

Lesson 02 Demo 04

Overloading Methods in Java

Objective: To depict how to overload methods in Java

Tools required: Eclipse IDE

Prerequisites: None

Steps to be followed:

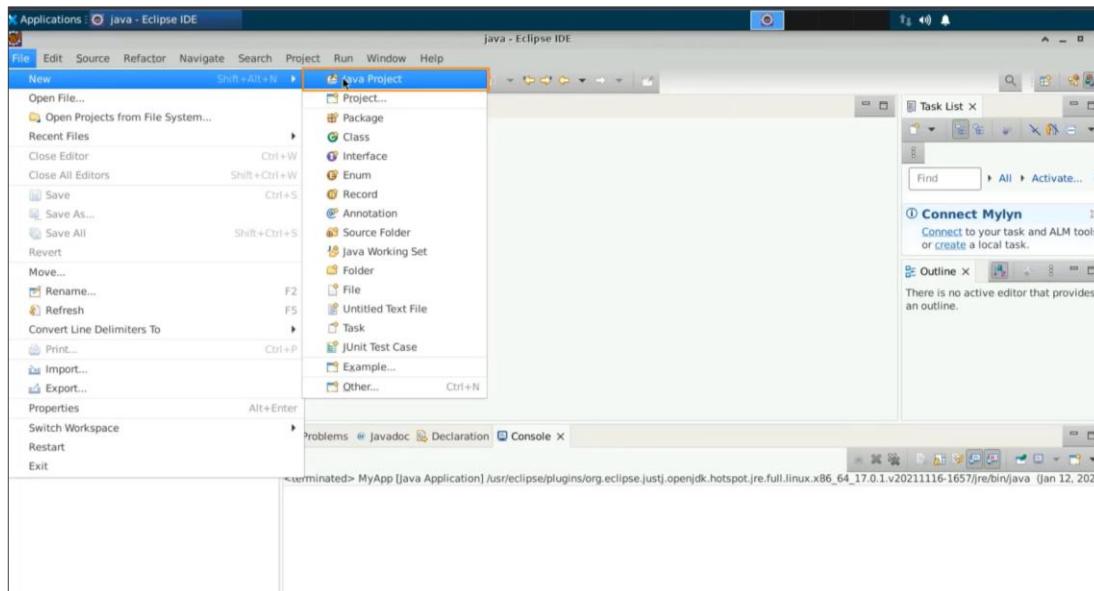
1. Write and implement method overloading rules in Java
2. Implement compiler time polymorphism
3. Test the overloaded methods
4. Run the code and execute the result
5. Write methods for authentication

Step 1: Write and implement method overloading rules in Java

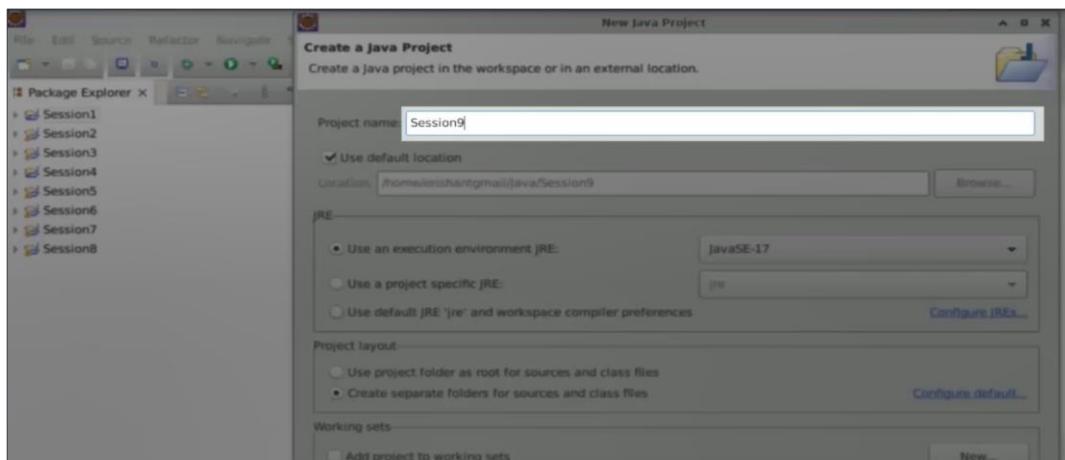
1.1 Open the **Eclipse IDE**



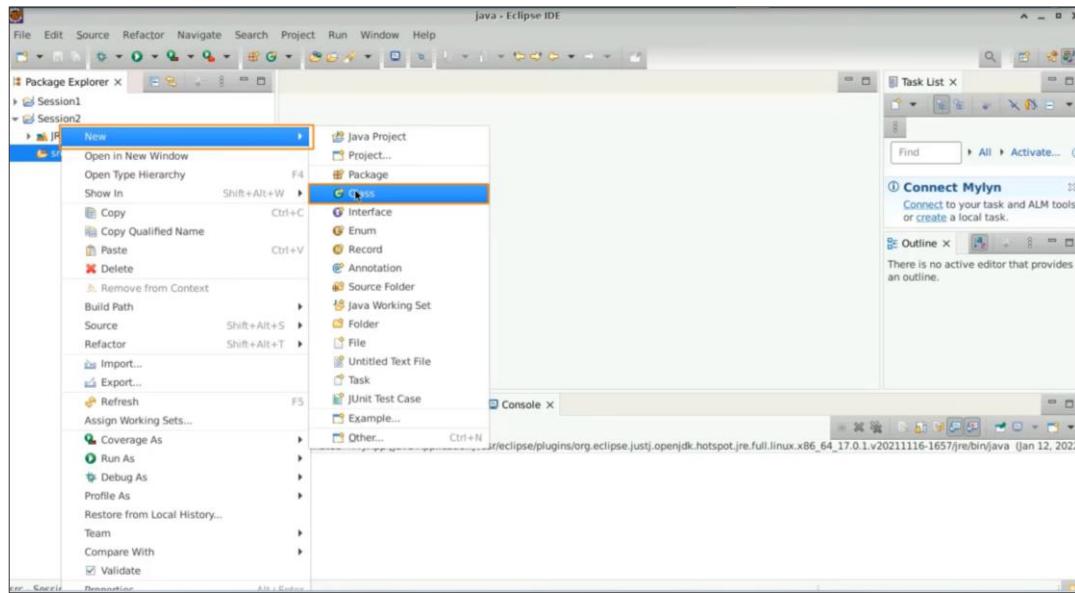
1.2. Select File, then New, and then Java project



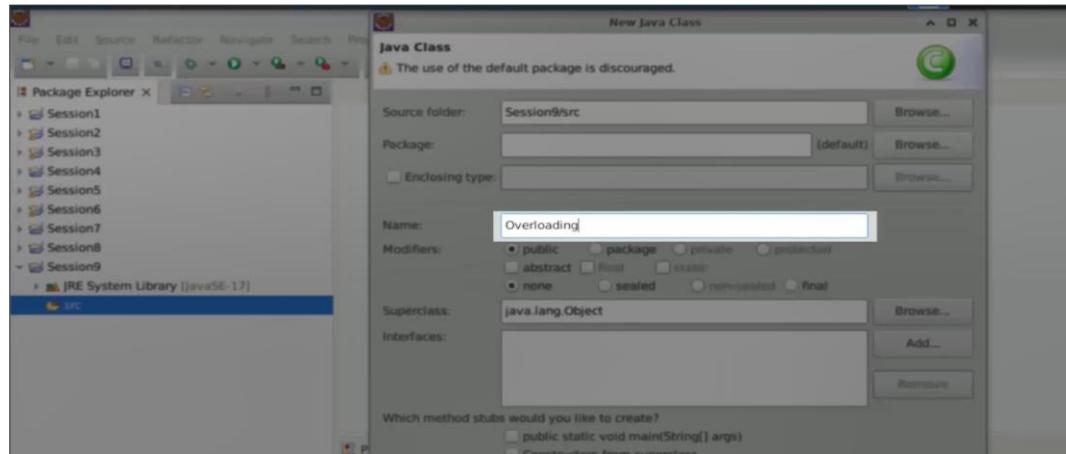
1.3 Name the project “Session9”, uncheck “Create a module info dot Java file”, and press Finish



1.4 With a **Session9** on the src, do a right-click and create a new class

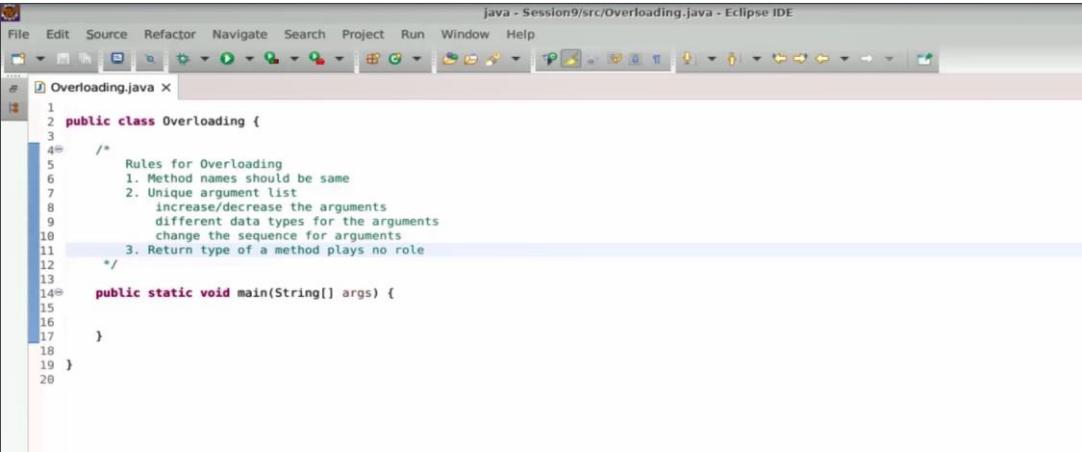


1.5 Name this class as an **Overloading**, then select the **main method**, and then select **finish**



Step 2: Implement compiler time polymorphism

2.1 To implement overloading, you need to follow certain rules. The very first rule is, that the method name should be the same, the second rule will be a Unique Argument List, which means you can increase or decrease the arguments. Then you can have different Data types for the arguments, then you can even change the sequence for arguments, and the third rule is the return type of a method Plays no role

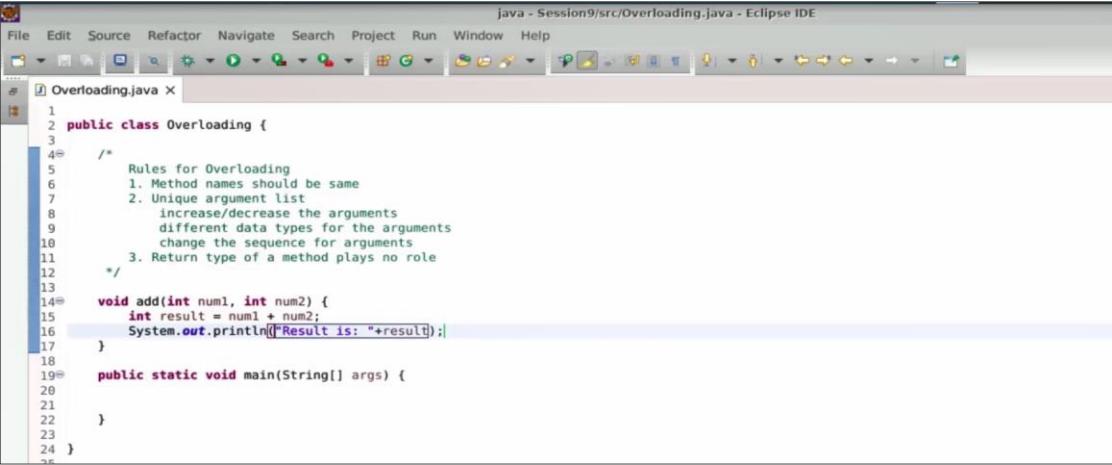


```

1  public class Overloading {
2
3     /*
4      * Rules for Overloading
5      * 1. Method names should be same
6      * 2. Unique argument list
7      *    increase/decrease the arguments
8      *    different data types for the arguments
9      *    change the sequence for arguments
10     3. Return type of a method plays no role
11    */
12
13    public static void main(String[] args) {
14
15    }
16
17 }
18
19
20
21
22
23
24
25

```

2.2 Create a method called add that takes two integer inputs, **integer1** and **integer2**. Write the equation **result = integer1 + integer2**. Then, print **Result is:** followed by the value of result



```

1  public class Overloading {
2
3     /*
4      * Rules for Overloading
5      * 1. Method names should be same
6      * 2. Unique argument list
7      *    increase/decrease the arguments
8      *    different data types for the arguments
9      *    change the sequence for arguments
10     3. Return type of a method plays no role
11    */
12
13    void add(int num1, int num2) {
14        int result = num1 + num2;
15        System.out.println("Result is: "+result);
16    }
17
18    public static void main(String[] args) {
19
20    }
21
22 }
23
24
25

```

2.3 If in case you try to create the same method with the same name and the same inputs, you will get an error, hence there cannot be ambiguity for a method. Hence, you can have a method overload

```

1  public class Overloading {
2
3     /*
4      * Rules for Overloading
5      * 1. Method names should be same
6      * 2. Unique argument list
7      *    increase/decrease the arguments
8      *    different data types for the arguments
9      *    change the sequence for arguments
10     * 3. Return type of a method plays no role
11   */
12
13  void add(int num1, int num2) {
14      int result = num1 + num2;
15      System.out.println("Result is: "+result);
16  }
17
18  void add(int num1, int num2) {
19      int result = num1 + num2;
20      System.out.println("Result is: "+result);
21  }
22
23
24  public static void main(String[] args) {
25

```

Step 3: Test the overloaded methods

3.1 Now, let us use x_1 and x_2 as the variables, and define the result as $x_1 + x_2 + 20$ to ensure it would not have any impact. The uniqueness of the method comes from the list of arguments or inputs you are trying to pass. If you try to execute this method, it would be confusing whether the two integers should be passed as number1, number2, or x_1 , x_2

```

1  public class Overloading {
2
3     /*
4      * Rules for Overloading
5      * 1. Method names should be same
6      * 2. Unique argument list
7      *    increase/decrease the arguments
8      *    different data types for the arguments
9      *    change the sequence for arguments
10     * 3. Return type of a method plays no role
11   */
12
13  void add(int num1, int num2) {
14      int result = num1 + num2;
15      System.out.println("Result is: "+result);
16  }
17
18  void add(int x1, int x2) {
19      int result = x1 + x2 + 20;
20      System.out.println("Result is: "+result);
21  }
22
23
24  public static void main(String[] args) {
25

```

3.2 Now in a scenario, let the Return type be an integer, where you will write Return the result. This will also not be a solution for overloading

```

1  public class Overloading {
2
3     /*
4      * Rules for Overloading
5      * 1. Method name should be same
6      * 2. Unique argument list
7      *    increase/decrease the arguments
8      *    different data types for the arguments
9      *    change the sequence for arguments
10     3. Return type of a method plays no role
11   */
12
13   void add(int num1, int num2) {
14       int result = num1 + num2;
15       System.out.println("Result is: "+result);
16   }
17
18   int add(int x1, int x2) {
19       int result = x1 + x2 + 20;
20       System.out.println("Result is: "+result);
21       return result;
22   }
23
24
25   public static void main(String[] args) {
26
27
28   }
29
30 }
31

```

3.3 Write integer number 3. With another integer number 3, you can have a different unique list. The method name is the same, and this is where overloading has been implemented. This is known as compile-time polymorphism, where the compiler will distinguish based on the method execution call, which definition is supposed to be executed

```

1  public class Overloading {
2
3     /*
4      * Rules for Overloading
5      * 1. Method name should be same
6      * 2. Unique argument list
7      *    increase/decrease the arguments
8      *    different data types for the arguments
9      *    change the sequence for arguments
10     3. Return type of a method plays no role
11   */
12
13   void add(int num1, int num2) {
14       int result = num1 + num2;
15       System.out.println("Result is: "+result);
16   }
17
18   /*int add(int x1, int x2) {
19       int result = x1 + x2 + 20;
20       System.out.println("Result is: "+result);
21       return result;
22   }
23 */
24
25   void add(int num1, int num2, int num3) {
26       int result = num1 + num2 + num3;
27       System.out.println("Result is: "+result);
28   }
29
30   public static void main(String[] args) {
31
32   }
33
34 }

```

3.4 Instead of adding the numbers with integers, you can add the numbers as doubles, and this will also help us to implement the method overloading. That is the same name with the different inputs

```

1  public class Overloading {
2
3      /*
4       * Rules For Overloading
5       * 1. Method names should be same
6       * 2. Unique argument list
7       *    increase/decrease the arguments
8       *    different data types for the arguments
9       *    change the sequence for arguments
10      3. Return type of a method plays no role
11  */
12
13  void add(int num1, int num2) {
14      int result = num1 + num2;
15      System.out.println("Result is: "+result);
16  }
17
18  /*int add(int x1, int x2) {
19      int result = x1 + x2 + 20;
20      System.out.println("Result is: "+result);
21      return result;
22  }*/
23
24  void add(int num1, int num2, int num3) {
25      int result = num1 + num2 + num3;
26      System.out.println("Result is: "+result);
27  }
28
29  void add(double num1, double num2) {
30      double result = num1 + num2;
31      System.out.println("Result is: "+result);
32  }
33
34
35}

```

3.5 Now, for the sequence of the arguments, there can be an integer number1 and a double number2, and then you can change the sequence to double number1 and integer number2

```

1  public class Overloading {
2
3      /*
4       * Method names should be same
5       * 1. Unique argument list
6       *    increase/decrease the arguments
7       *    different data types for the arguments
8       *    change the sequence for arguments
9       * 3. Return type of a method plays no role
10  */
11
12  void add(int num1, int num2) {
13      int result = num1 + num2;
14      System.out.println("Result is: "+result);
15  }
16
17  /*int add(int x1, int x2) {
18      int result = x1 + x2 + 20;
19      System.out.println("Result is: "+result);
20      return result;
21  }*/
22
23  void add(int num1, int num2, int num3) {
24      int result = num1 + num2 + num3;
25      System.out.println("Result is: "+result);
26  }
27
28  void add(double num1, double num2) {
29      double result = num1 + num2;
30      System.out.println("Result is: "+result);
31  }
32
33  void add(double num1, double num2) {
34      double result = num1 + num2;
35      System.out.println("Result is: "+result);
36  }
37
38  void add(int num1, double num2) {
39      double result = num1 + num2;
40      System.out.println("Result is: "+result);
41  }
42
43}

```

3.6 Create a reference variable and use method overloading. Write add, and inside the add method, you can pass two integers to add two numbers. With the same reference, you can call add to pass two doubles or floating-point numbers. Then with your reference, you can write **add(10, 20, 30)**. With the same reference, you can add one integer and the next one as a double. You can even use **reference.add(double, int)**

```

15     int result = num1 + num2;
16     System.out.println("Result is: "+result);
17   }
18
19*  /*int add(int x1, int x2) {
20*   int result = x1 + x2 + 20;
21*   System.out.println("Result is: "+result);
22*   return result;
23* }
24
25* void add(int num1, int num2, int num3) {
26*   int result = num1 + num2 + num3;
27*   System.out.println("Result is: "+result);
28* }
29
30* void add(double num1, double num2) {
31*   double result = num1 + num2;
32*   System.out.println("Result is: "+result);
33* }
34
35* void add(int num1, double num2) {
36*   double result = num1 + num2;
37*   System.out.println("Result is: "+result);
38* }
39
40* void add(double num1, int num2) {
41*   double result = num1 + num2;
42*   System.out.println("Result is: "+result);
43* }
44
45* public static void main(String[] args) {
46*
47*   Overloading ref = new Overloading();
48*   ref.add(10, 20);
49*   ref.add(22.5, 44.67);
50*   ref.add(10, 20, 30);
51*   ref.add(10, 2.2);
52*   ref.add(10, 30);
53* }

```

Step 4: Run the code and execute the result

4.1 Run this code, you can now see that all of the methods are executed and you are able to compute the result

```

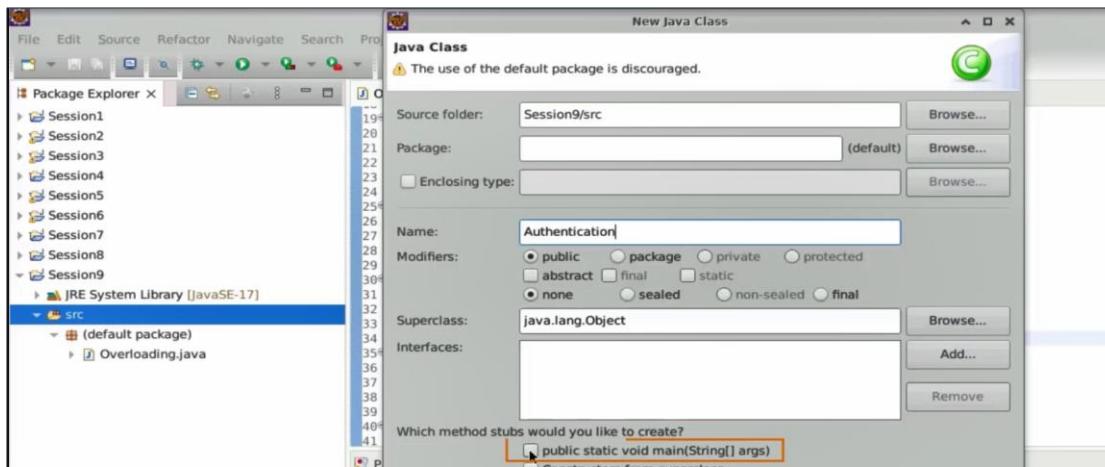
19*  /*int add(int x1, int x2) {
20*   int result = x1 + x2 + 20;
21*   System.out.println("Result is: "+result);
22*   return result;
23* }
24
25* void add(int num1, int num2, int num3) {
26*   int result = num1 + num2 + num3;
27*   System.out.println("Result is: "+result);
28* }
29
30* void add(double num1, double num2) {
31*   double result = num1 + num2;
32*   System.out.println("Result is: "+result);
33* }
34
35* void add(int num1, double num2) {
36*   double result = num1 + num2;
37*   System.out.println("Result is: "+result);
38* }
39
40* void add(double num1, int num2) {
41*   double result = num1 + num2;
42*   System.out.println("Result is: "+result);
43* }
44
45* public static void main(String[] args) {
46*
47*   Overloading ref = new Overloading();
48*   ref.add(10, 20);
49*   ref.add(22.5, 44.67);
50*   ref.add(10, 20, 30);
51*   ref.add(10, 2.2);
52*   ref.add(10, 30);
53* }

```

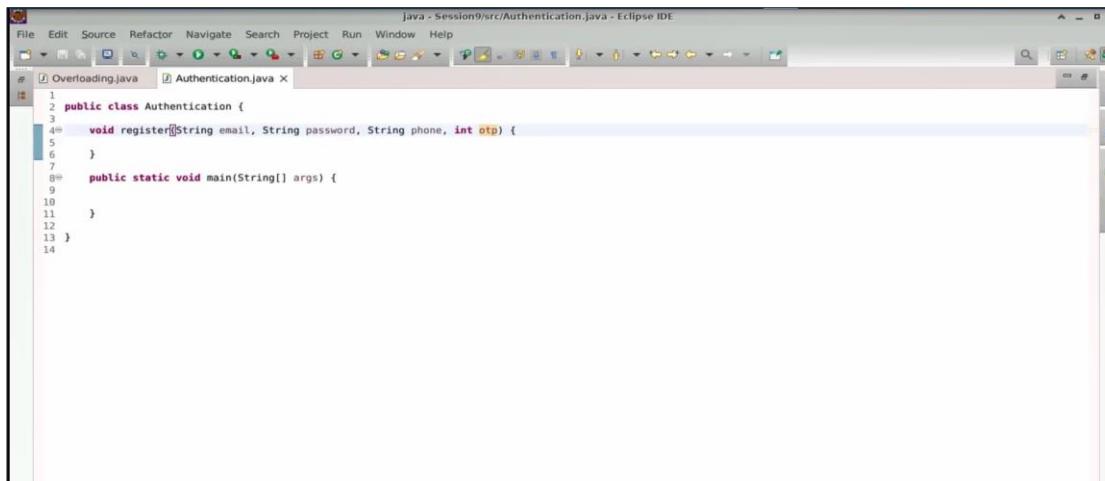
<terminated> Overloading [Java Application] /usr/local/eclipse/jee-neon/eclipse-workspace/Overloading/Overloading

Result is: 30
Result is: 67.17
Result is: 60
Result is: 12.2
Result is: 121.76

4.2 Create a new class and name this as authentication, with the main method.



4.3 If you want the user to register in your application, create a method called register. For the registration, there can be several inputs. The first input is email, and the second input is password. Next, you will pass one more input for example, this input is a phone number and an integer called OTP



4.4 Add a simple check: if the email is not empty using (`!email.isEmpty()`), then register the user with an email and password. Create an else-if case: if the phone number is not empty, register the user with the phone number and OTP

```

java - Session9/src/Authentication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Authentication.java X
1  public class Authentication {
2
3      void register(String email, String password, String phone, int otp) {
4
5          if(!email.isEmpty()) {
6              System.out.println("Register the User with Email and Password");
7          }else if(!phone.isEmpty()) {
8              System.out.println("Register the User with Phne and OTP");
9          }
10     }
11 }
12
13
14 public static void main(String[] args) {
15
16 }
17
18 }
19
20
21
22
23
24

```

4.5 In case you need to add more registration methods, you need to have a lot of other else If structures, and finally an else. So you can break down this method with the help of overloading

```

java - Session9/src/Authentication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Authentication.java X
1  public class Authentication {
2
3      void register(String email, String password, String phone, int otp) {
4
5          if(!email.isEmpty()) {
6              System.out.println("Register the User with Email and Password");
7          }else if(!phone.isEmpty()) {
8              System.out.println("Register the User with Phne and OTP");
9          }else if(true) {
10
11      }else {
12
13
14
15
16
17
18
19
20
21
22
23
24
}

```

4.6 You can write void register, and you can take the first thing as email and the second thing as a password. Next, print down the user registration with email and password

```

java - Session9/src/Authentication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Authentication.java
1 public class Authentication {
2     /*void register(String email, String password, String phone, int otp) {
3         if(!email.isEmpty()) {
4             System.out.println("Register the User with Email and Password");
5         }else if(!phone.isEmpty()) {
6             System.out.println("Register the User with Phne and OTP");
7         }else if(true) {
8             }
9         }*/
10     void register(String email, String password) {
11         System.out.println("User Registration with Email and Password");
12     }
13     public static void main(String[] args) {
14     }
15 }
16 */
17
18
19
20
21
22
23
24
25
26
27
28

```

4.7 The other one can be your string phone number and an integer OTP that is a One-time password. Here, user registration with phone and OTP

```

java - Session9/src/Authentication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Authentication.java
1 public class Authentication {
2     /*void register(String email, String password, String phone, int otp) {
3         if(!email.isEmpty()) {
4             System.out.println("Register the User with Email and Password");
5         }else if(!phone.isEmpty()) {
6             System.out.println("Register the User with Phne and OTP");
7         }else if(true) {
8             }
9         }*/
10     void register(String email, String password) {
11         System.out.println("User Registration with Email and Password");
12     }
13     void register(String phone, int otp) {
14         System.out.println("User Registration with Phone and OTP");
15     }
16     public static void main(String[] args) {
17     }
18 }
19
20
21
22
23
24
25
26
27
28
29
30
31
32

```

4.8 Next, you can even Register with Google, where you will take one Google email as input and print down this as user registration with Google. And you can have different ways to do registration for a user

```

File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Authentication.java
1  public class Authentication {
2
3      //void register(String email, String password, String phone, int otp) {
4
5          if(!email.isEmpty()) {
6              System.out.println("Register the User with Email and Password");
7          }else if(phone.isEmpty()) {
8              System.out.println("Register the User with Phone and OTP");
9          }else if(true) {
10
11      }
12  }
13
14
15 */
16
17     void register(String email, String password) {
18         System.out.println("User Registration with Email and Password");
19     }
20
21
22     void register(String phone, int otp) {
23         System.out.println("User Registration with Phone and OTP");
24     }
25
26     void registerWithGoogle(String googleEmail) {
27         System.out.println("User Registration with Google");
28     }
29
30     public static void main(String[] args) {
31
32
33 }
34

```

4.9 Like the way you got registration methods, similarly, you can create log-in methods, so you can log in the user with email and password, log in the user with the phone and OTP, and also, you can even log in as a user with a Google account

```

File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Authentication.java
11
12     }
13
14
15 */
16
17     void register(String email, String password) {
18         System.out.println("User Registration with Email and Password");
19     }
20
21
22     void register(String phone, int otp) {
23         System.out.println("User Registration with Phone and OTP");
24     }
25
26     void registerWithGoogle(String googleEmail) {
27         System.out.println("User Registration with Google");
28     }
29
30     void login(String email, String password) {
31         System.out.println("User Registration with Email and Password");
32     }
33
34     void login(String phone, int otp) {
35         System.out.println("User Registration with Phone and OTP");
36     }
37
38     void loginWithGoogle(String googleEmail) {
39         System.out.println("User Registration with Google");
40     }
41
42     public static void main(String[] args) {
43

```

Step 5: Write methods for authentication

5.1 The benefit is very simple, you have now divided your complex logic, and you have created different methods, and the user can anytime say auth dot register, which is a reference variable to your authentication object

```

11
12     }else {
13
14     }
15
16 */
17
18 void register(String email, String password) {
19     System.out.println("User Registration with Email and Password");
20 }
21
22 void register(String phone, int otp) {
23     System.out.println("User Registration with Phone and OTP");
24 }
25
26 void register(String googleEmail) {
27     System.out.println("User Registration with Google");
28 }
29
30 void login(String email, String password) {
31     System.out.println("User Registration with Email and Password");
32 }
33
34 void login(String phone, int otp) {
35     System.out.println("User Registration with Phone and OTP");
36 }
37
38 void login(String googleEmail) {
39     System.out.println("User Registration with Google");
40 }
41
42 public static void main(String[] args) {
43
44     Authentication auth = new Authentication();
45     //auth.register("john@example.com", "john123");
46     //auth.register("+91 99999 1111", 2345);
47     auth.register("john@gmail.com");
48 }

```

5.2 The logic can be written inside these methods. For example, if your googleEmail.endsWith ("@gmail.com"), only then you will register with Google, and in the else case, you can do some other processing

```

11
12     }else {
13
14     }
15
16 */
17
18 void register(String email, String password) {
19     System.out.println("User Registration with Email and Password");
20 }
21
22 void register(String phone, int otp) {
23     System.out.println("User Registration with Phone and OTP");
24 }
25
26 void register(String googleEmail) {
27     if(googleEmail.endsWith("@gmail.com")) {
28         System.out.println("User Registration with Google");
29     }else {
30         System.out.println("Invalid Google Account");
31     }
32 }
33
34 void login(String email, String password) {
35     System.out.println("User Registration with Email and Password");
36 }
37
38 void login(String phone, int otp) {
39     System.out.println("User Registration with Phone and OTP");
40 }
41
42 void login(String googleEmail) {
43     System.out.println("User Registration with Google");
44 }

```

5.3 Run the code, it states user registration with Google

The screenshot shows the Eclipse IDE interface with the title bar "Java - Session9/src/Authentication.java - Eclipse IDE". In the left panel, there are two tabs: "Overloading.java" and "Authentication.java X". The "Authentication.java" tab is active, displaying the following Java code:

```

19     System.out.println("User Registration with Email and Password");
20 }
21
22 void register(String phone, int otp) {
23     System.out.println("User Registration with Phone and OTP");
24 }
25
26 void register(String googleEmail) {
27     if(googleEmail.endsWith("@gmail.com")) {
28         System.out.println("User Registration with Google");
29     }else {
30         System.out.println("Invalid Google Account");
31     }
32 }
33
34 void login(String email, String password) {
35     System.out.println("User Registration with Email and Password");
36 }
37
38 void login(String phone, int otp) {
39     System.out.println("User Registration with Phone and OTP");
40 }
41
42 void login(String googleEmail) {
43     System.out.println("User Registration with Google");
44 }
45
46 public static void main(String[] args) {
47
48     Authentication auth = new Authentication();
49     //auth.register("john@example.com", "john123");
50     //auth.register("+91 99999 1111", 2345);
51     auth.register("john@gmail.com");

```

In the right panel, the "Console" tab is selected, showing the output from the run command: "<terminated> Authentication [Java Application] /usr/eclipse/plugins/org.eclipse.jst/jstopenj... User Registration with Google".

5.4 But if you use **John@example.com**, it states Invalid Google Account because you are still using the same register method, which takes the Google email as input

The screenshot shows the Eclipse IDE interface with the title bar "Java - Session9/src/Authentication.java - Eclipse IDE". In the left panel, there are two tabs: "Overloading.java" and "Authentication.java X". The "Authentication.java" tab is active, displaying the same Java code as the previous screenshot:

```

19     System.out.println("User Registration with Email and Password");
20 }
21
22 void register(String phone, int otp) {
23     System.out.println("User Registration with Phone and OTP");
24 }
25
26 void register(String googleEmail) {
27     if(googleEmail.endsWith("@gmail.com")) {
28         System.out.println("User Registration with Google");
29     }else {
30         System.out.println("Invalid Google Account");
31     }
32 }
33
34 void login(String email, String password) {
35     System.out.println("User Registration with Email and Password");
36 }
37
38 void login(String phone, int otp) {
39     System.out.println("User Registration with Phone and OTP");
40 }
41
42 void login(String googleEmail) {
43     System.out.println("User Registration with Google");
44 }
45
46 public static void main(String[] args) {
47
48     Authentication auth = new Authentication();
49     //auth.register("john@example.com", "john123");
50     //auth.register("+91 99999 1111", 2345);
51     auth.register("john@example.com");

```

In the right panel, the "Console" tab is selected, showing the output from the run command: "<terminated> Authentication [Java Application] /usr/eclipse/plugins/org.eclipse.jst/jstopenj... Invalid Google Account".

5.5 Now, with these different ways to do registration, you can also use the method called log-in, with the help of which you will execute your login method. Change all this as user log-in. Run this code. It states User login with email and password.

```

java - Session9/src/Authentication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Authentication.java
19     System.out.println("User Registration with Email and Password");
20 }
21 void register(String phone, int otp) {
22     System.out.println("User Registration with Phone and OTP");
23 }
24
25 void register(String googleEmail) {
26     if(googleEmail.endsWith("@gmail.com")) {
27         System.out.println("User Registration with Google");
28     } else {
29         System.out.println("Invalid Google Account");
30     }
31 }
32
33 void login(String email, String password) {
34     System.out.println("User Login with Email and Password");
35 }
36
37 void login(String phone, int otp) {
38     System.out.println("User Login with Phone and OTP");
39 }
40
41 void login(String googleEmail) {
42     System.out.println("User Login with Google");
43 }
44
45 public static void main(String[] args) {
46
47     Authentication auth = new Authentication();
48     //auth.register("john@example.com", "john123");
49     //auth.register("+91 99999 11111", 2345);
50     //auth.register("john@example.com");
51
52     I auth.login("john@example.com", "john123");
53
54 }
55
56 }

```

The screenshot shows the Eclipse IDE interface with the Java code for the `Authentication` class. The code includes methods for user registration (with email, phone, or Google account) and user login (with email, phone, or Google account). The `main` method creates an `Authentication` object and calls the `login` method with specific parameters. The `Console` view on the right shows the output: `User Login with Email and Password`.

5.6 If you use `auth.login` and mention `fionna@gmail.com` when you log in, it will state that the user logs in with Google

```

java - Session9/src/Authentication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Overloading.java Run Authentication.java X
19     System.out.println("User Registration with Email and Password");
20 }
21 void register(String phone, int otp) {
22     System.out.println("User Registration with Phone and OTP");
23 }
24
25 void register(String googleEmail) {
26     if(googleEmail.endsWith("@gmail.com")) {
27         System.out.println("User Registration with Google");
28     } else {
29         System.out.println("Invalid Google Account");
30     }
31 }
32
33 void login(String email, String password) {
34     System.out.println("User Login with Email and Password");
35 }
36
37 void login(String phone, int otp) {
38     System.out.println("User Login with Phone and OTP");
39 }
40
41 void login(String googleEmail) {
42     System.out.println("User Login with Google");
43 }
44
45 public static void main(String[] args) {
46
47     Authentication auth = new Authentication();
48     //auth.register("john@example.com", "john123");
49     //auth.register("+91 99999 11111", 2345);
50     //auth.register("john@example.com");
51
52     //auth.login("john@example.com", "john123");
53     auth.login("fionna@gmail.com");
54
55 }
56

```

This screenshot is similar to the previous one, but it highlights the line `auth.login("fionna@gmail.com");` in the `main` method. The `Console` view shows the output: `User Login with Google`.

By following the above steps, you have successfully depicted how to overload methods in Java.