

## 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:

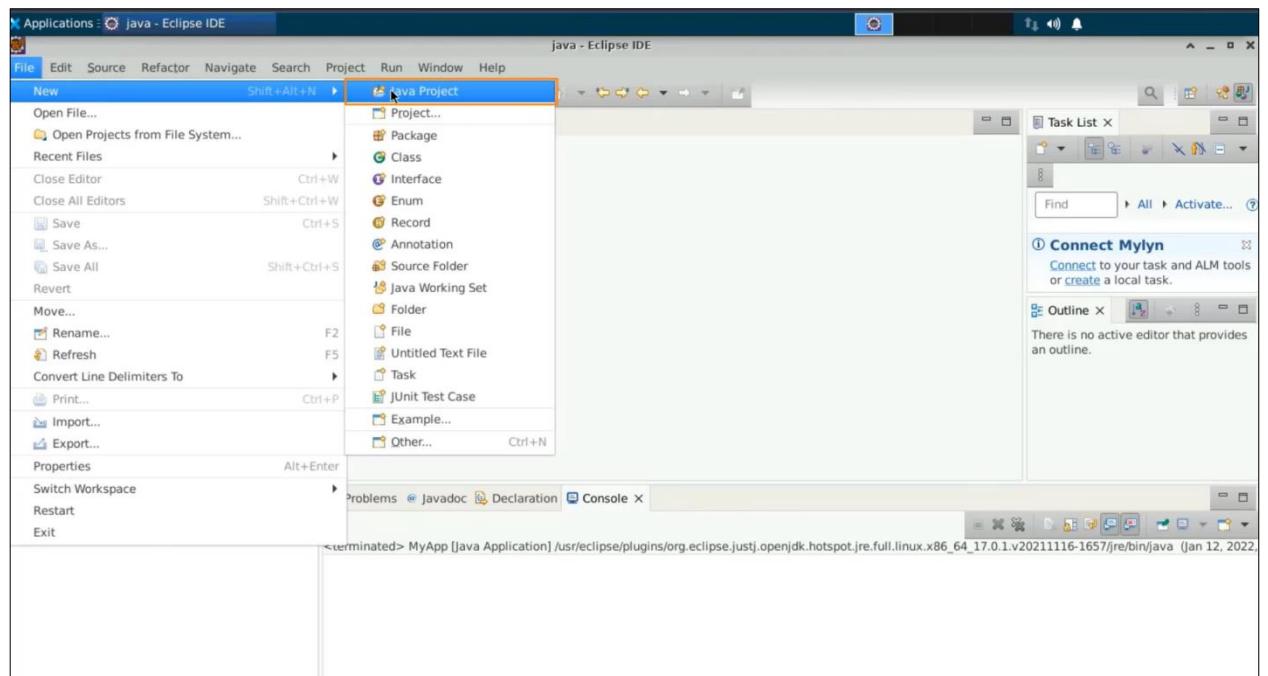
1. Create a new java project
2. Create another constructor which will be parameterized with the values initialized to the inputs
3. Create users with example data and execute the code
4. Implement a safe execution through the optional class
5. Create another optional object and rerun the code

## Step 1: Create a new java project

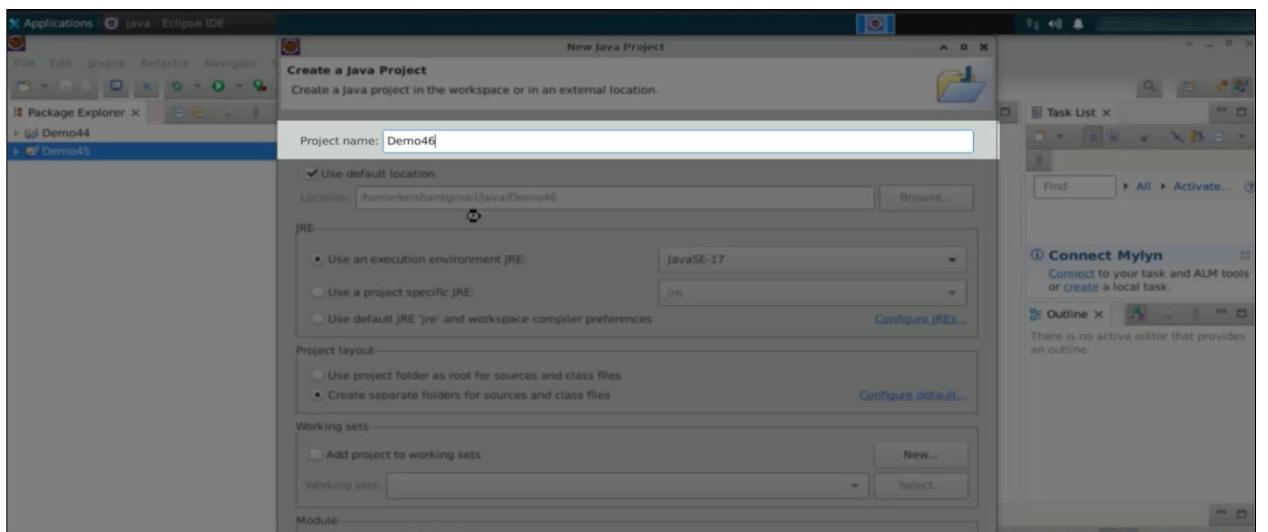
### 1.1 Open the Eclipse IDE



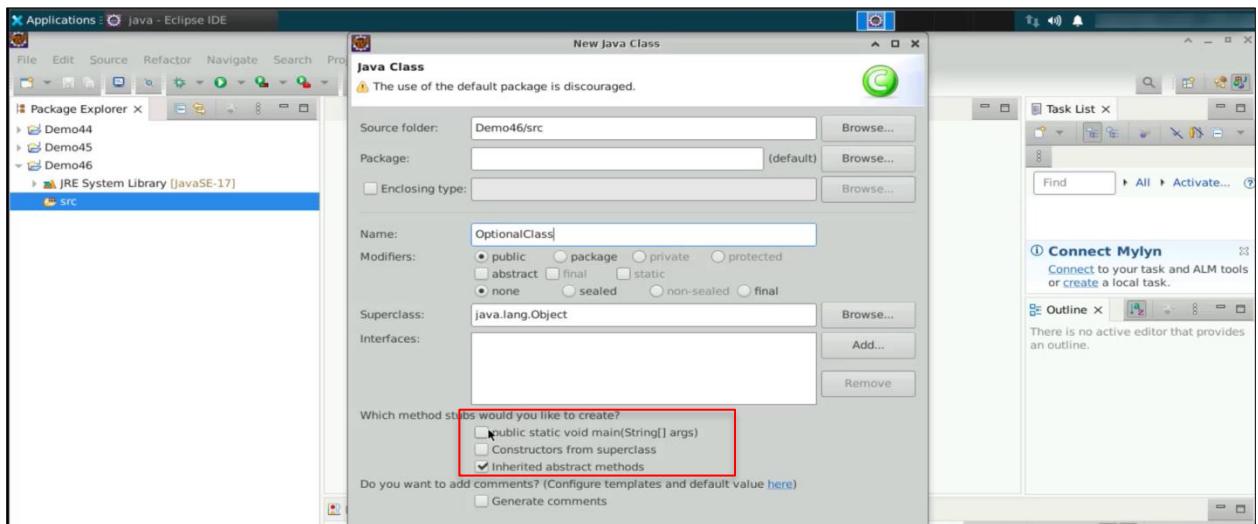
## 1.2 Select File, then New, and click Java project



## 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**

```
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  }
6
7 }
8
9 public class OptionalClass {
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13     }
14
15 }
16
17 }
18
```

The screenshot shows the Eclipse IDE code editor with 'OptionalClass.java' selected. The code defines a class 'User' with three string attributes: 'name', 'phone', and 'email'. It also contains an empty main method. The code editor interface is visible at the top, and the 'Outline' view is shown on the right.

## 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 *OptionalClass.java X
2
3 class User{
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
21 public class OptionalClass {
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

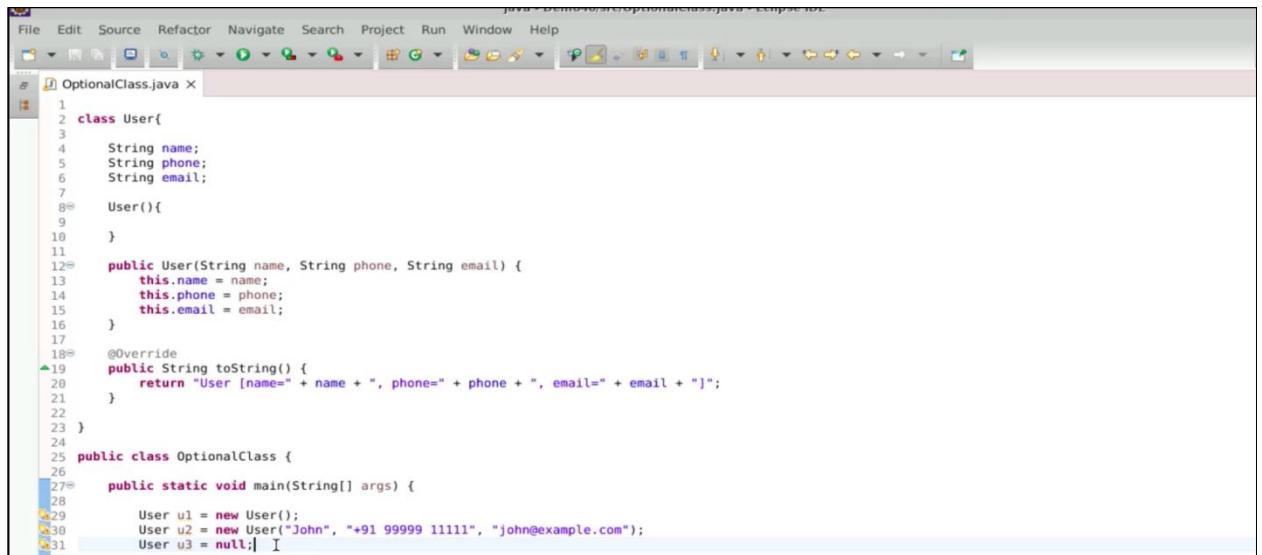
- 3.1 Create a new user object **u1** and assign it some example data. Similarly, create another User object **u2** with different example data

```

File Edit Source Refactor Navigate Search Project Run Window Help
1 *OptionalClass.java X
2
3 class User{
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
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
32     }
33 }

```

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

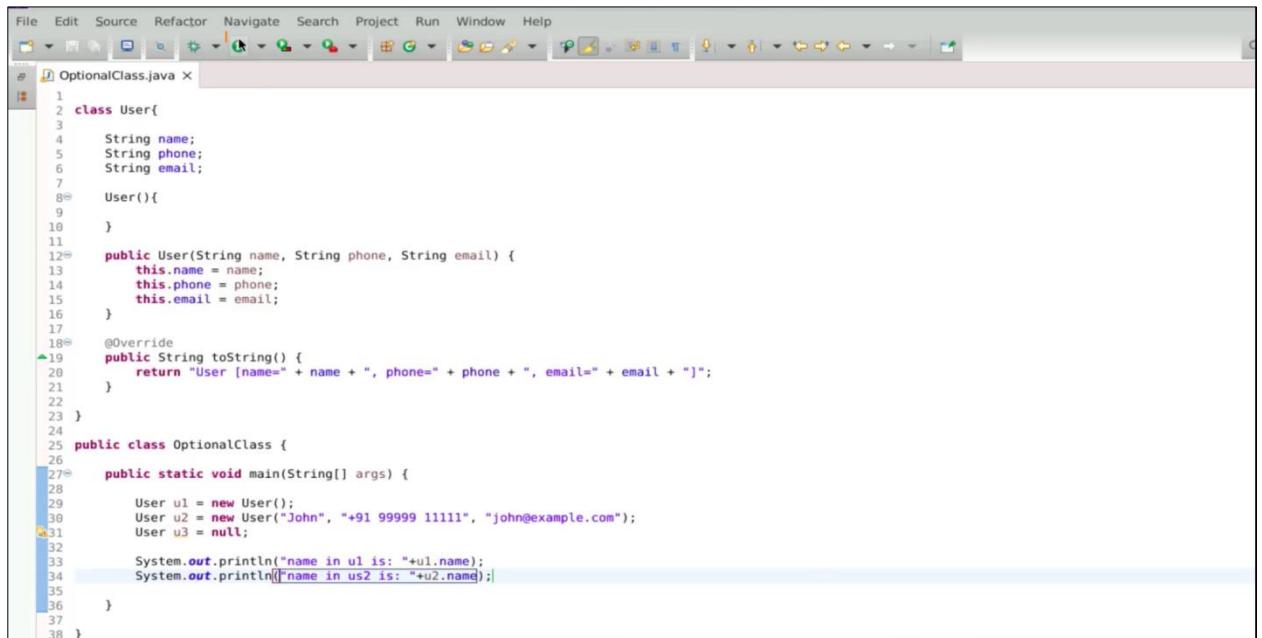


```

File Edit Source Refactor Navigate Search Project Run Window Help
java - Demo07/OptionalClass.java - Eclipse JDT
OptionalClass.java X
1 class User{
2     String name;
3     String phone;
4     String email;
5
6     User(){
7
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
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; I
}

```

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



```

File Edit Source Refactor Navigate Search Project Run Window Help
java - Demo07/OptionalClass.java - Eclipse JDT
OptionalClass.java X
1 class User{
2     String name;
3     String phone;
4     String email;
5
6     User(){
7
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
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);
34        System.out.println("name in u2 is: "+u2.name);
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
2 class User{
3
4     String name;
5     String phone;
6     String email;
7
8     User(){}
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
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     }
34
35 }
36
37
38 }
```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core.prefs  
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
2 class User{
3
4     String name;
5     String phone;
6     String email;
7
8     User(){}
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
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);
34         System.out.println("name in u2 is: "+u2.name);
35         System.out.println("name in u3 is: "+u3.name);
36
37     }
38 }
```

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

```

File Refactor Navigate Search Project Run Window Help
User.java Run OptionalClass
User{
    String name;
    String phone;
    String email;
}
{
    User(String name, String phone, String email) {
        this.name = name;
        this.phone = phone;
        this.email = email;
    }
}

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

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

```

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java X
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 interface. On the left, there's a code editor with Java code. The code defines a User class with fields name, phone, and email, and a constructor that initializes them. It also overrides the toString() method. Below it is an OptionalClass with a static main() method that creates three User objects: one null and two with values.

```
ser
ing name;
ing phone;
ing email;
r(){

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

erride
lic String toString() {
    return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";

class OptionalClass {
    lic static void main(String[] args) {

        User u1 = new User();
        User u2 = new User("John", "+91 99999 11111", "john@example.com");
        User u3 = null;
```

In the top right, there's a terminal window titled "Console X" showing the output of the application. It starts with the message "<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjd". Then it shows an exception: "Exception in thread \"main\" java.lang.NullPointerException: Cannot invoke \"OptionalClass.main(OptionalClass@1)" at OptionalClass.main(OptionalClass@1:33)".

- 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 Eclipse IDE interface. On the left, the code editor displays `OptionalClass.java` with the following content:

```
1  class User{
2      String name;
3      String phone;
4      String email;
5
6      User(){
7
8      }
9
10     public User(String name, String phone, String email) {
11         this.name = name;
12         this.phone = phone;
13         this.email = email;
14     }
15
16     @Override
17     public String toString() {
18         return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
19     }
20
21 }
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     }
38 }
```

On the right, the **Console** view shows the output of the application:

```
<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core.prefs
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 interface. On the left, there is a code editor window containing Java code. On the right, there is a 'Console' tab showing the output of the program's execution.

```

User.java x
ser{
    ing name;
    ing phone;
    ing email;
}

r(){

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

erride
lic String toString() {
    return "User [name=" + name + ", phone=" + phone + ", email=" + email + "]";
}

class OptionalClass {
    lic 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());
    }
}

```

The console output shows:

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.java\_11.0.1.v20191217-1200/bin <terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.java\_11.0.1.v20191217-1200/bin
a.lang.NullPointerException: Cannot read field "name" because "u3" is null
OptionalClass.java:35

## 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 interface. On the left, there is a code editor window containing Java code. On the right, there is a 'Console' tab showing the output of the program's execution.

```

File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java x
1  package com.simplilearn;
2
3  public class User {
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  }
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(u1.name);
39
40      }
41
42  }

```

The console output shows:

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 I
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(u1.name);
39         if(checkForName.isPresent()) {
40             System.out.println("name in u1 is: "+u1.name.toUpperCase());
41         }else {
42             System.out.println("Sorry, name in u1 is null");
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
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     }
46
47 }
48

```

Console X

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/compiler.jar  
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

```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java X
1
2
3
4
5
6
7
8
9
10
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**

```

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> 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**

```

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> checkForUser = Optional.ofNullable(u3);
46
47     }
48 }

```

Problems Javadoc Declaration Console X

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/compiler.log

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> 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}

```

#### 5.5 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> 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}

```

Problems Javadoc Declaration Console X

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/bin

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

```

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 1111", "john@example.com");
32     User u3 = new User("Fionna", "+91 99999 2222", "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> checkforUser = Optional.ofNullable(u3);
46     if(checkforUser.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

```

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 1111", "john@example.com");
32     User u3 = new User("Fionna", "+91 99999 2222", "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> checkforUser = Optional.ofNullable(u3);
46     if(checkforUser.isPresent()) {
    }
}

```

Problems Javadoc Declaration Console X

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/bin  
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



```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java X
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> 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     Optional<String> checkUserEmail = Optional.ofNullable(u2.email);
53     checkUserEmail.ifPresent(System.out::println); // print the data in case if its not null
54
55 }
```

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

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

- Left Panel (Code Editor):** Displays the file `OptionalClass.java`. The code uses `Optional` and `Optional.ofNullable` to handle nullable user objects.
- Right Panel (Terminal):** Shows the output of the application run. It prints the names of three users in uppercase: JOHN, FIONNA, and JOHN@example.com.

```
File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java X
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> checkForUser = Optional.ofNullable(u3);
46        if(checkForUser.isPresent()) {
47            System.out.println("name in u3 is:" + u3.name);
48        } else {
```

```
<terminated> OptionalClass [java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/jdt.launch
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

```

File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java Run OptionalClass
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 1111", 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> checkForUser = Optional.ofNullable(u3);
46     if(checkForUser.isPresent()) {
47         System.out.println("name in u3 is: "+u3.name);
48     } else {
        }
    }
}

```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core.prefs  
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

```

File Edit Source Refactor Navigate Search Project Run Window Help
OptionalClass.java X
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

```

The line of code highlighted in blue is: **checkUserEmail.ifPresent(System.out::println); // print the data in case if its not null**

## 5.12 When you run the code it shows no value present

```

File Edit Source Refactor Navigate Search Project Run Window Help
File 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", 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);

```

<terminated> OptionalClass [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/compiler.log  
name in u2 is: JOHN  
name in u3 is: Fionna  
Exception in thread "main" java.util.NoSuchElementException:  
at java.base/java.util.Optional.get(Optional.java:143)  
at OptionalClass.main(OptionalClass.java:54)

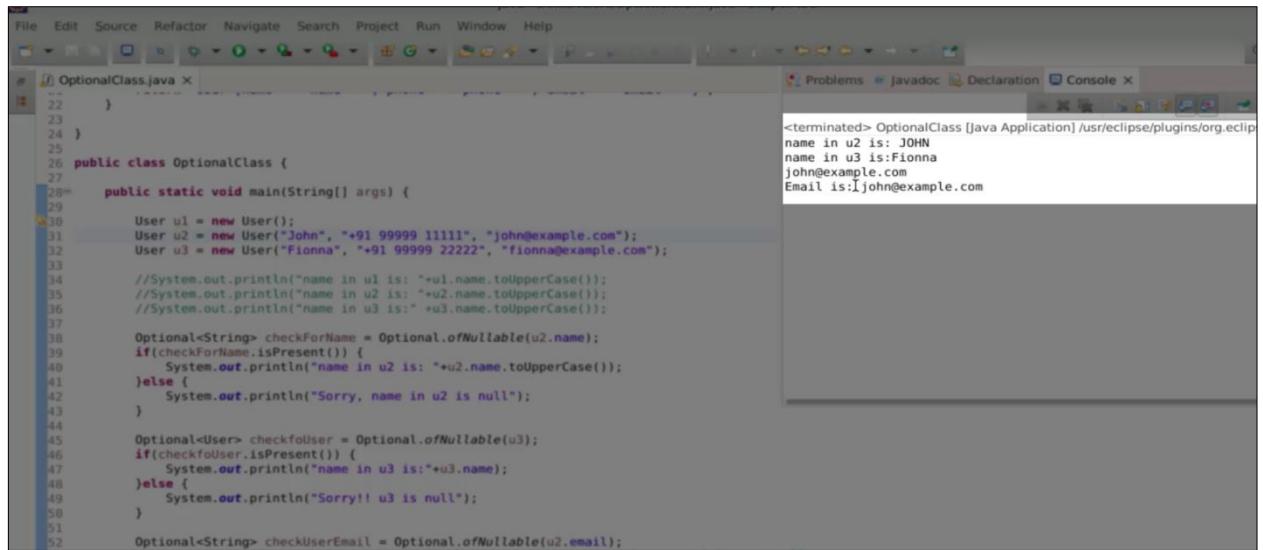
## 5.13 Add the email: john@example.com

```

File Edit Source Refactor Navigate Search Project Run Window Help
File 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);

```

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



The screenshot shows the Eclipse IDE interface. On the left, the code editor displays `OptionalClass.java` with Java code that creates three `User` objects (`u1`, `u2`, `u3`) and prints their names and emails. It uses `Optional.ofNullable()` to handle null values. The code is as follows:

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

In the bottom right corner, the `Console` view shows the program's output:

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

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