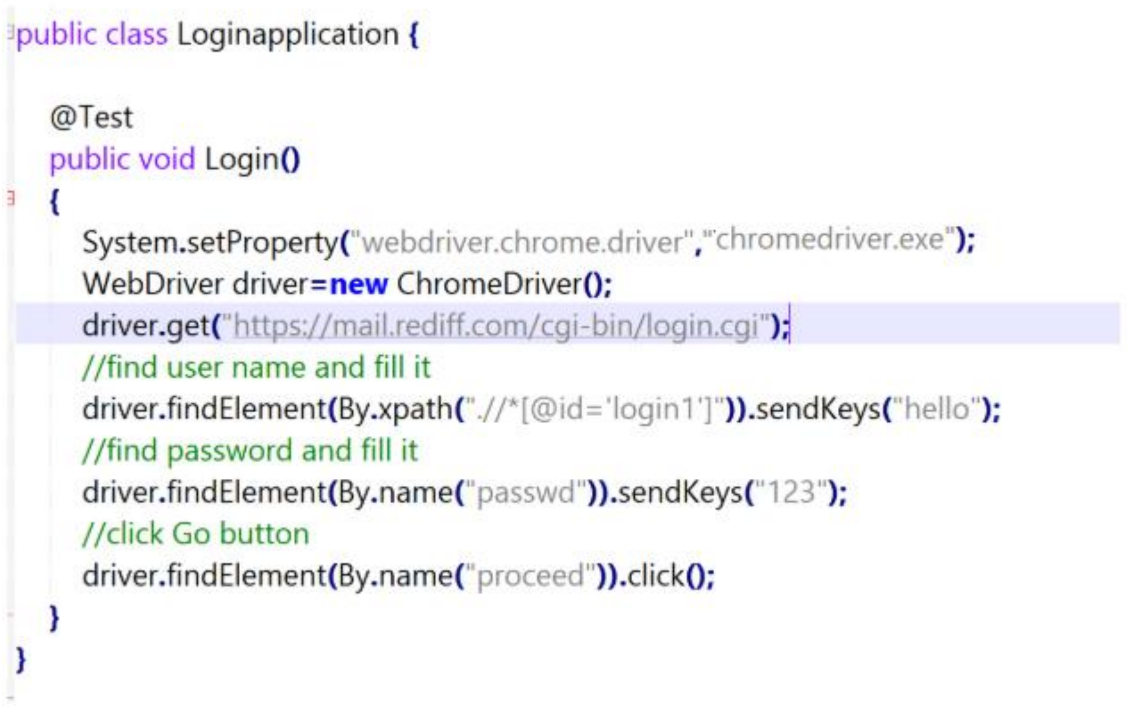
10. Demonstrate page object design pattern in Selenium.

Step 1: Explaining why POM is used

- Consider this simple script to log in to the Rediff mail website.

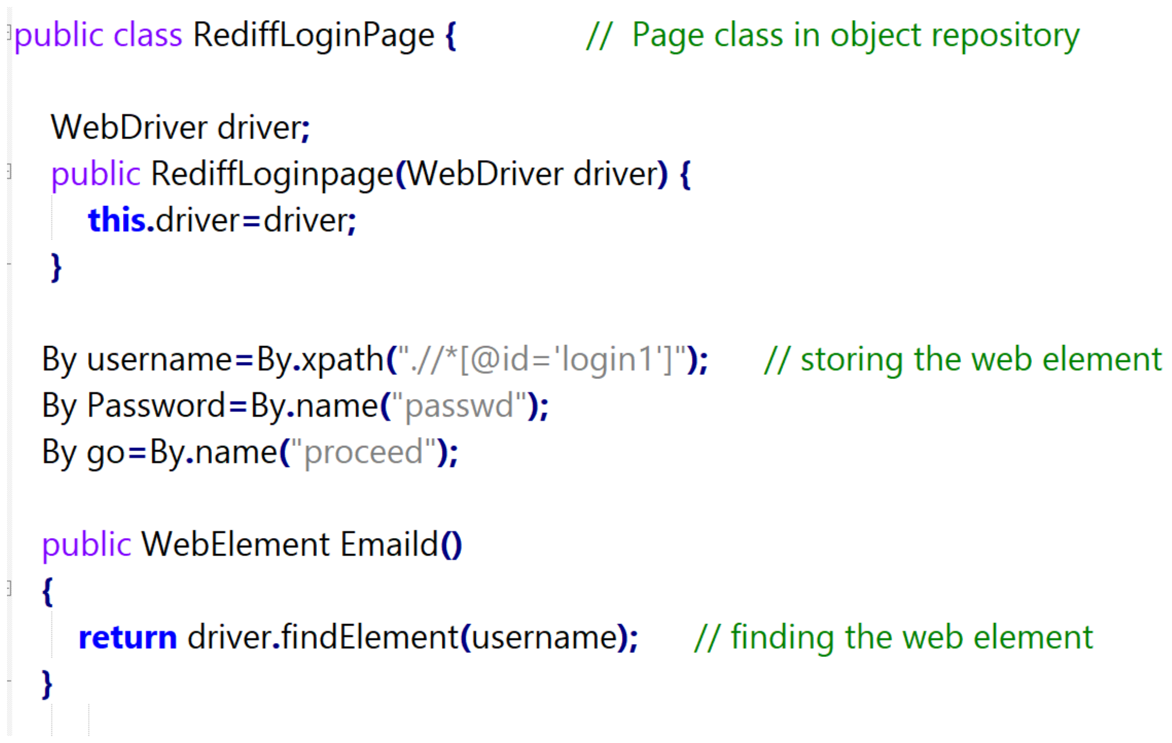


```
public class Loginapplication {

    @Test
    public void Login()
    {
        System.setProperty("webdriver.chrome.driver","chromedriver.exe");
        WebDriver driver=new ChromeDriver();
        driver.get("https://mail.rediff.com/cgi-bin/login.cgi");
        //find user name and fill it
        driver.findElement(By.xpath(".//*[@id='login1']")).sendKeys("hello");
        //find password and fill it
        driver.findElement(By.name("passwd")).sendKeys("123");
        //click Go button
        driver.findElement(By.name("proceed")).click();
    }
}
```

- As you can see, all we are doing is finding elements and filling values for those elements.
- This is a small script. Script maintenance looks easy. But with time the test suite will grow. As you add more and more lines to your code, things become tough.
- The main problem with script maintenance is that if 10 different scripts are using the same page element, with any change in that element, you need to change all the 10 scripts. This is time-consuming and error-prone.
- A better approach to script maintenance is to create a separate class file, which would find web elements, fill them, or verify them. This class can be reused in all the scripts using that element. In the future, if there is a change in the web element, we need to make the change in just 1 class file and not 10 different scripts.

Step 2: Implementing Page Object Model

- This is the basic structure of the Page Object Model (POM), where all Web Elements of the Application Under Test and the method that operate on these Web Elements, are maintained inside a class file.



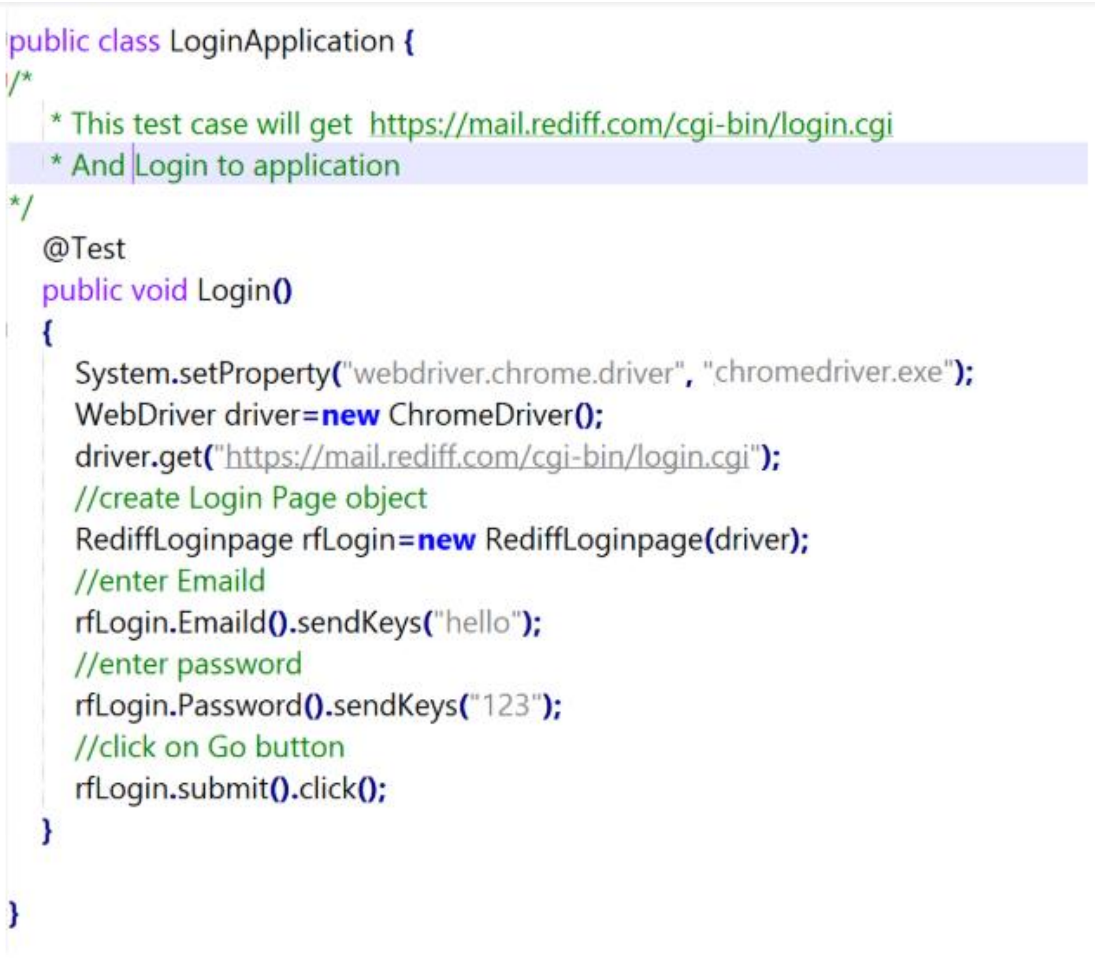
```
public class RediffLoginPage {            //  Page class in object repository

    WebDriver driver;
    public RediffLoginpage(WebDriver driver) {
        this.driver=driver;
    }

   By username=By.xpath(".//*[@id='login1']");      // storing the web element
   By Password=By.name("passwd");
   By go=By.name("proceed");

   public WebElement EmailId()
   {
       return driver.findElement(username);      // finding the web element
   }
```

Step 3: Writing code to implement the Page Object Model using a test case

- Test Case: Log in to Rediff mail. Here we will be dealing with Login page POM and Test script.
- Login Page:

```
public class RediffLoginPage {            //  Page class in object repository

  WebDriver driver;
  public RediffLoginpage(WebDriver driver) {
    this.driver=driver;
  }

  By username=By.xpath(".//*[@id='login1']");     // storing the web element
  By Password=By.name("passwd");
  By go=By.name("proceed");

  public WebElement EmailId()
  {
    return driver.findElement(username);     // finding the web element
  }

  public WebElement Password()
  {
    return driver.findElement(Password);
  }

  public WebElement submit()
  {
    return driver.findElement(go);
  }

}
```

- Test Case:



```
public class LoginApplication {
/*
  * This test case will get  https://mail.rediff.com/cgi-bin/login.cgi
  * And Login to application
*/
  @Test
  public void Login()
  {
    System.setProperty("webdriver.chrome.driver", "chromedriver.exe");
    WebDriver driver=new ChromeDriver();
    driver.get("https://mail.rediff.com/cgi-bin/login.cgi");
    //create Login Page object
    RediffLoginpage rfLogin=new RediffLoginpage(driver);
    //enter EmailId
    rfLogin.EmailId().sendKeys("hello");
    //enter password
    rfLogin.Password().sendKeys("123");
    //click on Go button
    rfLogin.submit().click();
  }

}
```

Step 4: Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

```
cd <folder path>
```

Initialize your repository using the following command:

```
git init
```

Add all the files to your git repository using the following command:

```
git add .
```

Commit the changes using the following command:

```
git commit .  -m “Changes have been committed.”
```

Push the files to the folder you initially created using the following command:

```
git push -u origin master
```