

# FRAMEWORK MOCK QUESTIONS

## 1.Create the Reusable method below actions

```
public class reusableMethods {
    WebDriver driver;

    public void explicitWait(WebElement element) {
        WebDriverWait driverWait = new WebDriverWait(driver, Duration.ofSeconds(30));
        driverWait.until(ExpectedConditions.visibilityOf(element));
    }

    public boolean isDisplayed(WebElement element) {
        boolean displayed = element.isDisplayed();
        return displayed;
    }

    public boolean isEnabled(WebElement element) {
        boolean enabled = element.isEnabled();
        return enabled;
    }

    // 1. insert value in textbox
    public void elementSendKeys(WebElement element, String data) {
        explicitWait(element);
        boolean isEnabled = isEnabled(element);
        boolean isDisplayed = isDisplayed(element);
        if (isEnabled && isDisplayed) {
            element.sendKeys(data);
        }
    }

    // 2. Click button
    public void elementClick(WebElement element) {
        explicitWait(element);
        boolean isDisplayed = isDisplayed(element);
        boolean isEnabled = isEnabled(element);
        if (isDisplayed && isEnabled) {
            element.click();
        }
    }
}
```

```

// 3. Take Screenshot
public File takescreenshot() {
    TakesScreenshot ts = (TakesScreenshot) driver;
    File screenshotAs = ts.getScreenshotAs(OutputType.FILE);
    return screenshotAs;
}

// 4. Get the text from webpage
public String elementGetText(WebElement element) {

    String res = element.getText();
    return res;
}

// 5. Select value from dropdown by text
public void elementDdnVisibleText(WebElement element, String data) {
    explicitWait(element);
    boolean isdisplayed = isdisplayed(element);
    if (isdisplayed) {
        Select select = new Select(element);
        select.selectByVisibleText(data);
    }
}

// 6. Select value from dropdown by index
public void elementDdnIndex(WebElement element, int index) {
    explicitWait(element);
    boolean isdisplayed = isdisplayed(element);
    if (isdisplayed) {
        Select select = new Select(element);
        select.selectByIndex(index);
    }
}

// 7. insert value in textbox using java script
public void elementSendkeysJs(WebElement element, String data) {
    explicitWait(element);
    boolean isdisplayed = isdisplayed(element);
    if (isdisplayed) {
        JavascriptExecutor js = (JavascriptExecutor) driver;
        js.executeScript("arguments[0].setAttribute('value','" + data + "'", element);
    }
}

// 8. Switch to child window
public void childWindow() {
    String windowHandle = driver.getWindowHandle();
    Set<String> windowHandles = driver.getWindowHandles();
    for (String string : windowHandles) {
        if (!windowHandle.equals(string)) {
            driver.switchTo().window(string);
        }
    }
}
}

```

```

// 9. Close window
public void closeCurrent() {
    driver.close();
}

// 10.Switch into frame by WebElement
public void frameSwitchByElement(WebElement element) {
    explicitWait(element);
    boolean isDisplayed = isDisplayed(element);
    if (isDisplayed) {
        driver.switchTo().frame(element);
    }
}
}

```

2. Write the correct order of execution in TestNG?

@AfterMethod,@BeforeClass,@AfterClass,@Test,@BeforeMethod,  
 @BeforeSuite,@AfterSuite,@BeforeGroups,@AfterGroups,@BeforeTest,  
 @AfterTest

Answer:

@BeforeSuite

@BeforeTest

@BeforeGroups

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterGroups

@AfterTest

@AfterSuite

### 3. Write a code to perform login[POM+Excel+Junit+BaseClass]

Excel : Input Datas

POM : Locators & Business logics

BaseClass : Resuable methods

Junit : Execution & Assertion

Locator Details:

id="username\_omr"

id="password\_omr"

xpath= "//button[text()='Login']"

name="greenstechomr"

Verification Details

After login set a verification message as "Welcome to Greens Technologys OMR"

Answer:

Junit:

```
public class JunitTest extends BaseClass {
    @Test
    public void login() throws IOException {
        LoginPage loginPage = new LoginPage();
        loginPage.login(getCellValuefromExcel("Login", 1, 1), getCellValuefromExcel("Login", 1, 2));
        SearchPage page = new SearchPage();
        String expectedWelcomeMessage = page.expectedWelcomeMessage();
        Assert.assertEquals("Verify the success message after login", expectedWelcomeMessage,
            "Welcome to Greens Technologys OMR");
    }
}
```

POM:

```
@Getter
@Setter
public class LoginPage extends BaseClass {
    public LoginPage() {
        PageFactory.initElements(driver, this);
    }

    @FindBy(id = "username_omr")
    private WebElement txtUsername;
    @FindBy(id = "password_omr")
    private WebElement txtPassword;
    @FindBy(xpath = "//button[text()='Login']")
    private WebElement btnLogin;

    public void login(String username, String password) {
        elementSendKeys(txtUsername, username);
        elementSendKeys(txtPassword, password);
        elementClick(btnLogin);
    }
}

@Getter
@Setter
public class SearchPage extends BaseClass {

    public SearchPage() {
        PageFactory.initElements(driver, this);
    }

    @FindBy(id = "greentechomr")
    private WebElement textWelcomeMessage;

    public String expectedWelcomeMessage() {
        return elementGetText(textWelcomeMessage);
    }
}
```

## BaseClass:

```
public class BaseClass {
    protected static WebDriver driver;

    public void explicitWait(WebElement element) {
        WebDriverWait driverWait = new WebDriverWait(driver, Duration.ofSeconds(30));
        driverWait.until(ExpectedConditions.visibilityOf(element));
    }

    public boolean isDisplayed(WebElement element) {
        boolean displayed = element.isDisplayed();
        return displayed;
    }

    public boolean isEnabled(WebElement element) {
        boolean enabled = element.isEnabled();
        return enabled;
    }

    // 1. insert value in textbox
    public void elementSendKeys(WebElement element, String data) {
        explicitWait(element);
        boolean isEnabled = isEnabled(element);
        boolean isDisplayed = isDisplayed(element);
        if (isEnabled && isDisplayed) {
            element.sendKeys(data);
        }
    }

    // 2. Click button
    public void elementClick(WebElement element) {
        explicitWait(element);
        boolean isDisplayed = isDisplayed(element);
        boolean isEnabled = isEnabled(element);
        if (isDisplayed && isEnabled) {
            element.click();
        }
    }

    // 3. Get the text from webpage
    public String elementGetText(WebElement element) {
        String res = element.getText();
        return res;
    }
}
```

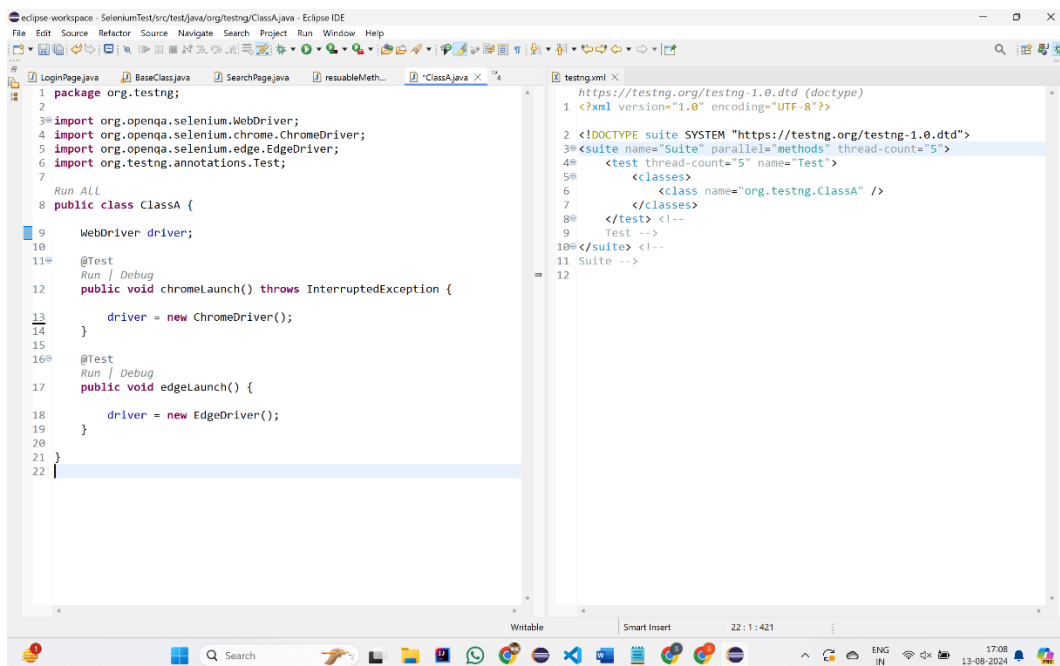
```
// 4. Get data from cell
public String getCellValueFromExcel(String sheetName, int rowIndex, int columnIndex) throws IOException {
    File file = new File("C:\\Users\\SRINIVASU\\eclipse-workspace\\Frameworkkk\\ExcelSheet\\TestNG.xlsx");
    FileInputStream fis = new FileInputStream(file);
    Workbook book = new XSSFWorkbook(fis);
    Sheet sheet = book.getSheet(sheetName);
    Row row = sheet.getRow(rowIndex);
    Cell cell = row.getCell(columnIndex);
    CellType type = cell.getCellType();
    String res = null;
    switch (type) {
        case STRING:
            res = cell.getStringCellValue();
            break;

        case NUMERIC:
            if (DateUtil.isCellDateFormatted(cell)) {
                Date dateCellValue = cell.getDateCellValue();
                SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yy");
                res = dateFormat.format(dateCellValue);
            } else {
                double numericCellValue = cell.getNumericCellValue();
                long round = Math.round(numericCellValue);
                if (round == numericCellValue) {
                    res = String.valueOf(round);
                } else {
                    res = String.valueOf(numericCellValue);
                }
            }
            break;

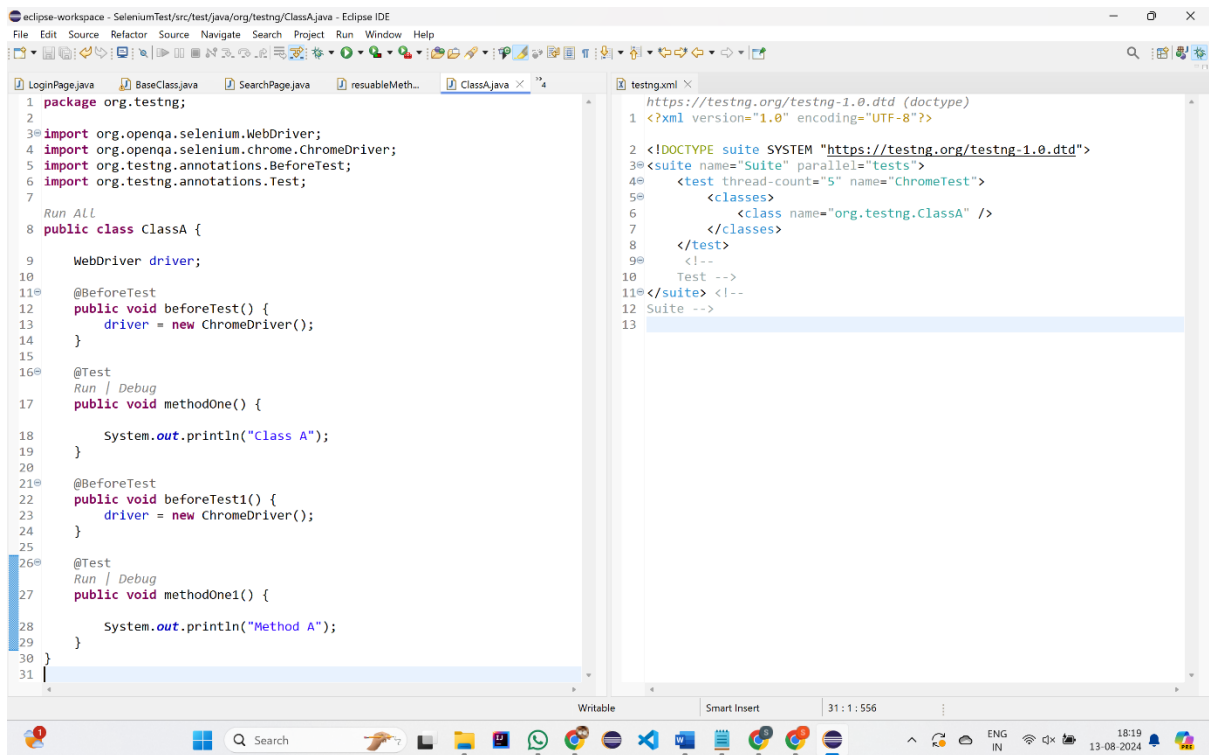
        default:
            break;
    }
    return res;
}
```

## 4. Parallel Execution methods, tests, classes in TestNG

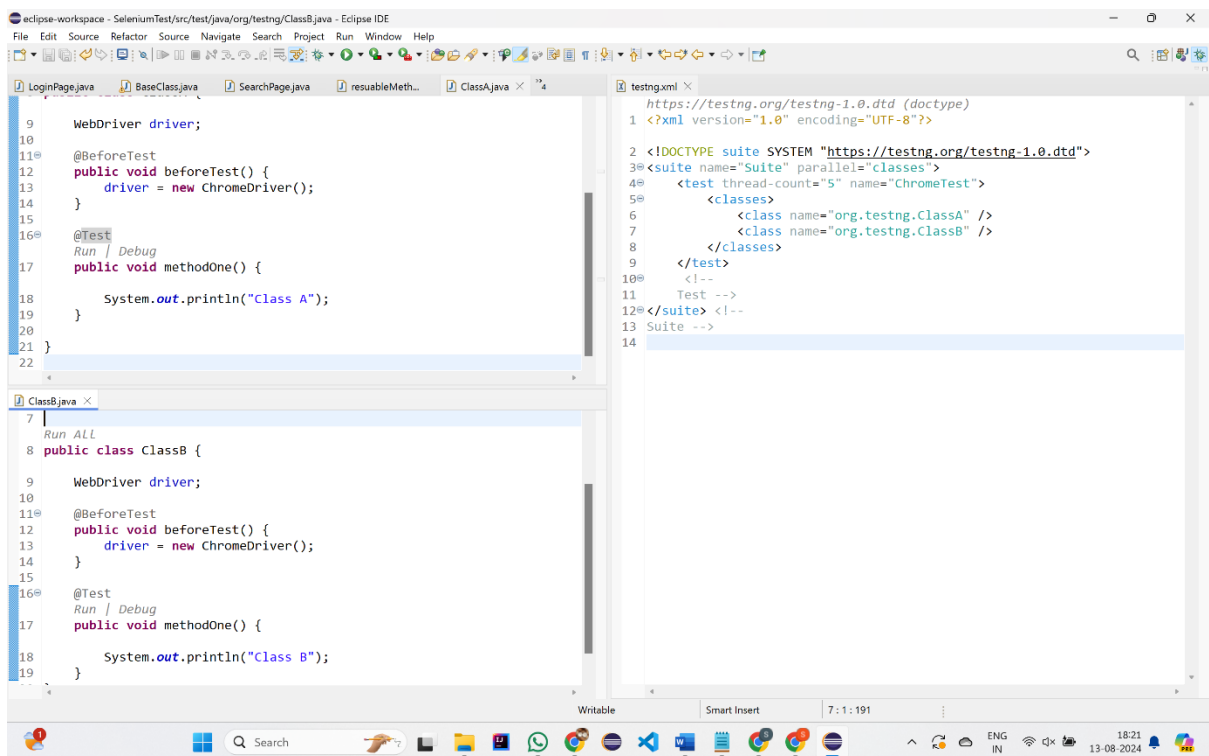
### Method Level:



## Tests level:



## Class Level:





## 5. Write a code to Data provider code

Run ALL

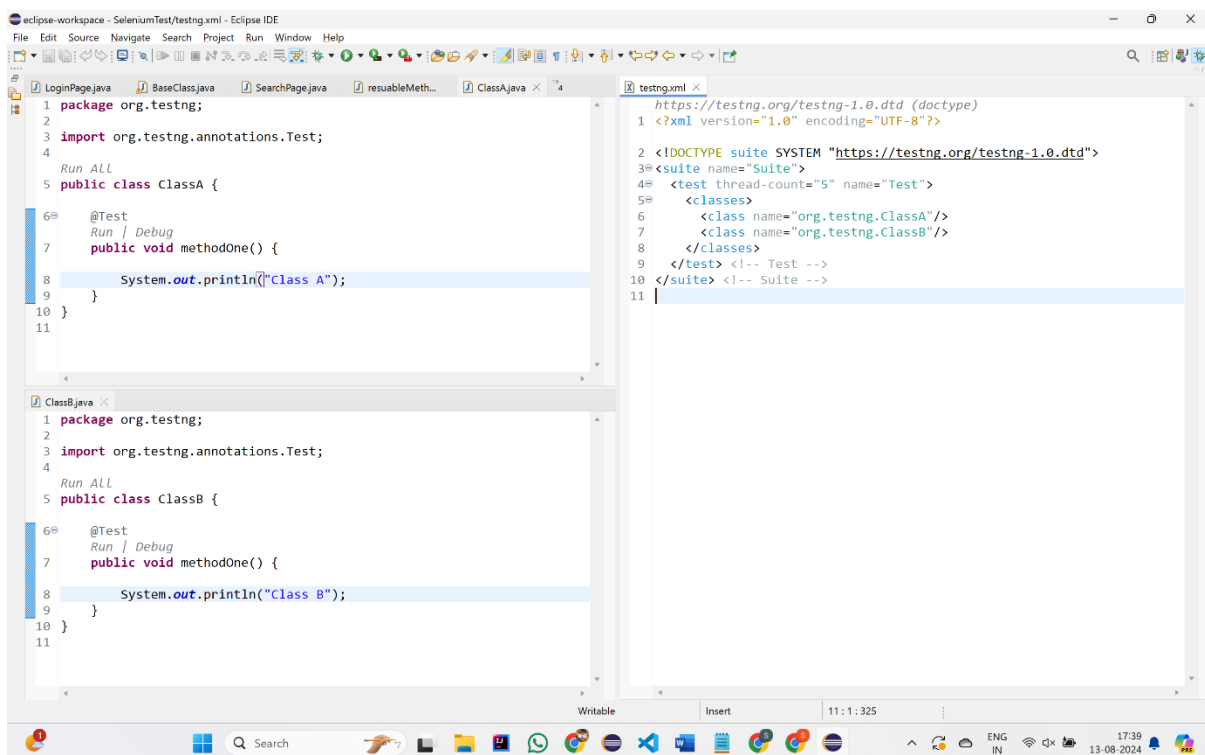
```
public class ClassA {

    @DataProvider(name = "userData")
    public Object[][] provideUserData() {
        return new Object[][] { { "John", 20 }, { "Age", 30 }, { "Doe", 40 } };
    }

    @Test(dataProvider = "userData")
    Run | Debug
    private void sampleTest(String name, int age) {

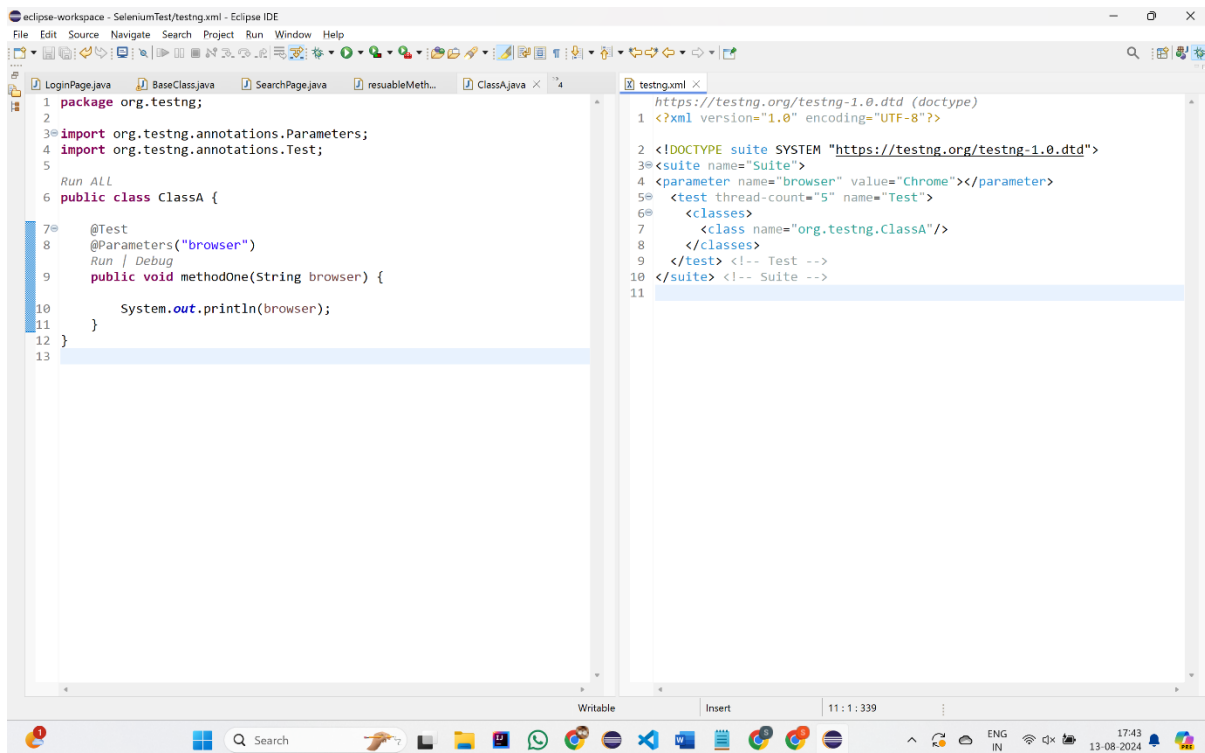
        System.out.println(name + age);
    }
}
```

## 6. Write a code to Run more than one classes in testng through TestNG.xml file

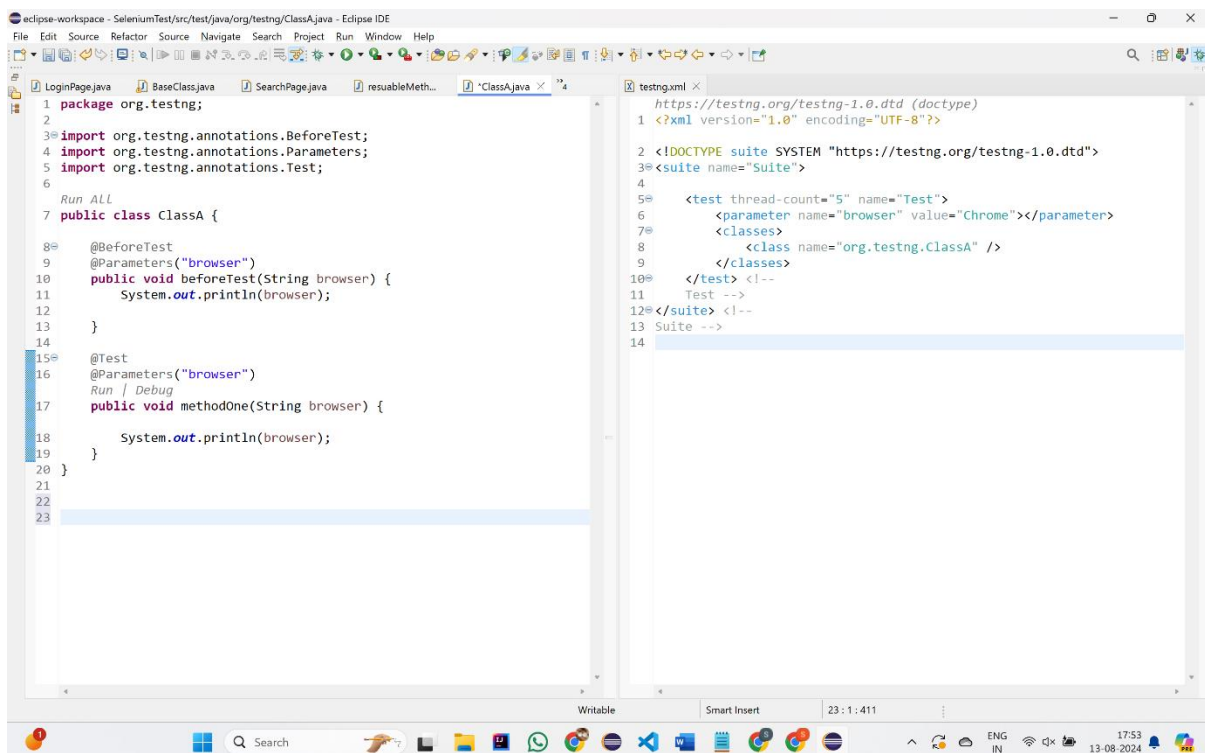


## 7. Pass parameter from testng.xml file[method/Suite/test/class]

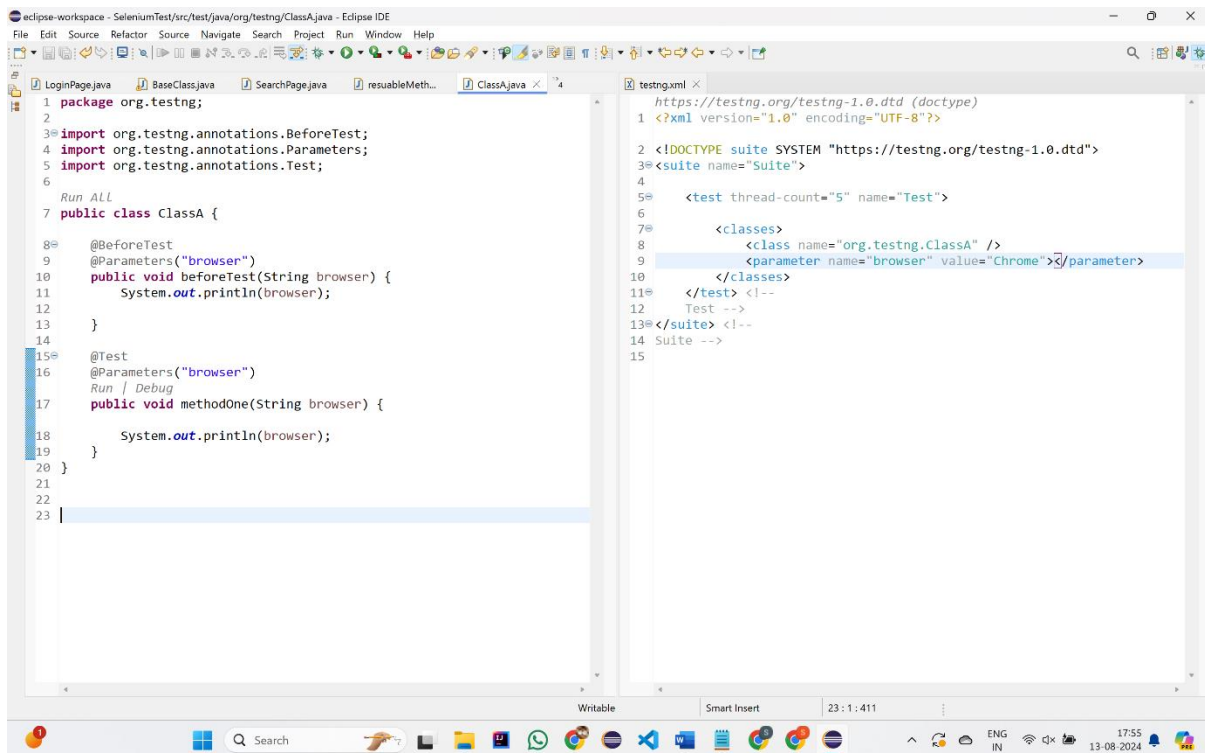
### Suite Level:



### Test Level:



# Class Level

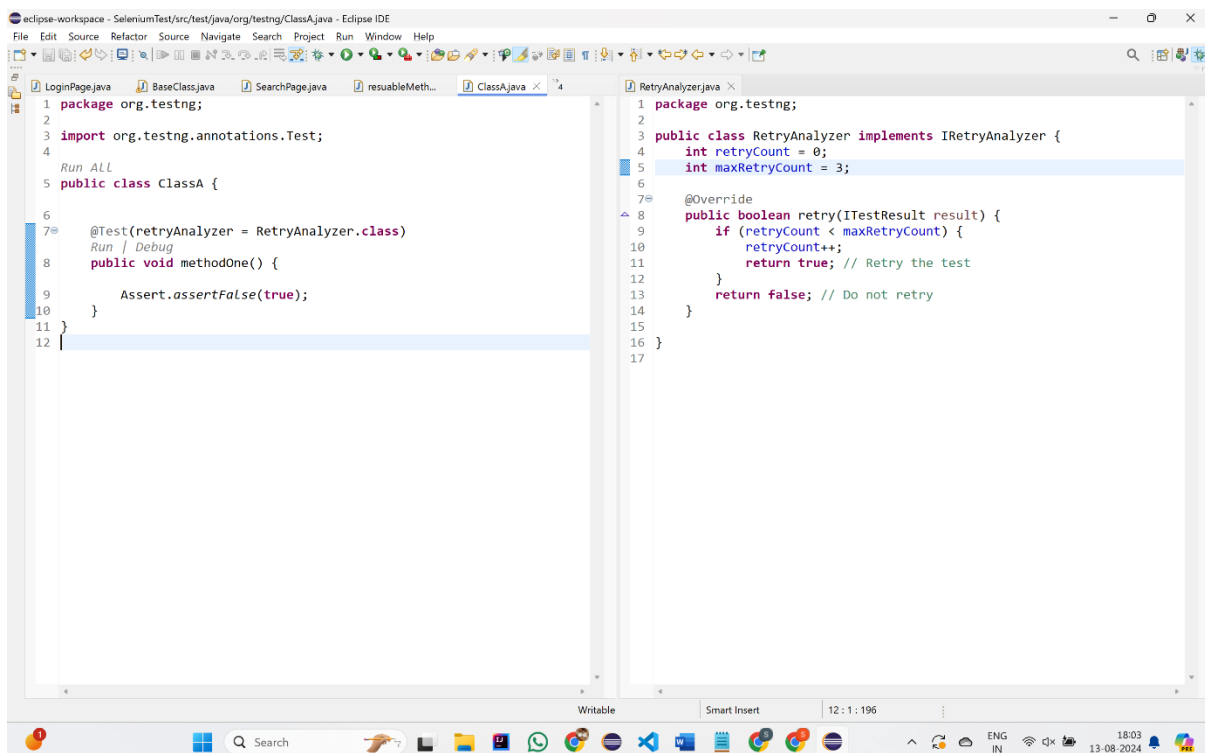


The screenshot shows the Eclipse IDE with two files open. The left file is `ClassA.java` and the right file is `testng.xml`.

```
1 package org.testing;
2
3 import org.testng.annotations.BeforeTest;
4 import org.testng.annotations.Parameters;
5 import org.testng.annotations.Test;
6
7 public class ClassA {
8     @BeforeTest
9     @Parameters("browser")
10    public void beforeTest(String browser) {
11        System.out.println(browser);
12    }
13
14
15    @Test
16    @Parameters("browser")
17    public void methodOne(String browser) {
18        System.out.println(browser);
19    }
20 }
21
22
23
```

```
1 https://testng.org/testng-1.0.dtd (doctype)
2 <?xml version="1.0" encoding="UTF-8"?>
3 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
4 <suite name="Suite">
5     <test thread-count="5" name="Test">
6         <classes>
7             <class name="org.testing.ClassA" />
8             <parameter name="browser" value="Chrome">/parameter>
9         </classes>
10    </test> <!--
11    Test -->
12 </suite> <!--
13 Suite -->
14
```

## 8. Write a code to rerun failed testcases in testing

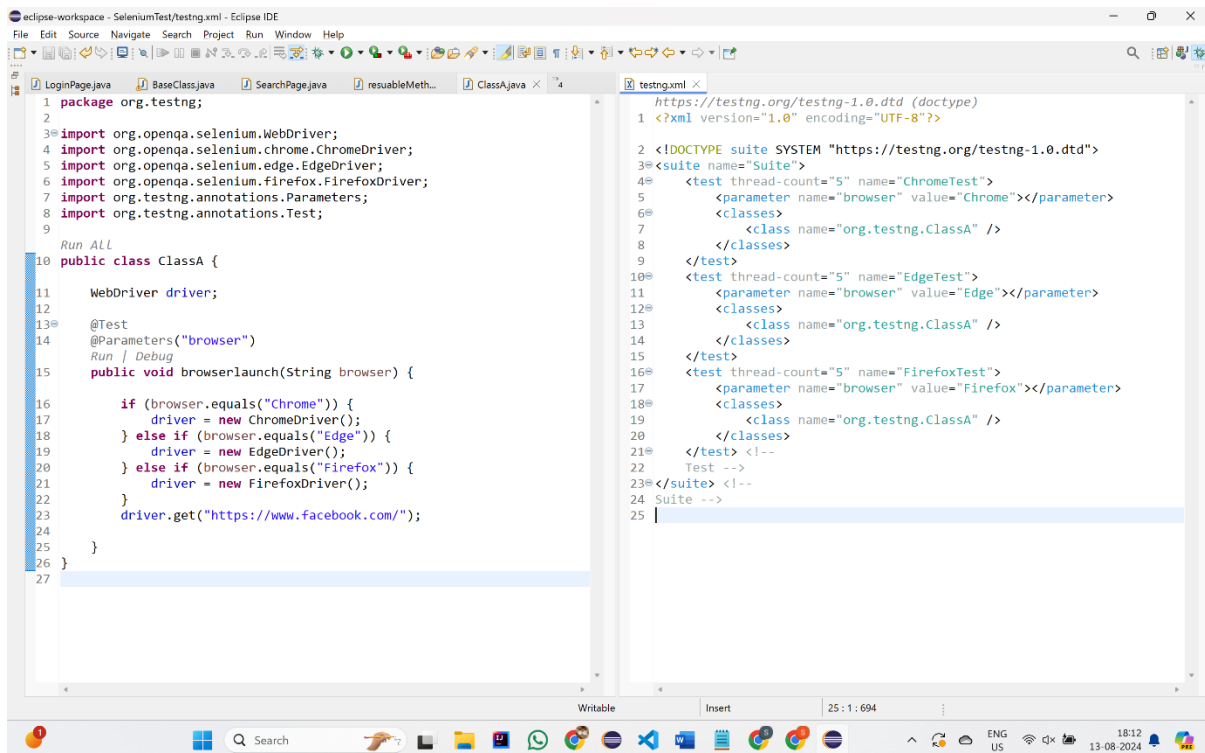


The screenshot shows the Eclipse IDE with two files open. The left file is `ClassA.java` and the right file is `RetryAnalyzer.java`.

```
1 package org.testing;
2
3 import org.testng.annotations.Test;
4
5 public class ClassA {
6
7     @Test(retryAnalyzer = RetryAnalyzer.class)
8     public void methodOne() {
9         Assert.assertFalse(true);
10    }
11 }
12
```

```
1 package org.testing;
2
3 public class RetryAnalyzer implements IRetryAnalyzer {
4     int retryCount = 0;
5     int maxRetryCount = 3;
6
7     @Override
8     public boolean retry(ITestResult result) {
9         if (retryCount < maxRetryCount) {
10             retryCount++;
11             return true; // Retry the test
12         }
13         return false; // Do not retry
14     }
15 }
16
17
```

## 9. Write a code for cross browser testing



## 10. Write a code for Datatable

```
public class ClassA {
    public void TwoDWithHeader(io.cucumber.datatable.DataTable dataTable) {
        List<Map<String, String>> asMaps = dataTable.asMaps();
        Map<String, String> map = asMaps.get(1);
        String value1 = map.get("value1");
        String value2 = map.get("value2");
        String value3 = map.get("value3");
        String value4 = map.get("value4");
        String value5 = map.get("value5");
    }
}
```