# Lesson 04 Demo 06

# Implementing Exception Handling

**Objective:** To implement exception handling in Java

**Tools required:** Eclipse IDE

**Prerequisites:** None

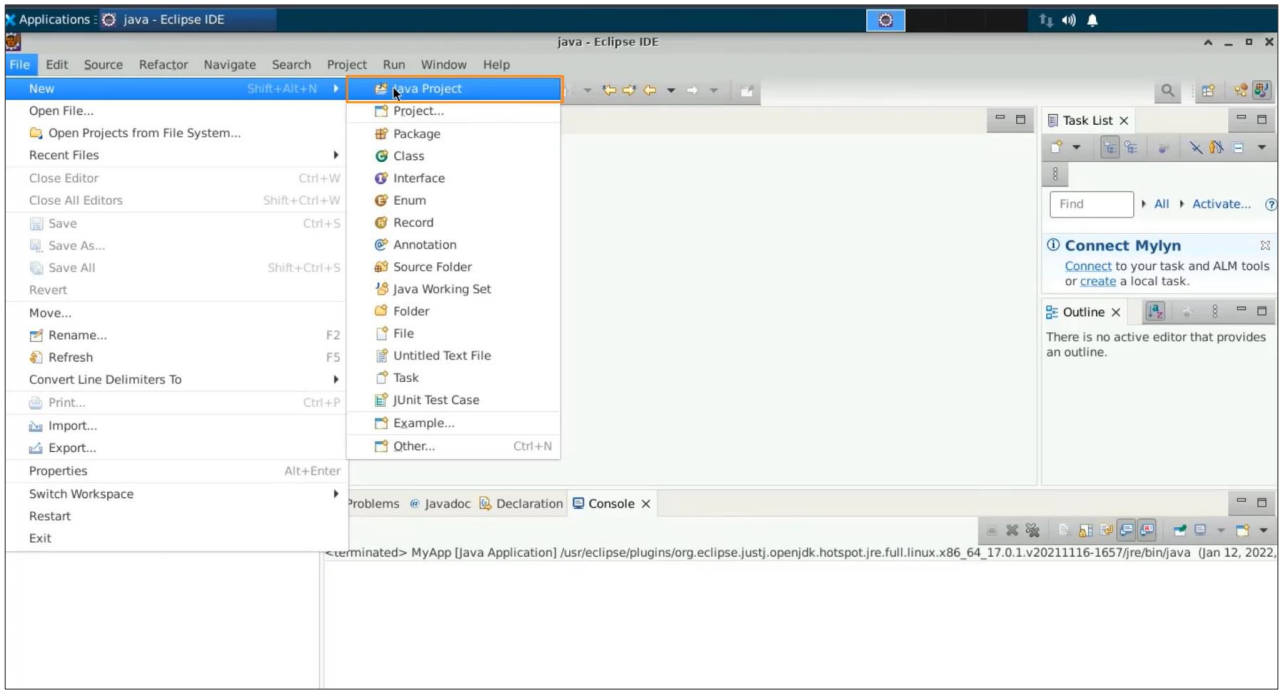
**Steps to be followed:**

1. Open IDE and create a new project
2. Create an array and take inputs from the user
3. Execute the code with sample data
4. Implement the method print stack trace
5. Use Try and the catch functionality and execute the code

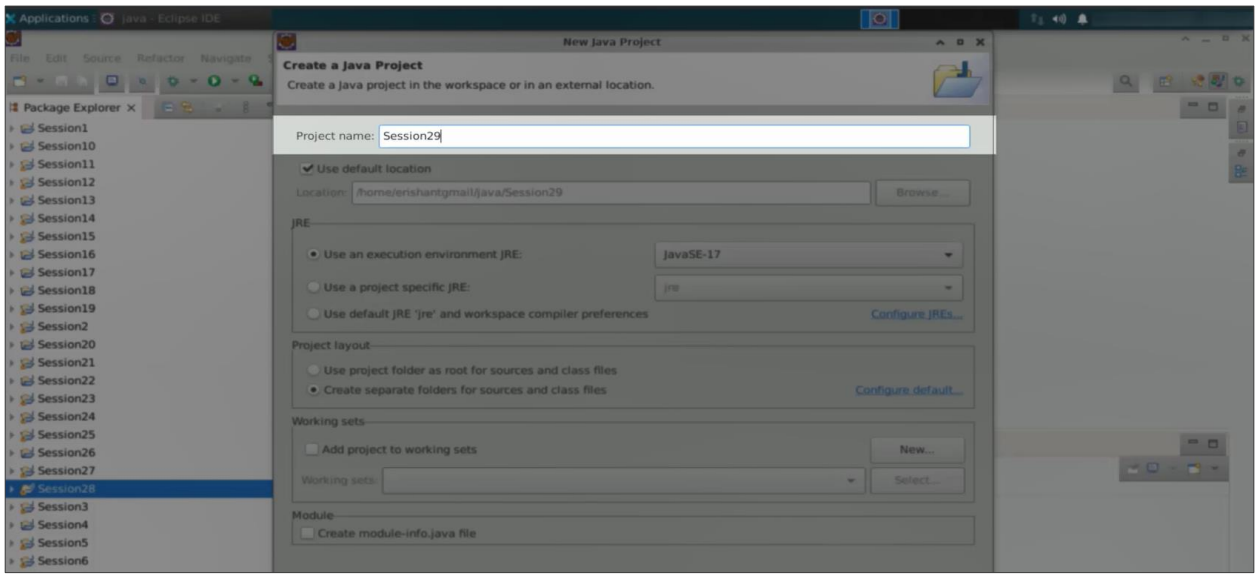
## Step 1: Open IDE and create a new project

1.1 Open the **Eclipse IDE**

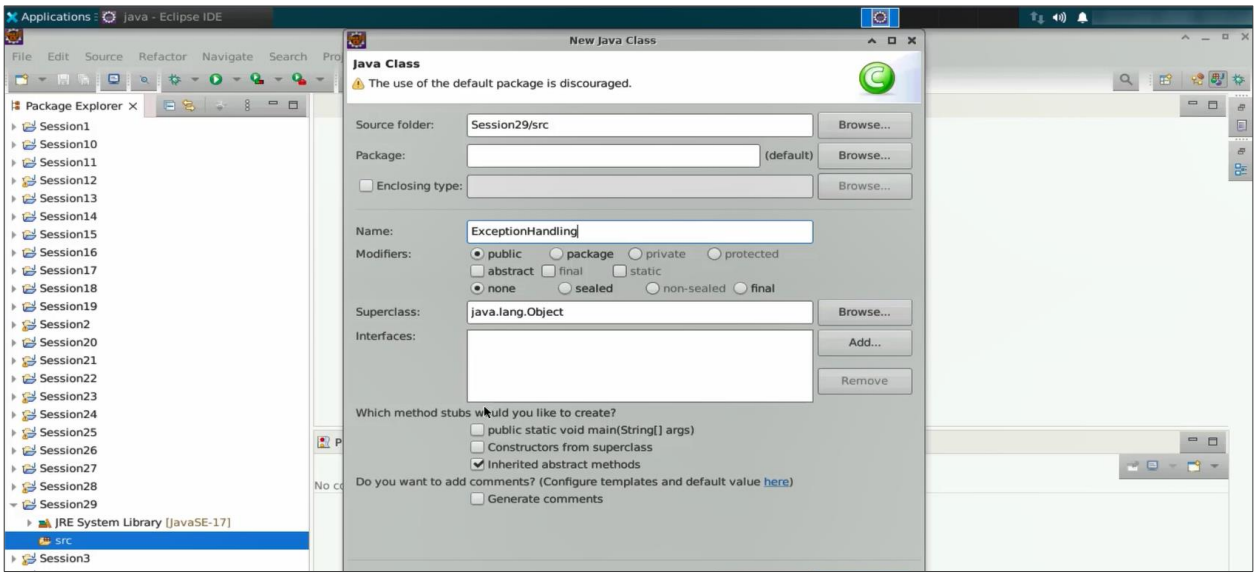
1.2 Select **File**, then **New,** and then **Java project**




1.3 Name the project "Session29", uncheck "**Create a module-info.java file**", and press Finish.

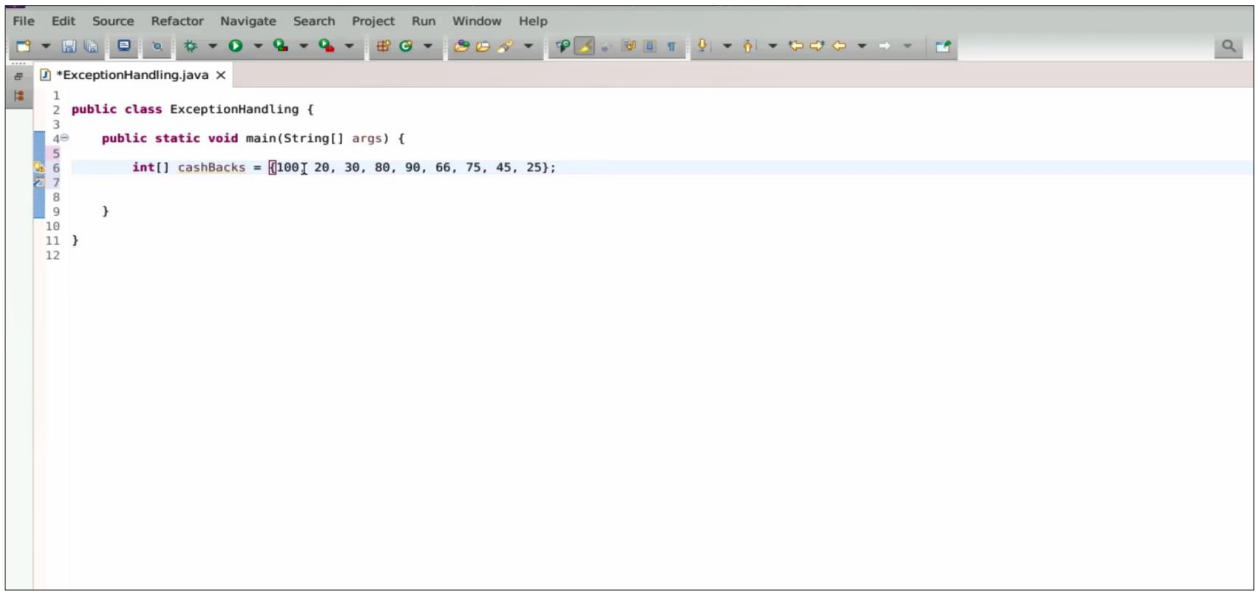


1.4 With a **Session29** on the src, do a right-click and create a **new class**. Name this class as an **ExceptionHandling**, then select the **main method,** and then select **finish.**

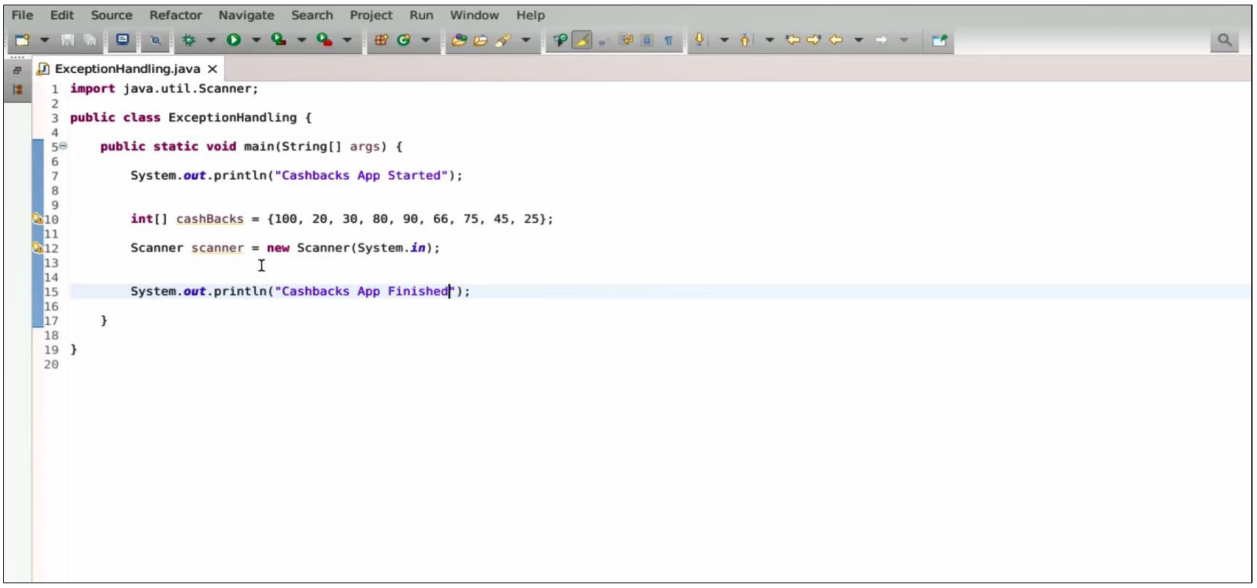



## Step 2: Create an array and take inputs from the user

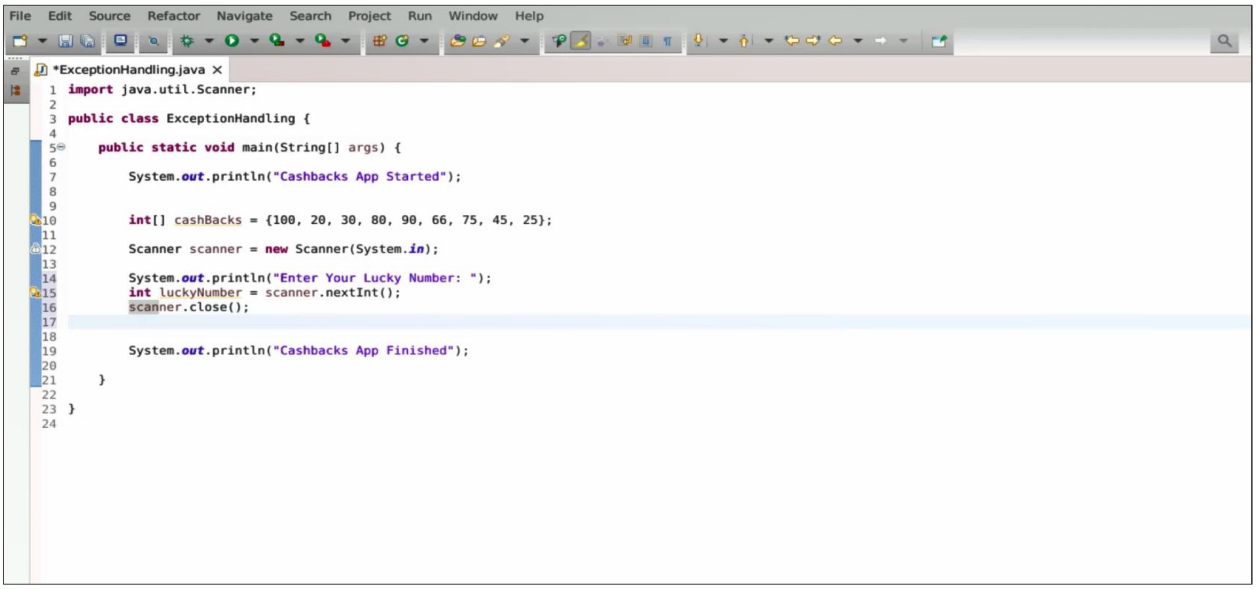
2.1 Create an array called cashbacks. You can have a few of the cashback as rewards.



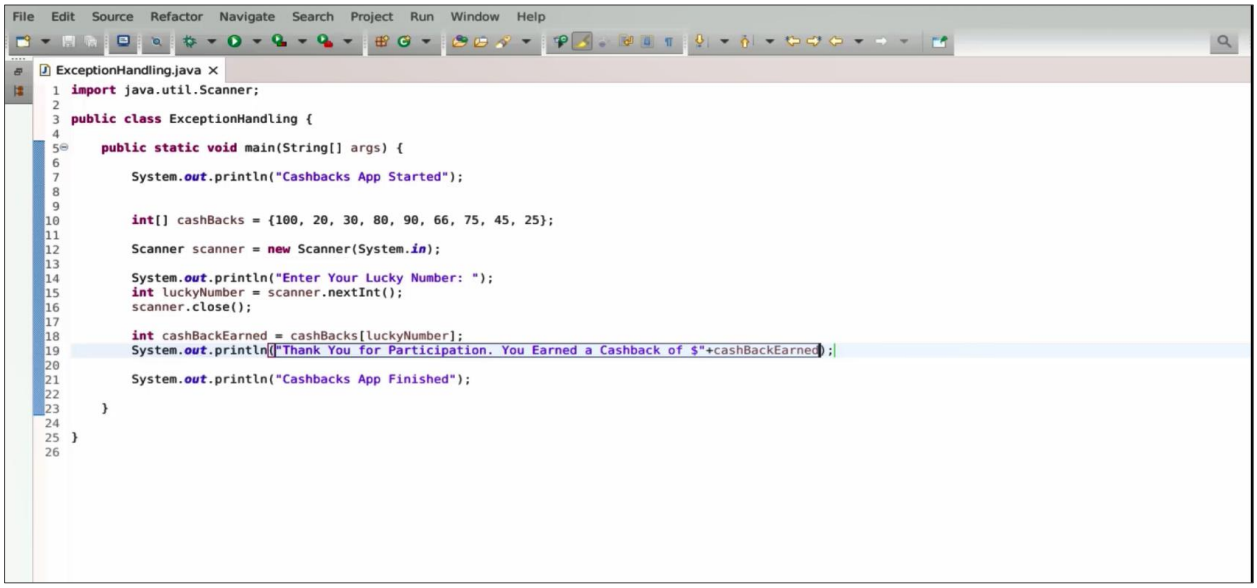
2.2 To take input from the user, you will use the Scanner class, initializing it as a new Scanner where you will pass System.in as the input. Here, use System.out.println("Cash backs app started"), and your last line of code will be System.out.println("Cash backs app finished").




2.3 Now, let us tell the user to give input by writing "Enter your lucky number." Then write "int luckyNumber = scanner.nextInt();" to read the integer from the console. Once you are done with it, you can use "scanner.close();" to close the scanner.

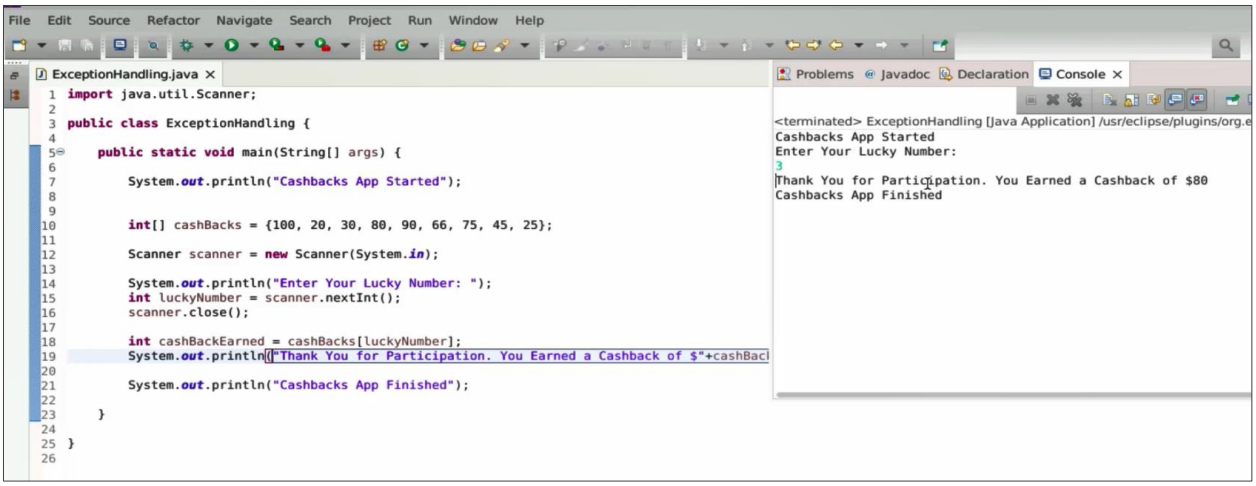


2.4 The cashback earned by the user would be the cashback of the lucky number. That is, whatever lucky number the user enters, it will give a cashback to the user. Next, print "Thank you for your participation, you earned a cashback of $" plus the cashback earned.

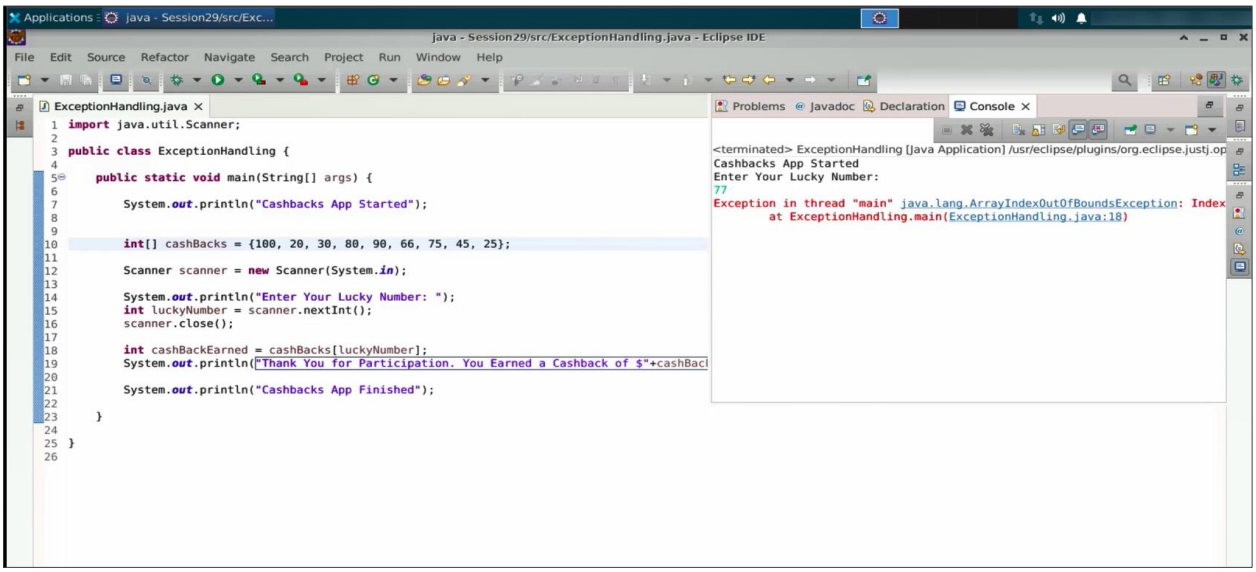



## Step 3: Execute the code with sample data

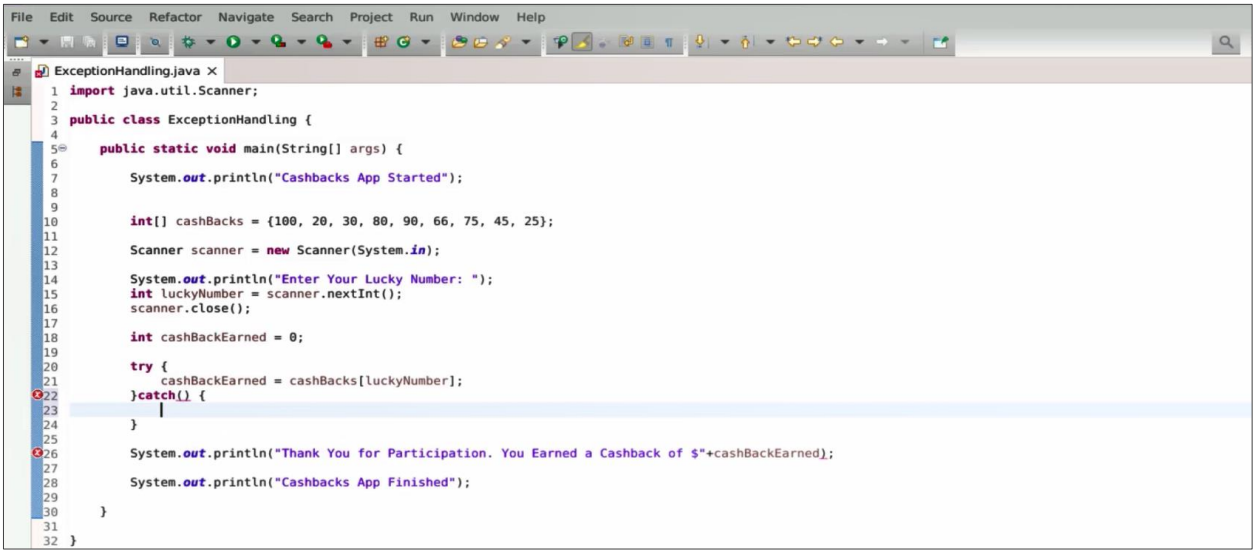
3.1 Run the program to see what happens. The Cashbacks app starts and prompts you to enter your lucky number. Enter three, hit enter, and it says, "Thank you for your participation, you earned a cashback of $80." The program automatically picks and gives you cashback based on the index.



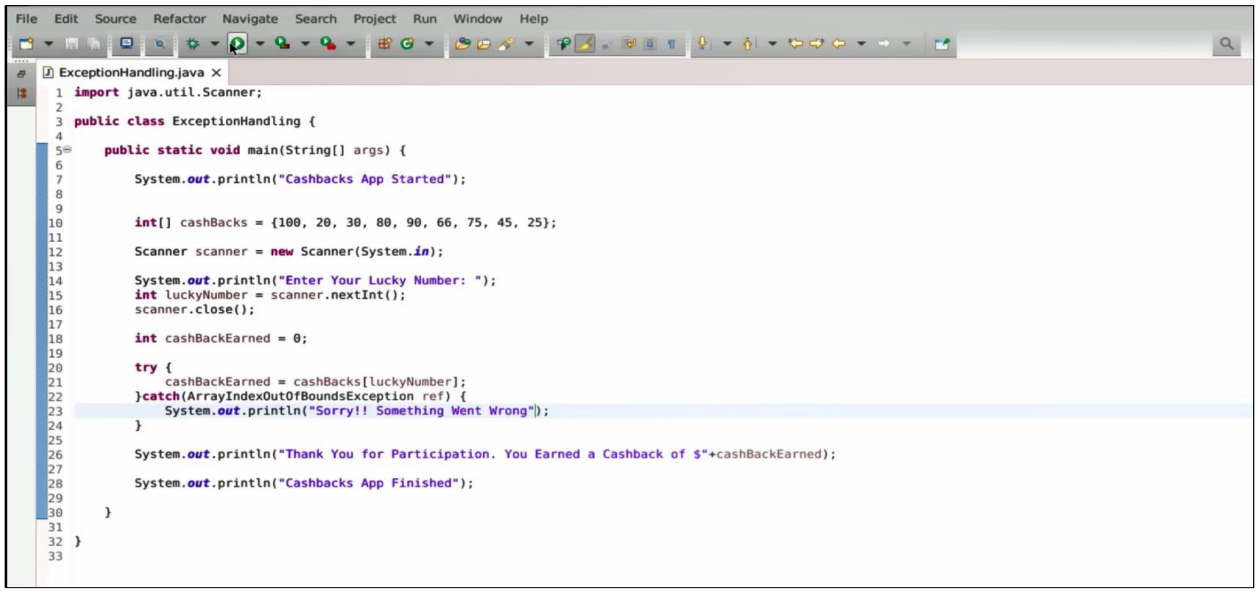
3.2 Consider a scenario where the prompt says, "Enter your lucky number," and the user enters a number like 77. Since 77 is out of the array's range, hitting enter will cause an error. This is not a compile-time error. When the error occurs, it disrupts and crashes your program, meaning your app started but never finished.



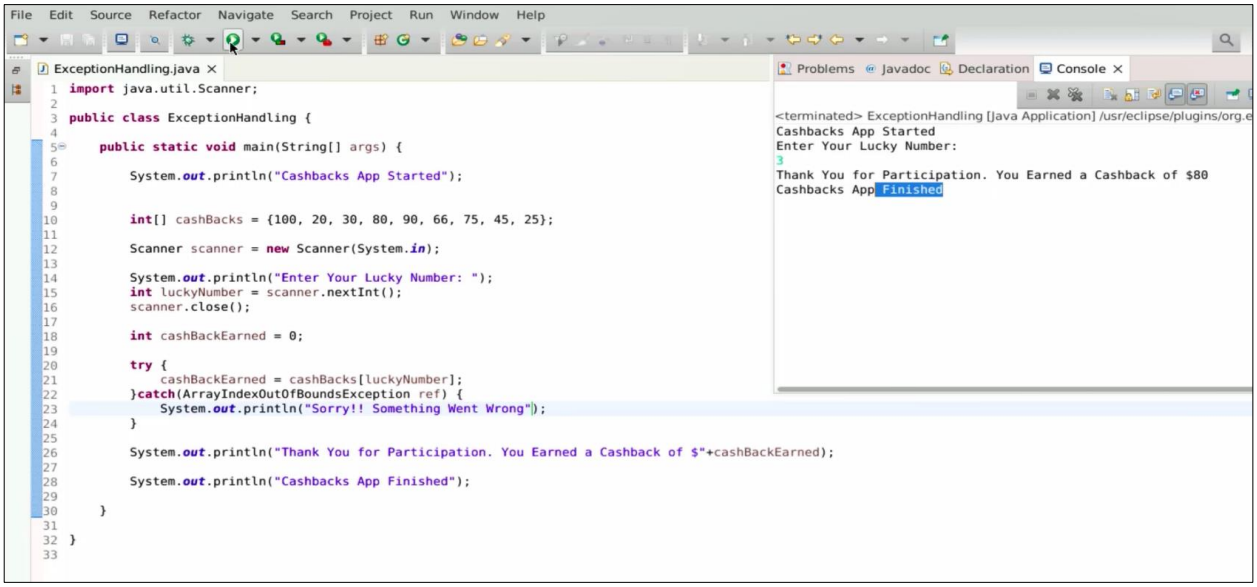

3.3 Now, you will create this integer cashback with a default value of zero, and you will surround this line of code with a try block. Whatever statement is written inside the try block, if any error occurs, you will navigate the user to a block called catch.



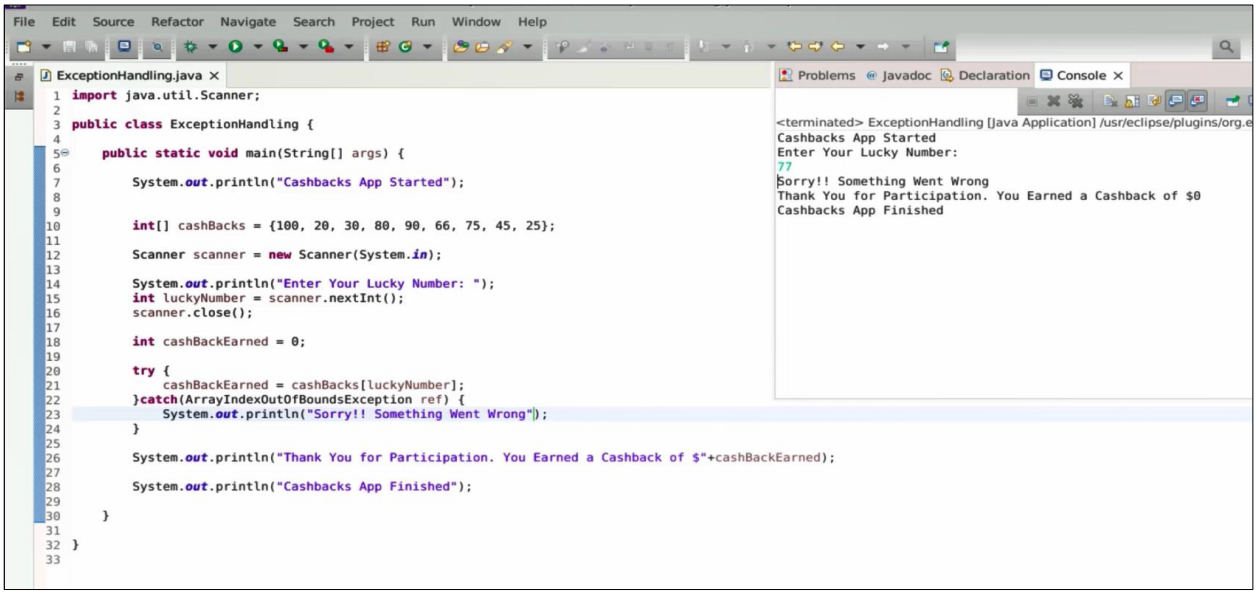
3.4 In your catch block, write "ArrayIndexOutOfBoundsException" and give a reference variable. Then use System.out.println("Sorry, something went wrong."). This try and catch program will now handle the exception.



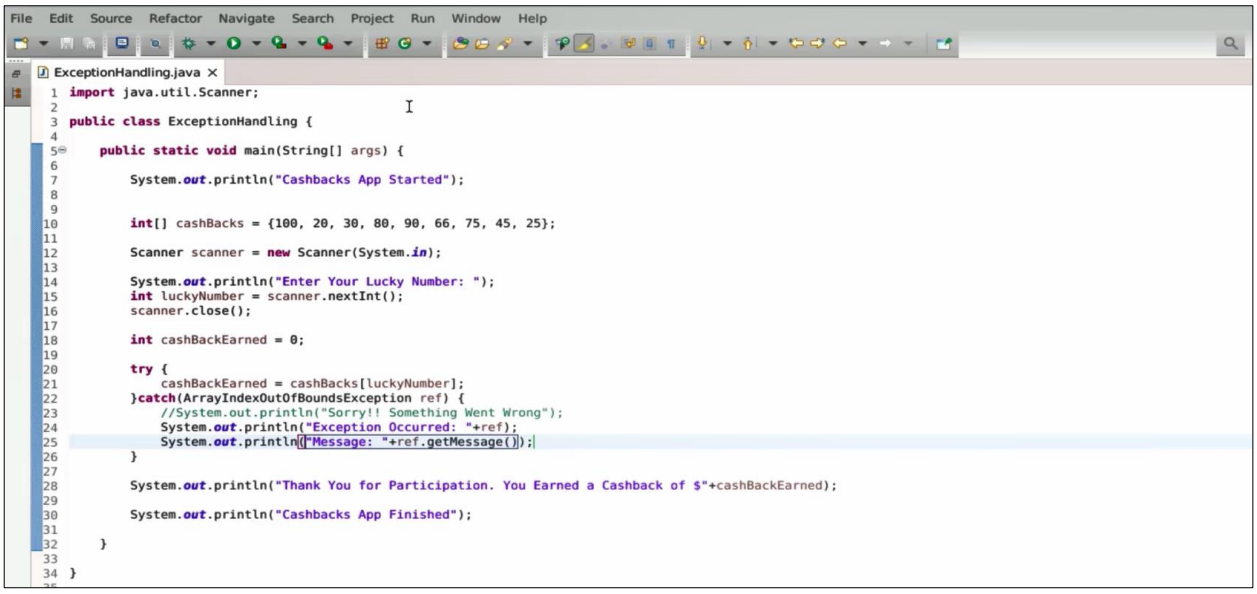

3.5 Run the code again and see what happens. This time when you enter the correct lucky number, your program flows in the best possible ways. You can see the CashBack app finished.



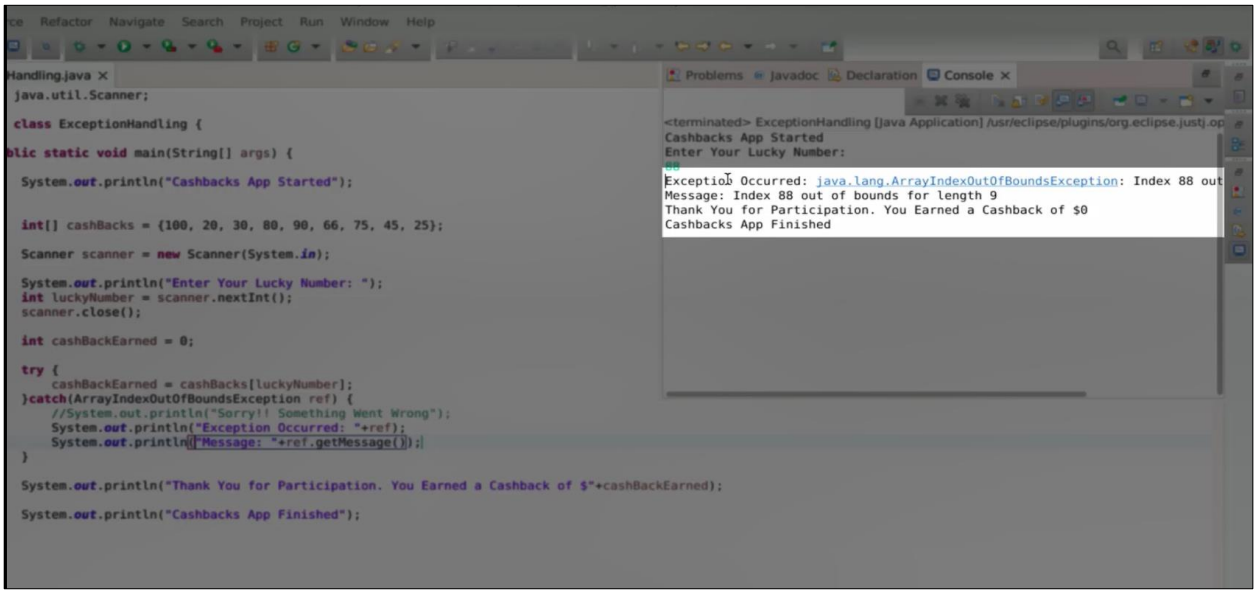
3.6 If the user will enter a wrong input 77 here, your array of cash backs will crash the program, but you can also see that your cashback app is also finishing, and you are even displayed with a message called Something went wrong, thank you for participation, you earned a cashback of zero dollars.




3.7 The other way around, when you are into development you can write exception occurred, and for the exception, you print the reference variable, that is, you can print down this reference variable, and if you want you can write a message here which you can obtain from your reference by executing the method called get message.

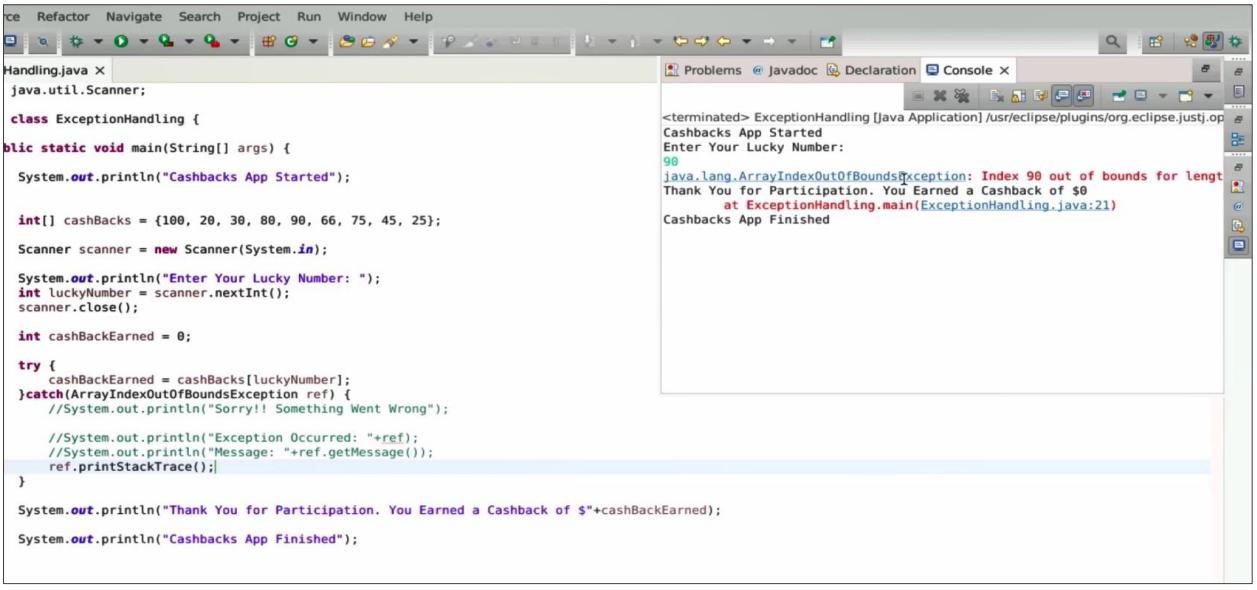


3.8 Now, when you run the same program and give the wrong input, you can see the exception occurred. This is the Error, and this is the message that index 88 is out of bounds for length number 9.



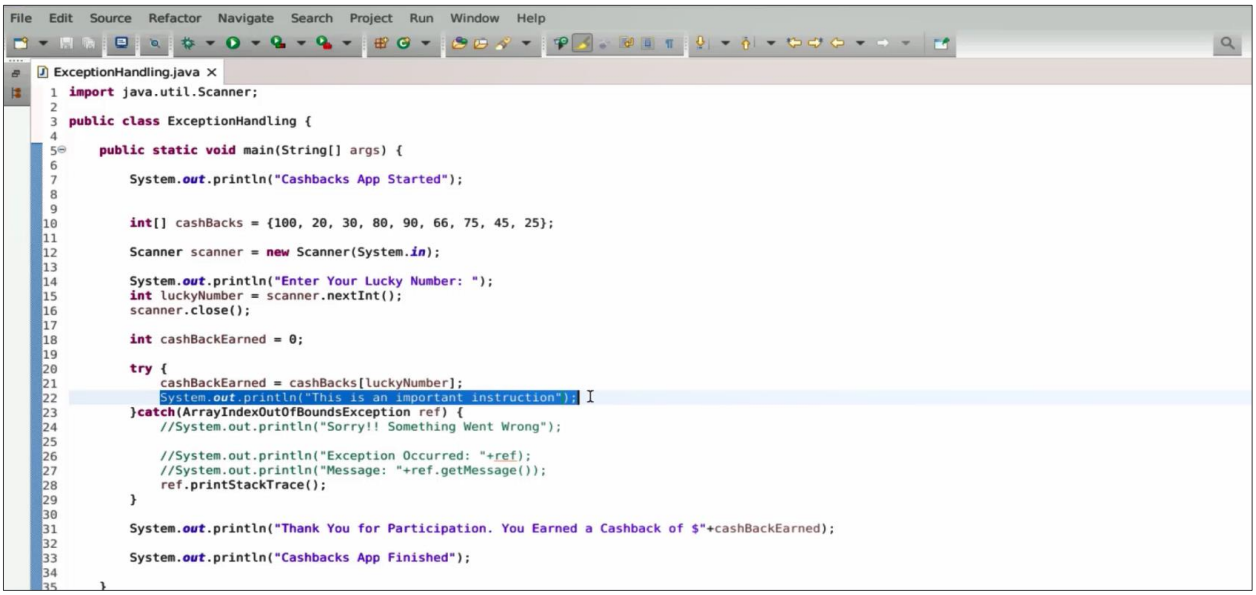

## Step 4: Implement the method print stack trace

4.1 You can execute another method known as printStackTrace. Write reference.printStackTrace(), and it will automatically print the entire trace of the exception. If you enter an incorrect number, you will see the exception occurred at line number 21 of your ExceptionHandling.java file. This was the exact line where the error occurred.

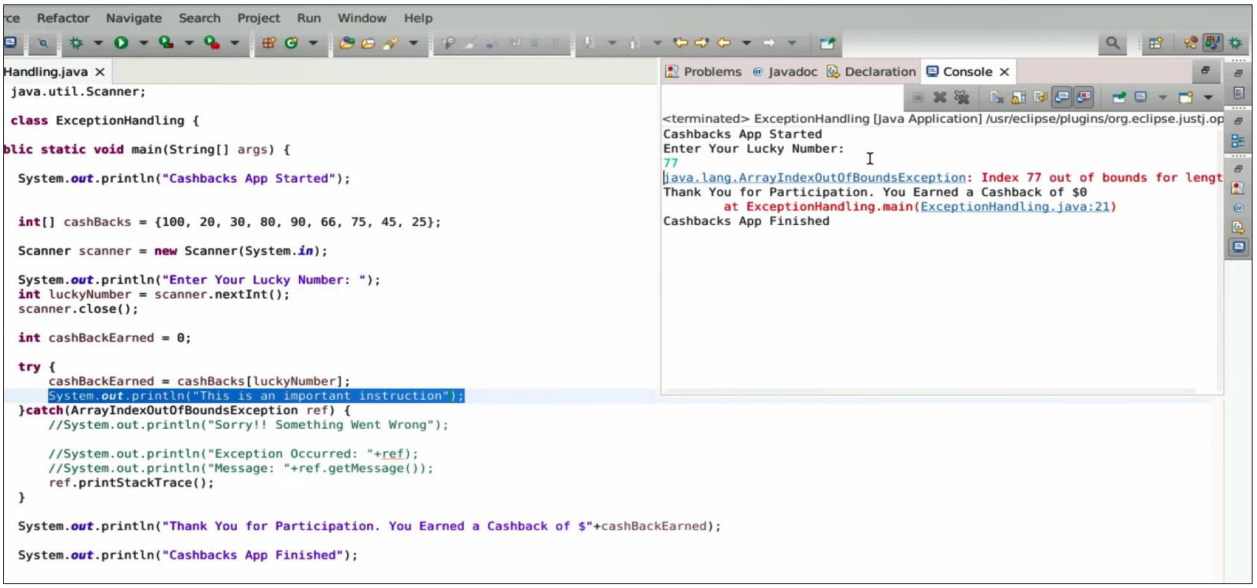


## Step 5: Use Try and the catch functionality and execute the code

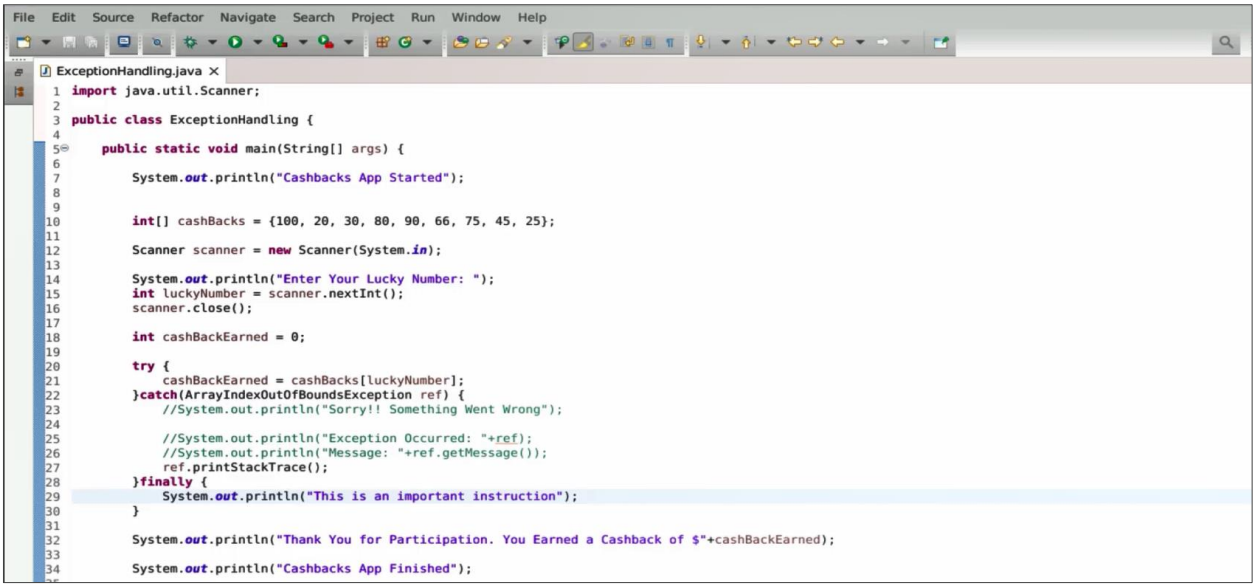
5.1 Try and catch blocks save your program from runtime errors, but remember, this does not mean you should surround your entire code with try and catch. There is one more thing you should know about try and catch. If you have some instruction written like this: "This is an important instruction.



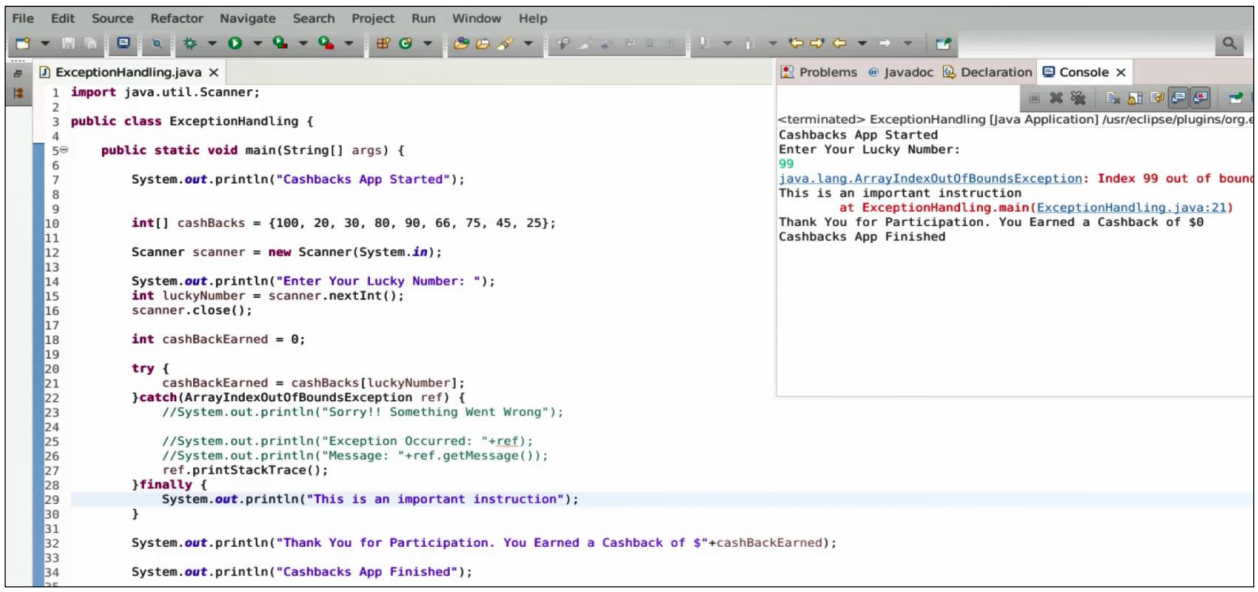
5.2 Re-run this same program. The moment you use the wrong number as input, you can see the program finishes in the best possible way. It states that the app is finished, but the important instruction is not executed. This means line number 21 was where you got the error, and from there, the control directly went to the catch block.



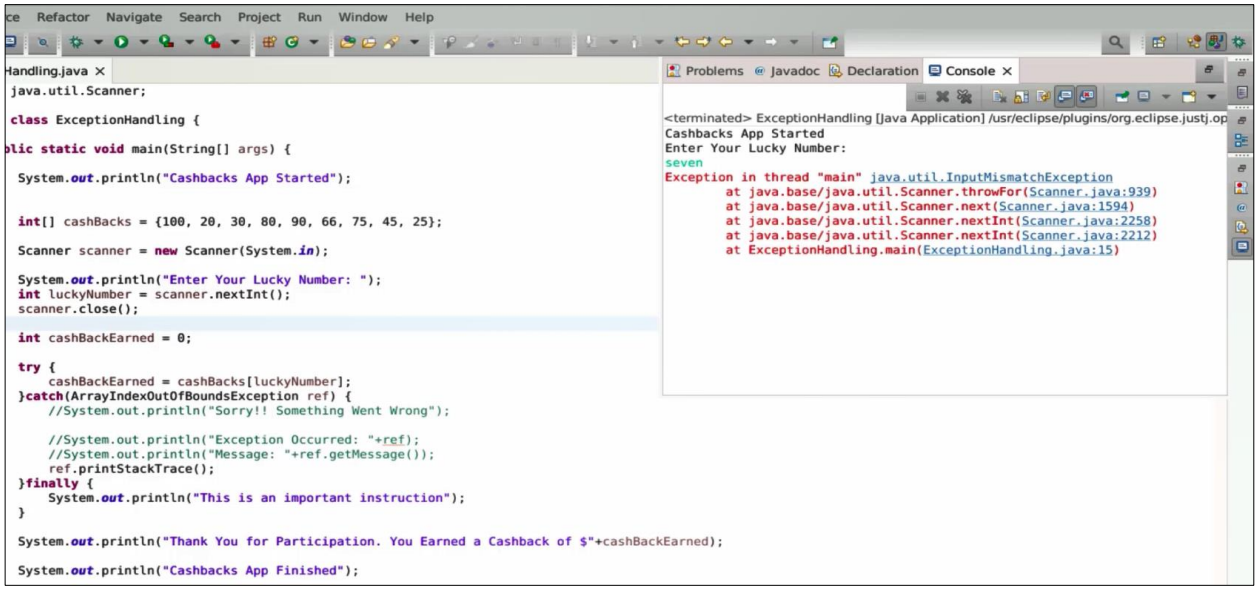

5.3 What can you do? Either you can write these instructions separately somewhere, or you get these instructions that are important and move them to the finally block. Finally, is a piece of block that will be executed. You have handled the exception, or you have not managed the exception.



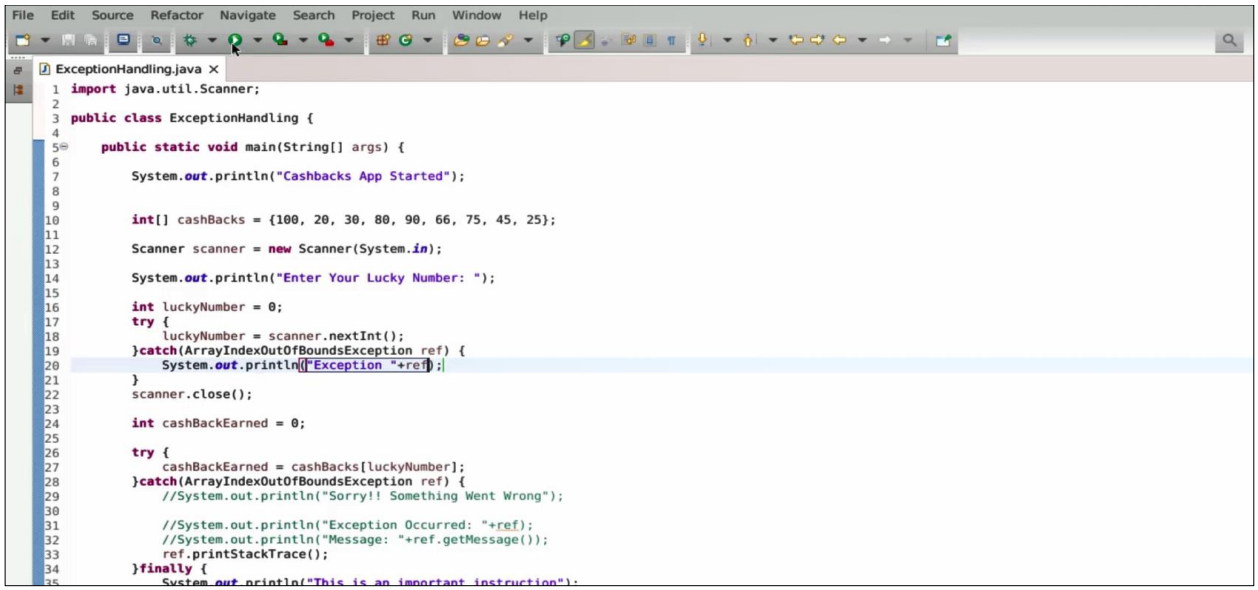
5.4 Re-run the program, and write enter your lucky number as 99, and now you can see this is an important instruction coming in your output.



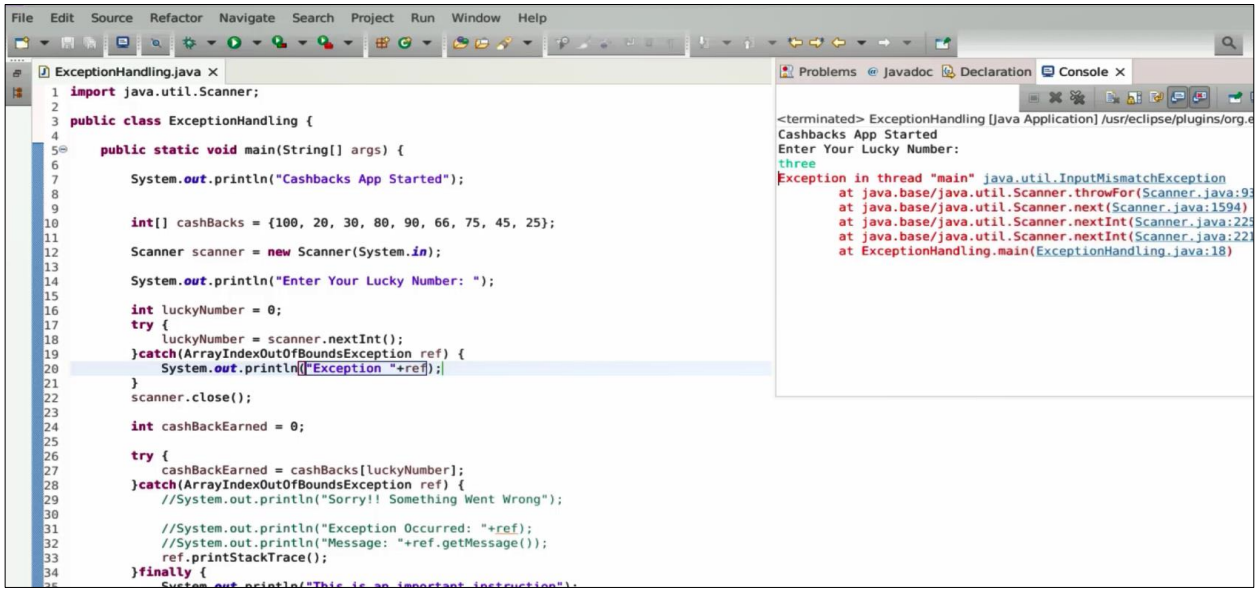

5.5 That is also possible, but if you run the program once again and if you try to add the lucky number as seven, this is illegal, and it is not possible to convert it to the numeric 7.



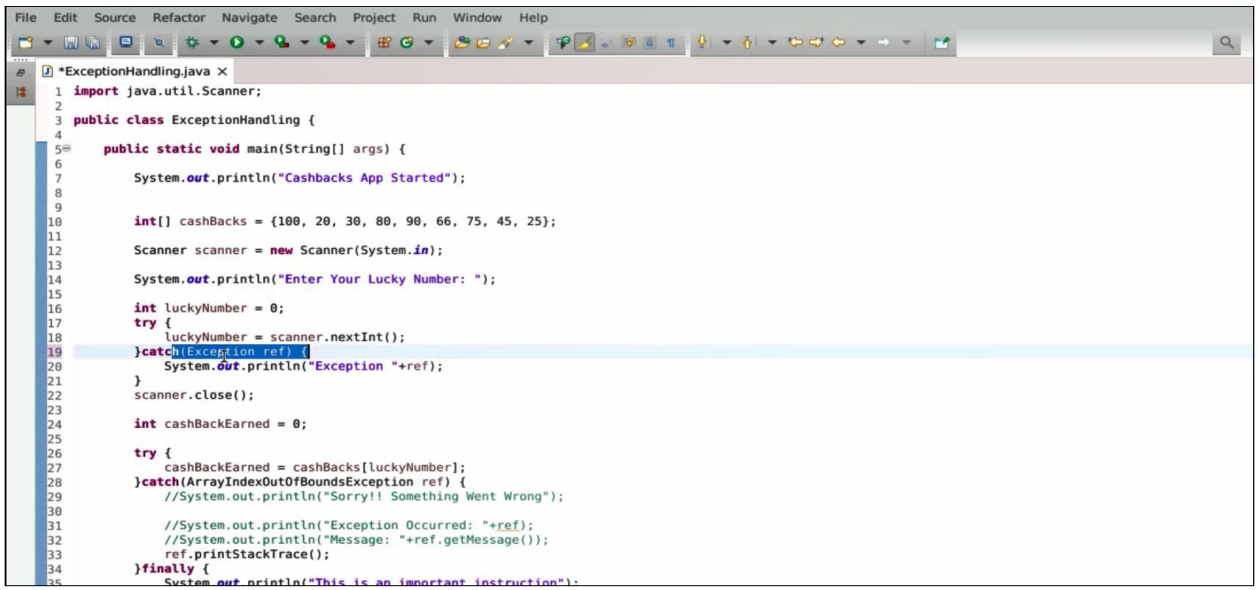
5.6 You have various possibilities where you can end up an error. You can surround this Line number 18 with the try and catch. You will put this in the try block and in the catch block, you will put the same array index out of bound exception as the reference, then Print exception,and then print the reference variable.



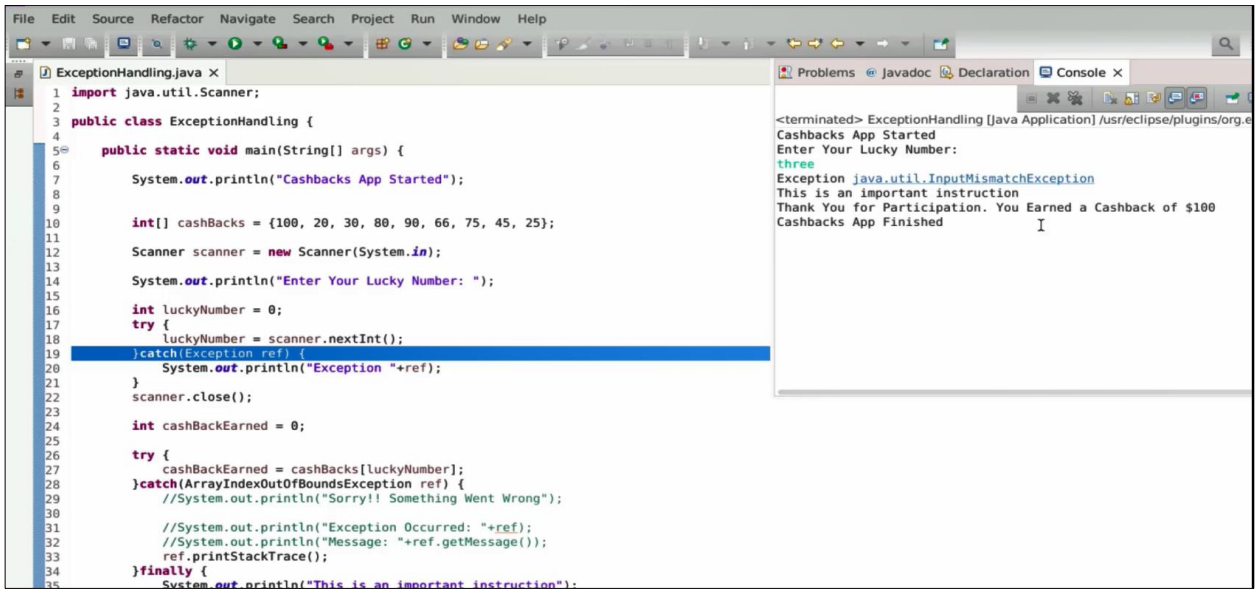

5.7 When you run the program, if you enter the correct number then everything works fine, but if you enter 3, you can see that even if you have placed the try and catch, still your program crashes. What is meant by this crash?



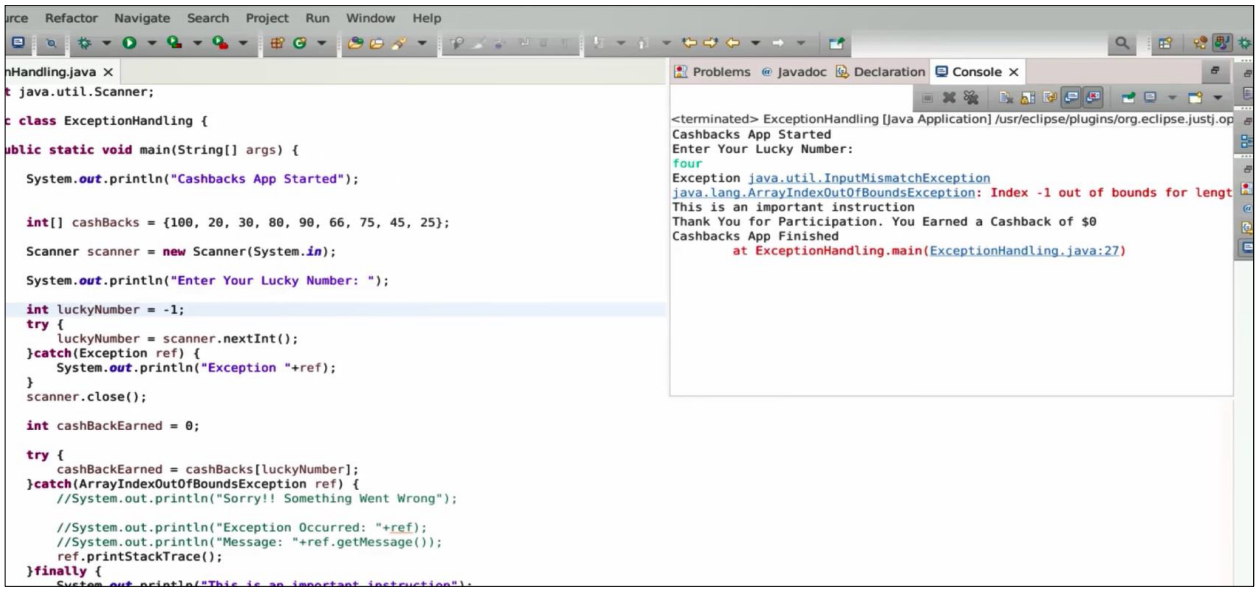
5.8 The crash is because you have handled the array index out of bounds exception and not this thing called input mismatch exception. In Java, the class exception is the parent to all the exceptions, that is, this class exception from the Java dot Lang is the parent to all the exceptions. So, you can use the polymorphic statement.



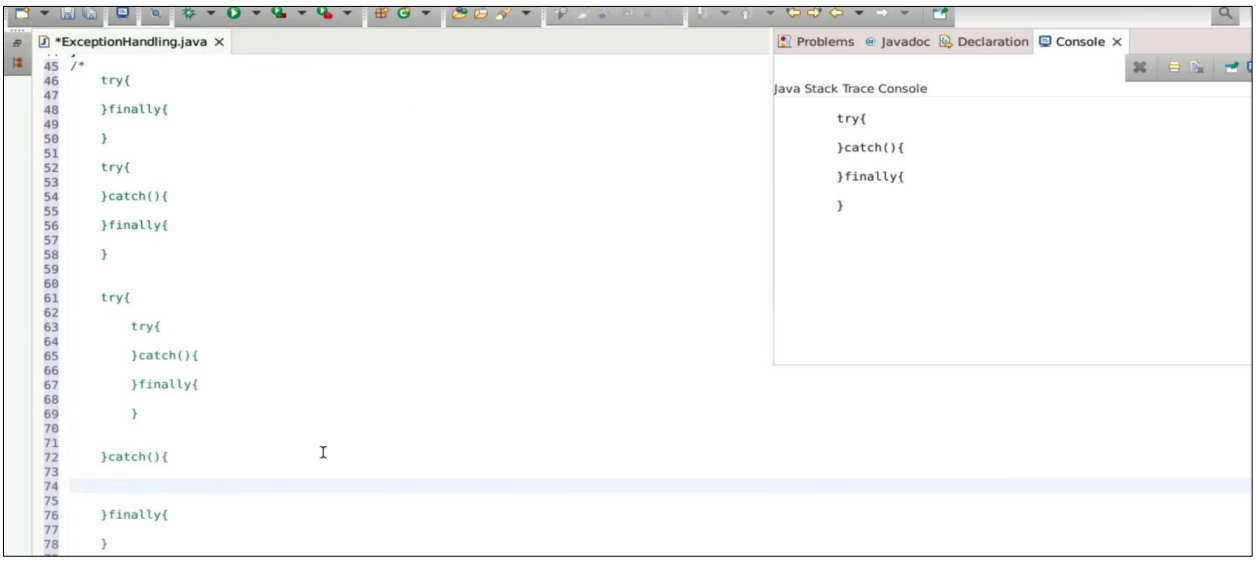

5.9 When you run the code again and enter the wrong number. You can see that it says, the exception is input mismatch exception. You get your finally block executed and you got a cashback of hundred dollars because the lucky number was by default 0.



5.10 You can change your lucky number as -1 here, and when you run the code here, and if you enter a wrong input like 4. So, here you are with the $0 output and the cashback app is also finishing. Thus, try and the catch block can be used along with the finally.




5.11 Finally is the block that executes regardless of whether the exception is managed. You can combine a try block with a finally block alone, but errors will not be handled, and the finally block will run before the program crashes. The try block alone cannot exist; use try with catch, then add a finally block. With try-catch-finally, you can even nest these blocks.




By using the following steps, you have successfully implemented exception handling in Java, demonstrating how to manage runtime errors effectively.