Exercise: Appium Mobile

1. "Summator" Page Object Model (POM)

1.1. Prerequisites

"Summator" Android App

You are given the following sample **Android mobile app** "Summator":
https://github.com/nakov/AndroidApp-Summator.

You can get its .apk file from here:
https://github.com/nakov/AndroidApp-Summator/releases/tag/v1.0.

You also have the .apk file added to the lecture's resources.

The Automated Testing Scenario

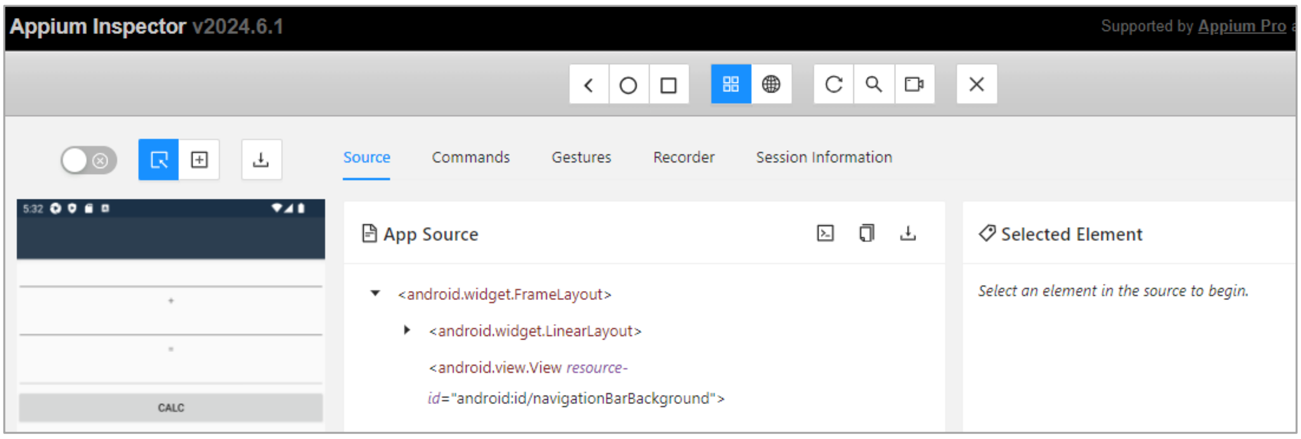
1. Open the **`Summator` app.**
2. Test with **valid** and **invalid data.**
    - With valid data: assert that result is **correct.**
    - With invalid data: assert that "**`error`**" is displayed in the result field

Configure Appium Inspector

- **Start** Appium Server (if you're using the Web version of Appium Inspector use `--allow-cors`)

```
appium        OR     appium --allow-cors
```

- Start your AVD in a separate CMD window
- Make sure that the Application for testing is installed on your AVD (drag & drop to install)
- Provide the **host and port of Appium server** in Appium Inspector (**`http://127.0.0.1:4723`**)
- Add the Desired Capabilities of the mobile device or emulator connected to the system
    - **`automationName`** – in our case **`UiAutomator2`**
    - **`platformName`** - get it using command **`appium driver list`**
    - **`platformVersion`** - get it using command **`adb shell getprop ro.build.version.release`**
    - **`appium:deviceName`** – get it using command **`adb devices`**
    - **`appium:app`** – path to the.apk file for testing on your computer

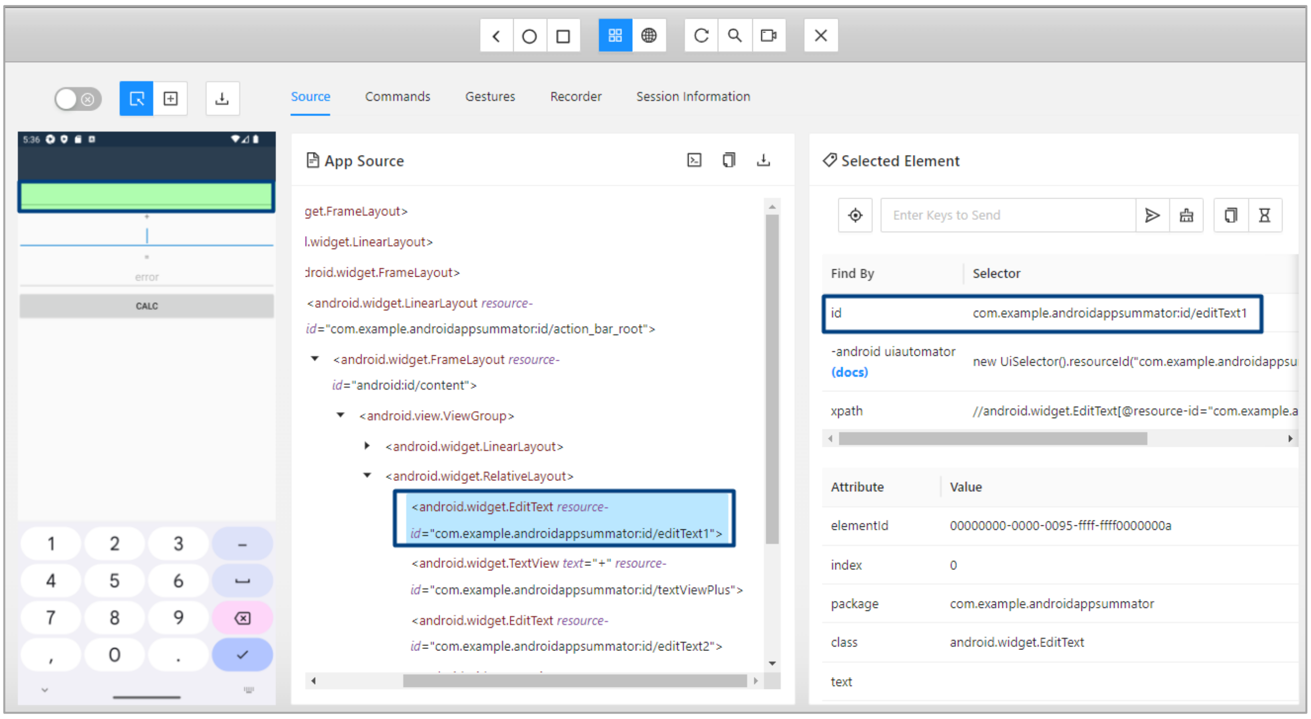


```
<android.widget.FrameLayout>
  <android.widget.LinearLayout>
  <android.view.View resource-id="android:id/navigationBarBackground">
```

Select an element in the source to begin.

Get the IDs of the elements

You can get the id of any element you need. Click on the element e.g., on the first number field, and its id will be shown under **"Selected Element":**



```
get.FrameLayout>
l.widget.LinearLayout>
droid.widget.FrameLayout>
<android.widget.LinearLayout resource-
id="com.example.androidappsummator:id/action_bar_root">
  <android.widget.FrameLayout resource-
  id="android:id/content">
    <android.view.ViewGroup>
      <android.widget.LinearLayout>
      <android.widget.RelativeLayout>
        <android.widget.EditText resource-
        id="com.example.androidappsummator:id/editText1">
        <android.widget.TextView text="+" resource-
        id="com.example.androidappsummator:id/textViewPlus">
        <android.widget.EditText resource-
        id="com.example.androidappsummator:id/editText2">
```

Find By	Selector
id	com.example.androidappsummator:id/editText1
-android uiautomator **(docs)**	new UiSelector().resourceId("com.example.androidappsu
xpath	//android.widget.EditText[@resource-id="com.example.a

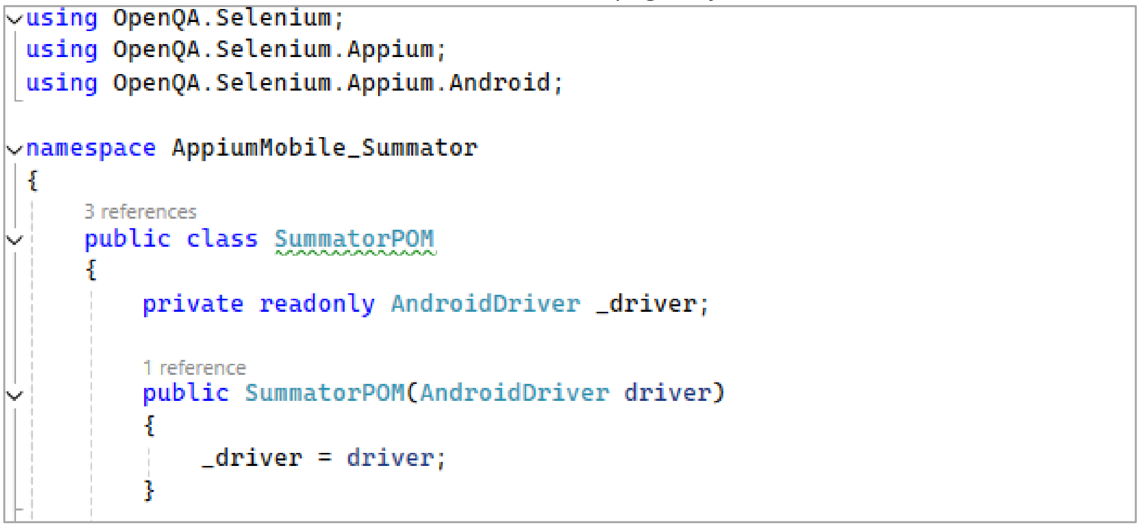
Attribute	Value
elementId	00000000-0000-0095-ffff-ffff0000000a
index	0
package	com.example.androidappsummator
class	android.widget.EditText
text	

You should do the same to get **ids** of all the elements you will need.

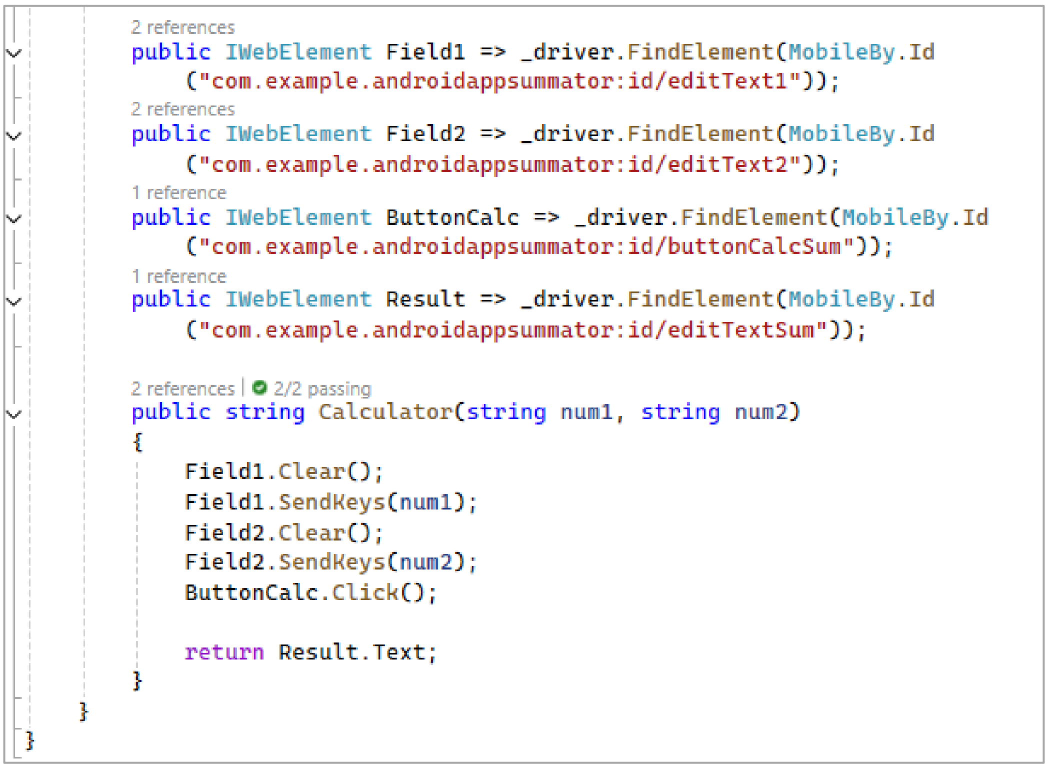
1.2. Refactoring "Summator" Tests from Non-POM to POM

- **Identify Elements and Actions**
  Identify the elements (field1, field2, buttonCalc, result) and actions (clearing fields, sending keys, clicking button) in the test.

- **Create the Page Object Class**
  Move the identified elements and actions to a new page object class

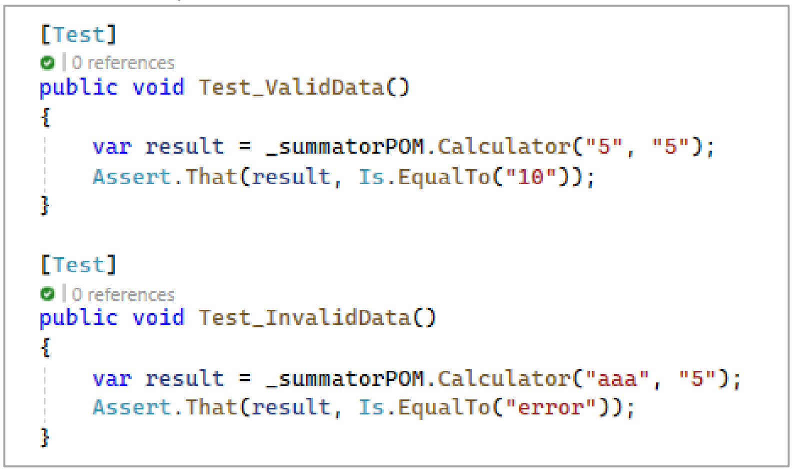


```
using OpenQA.Selenium;
using OpenQA.Selenium.Appium;
using OpenQA.Selenium.Appium.Android;

namespace AppiumMobile_Summator
{
    3 references
    public class SummatorPOM
    {
        private readonly AndroidDriver _driver;

        1 reference
        public SummatorPOM(AndroidDriver driver)
        {
            _driver = driver;
        }
```



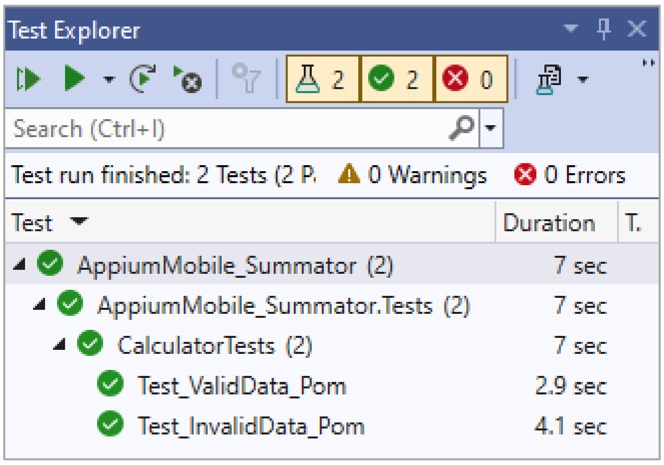
```
        2 references
        public IWebElement Field1 => _driver.FindElement(MobileBy.Id
            ("com.example.androidappsummator:id/editText1"));
        2 references
        public IWebElement Field2 => _driver.FindElement(MobileBy.Id
            ("com.example.androidappsummator:id/editText2"));
        1 reference
        public IWebElement ButtonCalc => _driver.FindElement(MobileBy.Id
            ("com.example.androidappsummator:id/buttonCalcSum"));
        1 reference
        public IWebElement Result => _driver.FindElement(MobileBy.Id
            ("com.example.androidappsummator:id/editTextSum"));

        2 references | 2/2 passing
        public string Calculator(string num1, string num2)
        {
            Field1.Clear();
            Field1.SendKeys(num1);
            Field2.Clear();
            Field2.SendKeys(num2);
            ButtonCalc.Click();

            return Result.Text;
        }
    }
}
```

- **Update Test Class -** Refactor the test class to use the page object methods
  - In the test class, instantiate the **SummatorPOM** object in the Setup method.
  - Replace the direct interactions with elements in the test methods with calls to the page object methods.



```
[Test]
| 0 references
public void Test_ValidData()
{
    var result = _summatorPOM.Calculator("5", "5");
    Assert.That(result, Is.EqualTo("10"));
}

[Test]
| 0 references
public void Test_InvalidData()
{
    var result = _summatorPOM.Calculator("aaa", "5");
    Assert.That(result, Is.EqualTo("error"));
}
```

1.3. Run the Tests



Test run finished: 2 Tests (2 P. ⚠ 0 Warnings ✖ 0 Errors

Test	Duration	T.
AppiumMobile_Summator (2)	7 sec	
AppiumMobile_Summator.Tests (2)	7 sec	
CalculatorTests (2)	7 sec	
Test_ValidData_Pom	2.9 sec	
Test_InvalidData_Pom	4.1 sec	

2. ColorNote App

We are working with a simple notepad Android application, called "ColorNote". The app allows users to create, edit, and delete notes. The app has a basic user interface and we will create automated tests to cover some testing scenarios.

2.1. Prerequisites

"ColorNote" Android App

The **.apk file** has been added to the exercise's resources

The Automated Testing Scenarios

- **Creating a New Note:**
  - Open the app.
  - Skip the tutorial if it appears.
  - Add a new text note with a specific title and content.
  - Verify that the note is created successfully.
- **Editing a Note:**
  - Open the app.
  - Skip the tutorial if it appears.
  - Add a new text note with a specific title and content.
  - Edit the newly created note.
  - Verify that the note is edited successfully.
- **Deleting a Note:**
  - Open the app.
  - Skip the tutorial if it appears.
  - Add a new text note with a specific title and content.
  - Delete the newly created note.
  - Verify that the note is deleted successfully.

As a part of the resources for this exercise you have a short **MP4 video** that shows what your tests should do.

Configure Appium Inspector

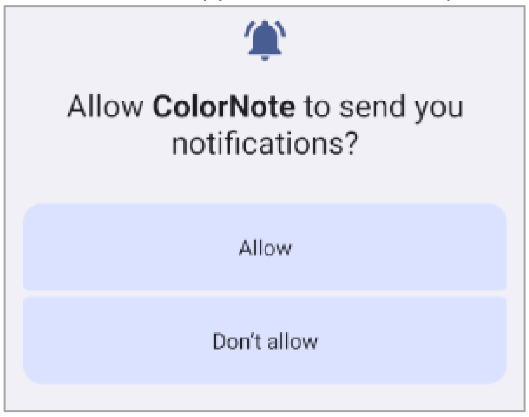
- **Start** Appium Server (if you're using the Web version of Appium Inspector use `--allow-cors`)

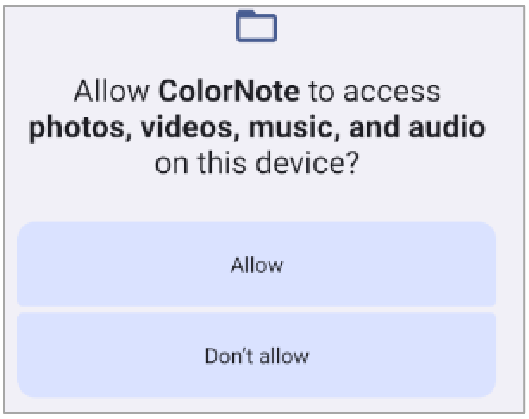
```
appium        OR     appium --allow-cors
```

- Start your AVD in a separate CMD window
- Make sure that the Application for testing is installed on your AVD (drag & drop to install)
- Provide the **host and port of Appium server** in Appium Inspector (**`http://127.0.0.1:4723`**)
- Add the Desired Capabilities of the mobile device or emulator connected to the system
  - **`automationName`** – in our case **`UiAutomator2`**
  - **`platformName`** - get it using command **`appium driver list`**
  - **`platformVersion`** - get it using command **`adb shell getprop ro.build.version.release`**
  - **`appium:deviceName`** – get it using command **`adb devices`**

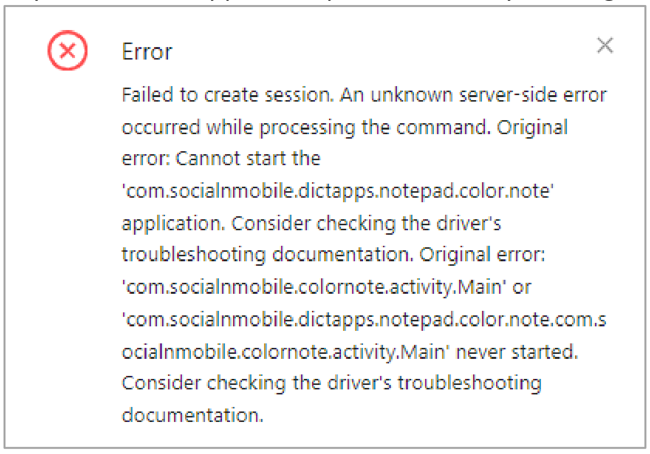
  - **`appium:app`** – path to the.apk file for testing on your computer

NOTE! The ColorNote app will ask for some permissions.




Allow **ColorNote** to access **photos, videos, music, and audio** on this device?
Allow
Don't allow

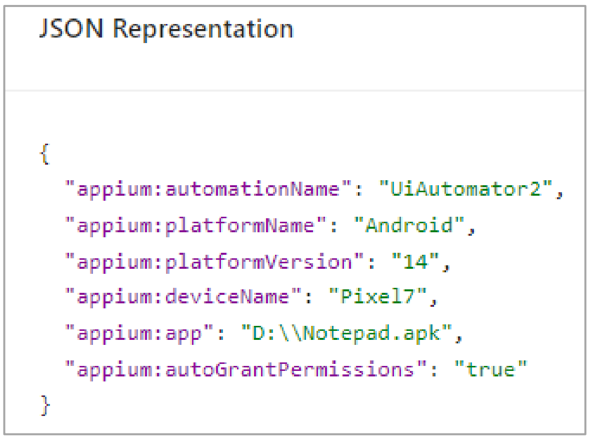
Because of them, when you try to start the Appium Inspector session you will get the following error:



Error

Failed to create session. An unknown server-side error occurred while processing the command. Original error: Cannot start the 'com.socialnmobile.dictapps.notepad.color.note' application. Consider checking the driver's troubleshooting documentation. Original error: 'com.socialnmobile.colornote.activity.Main' or 'com.socialnmobile.dictapps.notepad.color.note.com.socialnmobile.colornote.activity.Main' never started. Consider checking the driver's troubleshooting documentation.

- Add the following additional capability to the Desired Capabilities:
  - **`"appium:autoGrantPermissions": "true"`**

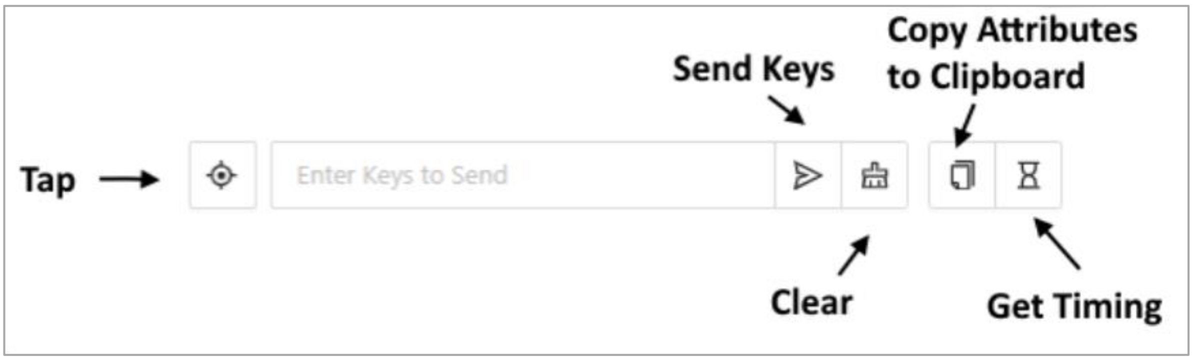


JSON Representation

```
{
  "appium:automationName": "UiAutomator2",
  "appium:platformName": "Android",
  "appium:platformVersion": "14",
  "appium:deviceName": "Pixel7",
  "appium:app": "D:\\Notepad.apk",
  "appium:autoGrantPermissions": "true"
}
```

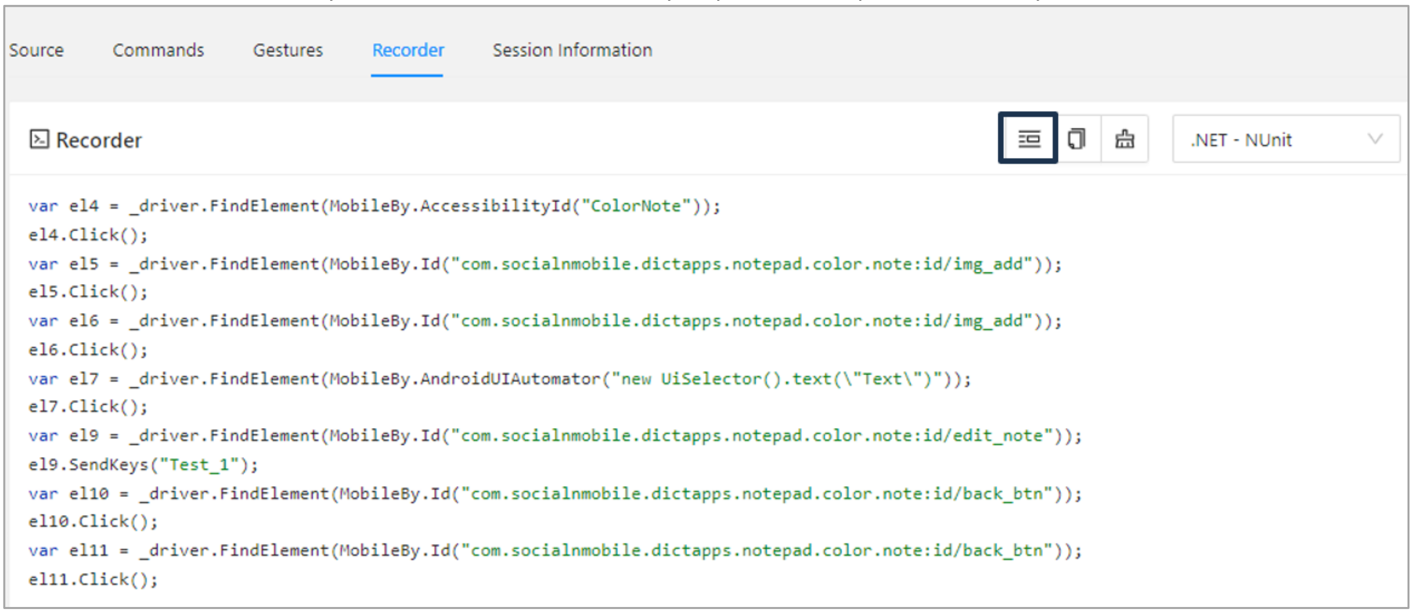
2.2. Recording via Appium Inspector

Click the Record Button on the Main Menu of Appium Inspector and follow the steps that has to be performed for the first test scenario "Creating a New Note", use the options from the "Select Element" panel to interact with the elements.



- **Open** ColorNote App
- **Tap** Skip for the offered "Step-by-Step Tutorial"
- Tap "Add Note" or "+" button to add a new Note
- Tap "Text" to create a Text Note
- Tap on the Yellow field with rows to start typing the note
- Type the Text of the note (in our case "Test_1")
- Tap on the "Tick" symbol to confirm the text of the note.
- Tap on the back arrow symbol to return to the main page where all notes are displayed.

Pause the recording in Appium Inspector. Navigate to "Recorder" and voila you have all the elements that you interacted with located for you, and all the actions that you performed, performed for you.



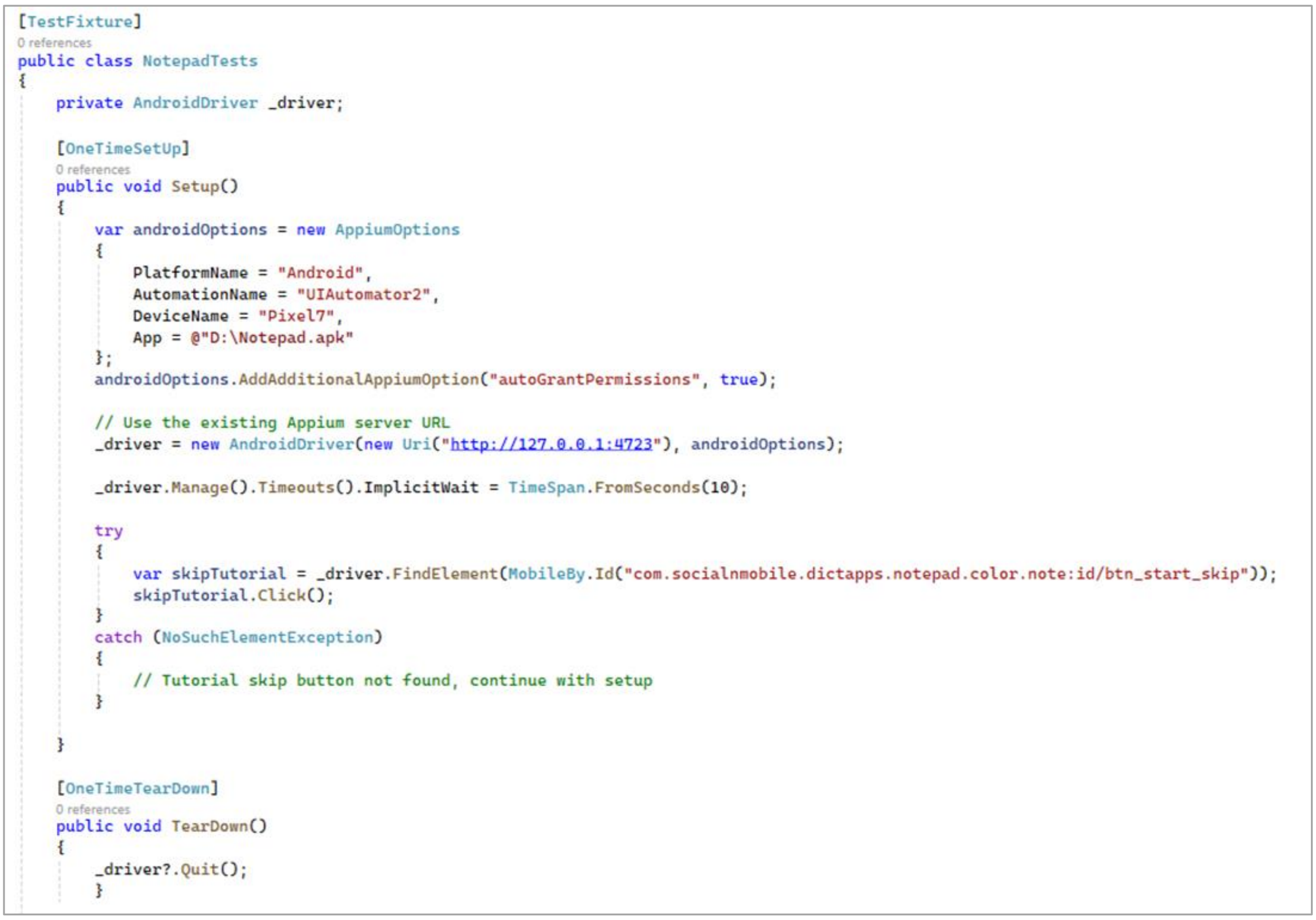
```
var el4 = _driver.FindElement(MobileBy.AccessibilityId("ColorNote"));
el4.Click();
var el5 = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/img_add"));
el5.Click();
var el6 = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/img_add"));
el6.Click();
var el7 = _driver.FindElement(MobileBy.AndroidUIAutomator("new UiSelector().text(\"Text\")"));
el7.Click();
var el9 = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/edit_note"));
el9.SendKeys("Test_1");
var el10 = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/back_btn"));
el10.Click();
var el11 = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/back_btn"));
el11.Click();
```

If you click "Show/Hide Boilerplate Code" you can see some code written for you. It is not perfect and you should never write code like that, but it is a great reference that you can use writing your own code. And no, absolutely not, you can't leave those elements called el1, el2, el3 etc.!

2.3. Writing Tests for "ColorNote" App

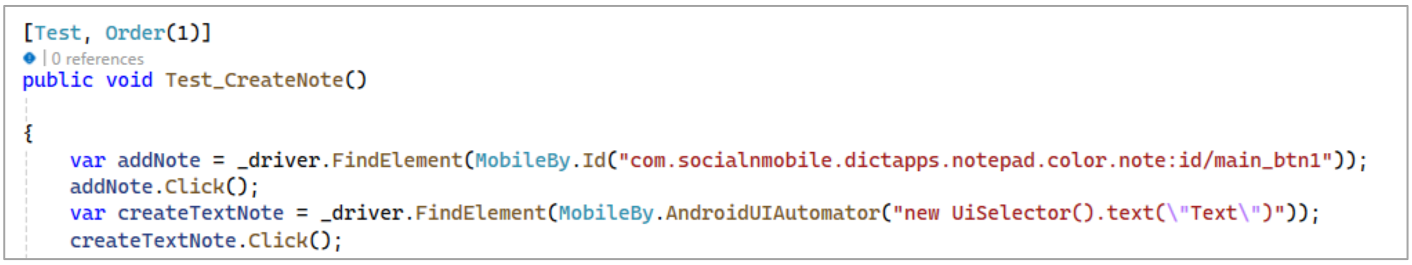
- Open Visual Studio and create a new NUnit Test Project.
- Install **`Appium.WebDriver`** from NuGet Package Manager.
- Import the Necessary Namespaces: **`OpenQA.Selenium.Appium`**; **`OpenQA.Selenium.Appium.Android`**; **`OpenQA.Selenium`**; etc.
- Define the Test Class:
  - Mark your class with [TestFixture] to indicate that it contains tests.
  - Define private fields for the Android driver and optional Appium server.
  - Create a method to set up the test environment. Mark it with [OneTimeSetUp].
    - Configure the desired capabilities for the Appium session using AppiumOptions.
    - Initialize the AndroidDriver with the Appium server URL.

- Set an implicit wait time.
- Handle any optional tutorial steps using a try-catch block.
- Create a method to clean up the test environment after tests are executed. Mark it with [OneTimeTearDown].

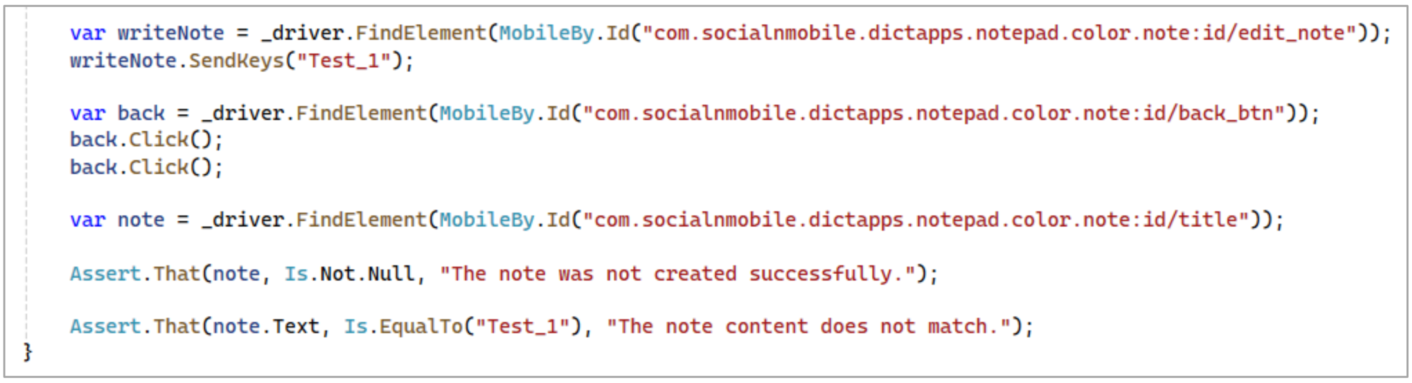


```
[TestFixture]
0 references
public class NotepadTests
{
    private AndroidDriver _driver;

    [OneTimeSetUp]
    0 references
    public void Setup()
    {
        var androidOptions = new AppiumOptions
        {
            PlatformName = "Android",
            AutomationName = "UIAutomator2",
            DeviceName = "Pixel7",
            App = @"D:\Notepad.apk"
        };
        androidOptions.AddAdditionalAppiumOption("autoGrantPermissions", true);

        // Use the existing Appium server URL
        _driver = new AndroidDriver(new Uri("http://127.0.0.1:4723"), androidOptions);

        _driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);

        try
        {
            var skipTutorial = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/btn_start_skip"));
            skipTutorial.Click();
        }
        catch (NoSuchElementException)
        {
            // Tutorial skip button not found, continue with setup
        }

    }

    [OneTimeTearDown]
    0 references
    public void TearDown()
    {
        _driver?.Quit();
        }
```

- **Creating Tests**
  - **Test to Create a Note:**
    - Locate the "Add Note" button and click it.
    - Select "Text" to create a text note.
    - Write the note content "Test_1".
    - Click the "Back" button twice to save and return to the main screen.
    - Verify that the note titled "Test_1" is displayed.

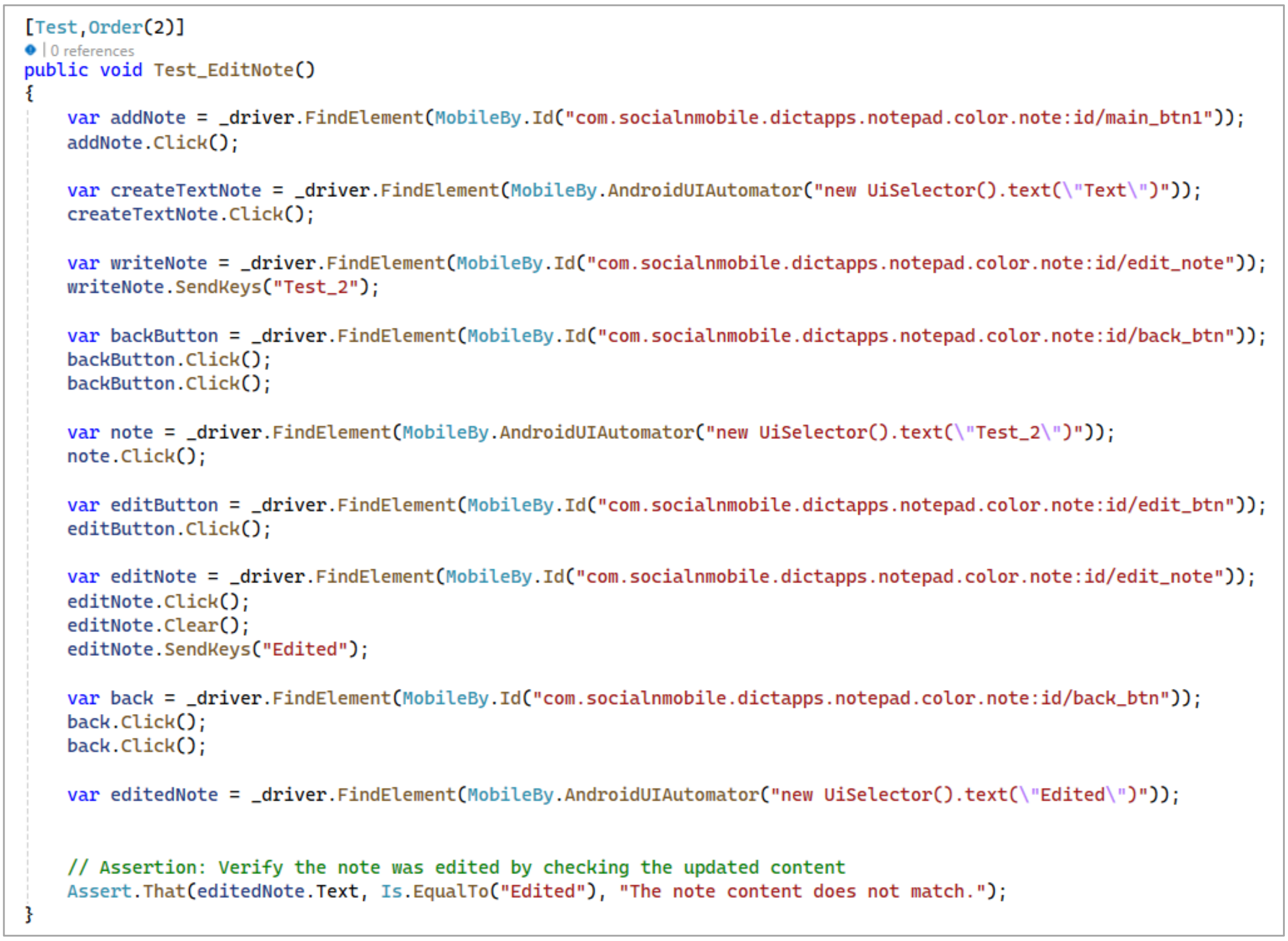


```
[Test, Order(1)]
● | 0 references
public void Test_CreateNote()

{
    var addNote = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/main_btn1"));
    addNote.Click();
    var createTextNote = _driver.FindElement(MobileBy.AndroidUIAutomator("new UiSelector().text(\"Text\")"));
    createTextNote.Click();
```



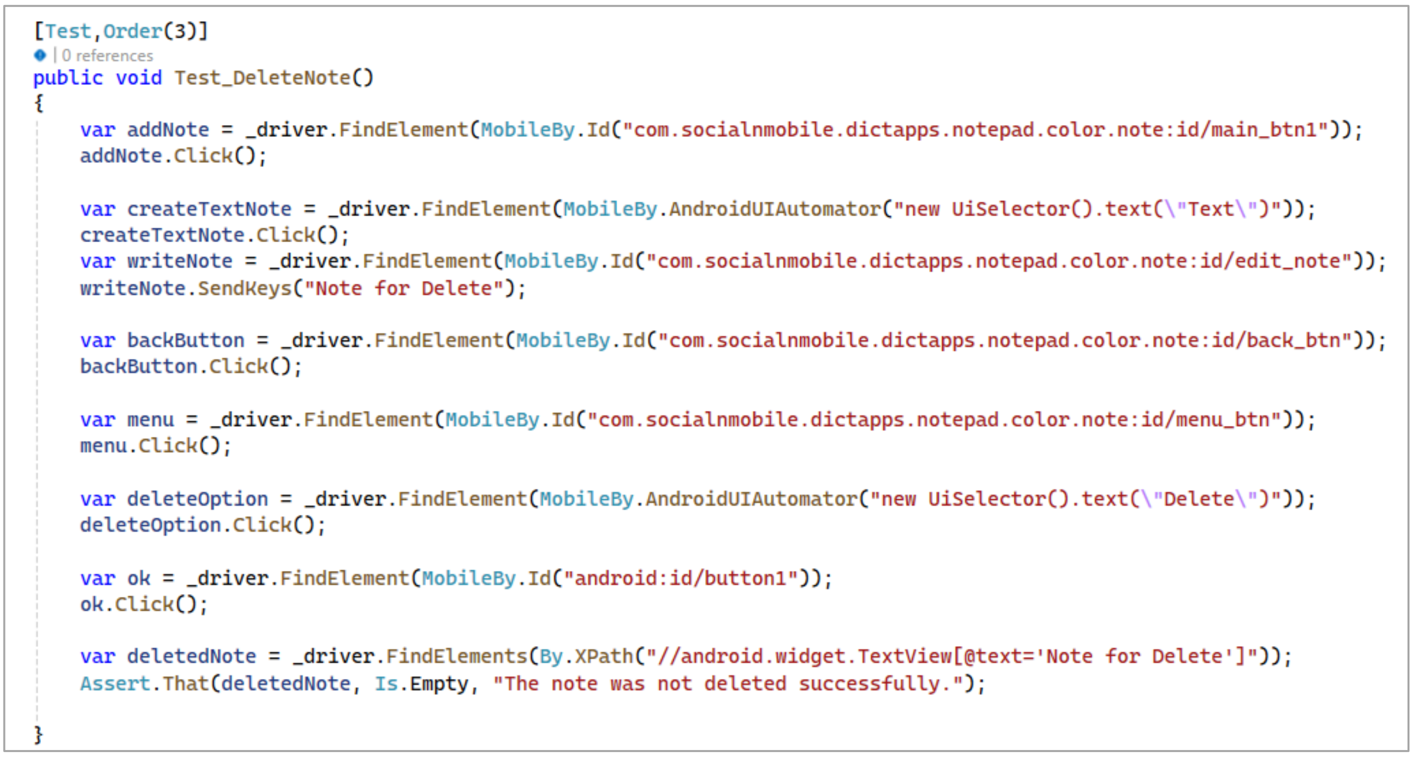
```
    var writeNote = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/edit_note"));
    writeNote.SendKeys("Test_1");

    var back = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/back_btn"));
    back.Click();
    back.Click();

    var note = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/title"));

    Assert.That(note, Is.Not.Null, "The note was not created successfully.");

    Assert.That(note.Text, Is.EqualTo("Test_1"), "The note content does not match.");
}
```

- **Test to Edit a Note:**
  - Repeat steps to create a new note titled "Test_2".
  - Find the note "Test_2" and click it.
  - Edit the note content to "Edited".
  - Save and return to the main screen.
  - Verify that the note content is updated to "Edited".



```
[Test,Order(2)]
0 references
public void Test_EditNote()
{
    var addNote = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/main_btn1"));
    addNote.Click();

    var createTextNote = _driver.FindElement(MobileBy.AndroidUIAutomator("new UiSelector().text(\"Text\")"));
    createTextNote.Click();

    var writeNote = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/edit_note"));
    writeNote.SendKeys("Test_2");

    var backButton = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/back_btn"));
    backButton.Click();
    backButton.Click();

    var note = _driver.FindElement(MobileBy.AndroidUIAutomator("new UiSelector().text(\"Test_2\")"));
    note.Click();

    var editButton = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/edit_btn"));
    editButton.Click();

    var editNote = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/edit_note"));
    editNote.Click();
    editNote.Clear();
    editNote.SendKeys("Edited");

    var back = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/back_btn"));
    back.Click();
    back.Click();

    var editedNote = _driver.FindElement(MobileBy.AndroidUIAutomator("new UiSelector().text(\"Edited\")"));


    // Assertion: Verify the note was edited by checking the updated content
    Assert.That(editedNote.Text, Is.EqualTo("Edited"), "The note content does not match.");
}
```

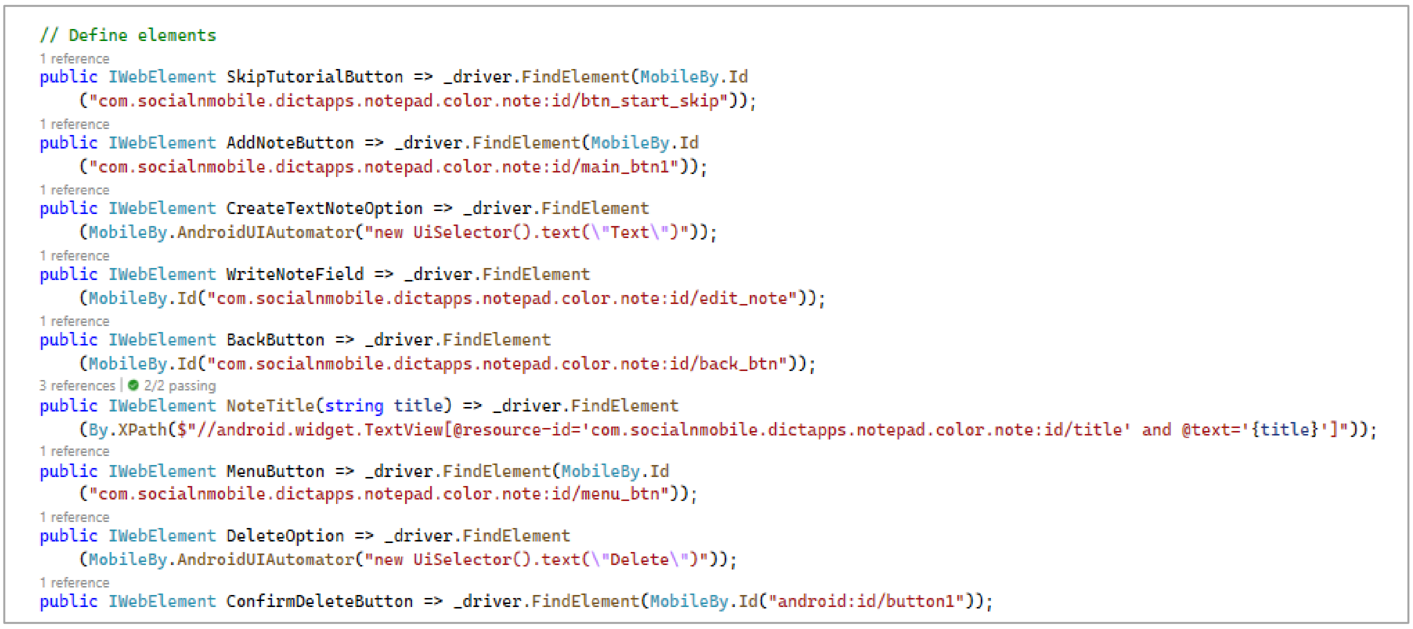
- **Test to Delete a Note:**
  - Repeat steps to create a new note titled "Note for Delete".
  - Open the menu and select "Delete".
  - Confirm the deletion.
  - Verify that the note is deleted by checking its absence.



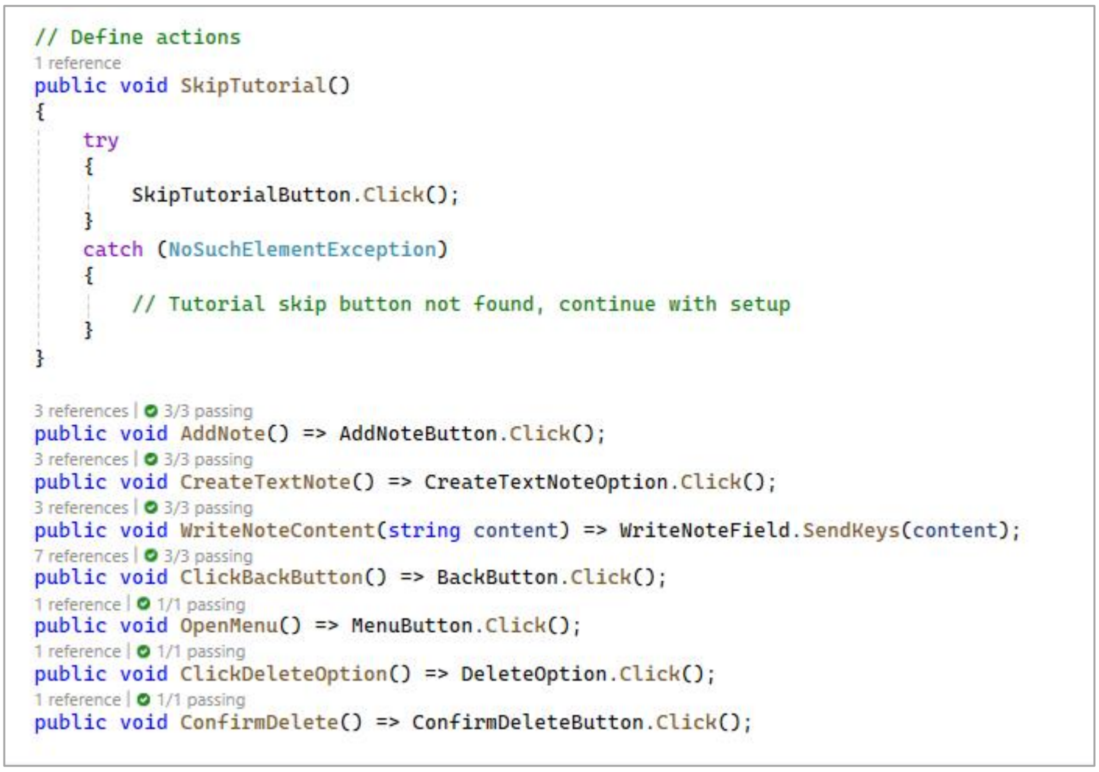
```
[Test,Order(3)]
● | 0 references
public void Test_DeleteNote()
{
    var addNote = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/main_btn1"));
    addNote.Click();

    var createTextNote = _driver.FindElement(MobileBy.AndroidUIAutomator("new UiSelector().text(\"Text\")"));
    createTextNote.Click();
    var writeNote = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/edit_note"));
    writeNote.SendKeys("Note for Delete");

    var backButton = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/back_btn"));
    backButton.Click();

    var menu = _driver.FindElement(MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/menu_btn"));
    menu.Click();

    var deleteOption = _driver.FindElement(MobileBy.AndroidUIAutomator("new UiSelector().text(\"Delete\")"));
    deleteOption.Click();

    var ok = _driver.FindElement(MobileBy.Id("android:id/button1"));
    ok.Click();

    var deletedNote = _driver.FindElements(By.XPath("//android.widget.TextView[@text='Note for Delete']"));
    Assert.That(deletedNote, Is.Empty, "The note was not deleted successfully.");

}
```

3. Refactoring ColorNote Tests to Use Page Object Model (POM)

- **Define Page Object Classes:**
  - Create a class for each screen or component, encapsulating elements and actions.



```
// Define elements
1 reference
public IWebElement SkipTutorialButton => _driver.FindElement(MobileBy.Id
    ("com.socialnmobile.dictapps.notepad.color.note:id/btn_start_skip"));
1 reference
public IWebElement AddNoteButton => _driver.FindElement(MobileBy.Id
    ("com.socialnmobile.dictapps.notepad.color.note:id/main_btn1"));
1 reference
public IWebElement CreateTextNoteOption => _driver.FindElement
    (MobileBy.AndroidUIAutomator("new UiSelector().text(\"Text\")"));
1 reference
public IWebElement WriteNoteField => _driver.FindElement
    (MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/edit_note"));
1 reference
public IWebElement BackButton => _driver.FindElement
    (MobileBy.Id("com.socialnmobile.dictapps.notepad.color.note:id/back_btn"));
3 references | ● 2/2 passing
public IWebElement NoteTitle(string title) => _driver.FindElement
    (By.XPath($"//android.widget.TextView[@resource-id='com.socialnmobile.dictapps.notepad.color.note:id/title' and @text='{title}']"));
1 reference
public IWebElement MenuButton => _driver.FindElement(MobileBy.Id
    ("com.socialnmobile.dictapps.notepad.color.note:id/menu_btn"));
1 reference
public IWebElement DeleteOption => _driver.FindElement
    (MobileBy.AndroidUIAutomator("new UiSelector().text(\"Delete\")"));
1 reference
public IWebElement ConfirmDeleteButton => _driver.FindElement(MobileBy.Id("android:id/button1"));
```



```
// Define actions
1 reference
public void SkipTutorial()
{
    try
    {
        SkipTutorialButton.Click();
    }
    catch (NoSuchElementException)
    {
        // Tutorial skip button not found, continue with setup
    }
}

3 references | 3/3 passing
public void AddNote() => AddNoteButton.Click();
3 references | 3/3 passing
public void CreateTextNote() => CreateTextNoteOption.Click();
3 references | 3/3 passing
public void WriteNoteContent(string content) => WriteNoteField.SendKeys(content);
7 references | 3/3 passing
public void ClickBackButton() => BackButton.Click();
1 reference | 1/1 passing
public void OpenMenu() => MenuButton.Click();
1 reference | 1/1 passing
public void ClickDeleteOption() => DeleteOption.Click();
1 reference | 1/1 passing
public void ConfirmDelete() => ConfirmDeleteButton.Click();
```

- **Update Test Class to Use POM:**
  - Replace direct element interactions with calls to the methods of the page object classes.
  - Example: Instead of _driver.FindElement, use _notepadPage.Element.

- **Create and Use Page Object Methods:**
  - Encapsulate common actions in methods within the page object class.
  - Example: AddNote, CreateTextNote, WriteNoteContent.