**PAGE OBJECT MODEL: (JAVA)**

It is a **design pattern** which creates/behaves like an object repository for web UI elements (locators) which can be readable and maintained easily and can be reused. It reduces duplication of code.

**Flow:**

* For each web page in the application there should be corresponding page class
* This **Page class (pure java classes)** will find all the locators of that web page and also contains Page methods (name should be given as per the task). which perform operations on those locators. Call them from the test in which you have to use. So the benefit from this will be if any changes in Page then you do not have to modify the test simply modify the respective page and that all.
* So, here you can create a layer between your test script and application page, which you have to automate. That is, it behaves like an **Object Repo** for web UI elements.

**Advantages:**

* Code becomes less and optimized because of the reusable page methods in the POM classes.
* Page Object Patten says operations and flows in the UI (page class) should be separated from verification (test). This concept makes our code cleaner and easy to understand.
* Object Repo is independent of test cases so, we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional Testing and at the same time with JBehave/Cucumber for acceptance testing.
* Methods get more realistic names which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

**PageFactory:**

Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver but it is very optimized.

* Here as well, we follow the concept of separation of Page Object Repository and Test Methods. Additionally, with the help of PageFactory class, we use annotations @FindBy to find WebElement. We use initElements method to initialize web elements

**Implementation of Page Object model using Selenium Webdriver:**

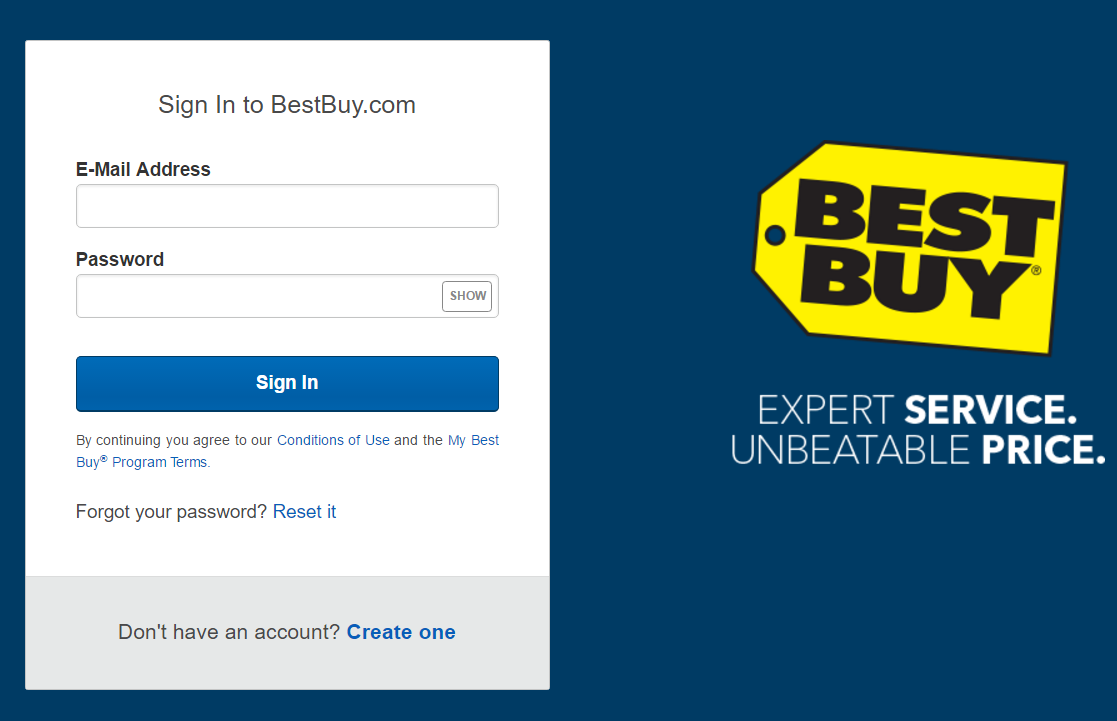
We have two choices and you can use any of it.

1. Page Object model without PageFactory
2. Page Object Model with Pagefactory.

**Page Object model without PageFactory:**

* Let’s take very basic scenario which you can relate to any application. Consider you have login page where Email Address, password, and Sign In button is present.

**Example**: Let’s take Best Buy site



**Step-1:**

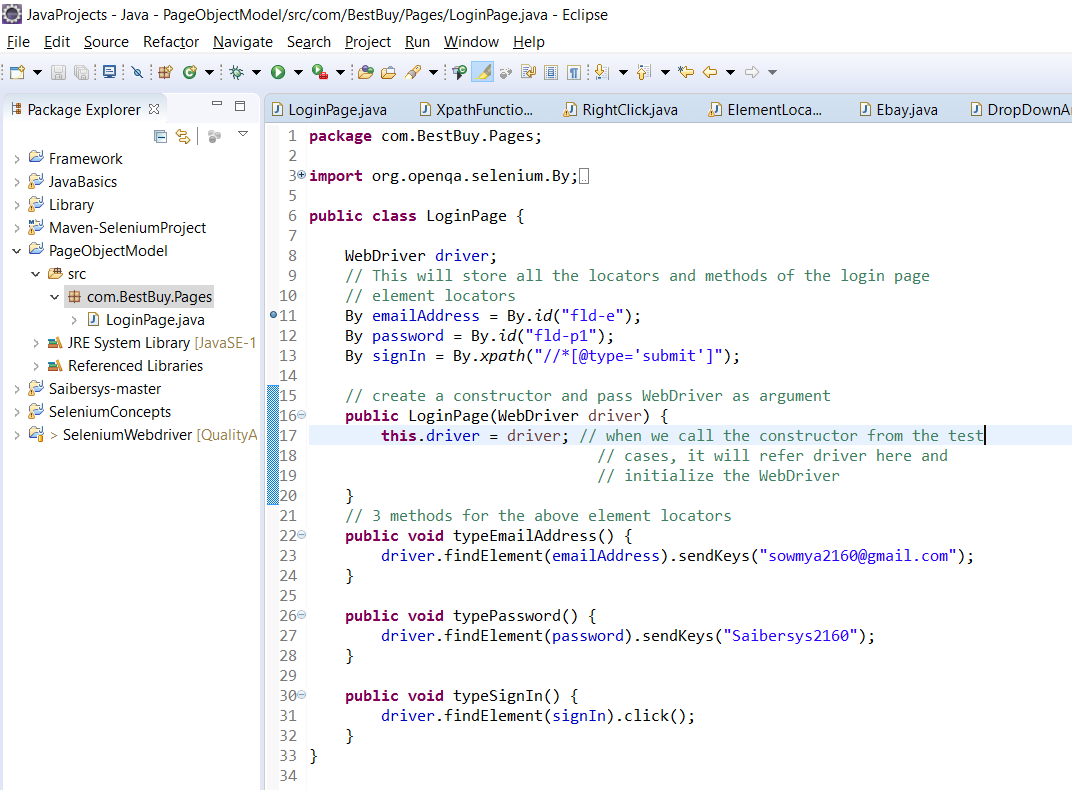
Now, for the above Login page will store three locators – email, password and SignIn and will create methods to access them.

* At first, create a project and create a package com.BestBuy.Pages -> create a LoginPage class inside it and write

3 element locators

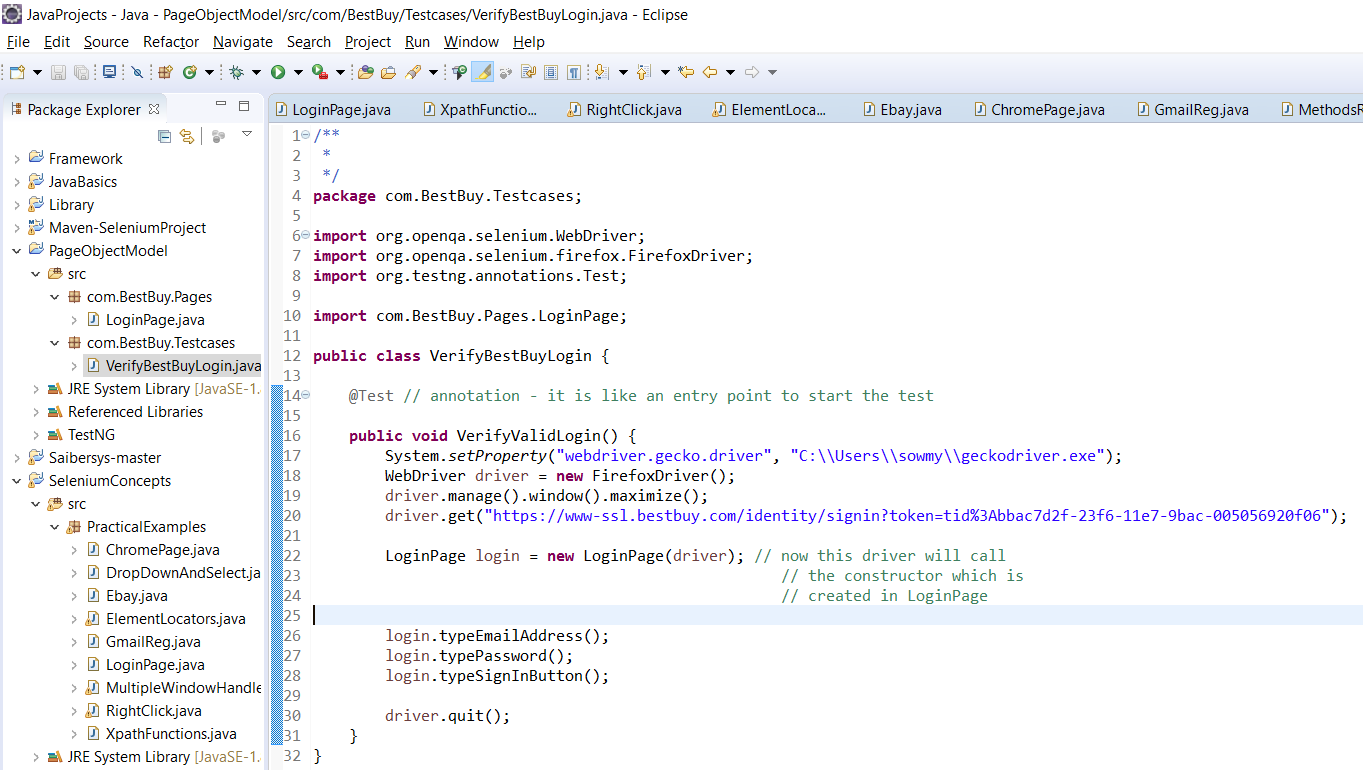
1 constructor

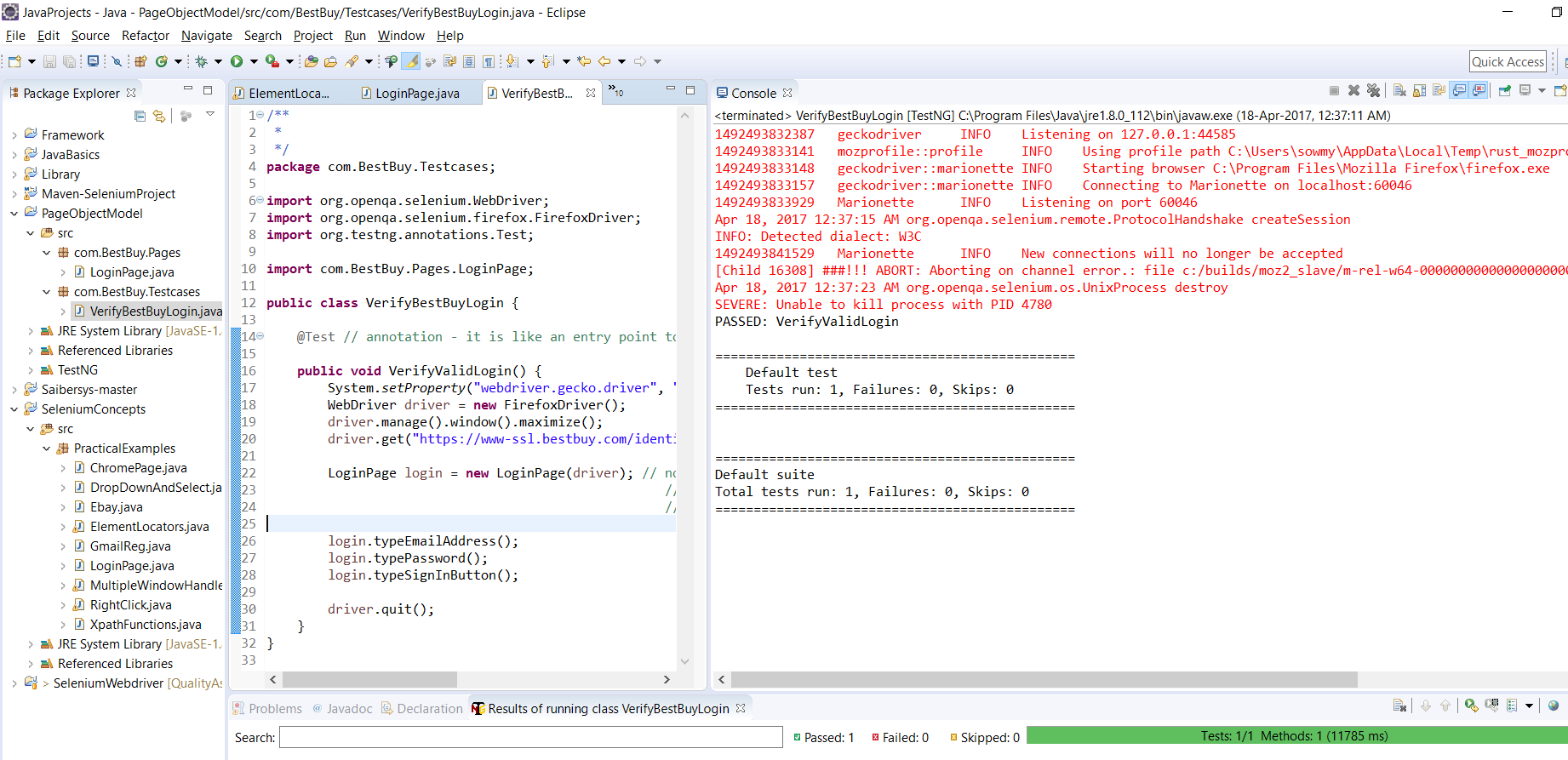
3 methods element locators



**Step-2:**

* Now, I want to design test case so I can use the Login class, which I created and can call the methods accordingly.
* For that create another package com.BestBuy.Testcases and create a VerifyBestBuyLogin class.
* Write the steps to invoke the website -> create an object for the java class LoginPage to call all the methods of it.
* Import TestNG to execute the test cases in the order which is given with the help of annotations
* After that execute and you can see the test cases as passed.





**Page Object Model with Pagefactory:**