

Lesson 06 Demo 07

Implementing Optional Class in Java

Objective: To implement the Optional class in Java to handle null values and avoid NullPointerExceptions

Tools Required: Eclipse IDE

Prerequisites: None

Steps to be followed:

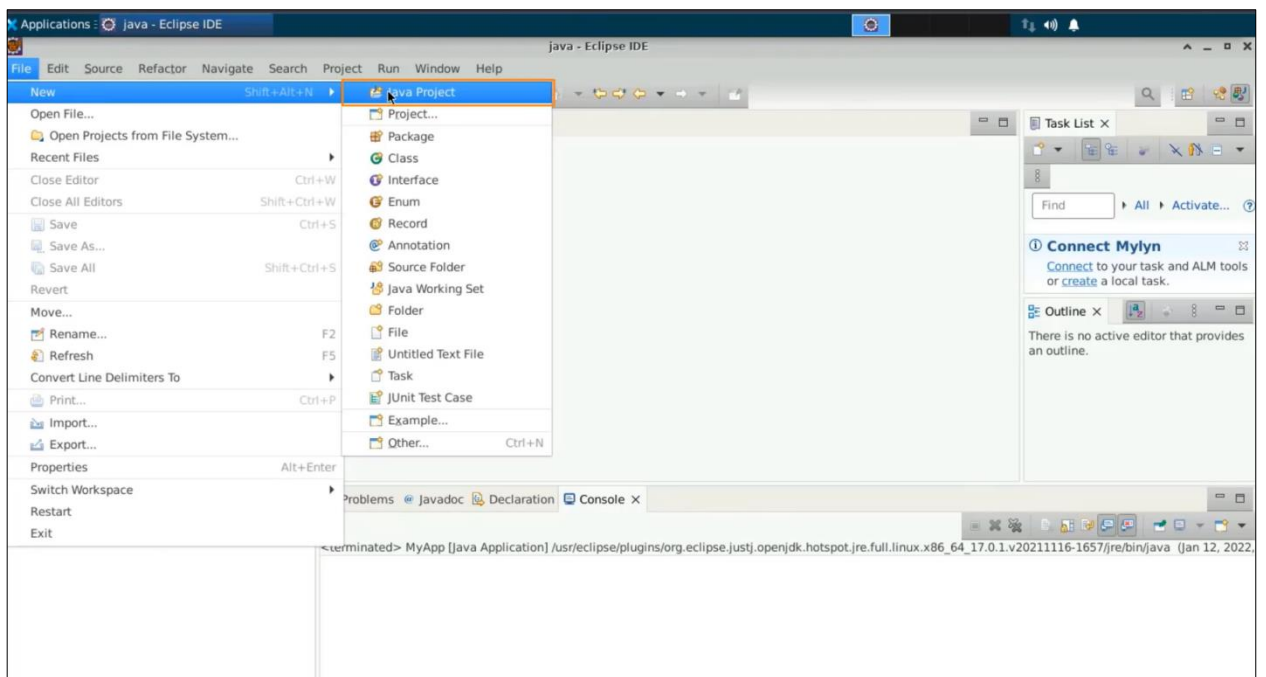
1. Create a new java project
2. Create another constructor which will be parameterized with the values initialized to the inputs
3. Create users with example data and execute the code
4. Implement a safe execution through the optional class
5. Create another optional object and rerun the code

Step 1: Create a new java project

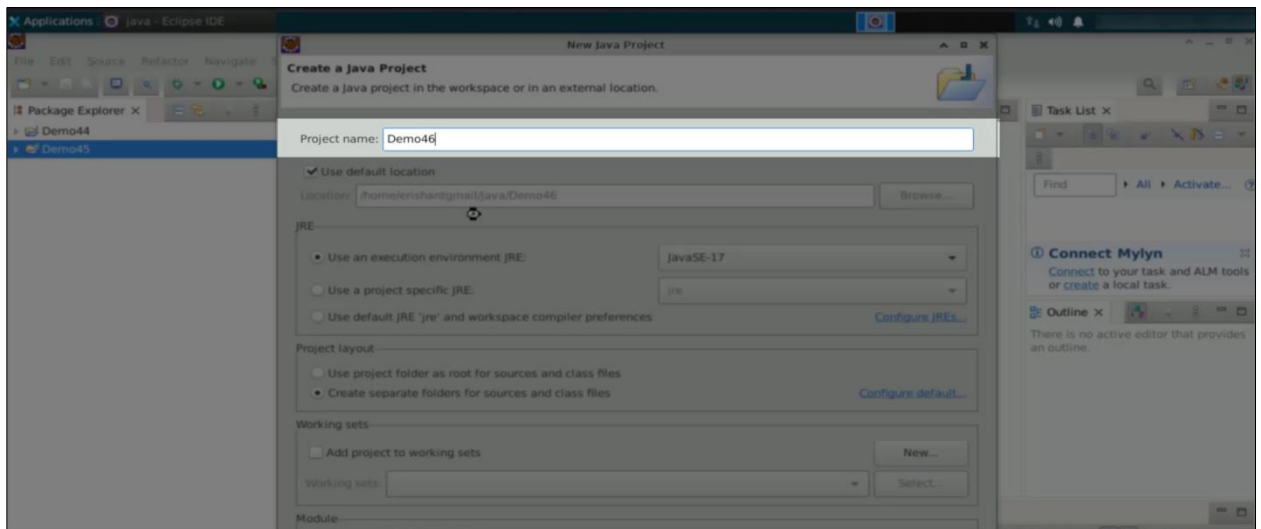
1.1 Open the Eclipse IDE



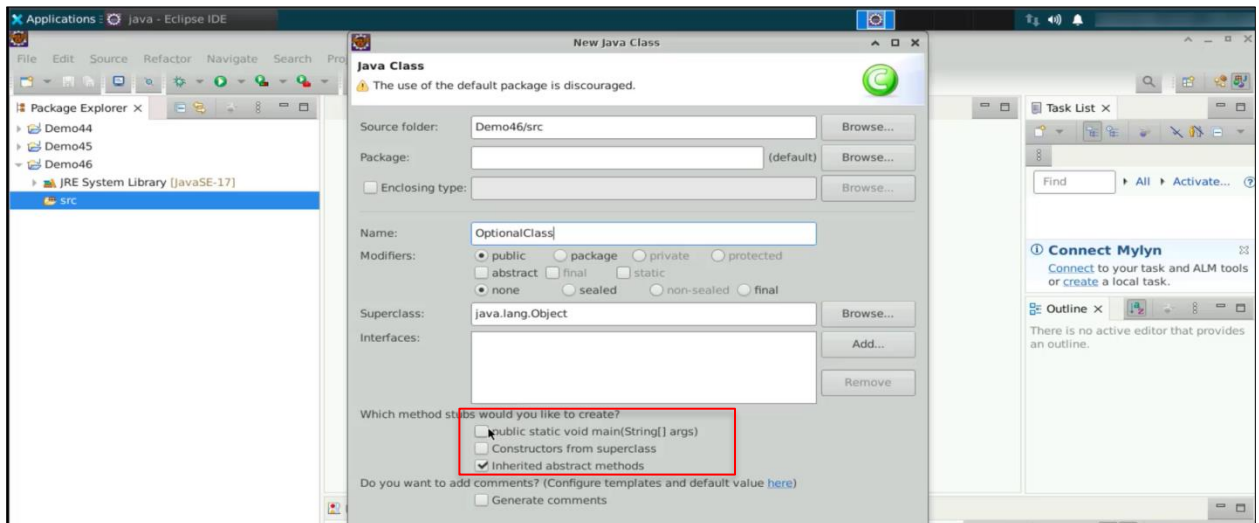
1.2 Select **File**, then **New**, and click **Java project**



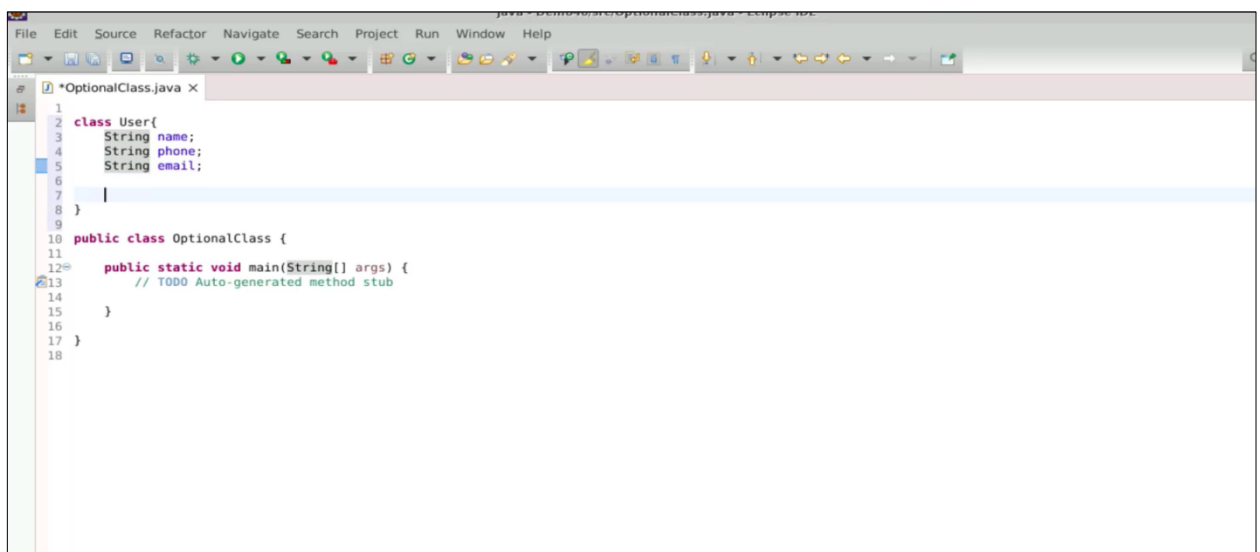
1.3 Name the project as **Demo46**, uncheck **Create a module-info.java file**, and click **Finish**



- 1.4 Right-click on the **Demo46** folder in the src directory, create a new class and name it **OptionalClass**, select the main method, and click **Finish**

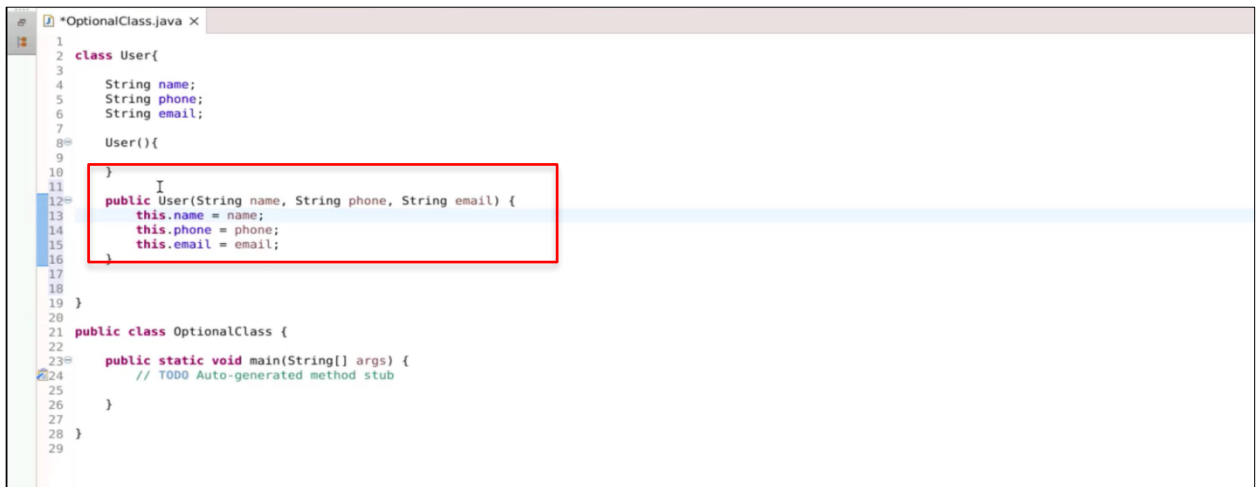


- 1.5 Create a class with the name **User** which includes attributes such as **name**, **phone**, and **email**



Step 2: Create another constructor which will be parameterized with the values initialized to the inputs

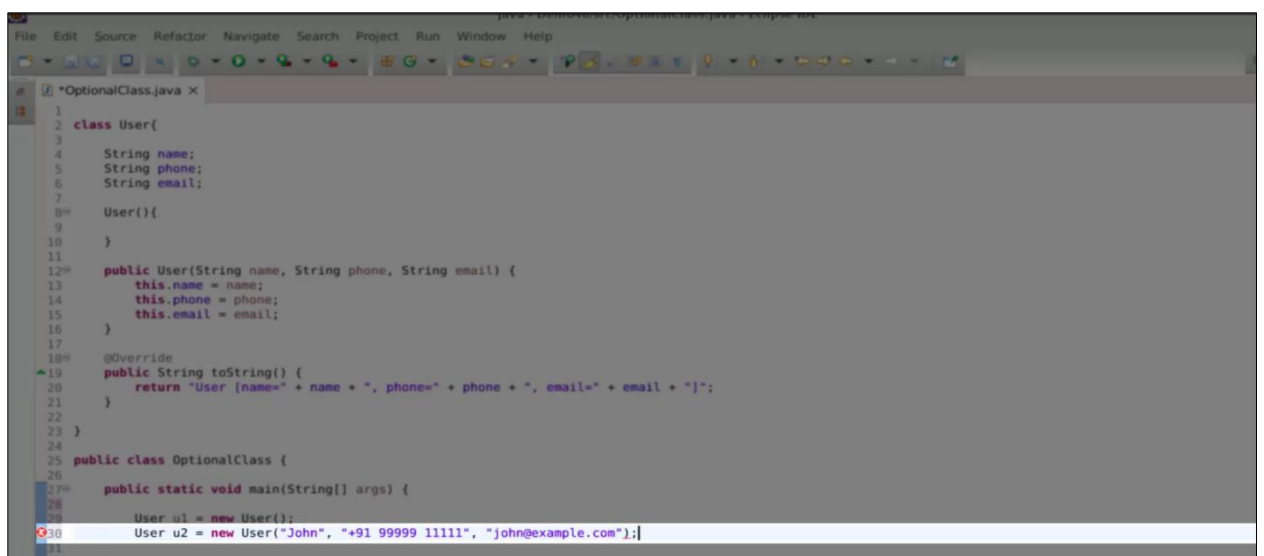
- 2.1 Create a default constructor for the **User** class without initializing the **name**, **phone**, and **email**. Then create another parameterized constructor with the values initialized to the inputs



```
1  class User{
2
3
4      String name;
5      String phone;
6      String email;
7
8      User(){
9
10     }
11
12     public User(String name, String phone, String email) {
13         this.name = name;
14         this.phone = phone;
15         this.email = email;
16     }
17
18 }
19
20 public class OptionalClass {
21
22
23     public static void main(String[] args) {
24         // TODO Auto-generated method stub
25     }
26
27 }
28
29
```

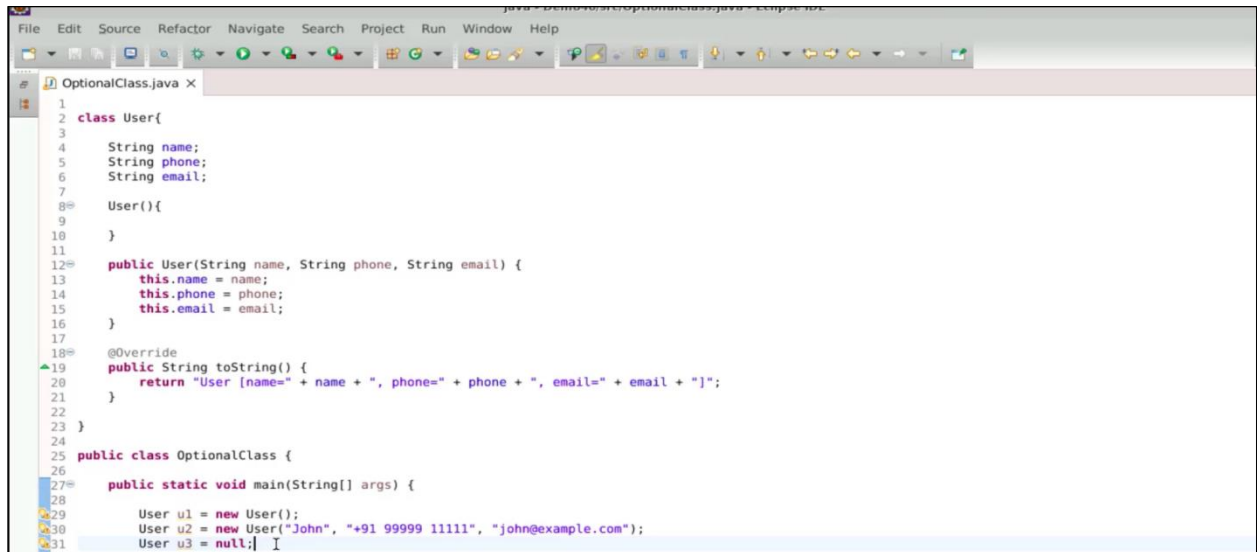
Step 3: Create users with example data and execute the code

- 3.1 Create a new user object **u1** and assign it some example data. Similarly, create another User object **u2** with different example data



```
1  class User{
2
3
4      String name;
5      String phone;
6      String email;
7
8      User(){
9
10     }
11
12     public User(String name, String phone, String email) {
13         this.name = name;
14         this.phone = phone;
15         this.email = email;
16     }
17
18     @Override
19     public String toString() {
20         return "User [name=" + name + ", phone=" + phone + ", email=" + email + "];"
21     }
22 }
23
24 public class OptionalClass {
25
26
27     public static void main(String[] args) {
28
29         User u1 = new User();
30         User u2 = new User("John", "+91 99999 11111", "john@example.com");
31     }
32
33 }
```

3.2 Create another user object **u3** and set it as null

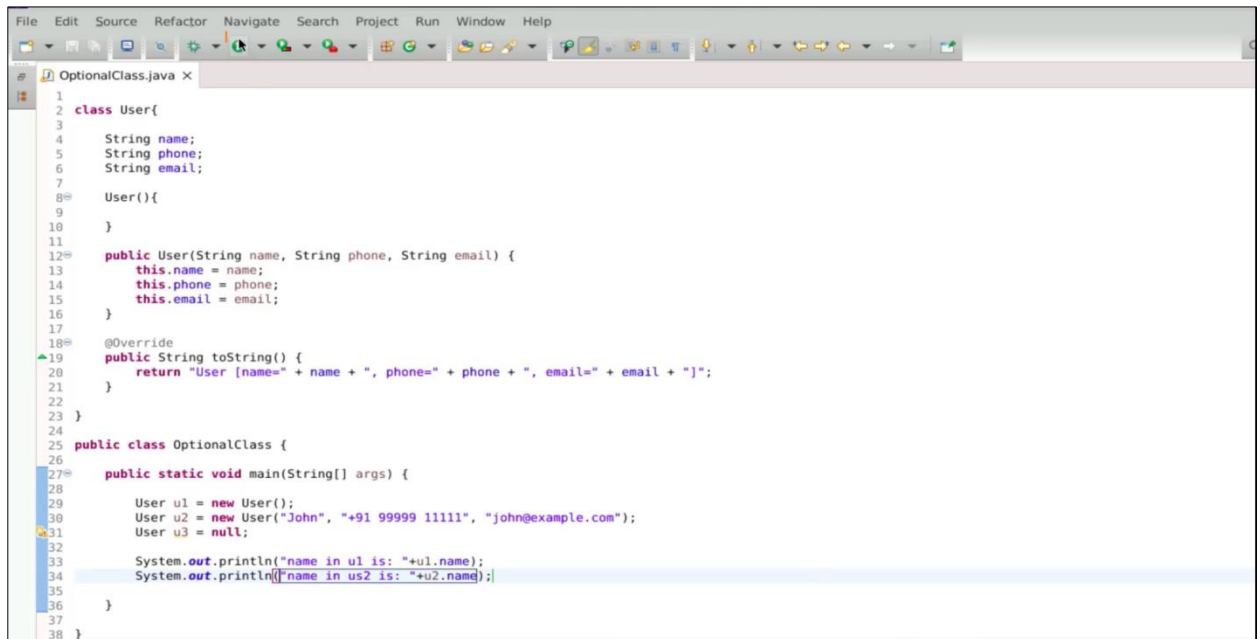


```

1  class User{
2
3
4      String name;
5      String phone;
6      String email;
7
8      User(){
9
10     }
11
12     public User(String name, String phone, String email) {
13         this.name = name;
14         this.phone = phone;
15         this.email = email;
16     }
17
18     @Override
19     public String toString() {
20         return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
21     }
22 }
23
24 public class OptionalClass {
25
26     public static void main(String[] args) {
27
28         User u1 = new User();
29         User u2 = new User("John", "+91 99999 11111", "john@example.com");
30         User u3 = null;
31     }
32 }

```

3.3 Print the name of **U1** using **System.out.println(U1.name)** and do the same for **U2**

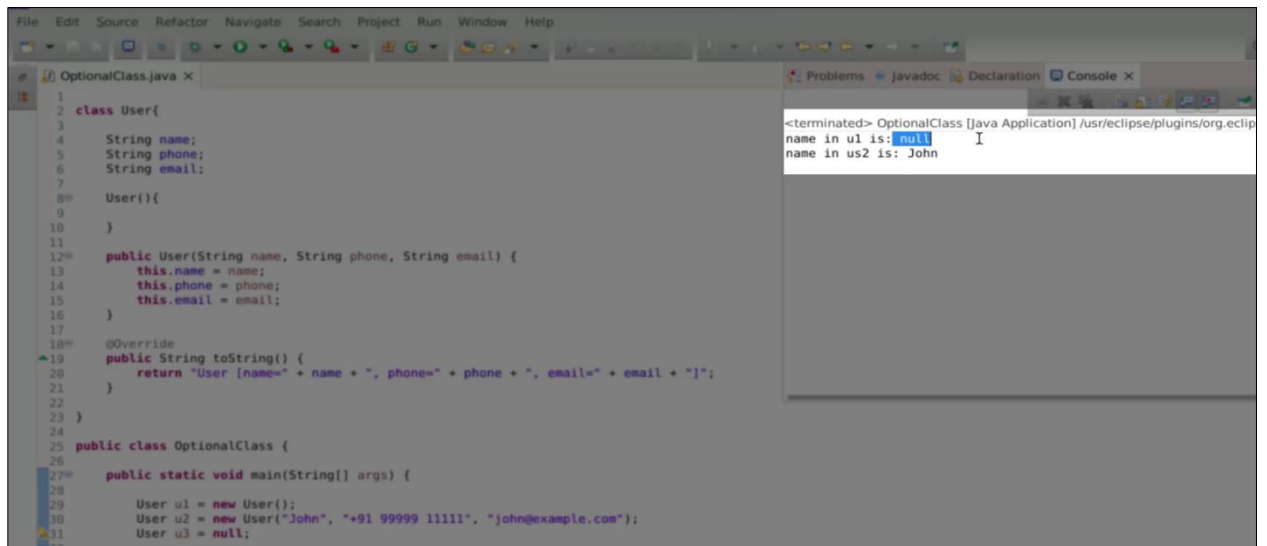


```

1  class User{
2
3
4      String name;
5      String phone;
6      String email;
7
8      User(){
9
10     }
11
12     public User(String name, String phone, String email) {
13         this.name = name;
14         this.phone = phone;
15         this.email = email;
16     }
17
18     @Override
19     public String toString() {
20         return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
21     }
22 }
23
24 public class OptionalClass {
25
26     public static void main(String[] args) {
27
28         User u1 = new User();
29         User u2 = new User("John", "+91 99999 11111", "john@example.com");
30         User u3 = null;
31
32         System.out.println("name in u1 is: "+u1.name);
33         System.out.println("name in u2 is: "+u2.name);
34     }
35 }
36
37
38 }

```

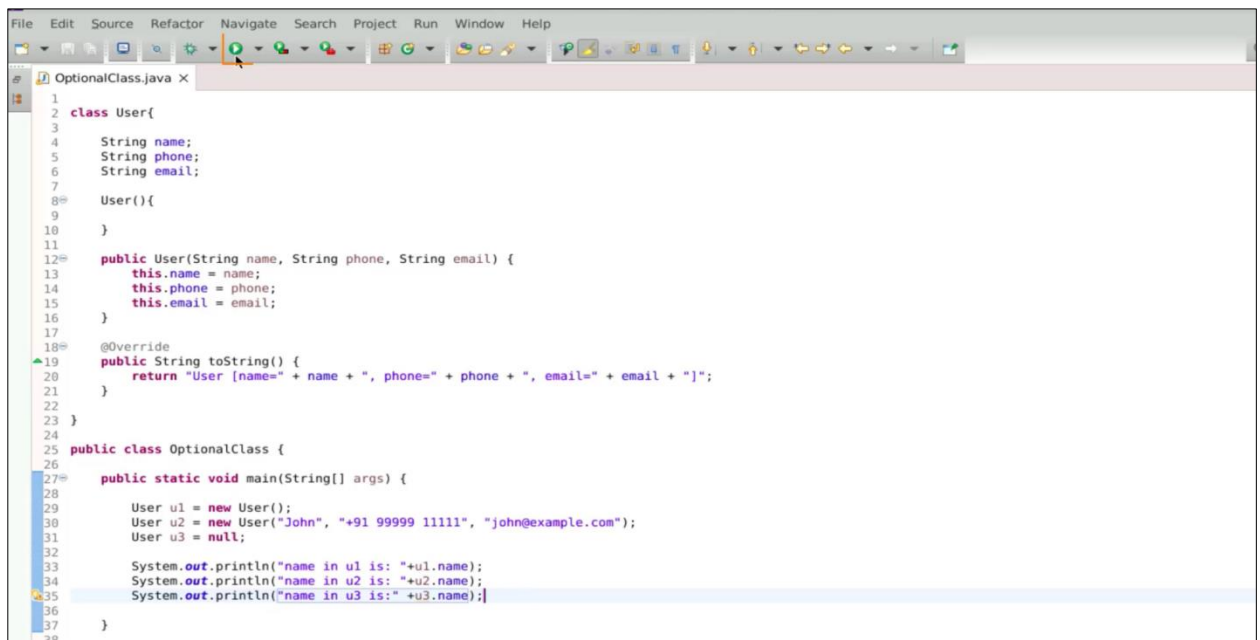
3.4 When you run the code, you will see that the name of **u1** is **null**, while the name of **u2** is displayed as **John**



```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java x
1 class User{
2     String name;
3     String phone;
4     String email;
5     User(){
6     }
7     public User(String name, String phone, String email) {
8         this.name = name;
9         this.phone = phone;
10        this.email = email;
11    }
12    @Override
13    public String toString() {
14        return "User {name=" + name + ", phone=" + phone + ", email=" + email + "}";
15    }
16    public class OptionalClass {
17        public static void main(String[] args) {
18            User u1 = new User();
19            User u2 = new User("John", "+91 99999 11111", "john@example.com");
20            User u3 = null;
21        }
22    }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
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 }
```

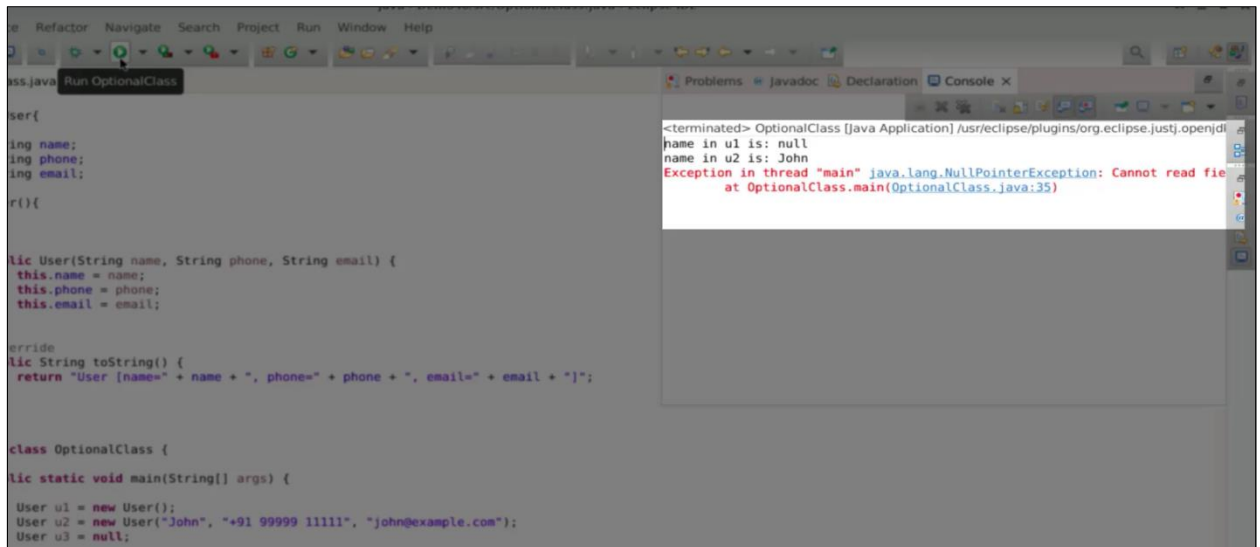
<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclip
name in u1 is: null
name in u2 is: John

3.5 Print the name of **u3** using **System.out.println(u3.name)**. Although **u3** does not refer to any object, you would not get any compile-time errors because the reference variable is resolved at runtime



```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java x
1 class User{
2     String name;
3     String phone;
4     String email;
5     User(){
6     }
7     public User(String name, String phone, String email) {
8         this.name = name;
9         this.phone = phone;
10        this.email = email;
11    }
12    @Override
13    public String toString() {
14        return "User {name=" + name + ", phone=" + phone + ", email=" + email + "}";
15    }
16    public class OptionalClass {
17        public static void main(String[] args) {
18            User u1 = new User();
19            User u2 = new User("John", "+91 99999 11111", "john@example.com");
20            User u3 = null;
21            System.out.println("name in u1 is: "+u1.name);
22            System.out.println("name in u2 is: "+u2.name);
23            System.out.println("name in u3 is: "+u3.name);
24        }
25    }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
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 }
```

3.6 Running the program will result in an exception at line number 35 when trying to access the name in **u3**



```

OptionalClass.java
Run OptionalClass

class User {
    String name;
    String phone;
    String email;

    User() {}

    public User(String name, String phone, String email) {
        this.name = name;
        this.phone = phone;
        this.email = email;
    }

    @Override
    public String toString() {
        return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
    }
}

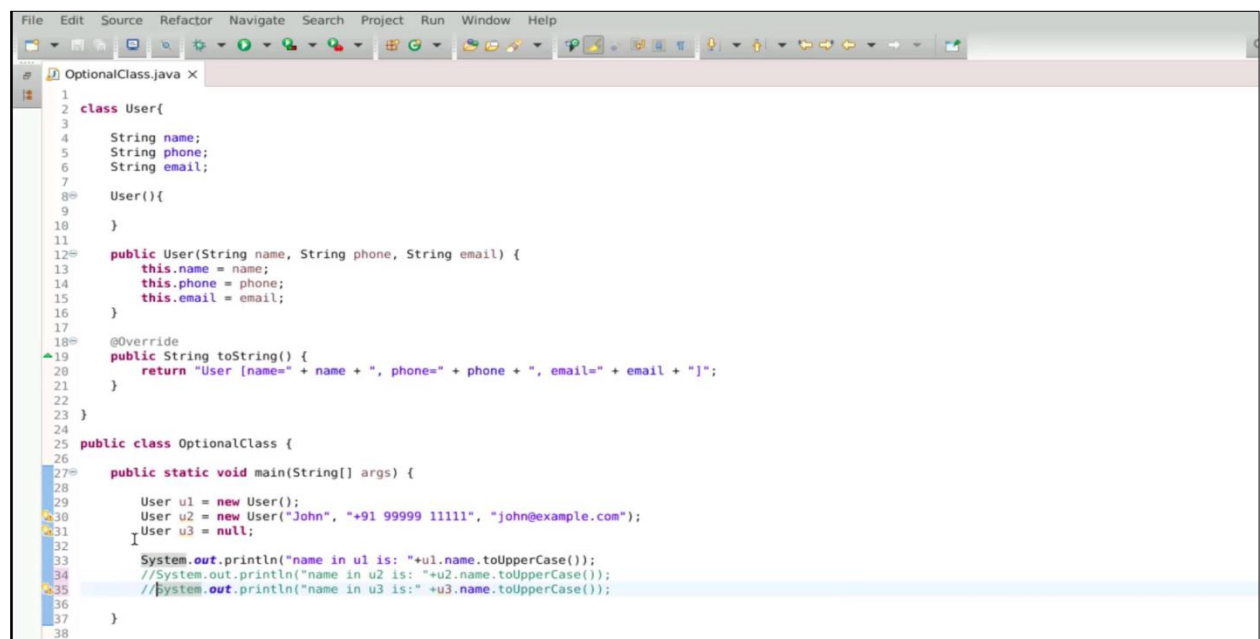
class OptionalClass {
    public static void main(String[] args) {
        User u1 = new User();
        User u2 = new User("John", "+91 99999 11111", "john@example.com");
        User u3 = null;

        System.out.println("name in u1 is: " + u1.name.toUpperCase());
        //System.out.println("name in u2 is: " + u2.name.toUpperCase());
        //System.out.println("name in u3 is: " + u3.name.toUpperCase());
    }
}

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk
name in u1 is: null
name in u2 is: John
Exception in thread "main" java.lang.NullPointerException: Cannot read field
at OptionalClass.main(OptionalClass.java:35)

```

3.7 If you try to execute the **toUpperCase()** method on the **u1** name, you will get a null Pointer exception because **u1.name** is null

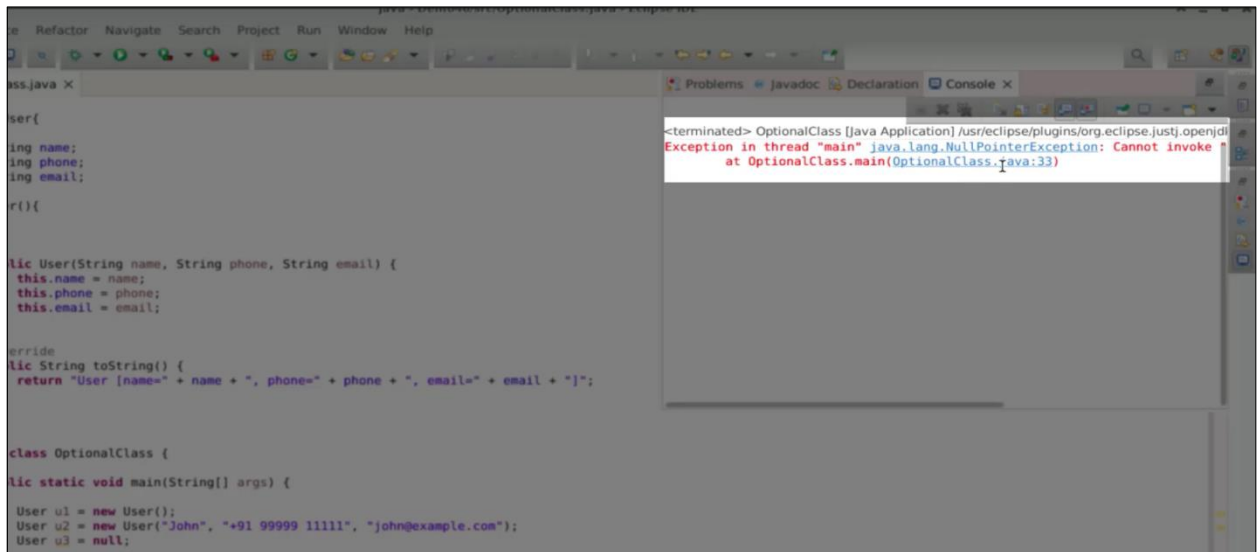


```

OptionalClass.java
1  class User{
2
3      String name;
4      String phone;
5      String email;
6
7      User(){
8
9      }
10
11     public User(String name, String phone, String email) {
12         this.name = name;
13         this.phone = phone;
14         this.email = email;
15     }
16
17     @Override
18     public String toString() {
19         return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
20     }
21
22 }
23
24 public class OptionalClass {
25
26     public static void main(String[] args) {
27
28         User u1 = new User();
29         User u2 = new User("John", "+91 99999 11111", "john@example.com");
30         User u3 = null;
31
32         System.out.println("name in u1 is: " + u1.name.toUpperCase());
33         //System.out.println("name in u2 is: " + u2.name.toUpperCase());
34         //System.out.println("name in u3 is: " + u3.name.toUpperCase());
35     }
36
37 }
38

```


3.8 If you comment out these two lines and run the code, you will receive an error message stating that you cannot invoke the **toUpperCase()** method because **u1.name** is null



The screenshot shows the Eclipse IDE with a Java file named `OptionalClass.java`. The code defines a `User` class with attributes `name`, `phone`, and `email`, and a `toString()` method. In the `OptionalClass` main method, three `User` objects are created: `u1` (default), `u2` (with values "John", "+91 99999 11111", "john@example.com"), and `u3` (null). The `toString()` method is called for each. The console shows a `NullPointerException: Cannot invoke` error at line 33, which corresponds to the `u1.name.toUpperCase()` call.

```
OptionalClass.java X
class User {
    String name;
    String phone;
    String email;

    User() {}

    public User(String name, String phone, String email) {
        this.name = name;
        this.phone = phone;
        this.email = email;
    }

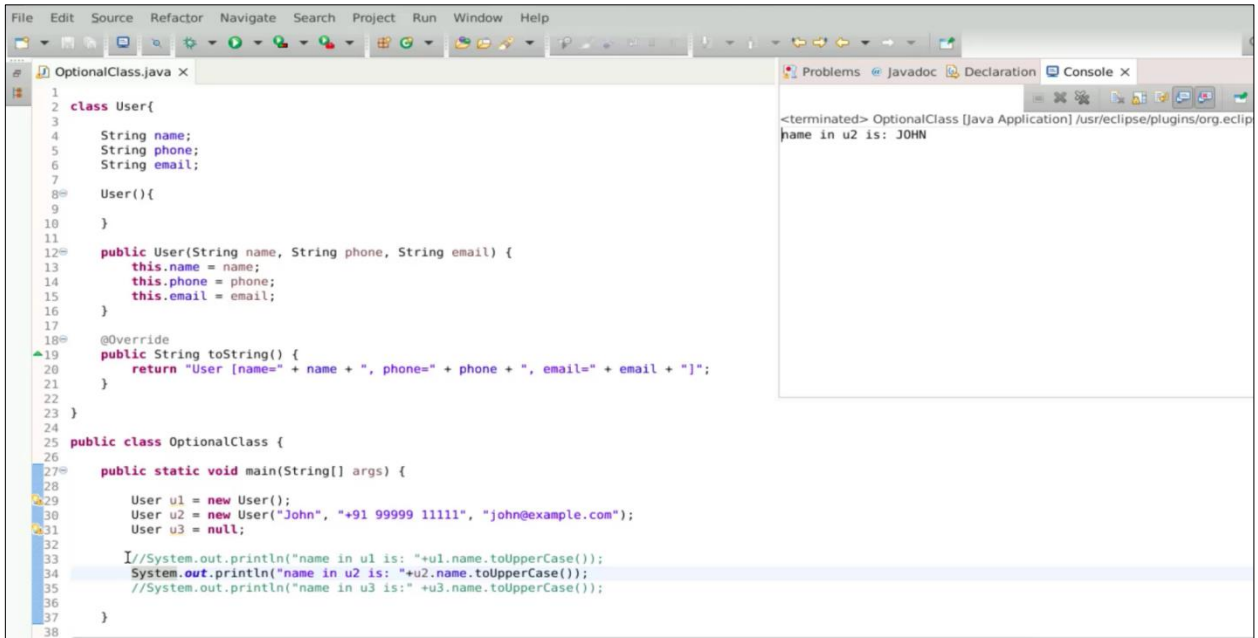
    @Override
    public String toString() {
        return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]\n";
    }
}

public class OptionalClass {
    public static void main(String[] args) {
        User u1 = new User();
        User u2 = new User("John", "+91 99999 11111", "john@example.com");
        User u3 = null;

        System.out.println("name in u1 is: " + u1.name.toUpperCase());
        System.out.println("name in u2 is: " + u2.name.toUpperCase());
        System.out.println("name in u3 is: " + u3.name.toUpperCase());
    }
}
```

Console: <terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.just.openid
Exception in thread "main" java.lang.NullPointerException: Cannot invoke
at OptionalClass.main(OptionalClass.java:33)

3.9 If you uncomment line number 34 and run the code, you will see that the output **name in u2 is: John**, and it is displayed in uppercase



The screenshot shows the same Eclipse IDE with the `OptionalClass.java` file. Line 34, which was previously commented out, is now active. The console shows the output: "name in u2 is: JOHN".

```
OptionalClass.java X
1 class User {
2     String name;
3     String phone;
4     String email;
5
6     User() {}
7
8     public User(String name, String phone, String email) {
9         this.name = name;
10        this.phone = phone;
11        this.email = email;
12    }
13
14    @Override
15    public String toString() {
16        return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]\n";
17    }
18
19 public class OptionalClass {
20     public static void main(String[] args) {
21         User u1 = new User();
22         User u2 = new User("John", "+91 99999 11111", "john@example.com");
23         User u3 = null;
24
25         System.out.println("name in u1 is: " + u1.name.toUpperCase());
26         System.out.println("name in u2 is: " + u2.name.toUpperCase());
27         System.out.println("name in u3 is: " + u3.name.toUpperCase());
28     }
29 }
```

Console: <terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.just.openid
name in u2 is: JOHN

- 3.10 When trying to access the name of u3 using **u3.name.toUpperCase()**, it will crash because u3 itself is null, showing the error **Cannot read the field name as U3 is null**

The screenshot shows the Eclipse IDE with a Java file named `OptionalClass.java` open. The code defines a `User` class with attributes `name`, `phone`, and `email`, and a static `main` method. In the `main` method, three `User` objects are created: `u1`, `u2`, and `u3`. `u1` and `u2` are initialized with values, while `u3` is set to `null`. The code then prints the names of `u1`, `u2`, and `u3` using `toUpperCase()`. The console shows a `NullPointerException` error: `Cannot read field 'name' because 'u3' is null`. The error message is highlighted in blue in the console window.

Step 4: Implement a safe execution through the optional class

- 4.1 Create the **Optional** object for the name variable using **Optional.ofNullable(u1.name)**. The **ofNullable()** method checks whether the name in the **User** object is null or not

The screenshot shows the Eclipse IDE with the same `OptionalClass.java` file. The code is identical to the previous screenshot, but with an additional line of code added in the `main` method: `Optional<String> checkForName = Optional.ofNullable(u1.name);`. This line is highlighted in blue. The code also includes comments for the `println` statements.

4.2 Use **isPresent** to check if the name is present. If it is present, print **name in u1** is followed by **u1.name.toUpperCase()**. Otherwise, print **Sorry, name in u1 is null**. This approach provides safer execution compared to the previous approach

```

File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java x
11 }
12
13 public User(String name, String phone, String email) {
14     this.name = name;
15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
22 }
23 }
24
25 public class OptionalClass {
26
27     public static void main(String[] args) {
28
29         User u1 = new User();
30         User u2 = new User("John", "+91 99999 11111", "john@example.com");
31         User u3 = null;
32
33         //System.out.println("name in u1 is: " + u1.name.toUpperCase());
34         //System.out.println("name in u2 is: " + u2.name.toUpperCase());
35         //System.out.println("name in u3 is: " + u3.name.toUpperCase());
36
37         Optional<String> checkForName = Optional.ofNullable(u1.name);
38         if(checkForName.isPresent()) {
39             System.out.println("name in u1 is: " + u1.name.toUpperCase());
40         } else {
41             System.out.println("Sorry, name in u1 is null");
42         }
43     }
44 }
45
46
47 }
48

```

4.3 If you replace the name with **u2**, you will see that it prints **name in U2 is:JOHN**

```

File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java Run OptionalClass
11 }
12
13 public User(String name, String phone, String email) {
14     this.name = name;
15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
22 }
23 }
24
25 public class OptionalClass {
26
27     public static void main(String[] args) {
28
29         User u1 = new User();
30         User u2 = new User("John", "+91 99999 11111", "john@example.com");
31         User u3 = null;
32
33         //System.out.println("name in u1 is: " + u1.name.toUpperCase());
34         //System.out.println("name in u2 is: " + u2.name.toUpperCase());
35         //System.out.println("name in u3 is: " + u3.name.toUpperCase());
36
37         Optional<String> checkForName = Optional.ofNullable(u2.name);
38         if(checkForName.isPresent()) {
39             System.out.println("name in u2 is: " + u2.name.toUpperCase());
40         } else {
41             System.out.println("Sorry, name in u2 is null");
42         }
43     }
44 }
45
46
47 }
48

```

Problems Javadoc Declaration Console X

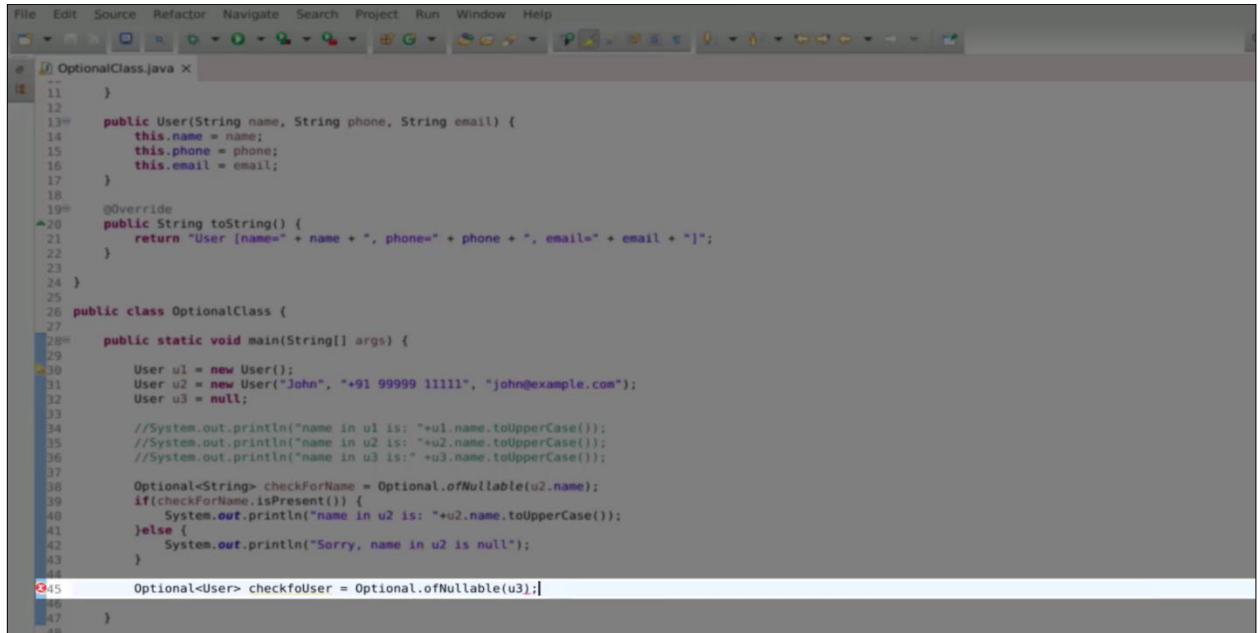
```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclips
name in u2 is: JOHN

```

Step 5: Create another optional object and rerun the code

- 5.1 Now, create one more Optional object, but this time it will work specifically on the User object. Enter **Optional<User>** to define the type of the **Optional**. Check for the User object by using **Optional.ofNullable()** and pass **u3** inside the **ofNullable()** method



```
OptionalClass.java x
11 }
12
13 public User(String name, String phone, String email) {
14     this.name = name;
15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = null;
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46
47     }
48 }
```

5.2 Next, type an **if** check for **user.isPresent()**. This condition checks if there is a user object available and it is not null. If it is present, you can say **name in u3** is followed by **U3.name**. In the **else** case, type **Sorry, u3 is null**. This is a small implementation using Optional

```

OptionalClass.java X
15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "];"
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = null;
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkForUser = Optional.ofNullable(u3);
46         if(checkForUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51     }
52 }

```

5.3 When you run the code it shows: **Cannot read field name because U3 is null**

```

OptionalClass.java X
15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "];"
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = null;
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkForUser = Optional.ofNullable(u3);
46         if(checkForUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51     }
52 }

```

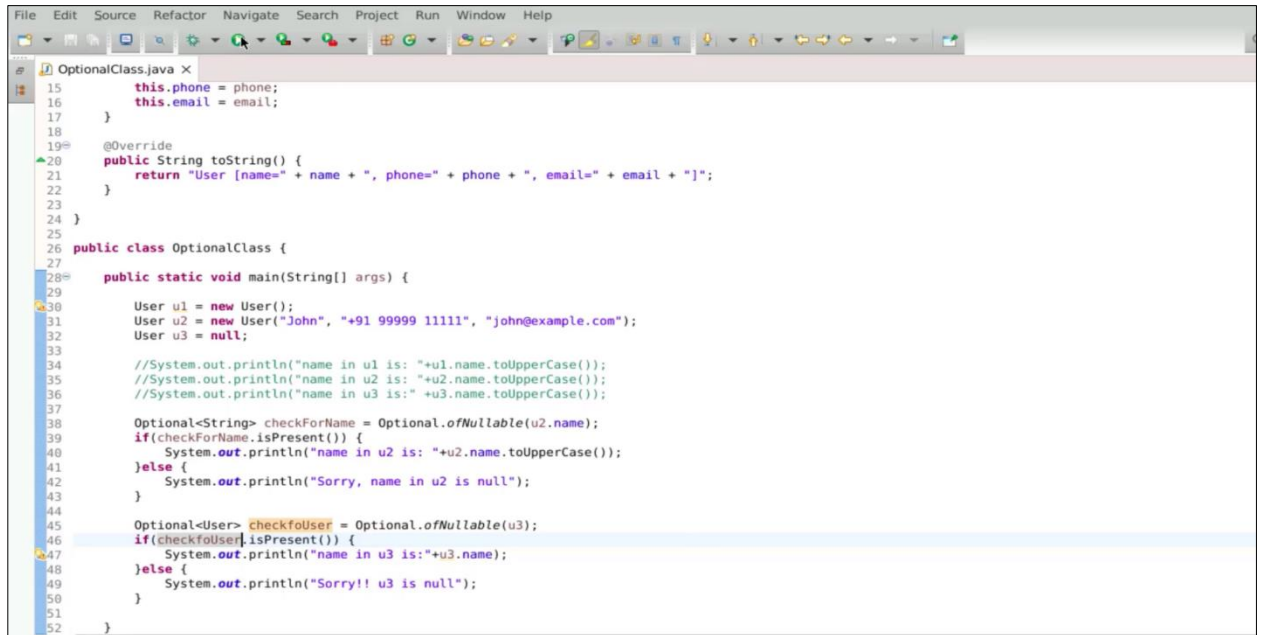
Console X

```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclips
a.lang.NullPointerException: Cannot read field "name" because
OptionalClass.java:47

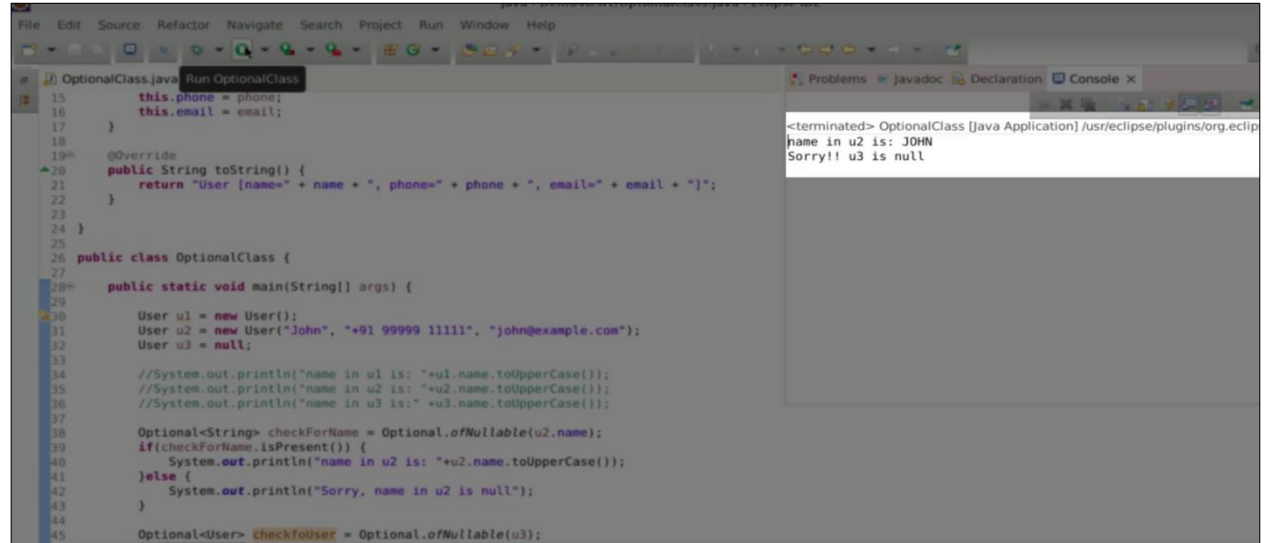
```

5.4 Here, change this to **checkforUser** instead of check for **name**



```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java X
15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "];"
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = null;
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51     }
52 }
```

5.5 Now, re-run the code and here you can see it shows **Sorry!! u3 is null**

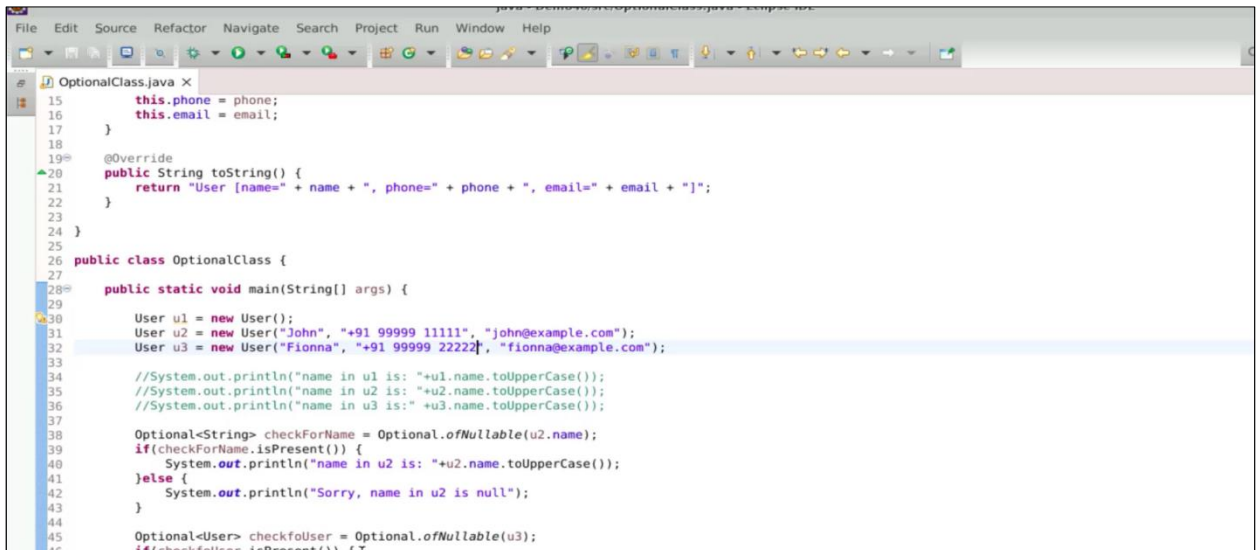


```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java Run OptionalClass
15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "];"
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = null;
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51     }
52 }
```

```
<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclips
name in u2 is: JOHN
Sorry!! u3 is null
```


5.6 Create a new user object created and add the given data:

Name= **Fionna**, Email: **fionna@example.com**, and phone number

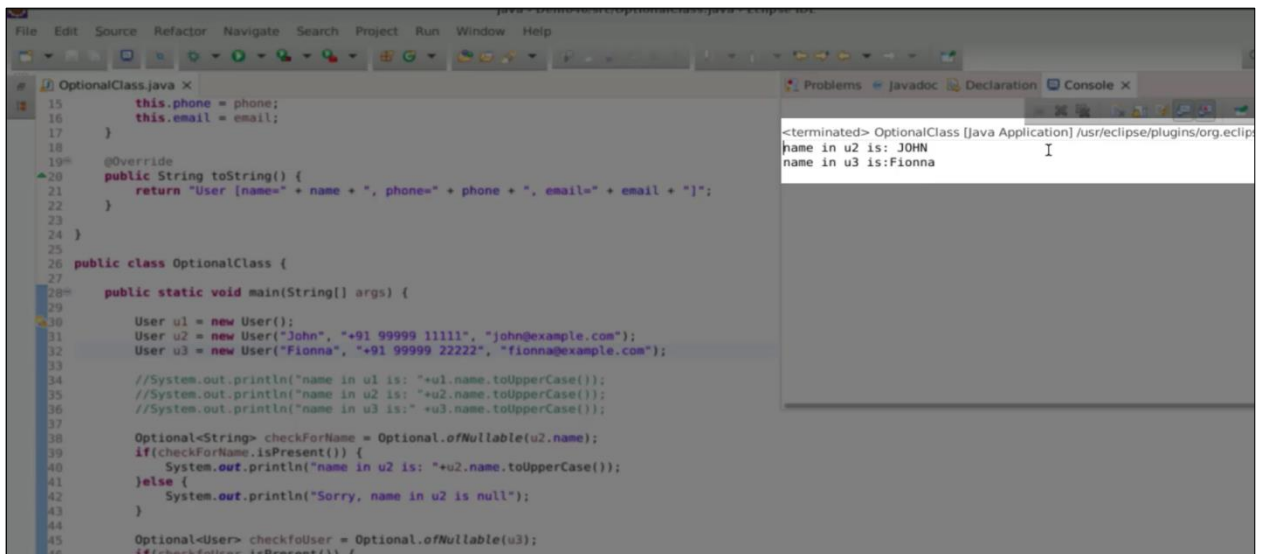


```

15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {

```

5.7 Now, since this is not null, when you run the program, it will give you the name. You can even check for the name within this method



```

15     this.phone = phone;
16     this.email = email;
17 }
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {

```

```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse
name in u2 is: JOHN
name in u3 is: Fionna

```

5.8 Enter **Optional<User>** and then check user by using **Optional.ofNullable()** and pass **u2** as the parameter. Next, for the **checkUser**, execute the **ifPresent()** method and pass a lambda function. Enter **ifPresent(System.out::println)** to print the data if it is not null. It's better to convert it to a string, so use **String.valueOf(checkUser.email.orElse(null))** to handle the email which may be null. Finally, print the data using **checkUser.email**. If the email is not null, it will be printed

```

18
19
20 @Override
21 public String toString() {
22     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "];
23 }
24
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51
52         Optional<String> checkUserEmail = Optional.ofNullable(u2.email);
53         checkUserEmail.ifPresent(System.out::println); // print the data in case if its not null
54
55     }
56 }

```

5.9 Run this, and you will see **john@example.com** printed

```

18
19
20 @Override
21 public String toString() {
22     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "];
23 }
24
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51
52         Optional<String> checkUserEmail = Optional.ofNullable(u2.email);
53         checkUserEmail.ifPresent(System.out::println); // print the data in case if its not null
54
55     }
56 }

```

```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse
name in u2 is: JOHN
name in u3 is: Fionna
john@example.com

```


5.10 Let us consider that you have not supplied an email and have given **null** as the email for **u2**. When you run the code, you won't be able to see the email of the user printed. With this simple approach of method referencing you have printed your data

```

OptionalClass.java
18
19 @Override
20 public String toString() {
21     return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
22 }
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", null);
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49
50         }
51     }
52 }
53
54
55
56
57
58
59

```

```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclips
name in u2 is: JOHN
name in u3 is: Fionna

```

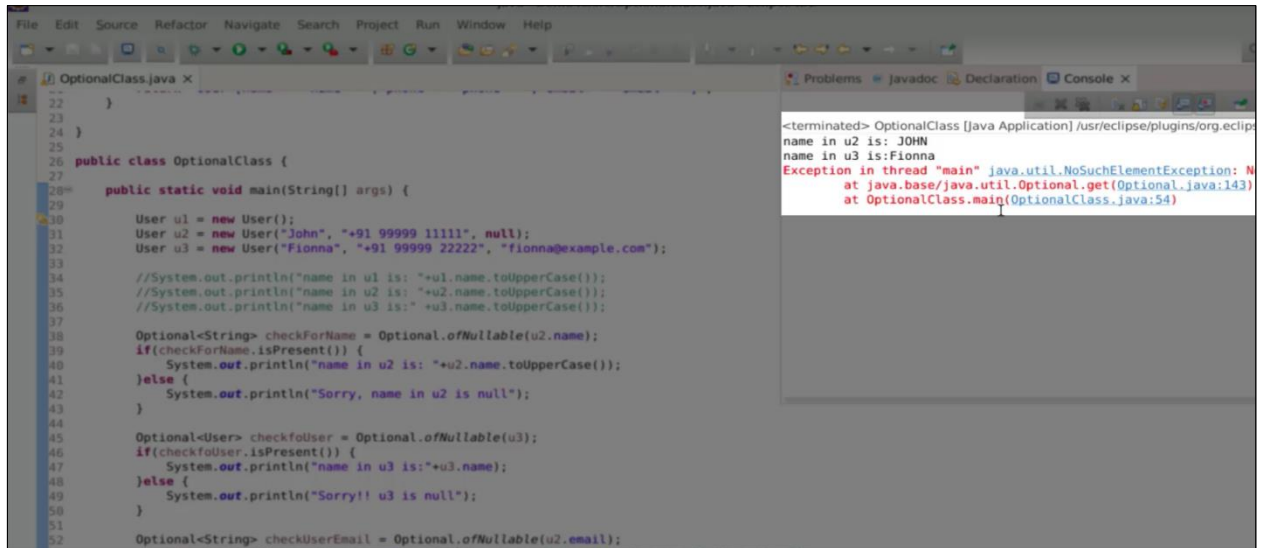
5.11 You can even use another method:
System.out.println("Email is: "+checkUser.email.get()) to extract the email data.
 The **get()** method returns the optional email, which may be null or not

```

OptionalClass.java
22
23
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", null);
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51
52         Optional<String> checkUserEmail = Optional.ofNullable(u2.email);
53         checkUserEmail.ifPresent(System.out::println); // print the data in case if its not null
54         System.out.println("Email is: "+checkUserEmail.get());
55
56     }
57 }
58
59

```

5.12 When you run the code it shows no value present



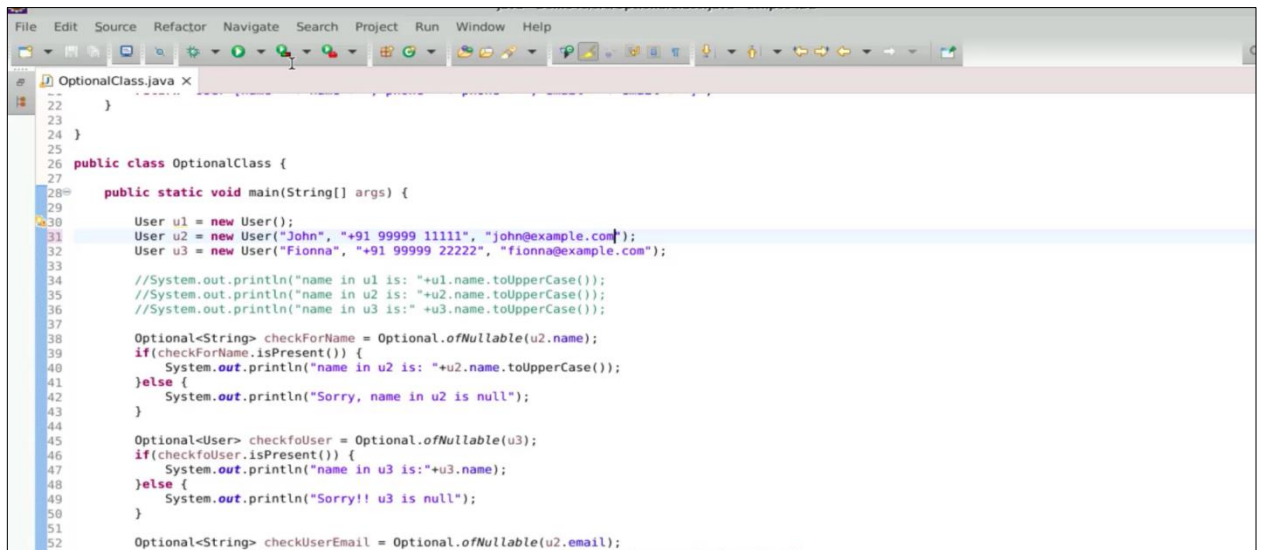
The screenshot shows the Eclipse IDE with a Java file named `OptionalClass.java`. The code defines a `User` class and an `OptionalClass` with a `main` method. In the `main` method, three `User` objects are created: `u1` (John, +91 99999 11111, null), `u2` (John, +91 99999 11111, null), and `u3` (Fionna, +91 99999 22222, fionna@example.com). The code then checks for the presence of `u2` and `u3` using `Optional` classes. The console output shows the names of `u1` and `u3`, but for `u2`, it says "name in u2 is: JOHN". An exception is thrown: `Exception in thread "main" java.util.NoSuchElementException: No element present.`

```
OptionalClass.java
22 }
23 }
24 }
25 }
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", null);
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51
52         Optional<String> checkUserEmail = Optional.ofNullable(u2.email);
```

Console Output:

```
<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse
name in u2 is: JOHN
name in u3 is: Fionna
Exception in thread "main" java.util.NoSuchElementException: No
at java.base/java.util.Optional.get(Optional.java:143)
at OptionalClass.main(OptionalClass.java:54)
```

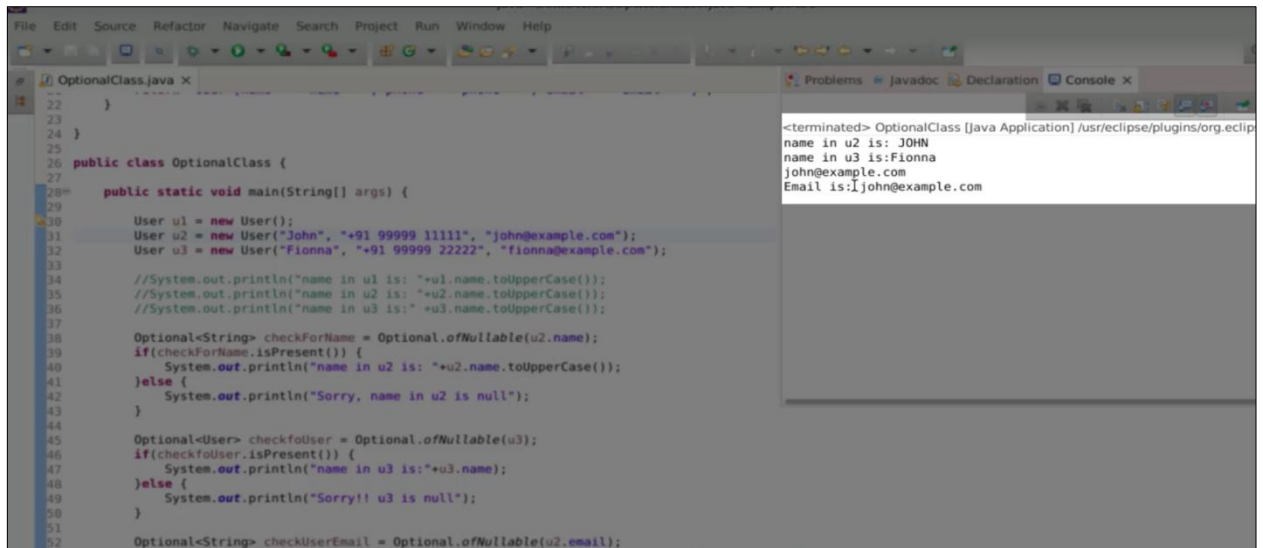
5.13 Add the email: john@example.com



The screenshot shows the Eclipse IDE with the same `OptionalClass.java` file. The code is updated to include the email for `u2`. The `User` class and `OptionalClass` are the same as in the previous screenshot. The `main` method now creates `u2` with the email `john@example.com`. The console output shows the names of `u1` and `u3`, and for `u2`, it says "name in u2 is: JOHN".

```
OptionalClass.java
22 }
23 }
24 }
25 }
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         } else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         } else {
49             System.out.println("Sorry!! u3 is null");
50         }
51
52         Optional<String> checkUserEmail = Optional.ofNullable(u2.email);
```

5.14 Re-run the code, and you can see that with the **get()** method, you can extract the data. However, if it is null, it will crash



```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java X
22 }
23 }
24 }
25
26 public class OptionalClass {
27
28     public static void main(String[] args) {
29
30         User u1 = new User();
31         User u2 = new User("John", "+91 99999 11111", "john@example.com");
32         User u3 = new User("Fionna", "+91 99999 22222", "fionna@example.com");
33
34         //System.out.println("name in u1 is: "+u1.name.toUpperCase());
35         //System.out.println("name in u2 is: "+u2.name.toUpperCase());
36         //System.out.println("name in u3 is: "+u3.name.toUpperCase());
37
38         Optional<String> checkForName = Optional.ofNullable(u2.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u2 is: "+u2.name.toUpperCase());
41         }else {
42             System.out.println("Sorry, name in u2 is null");
43         }
44
45         Optional<User> checkfoUser = Optional.ofNullable(u3);
46         if(checkfoUser.isPresent()) {
47             System.out.println("name in u3 is: "+u3.name);
48         }else {
49             System.out.println("Sorry!! u3 is null");
50         }
51
52         Optional<String> checkUserEmail = Optional.ofNullable(u2.email);
53     }
54 }
```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclips
name in u2 is: JOHN
name in u3 is: Fionna
john@example.com
fionna@example.com
Email is: Ijohn@example.com

By following these steps, you have successfully implemented the Optional class in Java to handle null values and avoid NullPointerExceptions.