Lesson 04 Demo 02

Implementing Anonymous Classes

Objective: To demonstrate the implementation of anonymous classes **Tools required:** Eclipse IDE **Prerequisites:** None

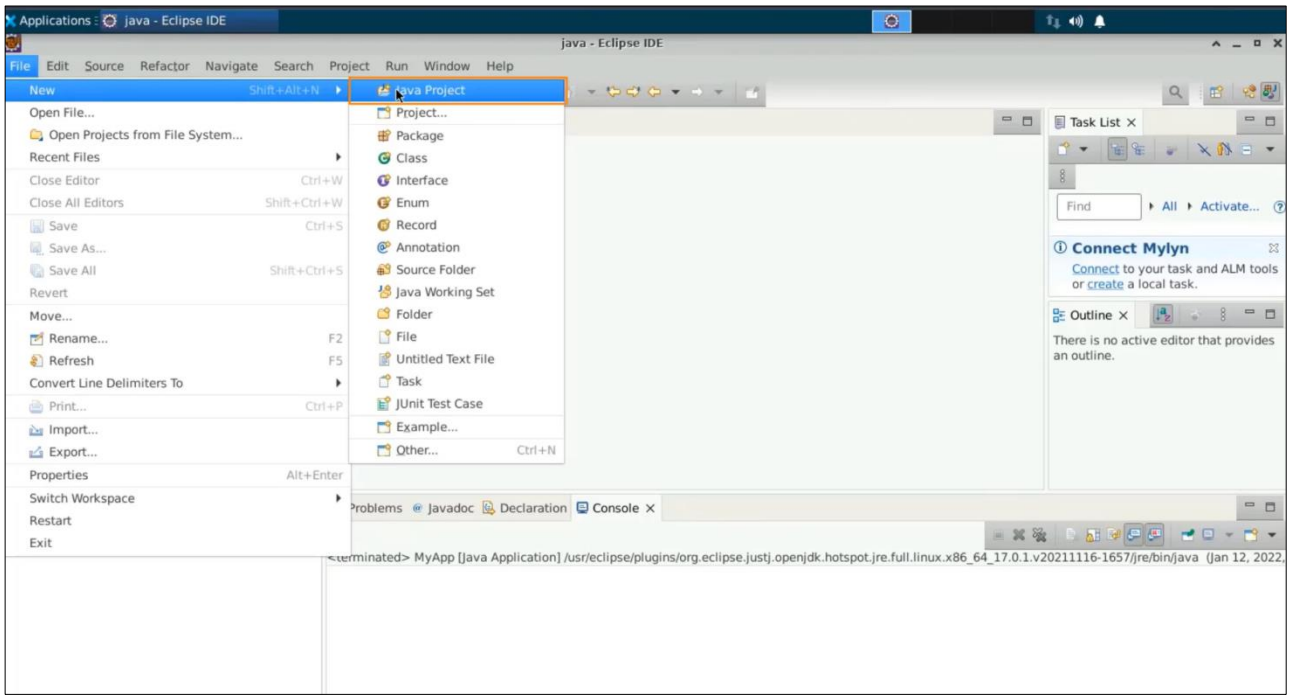
Steps to be followed:

1. Open the IDE and create a new project
2. Write an interface and execute the code with example data
3. Execute the code and print the message accordingly
4. Override the methods called on success and failure
5. Use anonymous classes

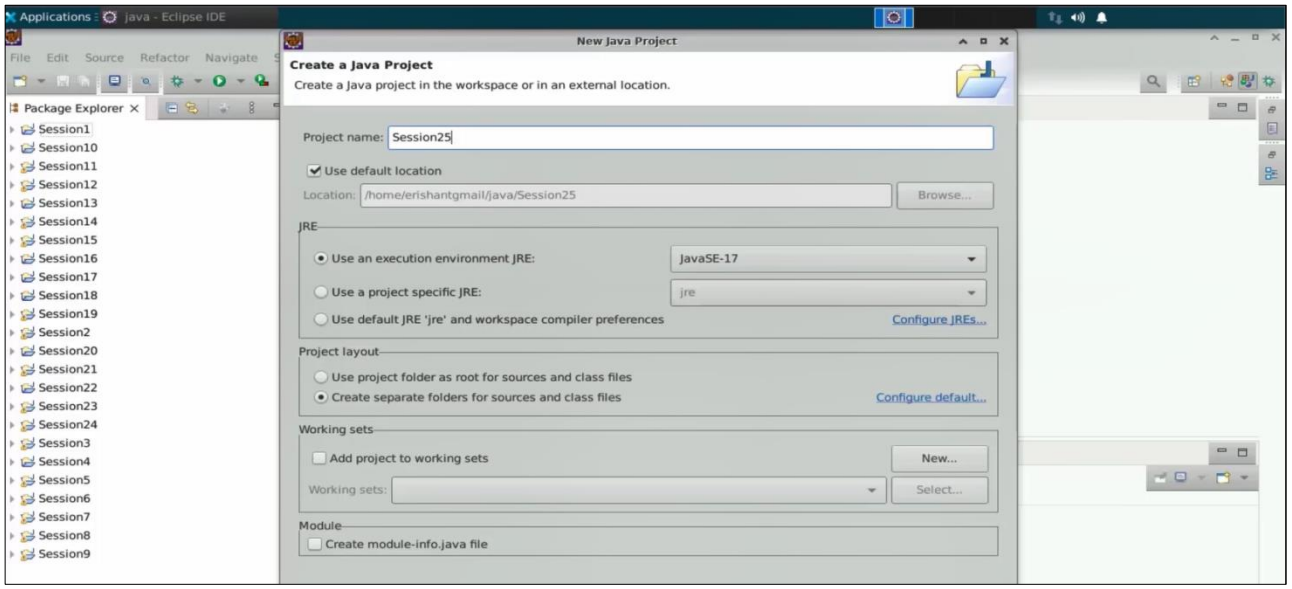
Step 1: Open IDE and create a new project

1.1 Open the **Eclipse IDE**

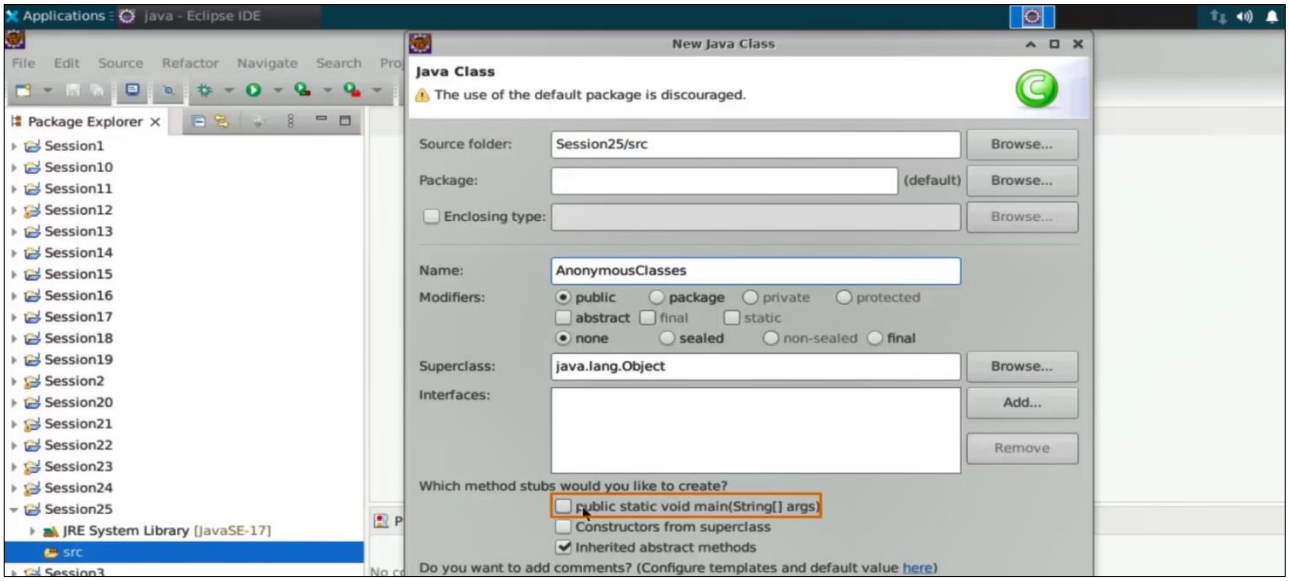
1.2 Select **File**, then **New,** and then **Java project**



1.3 Name the project "Session25", uncheck "Create a module-info.java file", and press Finish.

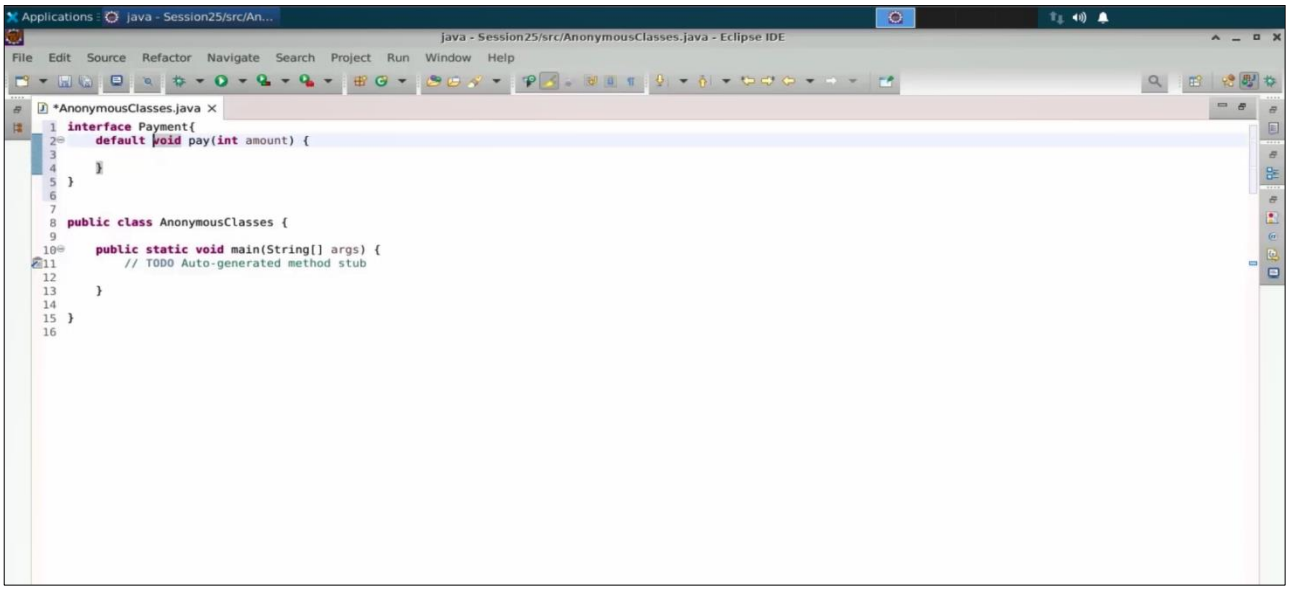


1.4 With a **Session25** on the src, do a right-click and create a **new class**. Name this class as an **AnonymousClasses**, then select the **main method,** and then select **finish.**

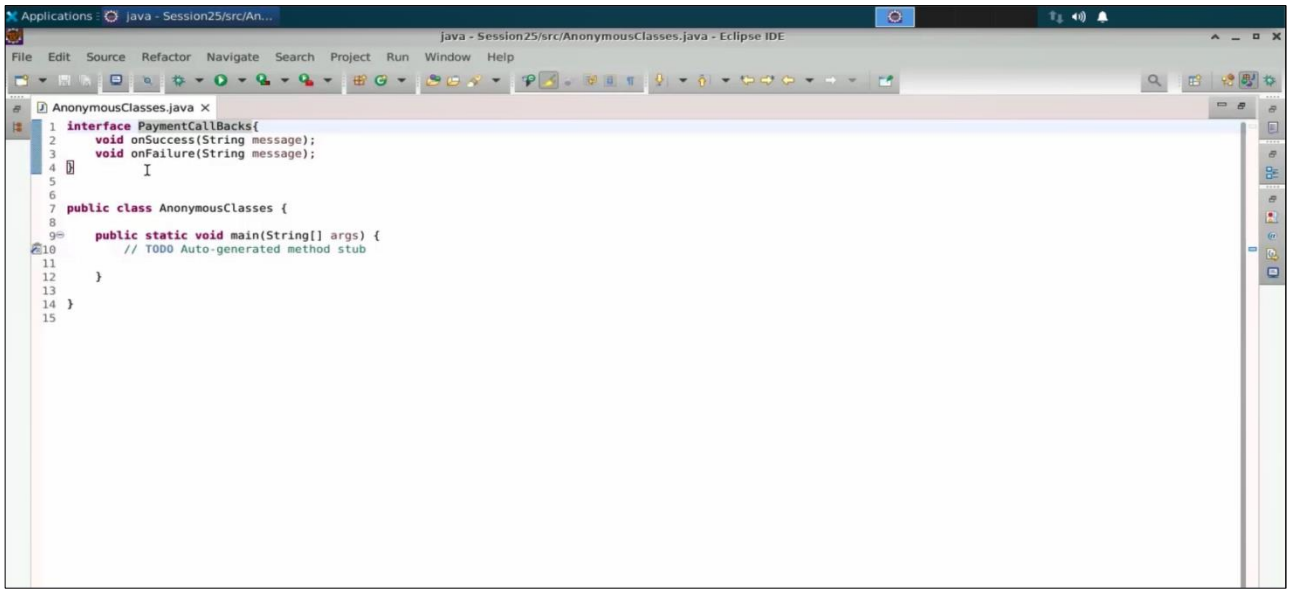


Step 2: Write an interface and execute the code with example data

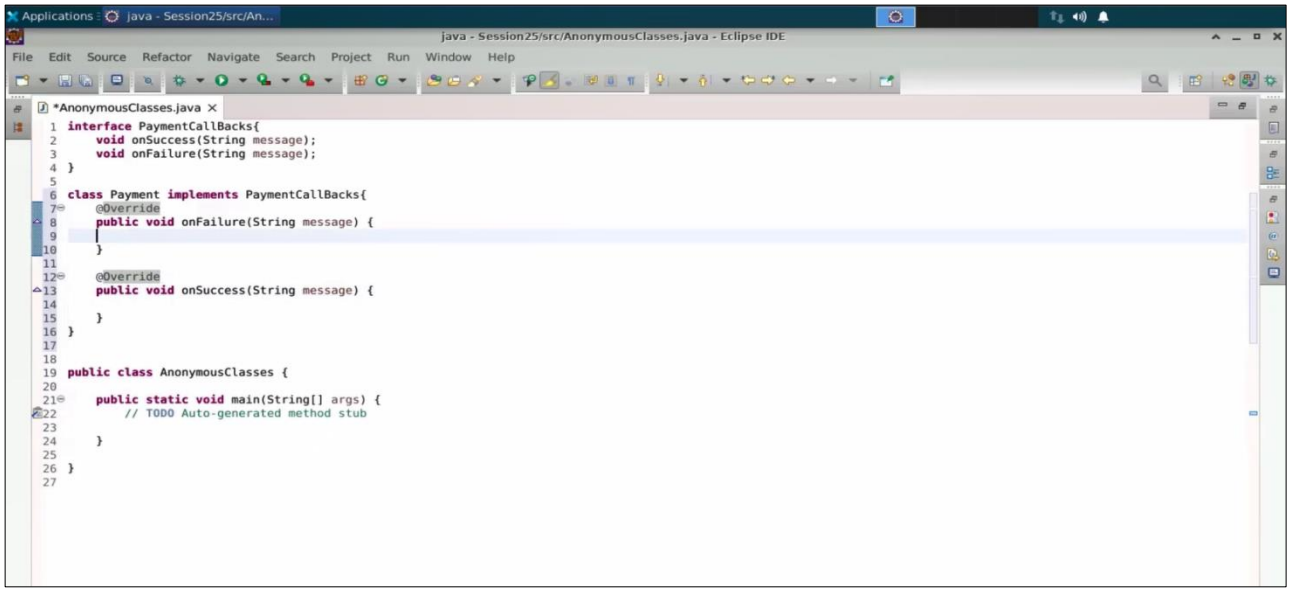
2.1 Let us write one of the interfaces, which goes interface as the payment. Then define a method called pay an amount. And mark this method as default, since you want the definition.



2.2 You can make it simpler, with the method called void on success. And a method called void on failure. Hence, there are two methods for the payment interface. Let us select the input as the message for both and these are like payment callbacks.

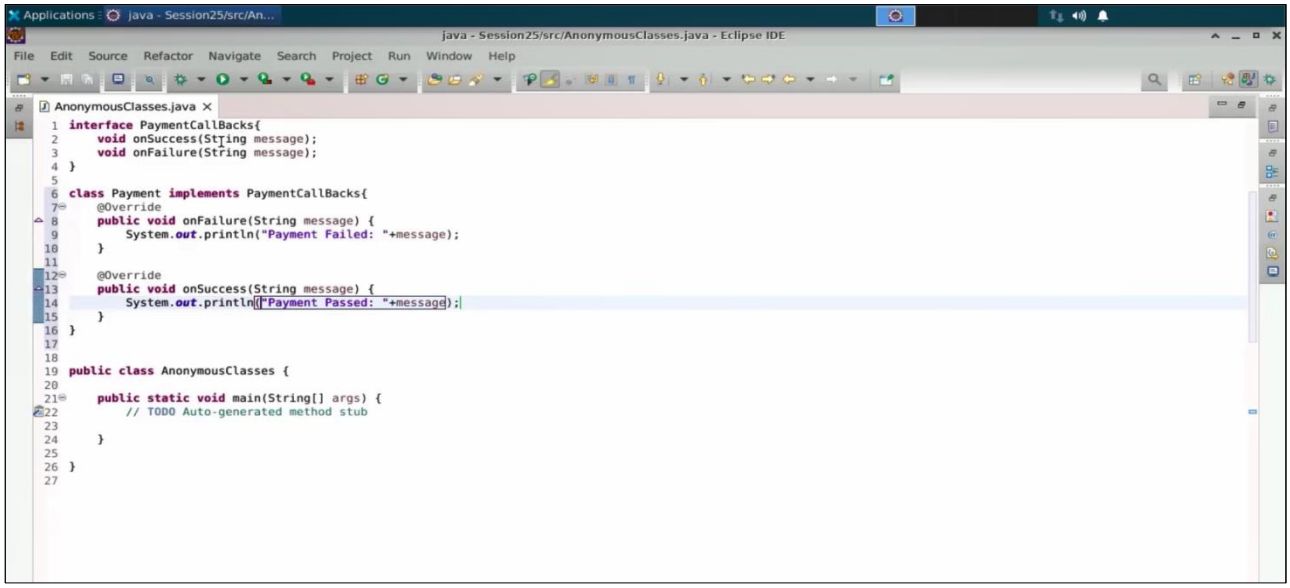


2.3 Now, for the class called `Payment`, you can implement the payment callbacks. When you implement the callbacks, you need to override the `onFailure` and `onSuccess` methods.

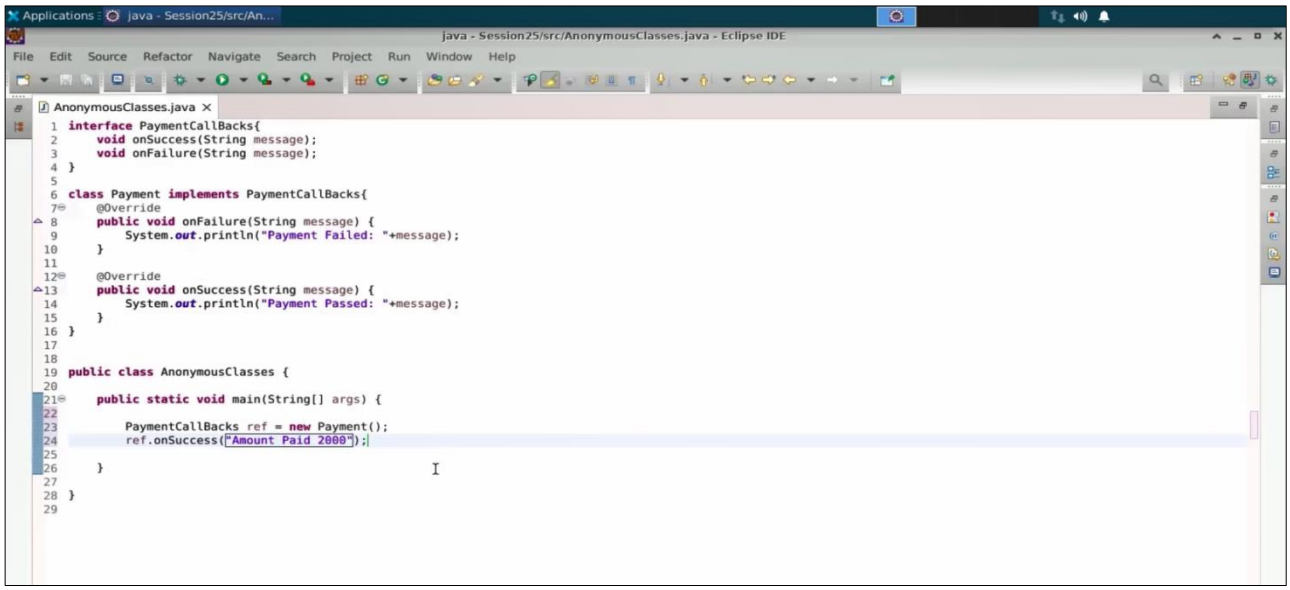


Step 3: Execute the code and print the message accordingly

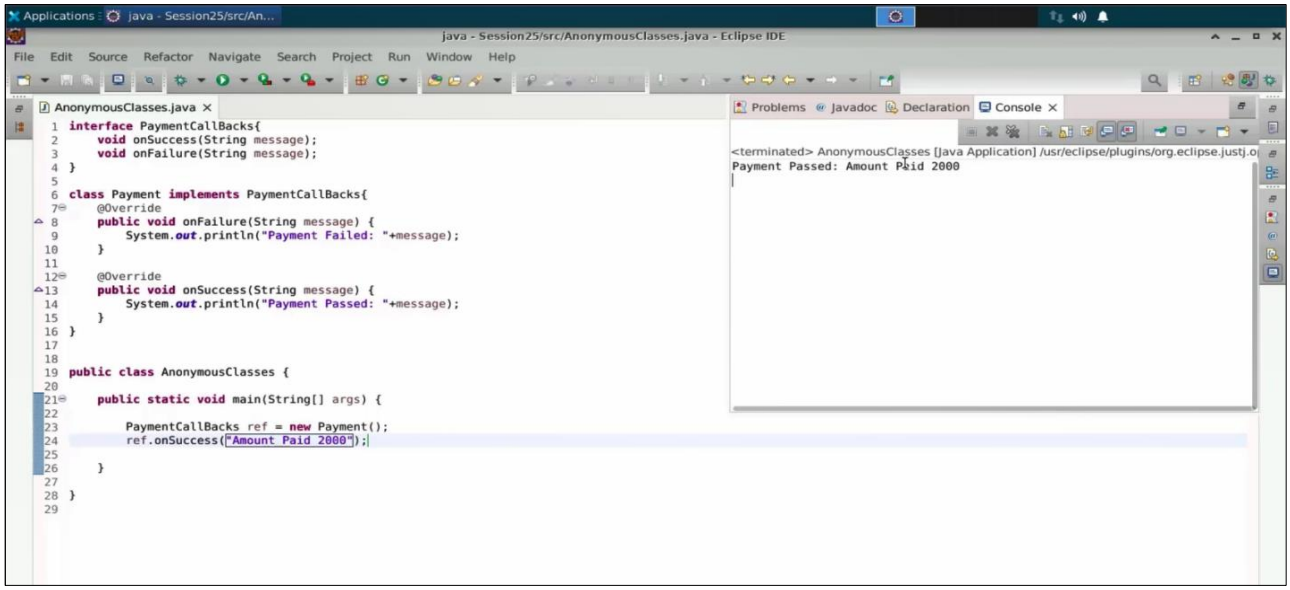
3.1 Let us print down payment failed along with the message. Next, write as payment passed and along with it, also print the message.



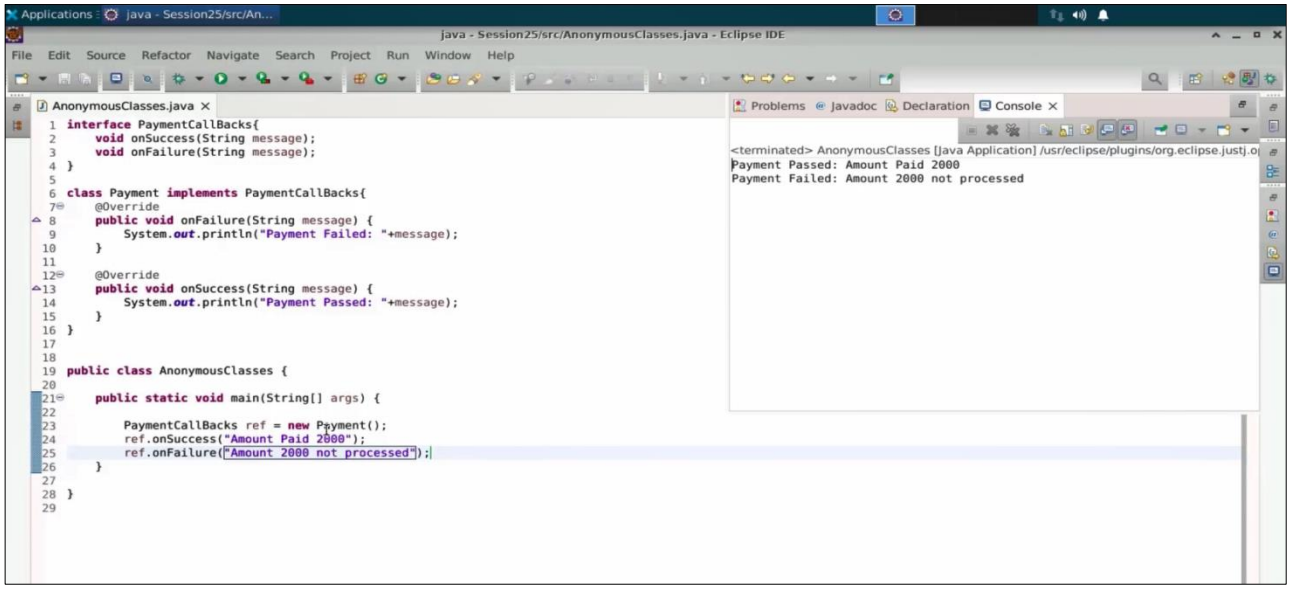
3.2 For the polymorphic statement, the payment callbacks can have a reference variable that can refer to the object of payment. And here you can use the reference variable to execute the method called on success and let us write the amount paid as 2000.



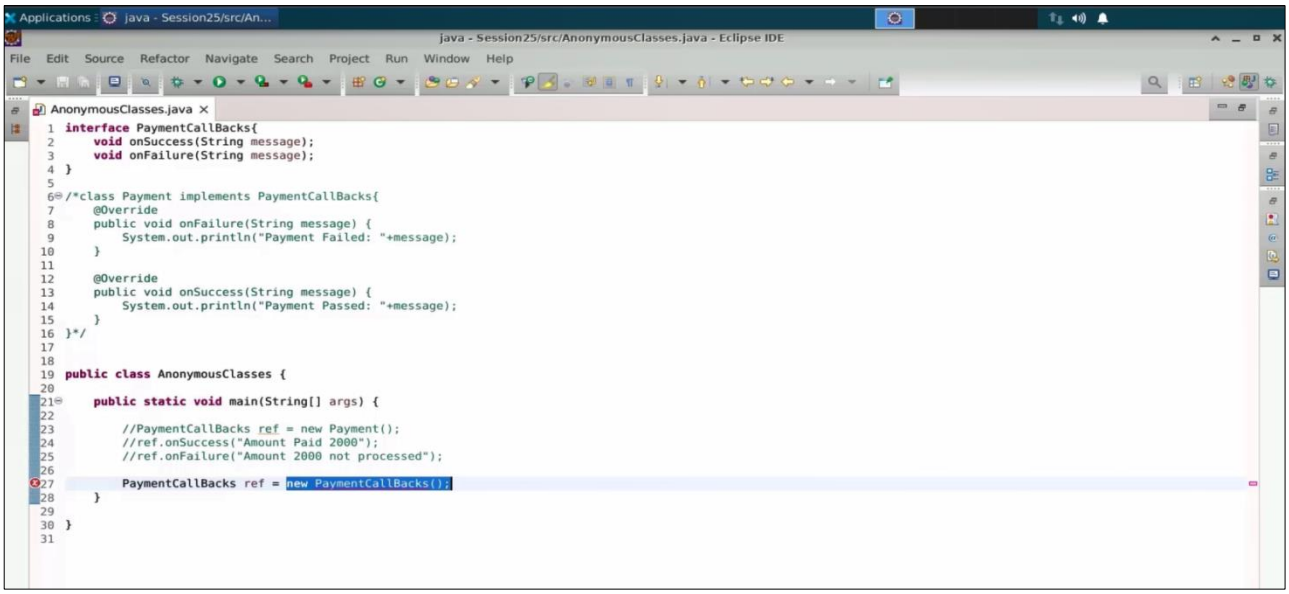
3.3 Hence, when you run the code, it shows payment passed and amount as 2000 like a message.



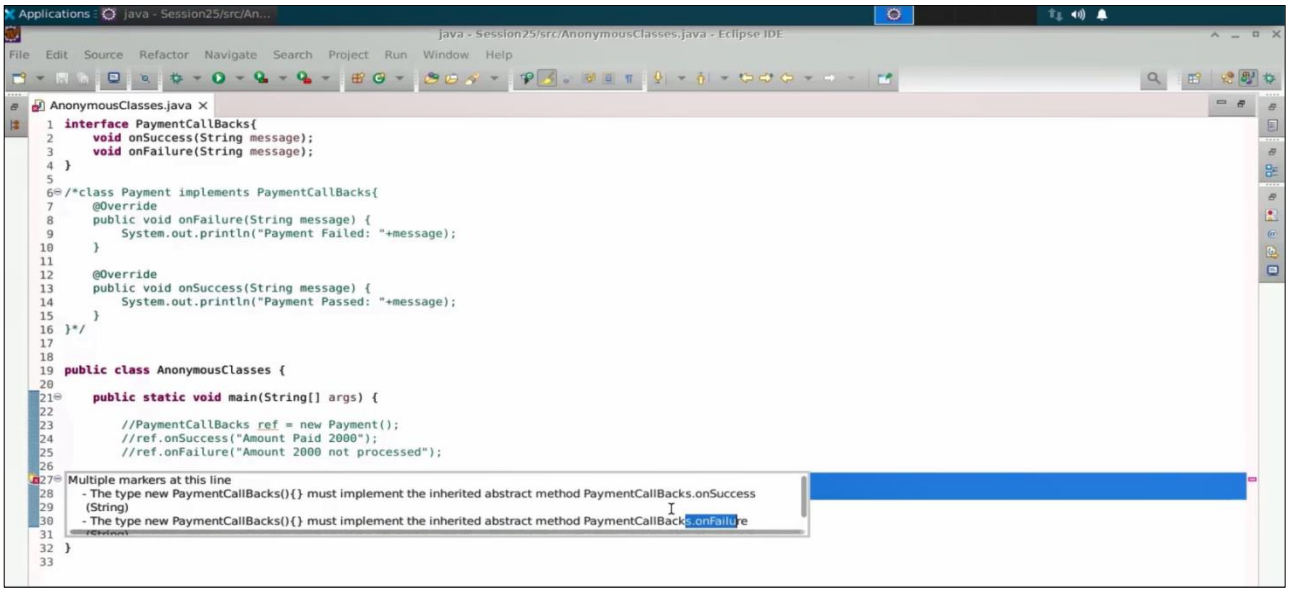
3.4 Similarly, you can write `reference.onFailure` with the message "Amount 2000 not processed." Thus, you have executed both methods. You only need a single object of `Payment`. This is how the use case goes.



3.5 If there is only one single object of payment and you do not need multiple objects of payment. In such kind of situation, you can eliminate, this entire class. Then you can come here and give as the interface payment callbacks create a reference variable as a new payment callback. This is like you are creating an object.

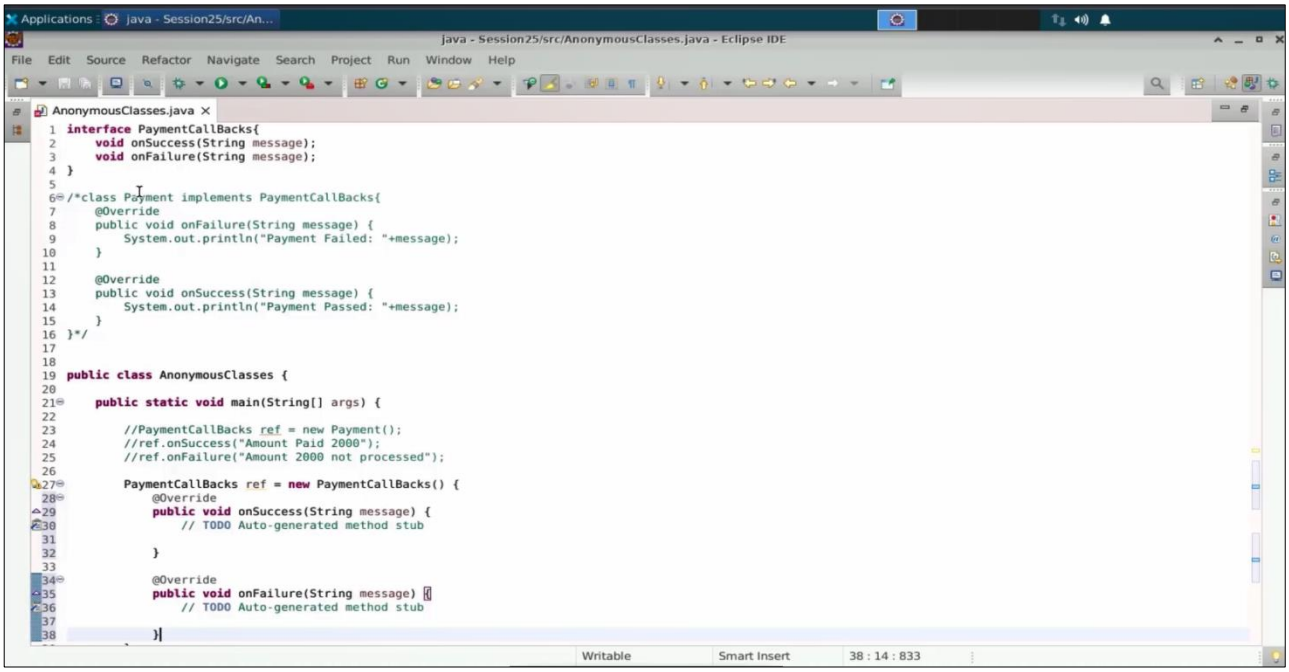


3.6 With this kind of syntax, you can use these two brackets and come down here. This is one of the syntaxes where you are trying to create an anonymous class. If you notice here, you are getting an error message that says that on success and on failure are not implemented.

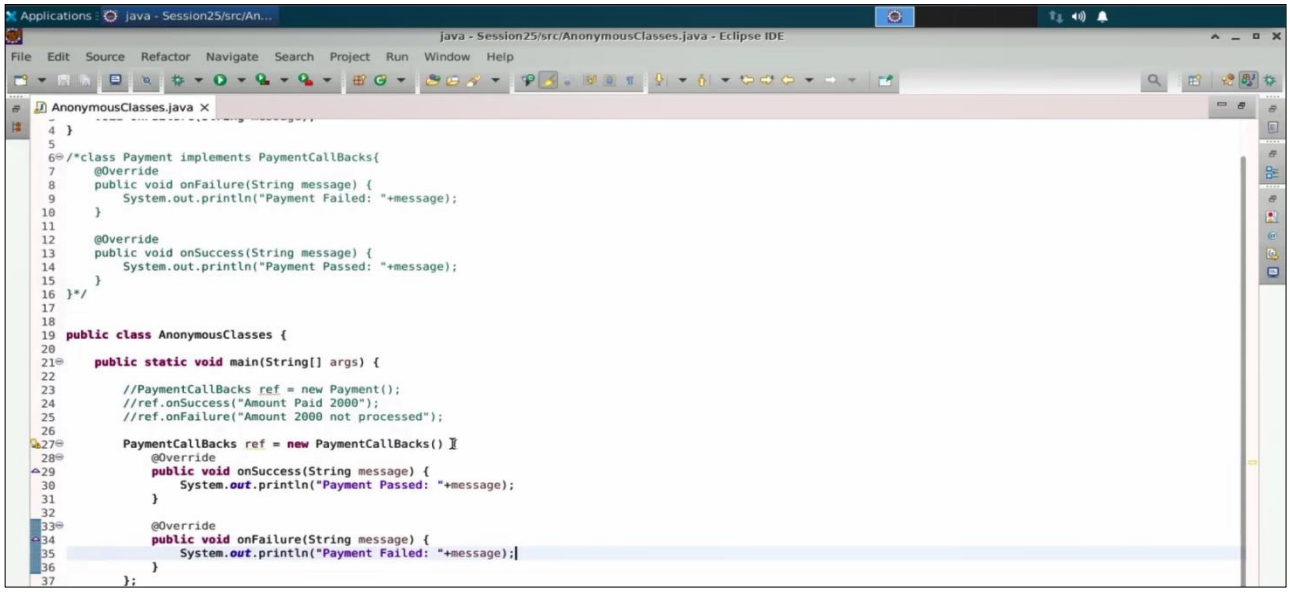


Step 4: Override the methods called on success and failure

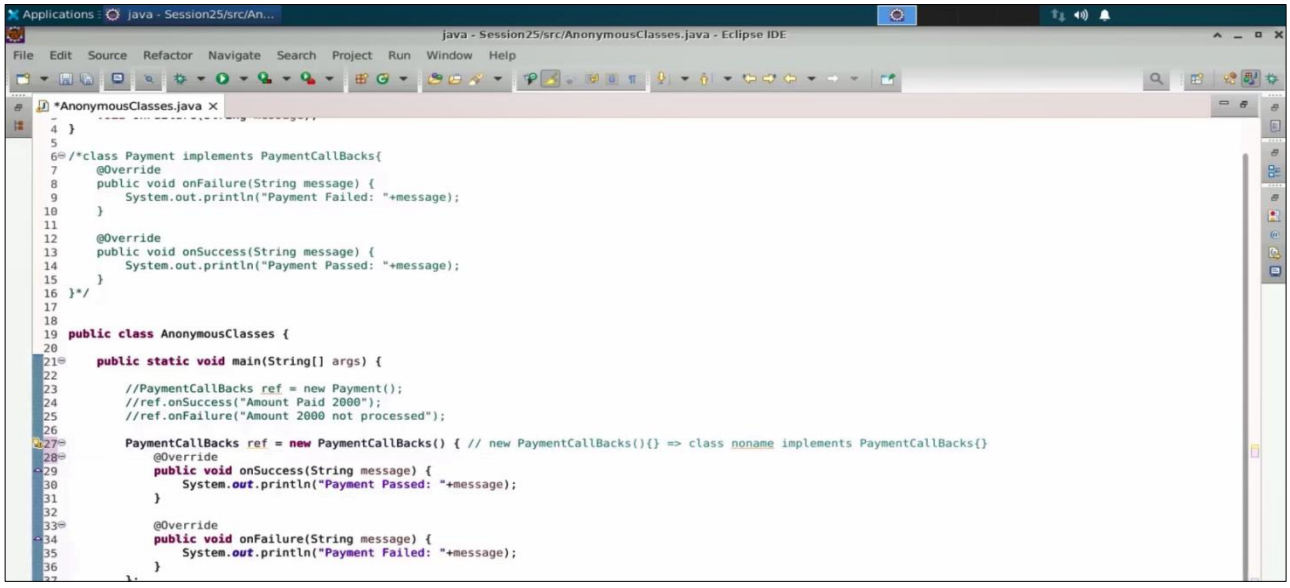
4.1 Let us come here and override the method called on success. And then override another method called on failure. Hence, here you are with no more errors.



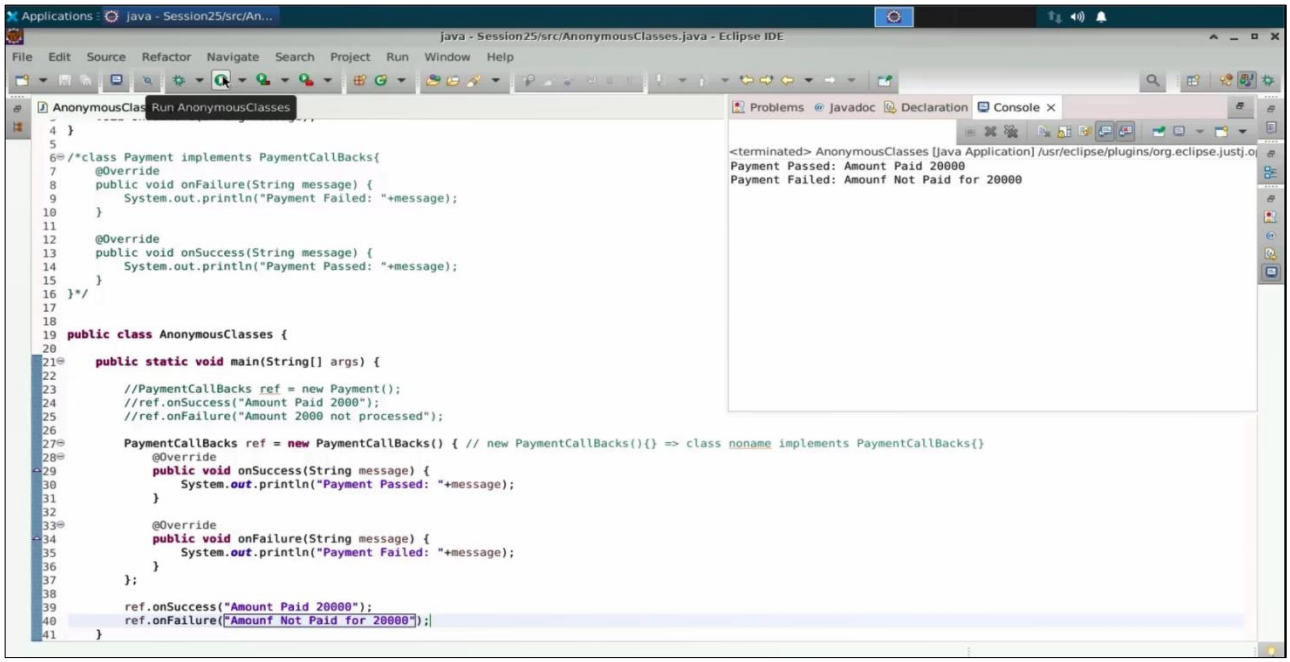
4.2 Come here and write as the payment failed, then the message. Then, this is payment passed and this can be given as payment failed. What you observe is that you have created one of the classes which starts here and finishes here with the semicolon at the end. This is known as an anonymous class. It has no name. And you got the definitions of two methods created.



4.3 This new payment callback is an anonymous class, meaning there is a class with no name implementing your payment callbacks. The new payment callbacks are replaced with this anonymous class implementing the payment callback. Thus, a new class is created at runtime, an object of that class is instantiated, and the reference to that object is stored in this reference variable.

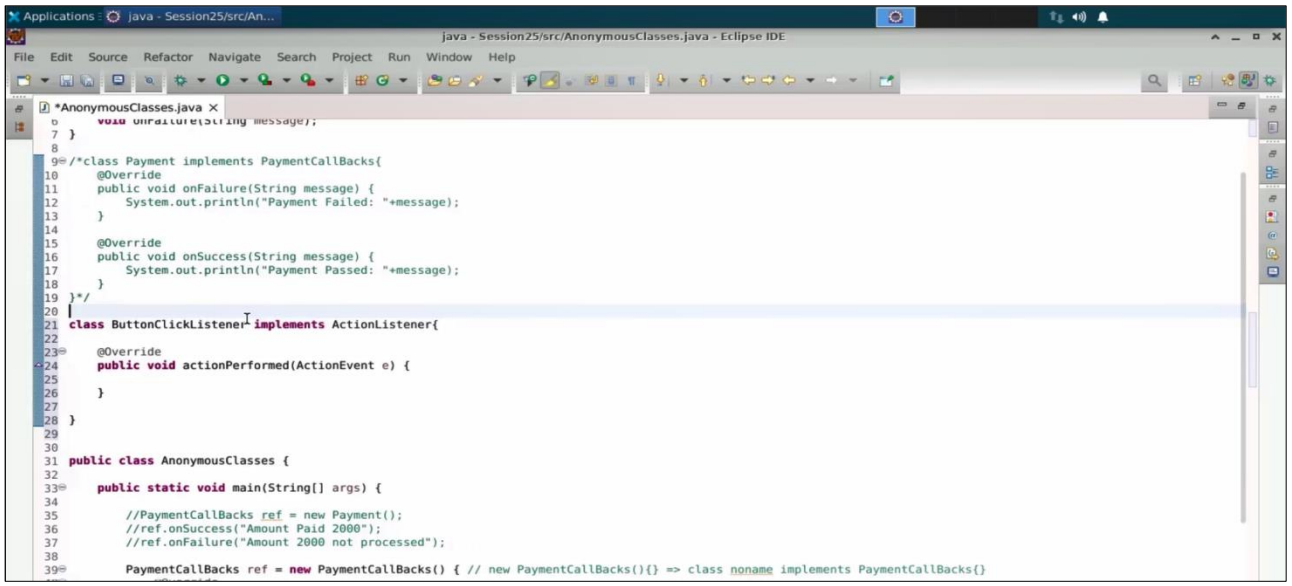


4.4 You can use the same reference variable and execute the success and the failure. Hence, The amount paid is 20000 and then you can give as on failure as well which says amount not paid for 20,000. Run the code here. You get the output coming as payment passes and the payment fails.

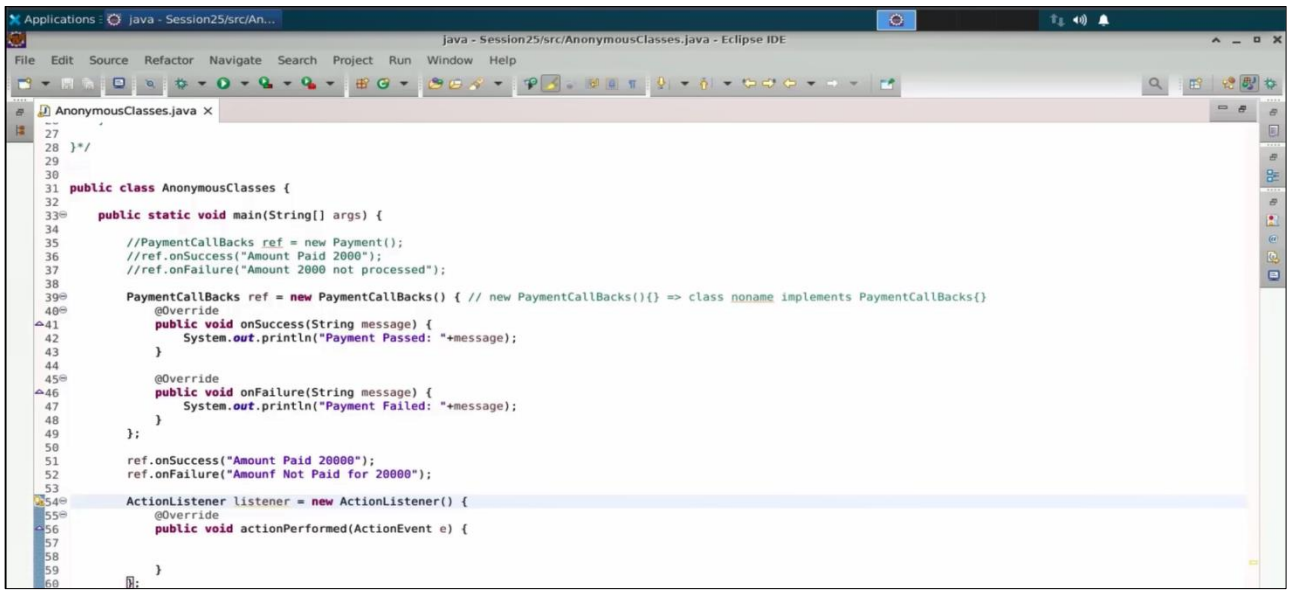


Step 5: Use anonymous classes

5.1 You can use anonymous classes in various scenarios, such as in the graphical user interfaces (GUIs) of Java. For instance, there's an interface known as `ActionListener` that allows you to control buttons and detect when they are clicked. The `ActionListener` interface has a single method called `actionPerformed`. If you create a class called `ButtonClickListener`, it will implement `ActionListener`. By adding the unimplemented method, `actionPerformed`, you can handle the button click events.



5.2 You must come up and create a separate class that contains only one single method implementation and then use the object of this class. Rather than writing a class, you can always come back and write as there is a listener, which is going to be an object of an anonymous class with the action performed.



By following the above steps, you have successfully implemented anonymous classes, allowing you to create concise and efficient instances of classes.