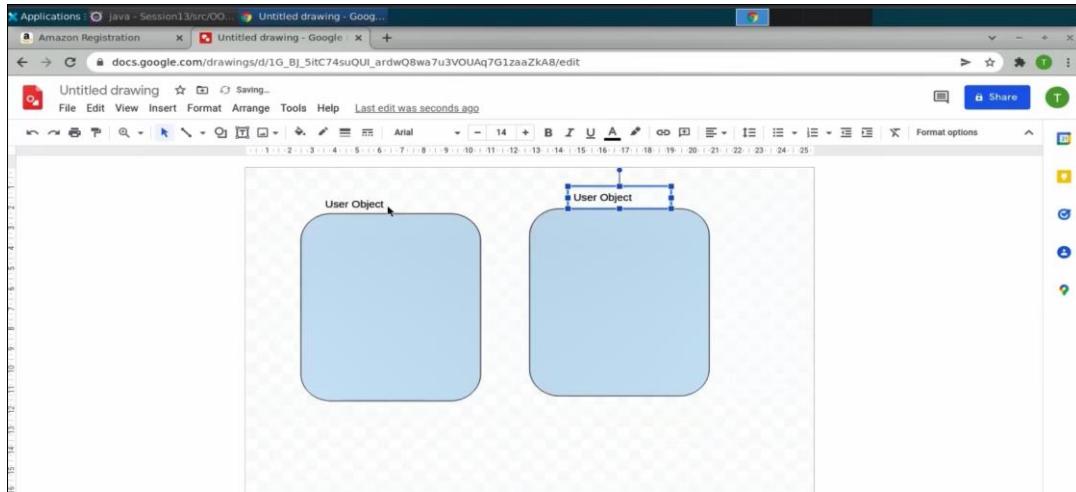
```

terminated> OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot
user1 is: User@5fcfe4b2
user2 is: User@a30797

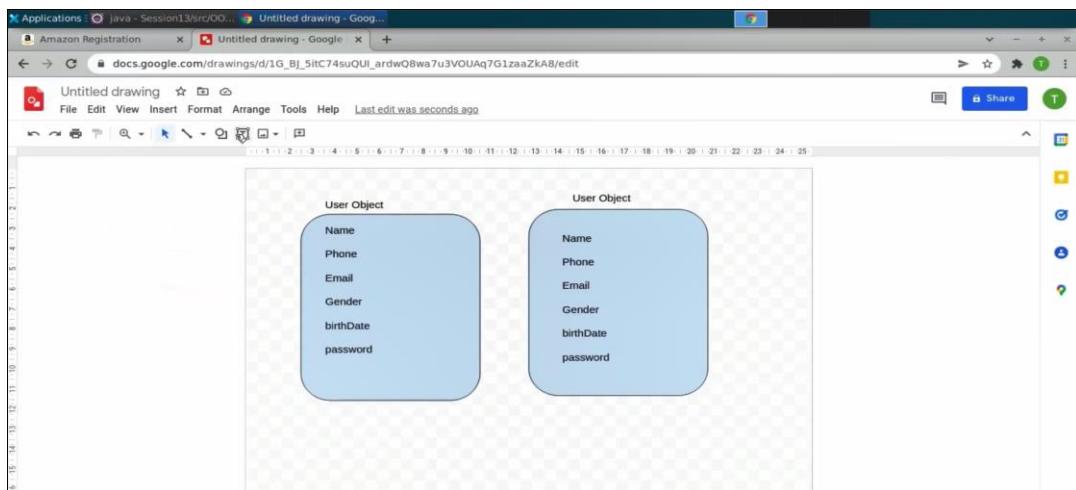
```

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

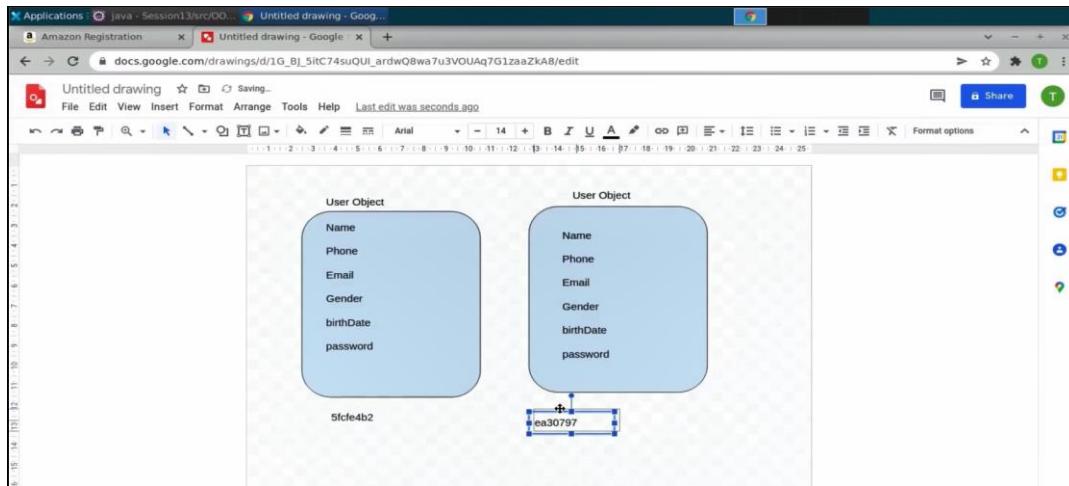


5.2 Inside the object, you will have the attributes which are the name, the phone, email, gender, birth date, and then the password

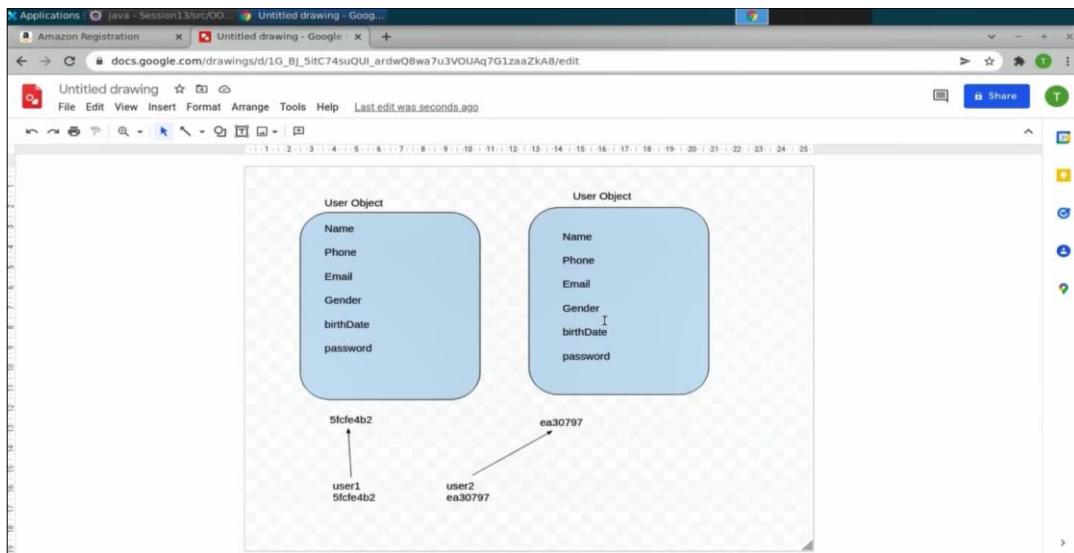


Step 6: Assign hash codes to the objects

- 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

```

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         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 }

```

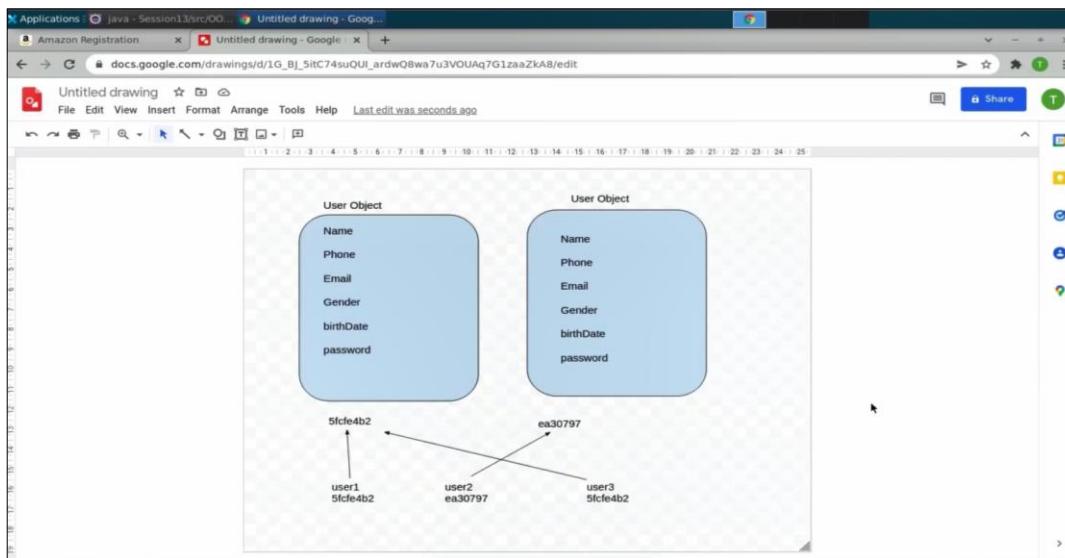
7.2 Now if you print **user 3**, you will notice that user one and user three have the same hash codes. Every time you run the program, hash codes may be different, so do not get confused on the part that why the hash codes are coming differently now

```

<terminated> OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jdk11/bin/java -Dfile.encoding=UTF-8 -cp . OOPS
user1 is: User@4c98385c
user2 is: User@73adffcc
user3 is: User@4c98385c

```

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

```

1 package com.simplilearn;
2
3 public class User {
4     String name;
5     String phone;
6     String email;
7     char gender;
8     Date birthDate;
9     String password;
10 }
11
12 public class OOPS {
13     public static void main(String[] args) {
14         // 3. From the class create the real object in memory
15         User user1 = new User();
16         User user2 = new User();
17         User user3 = new User();
18
19         System.out.println("user1 is: "+user1);
20         System.out.println("user2 is: "+user2);
21         System.out.println("user3 is: "+user3);
22
23         System.out.println("Reading Data from user1");
24         System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email);
25
26         System.out.println("Reading Data from user2");
27         System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email);
28
29         System.out.println("Reading Data from user3");
30         System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email);
31     }
32 }

```

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

```

 1 Applications java - Session13/src/OOPS.java Untitled drawing - Goog... Java - Session13/src/OOPS.java - Eclipse IDE
 2 File Edit Source Refactor Navigate Search Project Run Window Help
 3 Problems Javadoc Declaration Console X
 4
 5 OOPS.java X
 6 class User{
 7
 8     // Attributes: Property of Object
 9     String name;
10     String phone;
11     String email;
12     char gender;
13     Date birthdate;
14     String password;
15 }
16
17 public class OOPS {
18
19     public static void main(String[] args) {
20
21         // 3. From the class create the real object in memory
22         User user1 = new User();
23         User user2 = new User();
24
25         User user3 = user1; // Reference Copy Operation
26
27         System.out.println("user1 is: "+user1);
28         System.out.println("user2 is: "+user2);
29         System.out.println("user3 is: "+user3);
30
31         System.out.println("Reading Data from user1");
32         System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email);
33
34         System.out.println("Reading Data from user2");
35         System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email);
36
37         System.out.println("Reading Data from user3");
38         System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email);
39
40     }
41
42 }

```

The console output shows:

```

<terminated> OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.glass/jdk1.8.0_251/bin/java
user1 is: User@e4c9838c
user2 is: User@7aa8dfcc
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

```

 1 Applications java - Session13/src/OOPS.java Untitled drawing - Goog... Java - Session13/src/OOPS.java - Eclipse IDE
 2 File Edit Source Refactor Navigate Search Project Run Window Help
 3 Problems Javadoc Declaration Console X
 4
 5 OOPS.java X
 6 class User{
 7
 8     // Attributes: Property of Object
 9     String name;
10     String phone;
11     String email;
12     char gender;
13     Date birthdate;
14     String password;
15 }
16
17 public class OOPS {
18
19     public static void main(String[] args) {
20
21         // 3. From the class create the real object in memory
22         User user1 = new User();
23         User user2 = new User();
24
25         User user3 = user1; // Reference Copy Operation
26
27         System.out.println("user1 is: "+user1);
28         System.out.println("user2 is: "+user2);
29         System.out.println("user3 is: "+user3);
30
31         // Operations on Object
32         // 1. Write/Update i.e. Set the data
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         System.out.println("Reading Data from user2");
38         System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email);
39
40         System.out.println("Reading Data from user3");
41         System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email);
42
43     }
44
45 }

```

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"

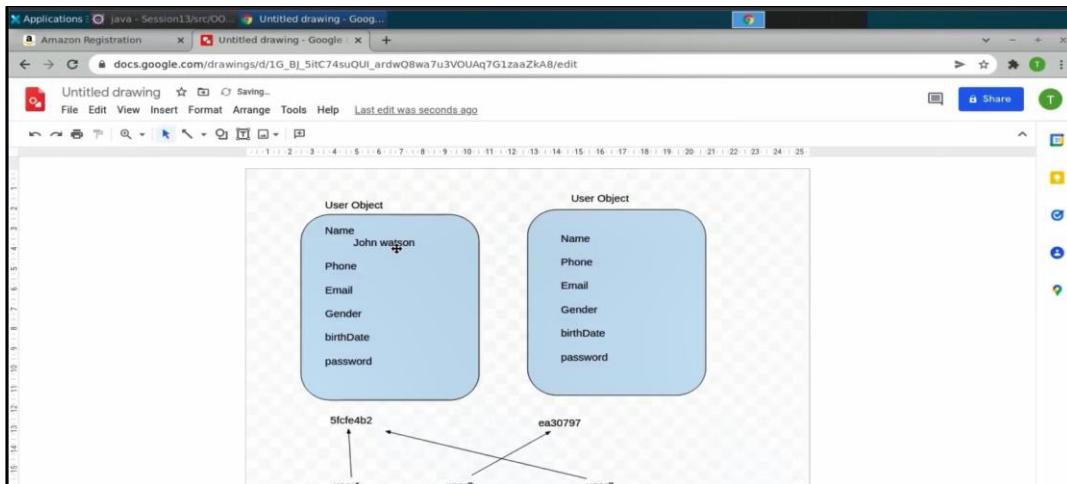
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - Java - Session13/src/OOPS... - Untitled drawing - Google...
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and navigation.
- Code Editor:** The main window displays Java code for a User class and a main method demonstrating object creation and modification.

```
File Edit Source Refactor Navigate Search Project Run Window Help

e [ OOPS.java x
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         // 3. From the class create the real object in memory
33         User user1 = new User();
34         User user2 = new User();
35
36         User user3 = user1; // Reference Copy Operation
37
38         System.out.println("user1 is: "+user1);
39         System.out.println("user2 is: "+user2);
40         System.out.println("user3 is: "+user3);
41
42         // Operations on Object
43         // 1. Write/Update i.e. Set the data
44         user1.name = "John";
45         user3.name = "John Watson";
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     }
51 }
```

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 // 2. Create the class
    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 1111";
   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     }
   38
   39 }

```

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 // 2. Create the class
    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 1111";
   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     }
   38
   39 }

```

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
# OOPS.java X
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

```

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
# OOPS.java X
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

```

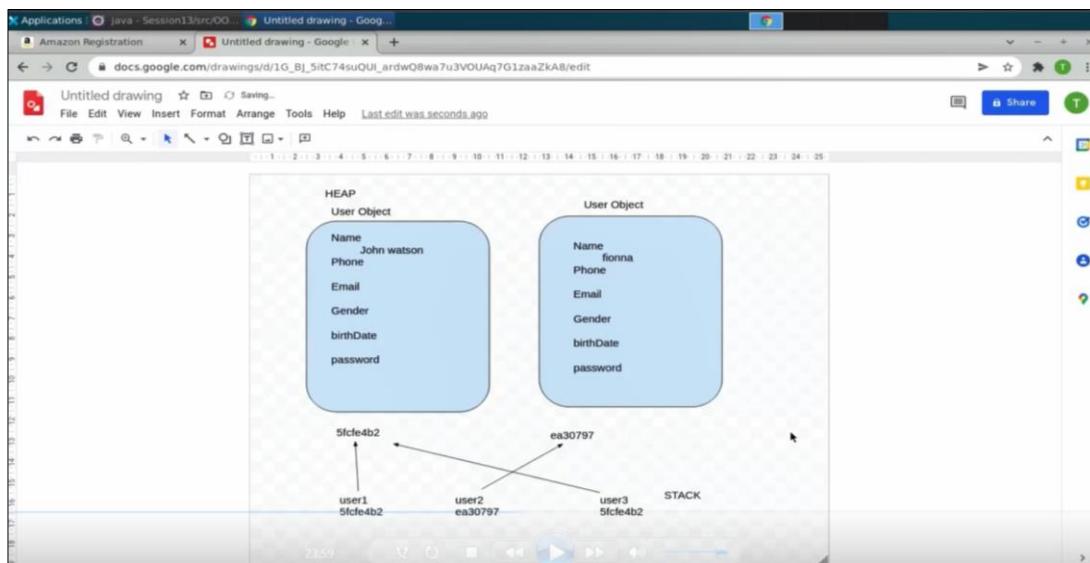
    35     User user1 = new User();
    36     User user2 = new User();
    37
    38     User user3 = user1; // Reference Copy Operation
    39
    40     System.out.println("user1 is: "+user1);
    41     System.out.println("user2 is: "+user2);
    42     System.out.println("user3 is: "+user3);
    43
    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 1111";
    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);
  
```

The console output shows:

```

<terminated> OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justj/openjdk.hotspot
user1 is: User@4c98385c
user2 is: User@73a8dfcc
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
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
  
```

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 //OOPS.java X
2 import java.util.Date;
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         I
30         name = "";
31         phone = "NA";
32         email = "demo@example.com";
33     }
34 }

```

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         I
30         name = "";
31         phone = "NA";
32         email = "demo@example.com";
33         gender = 'F';
34         birthDate = new Date();
35         password = "password";
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

```

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



The screenshot shows the Eclipse IDE interface with the title bar "java - Session13/src/OOPS.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The code editor displays Java code related to Object-Oriented Programming (OOPS) concepts:

```
File Edit Source Refactor Navigate Search Project Run Window Help

# *OOPS.java X
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
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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session13/src/OOPS.java - Untitled drawing - Google Chrome
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Java development tools (New, Open, Save, Cut, Copy, Paste, Find, etc.)
- Code Editor:** Displays the `OOPS.java` file containing Java code for creating objects and printing their details.
- Output View:** Shows the execution results of the code, including user IDs and their details, and a message indicating the object can be called at +91 99999 11111 and emailed at john.watson@example.com.

```
System.out.println("user1 is: "+user1);
System.out.println("user2 is: "+user2);
System.out.println("user3 is: "+user3);
System.out.println("user4 is: "+user4);

// Operations on Object
// 1. Write/Update i.e. Set the data
user1.name = "john watson";
user1.name = "john watson";

user3.phone = "+91 99999 11111";
user3.email = "john.watson@example.com";
user1.birthDate = new Date();
user1.password = "john@123";

user2.name = "fionna";
user2.phone = "+91 98765 12345";
user2.email = "fionna@example.com";
user2.birthDate = new Date();
user2.password = "fionna@123";

// 2. Read the Data
System.out.println("Reading Data from user1");
System.out.println(user1.name+" can be called at "+user1.phone+" and can be emailed at "+user1.email+" birth date: "+user1.birthDate);

System.out.println("Reading Data from user2");
System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);

System.out.println("Reading Data from user3");
System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);

System.out.println("Reading Data from user4");
System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
```

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 User(String n, String phone, String e, char gender, Date birthDate, String password){
 39     name = n;
 40     this.phone = phone;
 41     email = e;
 42     this.gender = gender;
 43     this.birthDate = birthDate;
 44     this.password = password;
 45 }
 46
 47 }
 48
 49 public class OOPS {
 50
 51     public static void main(String[] args) {
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1298
 1299
 1300
 1301
 1302
 1303
 1304
 1305
 1306
 1307
 1308
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1510
 1511
 1512
 1513
 1514
 1515
 1516
 1517
 1518
 1519
 1520
 1521
 1522
 1523
 1524
 1525
 1526
 1527
 1528
 1529
 1530
 1531
 1532
 1533
 1534
 1535
 1536
 1537
 1538
 1539
 1540
 1541
 1542
 1543
 1544
 1545
 1546
 1547
 1548
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1559
 1560
 1561
 1562
 1563
 1564
 1565
 1566
 1567
 1568
 1569
 1570
 1571
 1572
 1573
 1574
 1575
 1576
 1577
 1578
 1579
 1580
 1581
 1582
 1583
 1584
 1585
 1586
 1587
 1588
 1589
 1590
 1591
 1592
 1593
 1594
 1595
 1596
 1597
 1598
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676
 1677
 1678
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1789
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889

```

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     user3.name = "john";
71     user3.name = "john watson";
72
73     user3.phone = "+91 99999 1111";
74     user3.email = "john.watson@example.com";
75     user3.birthDate = new Date();
76     user3.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     user3.name = "john";
71     user3.name = "john watson";
72
73     user3.phone = "+91 99999 1111";
74     user3.email = "john.watson@example.com";
75     user3.birthDate = new Date();
76     user3.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");

```

Output:

```

<terminated>- OOPS [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot
user1 is: User@ea80797
user2 is: User@8d25a40
user3 is: User@ea80797
user4 is: User@1b701d1
Reading Data from user1
john can be called at +91 99999 1111 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 1111 and can be emailed at john.watson@example.com
Reading Data from user4
Leo can be called at +91 98765 12345 and can be emailed at leo@example.com
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**

```

1 Applications : java - Session13/src/OOPS.java - Untitled drawing - Google Chrome
2 File Edit Source Refactor Navigate Search Project Run Window Help
3
4 OOPS.java
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     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
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
63     // 3. From the class create the real object in memory
64     User user1 = new User();

```

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 90909", "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

```

 66 User user3 = user1; // Reference Copy Operation
 67
 68 User user4 = new User();
 69 User user5 = new User("Leo", "+91 98765 90909", "leo@example.com", 'M', new Date());
 70
 71 System.out.println("user1 is: "+user1);
 72 System.out.println("user2 is: "+user2);
 73 System.out.println("user3 is: "+user3);
 74 System.out.println("user4 is: "+user4);
 75 System.out.println("user5 is: "+user5);
 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");
100 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
101
102 System.out.println("Reading Data from user3");
103 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);

```

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

```

 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   void showUserData() {
 58     System.out.println("-----");
 59     System.out.println(name+" can be called at "+phone);
 60     System.out.println(name+" can be emailed at "+email);
 61     System.out.println(name+" has a gender: "+gender+" and was born on "+birthDate);
 62     System.out.println("-----");
 63   }
 64
 65 }
 66
 67 public class OOPS {
 68
 69   public static void main(String[] args) {
 70
 71     // 3. From the class create the real object in memory
 72     User user1 = new User();
 73     User user2 = new User();
 74
 75     User user3 = user1; // Reference Copy Operation
 76
 77     User user4 = new User();

```

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



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session13/src/OOPS.java - Untitled drawing - Google...
- Toolbar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Code Editor:** The file OOPS.java is open, displaying Java code related to user objects and their data.
- Code Content:**

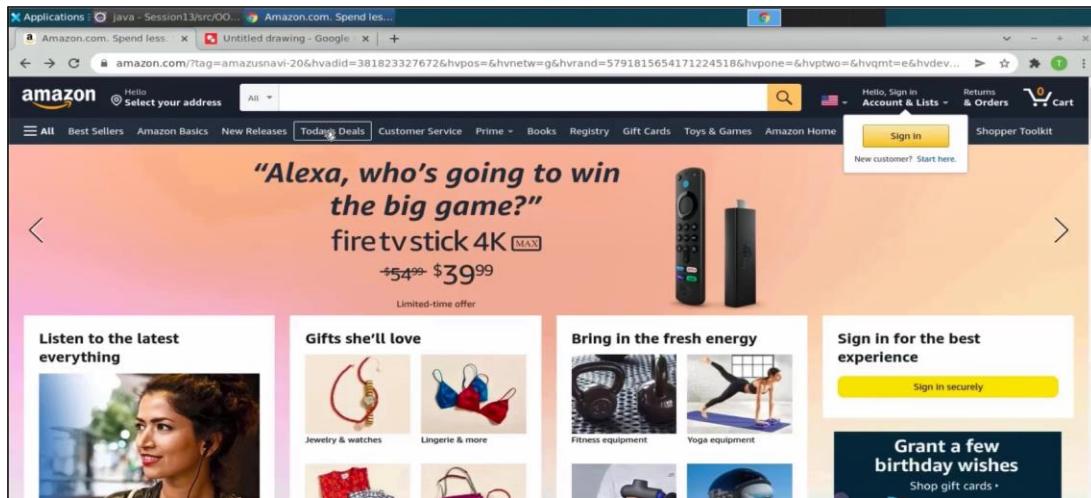
```
user3.name = "john watson";
90
91 user3.phone = "+91 99999 1111";
92 user3.email = "john.watson@example.com";
93 user1.birthDate = new Date();
94 user1.password = "john@123";
95
96 // user2.name = "fionna";
97 // user2.phone = "+91 98765 12345";
98 // user2.email = "fionna@example.com";
99 // user2.birthDate = new Date();
100 // user2.password = "fionna@123";
101
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();
```

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

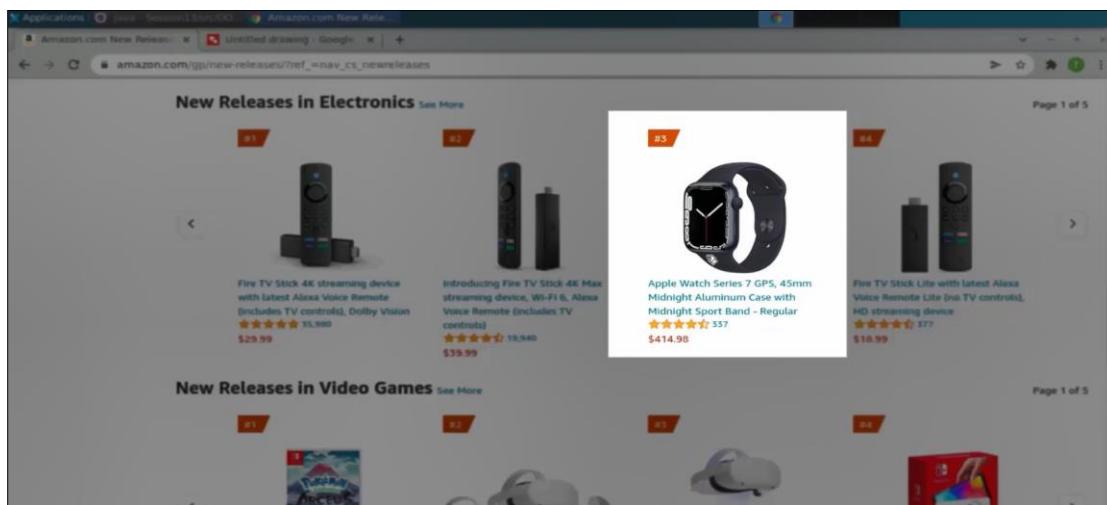
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Applications - java - Session13/src/OOPS.java - Untitled drawing - Google...
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Side:** A package explorer window showing a single file named "OOPS.java X".
- Code Editor:** The code for "OOPS.java" is displayed. It defines a class User with various fields and methods. It also contains several System.out.println statements to print user details and their corresponding calling methods.
- Console View:** The "Console" tab is selected in the top right. The output shows the results of the println statements, including:
 - "john watson can be called at +91 99999 1111"
 - "john watson can be emailed at john.watson@example.com"
 - "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"
- Bottom Status Bar:** Shows the current file path as "Session13/src/OOPS.java" and the line number "118".

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



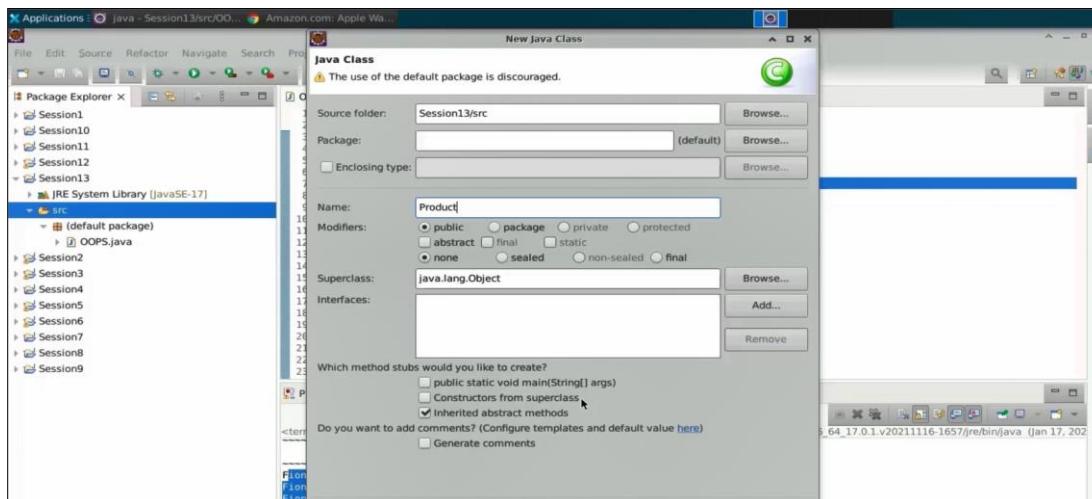
13.6 You can click on one of the products called the Apple Watch series.



- 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



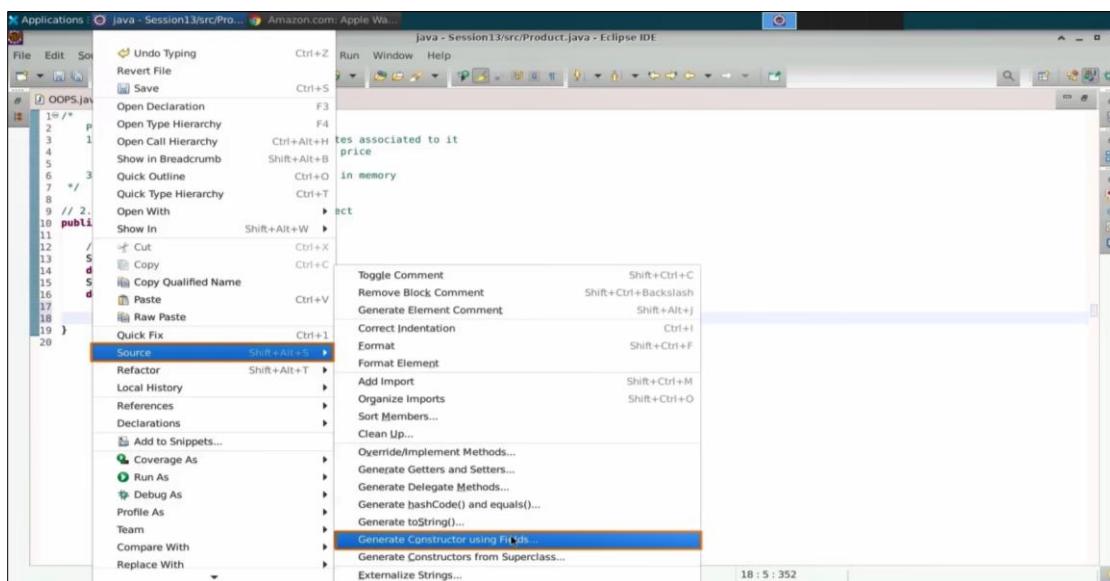
13.9 For the product object, the first key step is to identify the object and list the attributes associated with it. So, the object should be named **Product**, and it should have a name, ratings, description, and price. Let us now define these attributes. The name is going to be a string. The ratings 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

```

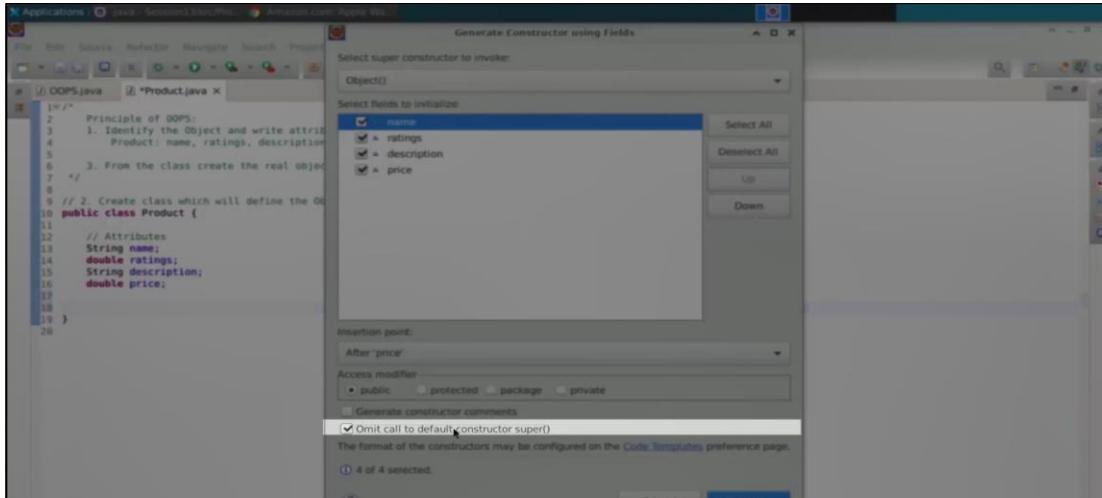
11 /*
2 * Principle of OOPS:
3 * 1. Identify the Object and write attributes associated to it
4 *     Product: name, ratings, description, price
5 *
6 * 3. From the class create the real object in memory
7 */
8
9 // 2. Create class which will define the Object
10 public class Product {
11
12     // Attributes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

```

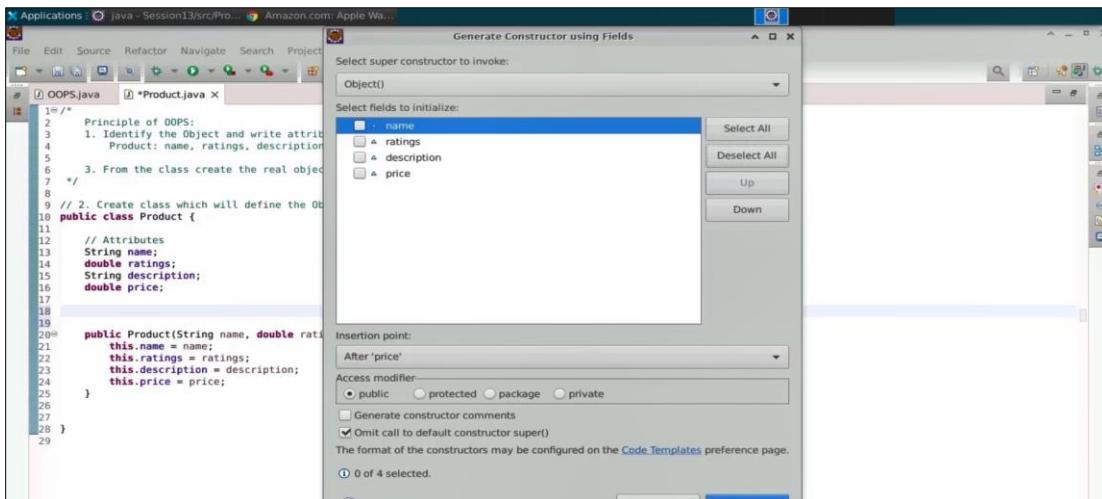
13.10 In order to make things a bit simpler, you can do right click and select **source > generate the constructor using the fields**



- 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**

The screenshot shows the Eclipse IDE interface with the title bar "java - Session 13/src/Pro..." and "Amazon.com: Apple Wa...". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The left sidebar shows the package structure with "OOPS.java" and "Product.java X". The main editor area contains the following Java code:

```
9 // 2. Create class which will define the Object
10 class Product {
11
12     // Attributes
13     String name;
14     double ratings;
15     String description;
16     double price;
17
18     // Default Constructor
19     Product() {
20
21     }
22
23     // Parameterized Constructor
24     Product(String name, double ratings, String description, double price) {
25         this.name = name;
26         this.ratings = ratings;
27         this.description = description;
28         this.price = price;
29     }
30
31     void setProductData(String name, double ratings, String description, double price) {
32         this.name = name;
33         this.ratings = ratings;
34         this.description = description;
35         this.price = price;
36     }
37
38     void showProductData() {
39         System.out.println("Product "+name+" has "+ratings+" ratings");
40         System.out.println("Product "+name+" is priced at $"+price);
41         System.out.println("Product "+name+" description is "+description);
42         System.out.println("-----");
43     }
44
45
46 }
```

- 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**

```
File Edit Source Refactor Navigate Search Project Run Window Help
*OOPS.java | Product.java
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 System.out.println("Reading Data from user2");
108 System.out.println(user2.name+" can be called at "+user2.phone+" and can be emailed at "+user2.email+" birth date: "+user2.birthDate);
109 System.out.println("Reading Data from user3");
110 System.out.println(user3.name+" can be called at "+user3.phone+" and can be emailed at "+user3.email+" birth date: "+user3.birthDate);
111 System.out.println("Reading Data from user4");
112 System.out.println(user4.name+" can be called at "+user4.phone+" and can be emailed at "+user4.email+" birth date: "+user4.birthDate);
113 System.out.println("Reading Data from user5");
114 System.out.println(user5.name+" can be called at "+user5.phone+" and can be emailed at "+user5.email+" birth date: "+user5.birthDate);/*
115 user1.showuserData();
116 user2.showuserData();
117 user3.showuserData();
118 user4.showuserData();
119 user5.showuserData();
120
121 Product p1 = new Product();
122 Product p2 = new Product("iphone 11 pro max", 4.7, "Apple iPhone with 3 cameras", 800);
123 Product p3 = new Product();
124 p3.setProductData("Ultraboost Shoe", 5.0, "Adidas comfortable Ultraboost Shoes 21", 200);
125
126
127
128
129
130
131
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

```

File Edit Source Refactor Navigate Search Project Run Window Help
# OOPS.java X Product.java
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
138     //3. Delete the data -> Automatically
139     System.gc();

```

Writable Smart Insert 135 : 30 : 4161

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<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 98909
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.