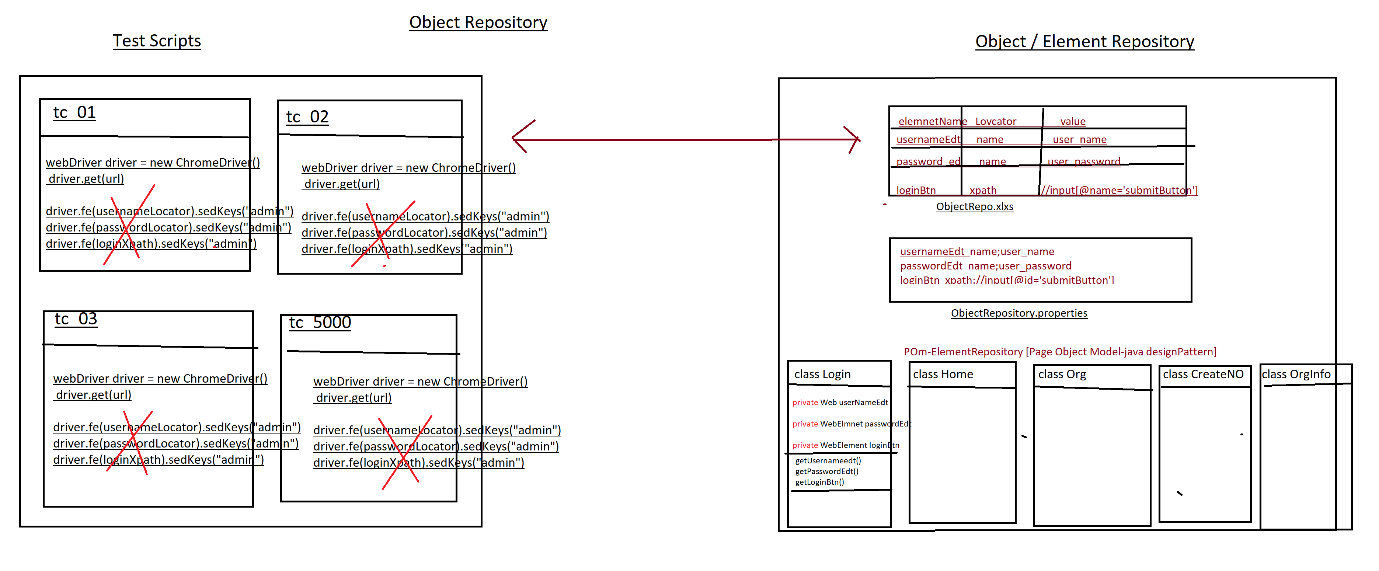
1. What is Object/Elements/POM Repository

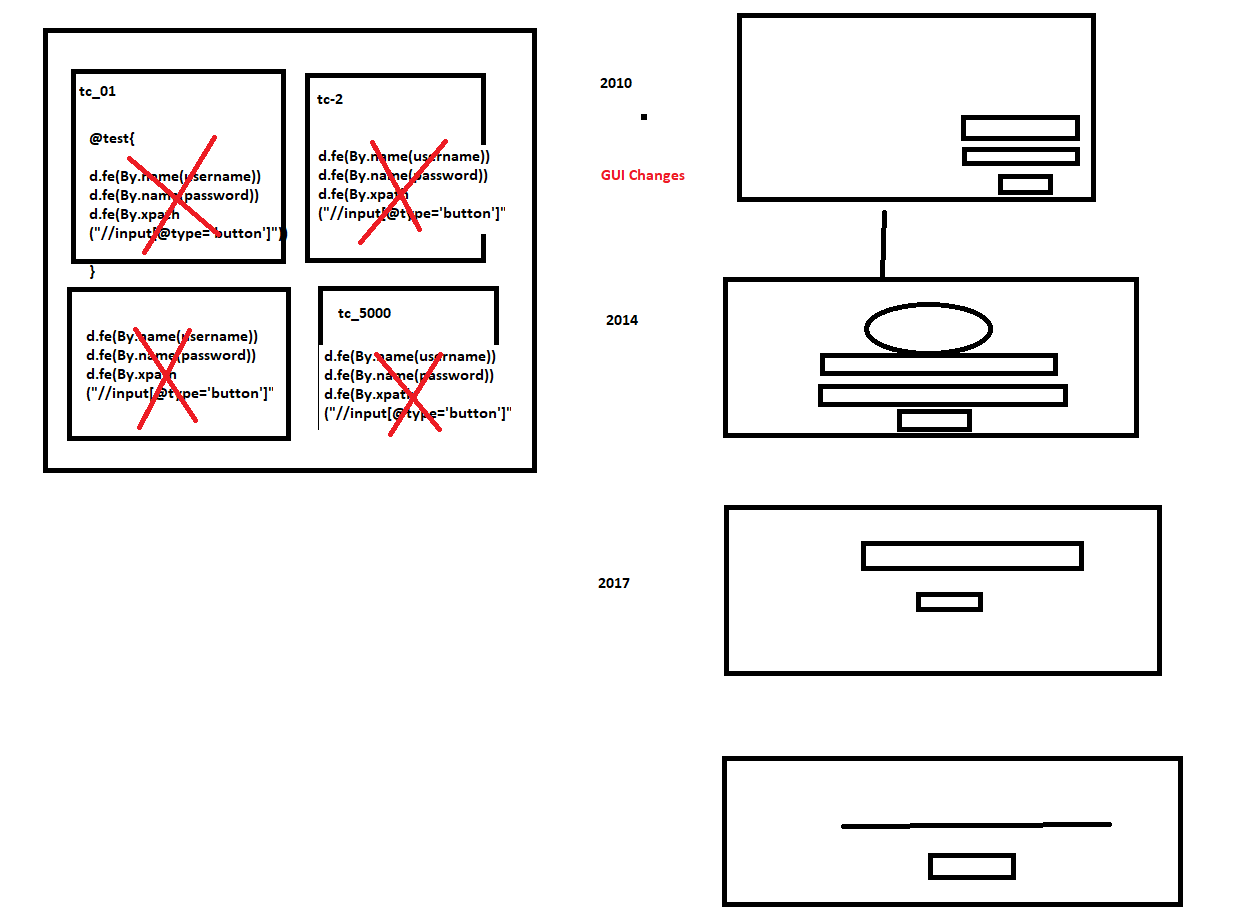
Its collection of elements locators & business libraries in one place & its developed using POM design pattern



1. Why Object repository ?

As per the rule of the automation, we should not hardcode[fixed]elements with in test Scripts instead, we should get elements from Object Repository , because in Agile process due to frequent requirement changes , modification & maintenance of elements is tedious job

EG : below is the example of Gmail application for GUI changes



1. What is the advantages repository?
2. Reusability of elements, no need to Rewrite xpath & other locators again & again
3. Modification in Repository is easy, when GUI changes frequently
4. Maintenance is easy, because all the elements we kept in one place
5. Test Script Code Optimized via business reusable libraries
6. More Readability
7. Test Script development is faster due to business lib
8. Test Script is more robust
9. Handle Stale Elements Exception.
10. What is POM?

POM is a java design pattern preferred by google to develop object repository.

1. Why POM ?

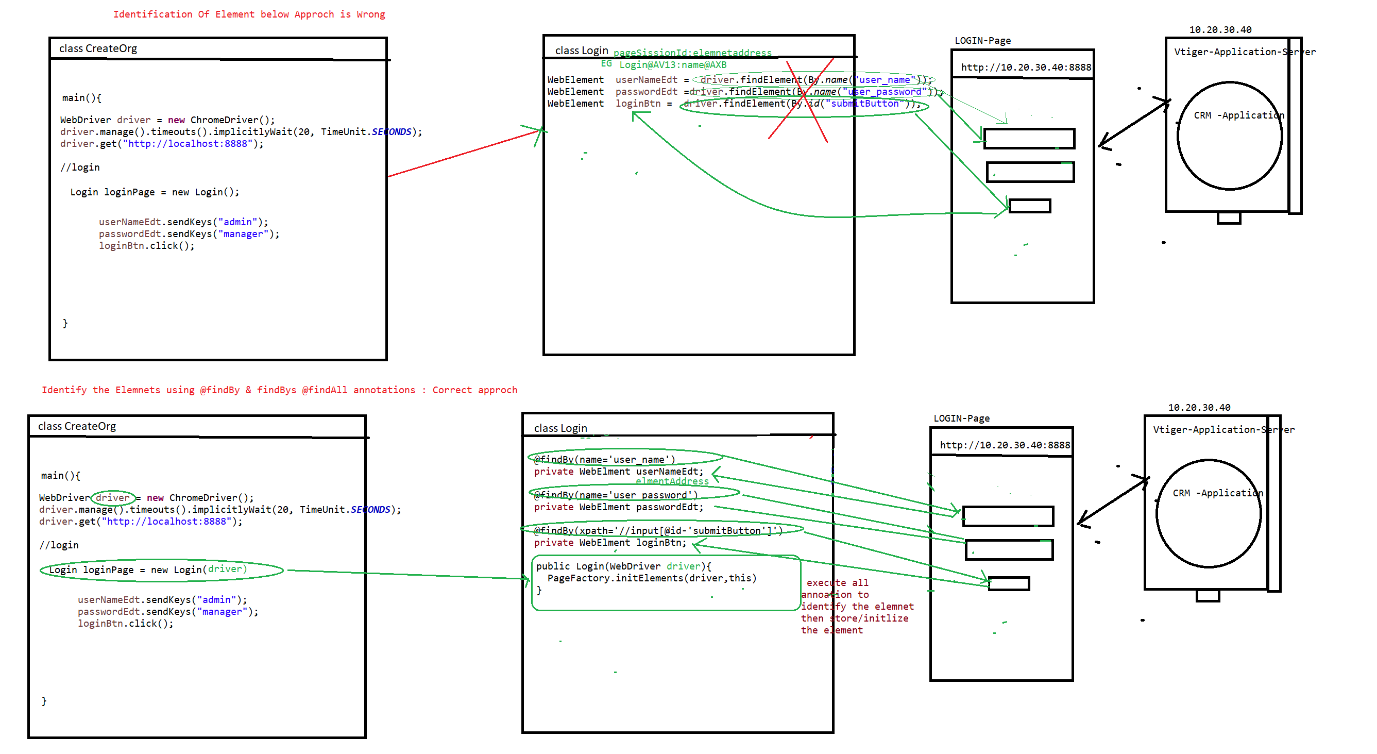
It’s a well-organized structured design pattern, where we can maintain all the web elements in page wise, due to POM design pattern maintains & modification is easy & faster.

1. Advantages of POM:
2. Well organized structure
3. Handle stale element exception.
4. maintains & modification of element is easy
5. We can directly store Web Elements in java class
6. Better fit for Agile processes
7. Support Auto healing feature
8. Why @FindBy & @Findall annotation instead of driver.findElement(“locator”)

Ans : to avoid staleElementReferenceException & also we can use Auto heling feature in @findAll annoation

1. What is staleElementReferenceException ?

It’s one of selenium Exception , whenever webdriver try to identify an element , element was available in GUI, but at time of performing an action on the elements element was not recognized due to page got refreshed or elements may become old or element not attached to page in such case we get staleElementReferenceException



1. Rules of POM

Rule 1 : create separte java class for every page in a application & class name should be same page name

Rule 2 : Identify all the elements using @findBy & @findAll , @findbys annotations & store them in speific pom / java class (Element declartion)

Rule3 : For Every POM class create Constructor to get an Object of the POM class , inside the Constructor use Pagefactory.initElement() to initialize the Page Elements at the time Object creation, in order to initialize all the page Elements we should use Pagefactory.initElement() (Element initialization)

Rule 4 : declare all the WebElements as private & provide getters methods to accesses elements in testScripts class [this processes is called Encapsulation]

Note : to create getters mtds inside the java class fallow below steps

=>place cursor inside the class🡺 Right click🡺source 🡺generate getters & setters 🡺 select the getters check box 🡺 click on ok button

Rule 5 : Go to every Page & identify the reusable business libraries & implement them in same POM class [Utilization]

There are two ways , we can utilize the POM elements in testScropts

1. Using getters() // single elements access
2. Using business libraries // multiple elements access
3. Difference between POM & PageFactory design pattern?

POM is java design pattern, where will maintain all the Web element locator in well-organized manner

Pagefactory it’s an extended design pattern of POM , which is used to create an Object to POM classes , & at the time of object creation it will execute all @findBy @findbys annotation then initialize all the elements

Difference between @findBy , @findAll &@findBys annotation

All annotation available in Selenium webdriver, its traditional ways to identify the elements in GUI.

@findBy : used to identify the element using one locator or one condition

@findAll : it contains multiple @findBy annotation , it mean we can identify the same element using multiple locator (multiple conditions) , it will use OR condition during execution of locator

@findALL({ @findBy(@id=’username’) , @findBy(name=’user’)})

Private Webelements userNAmeEdt;

Note : using above concepts we can achieve **Autohealing** technique

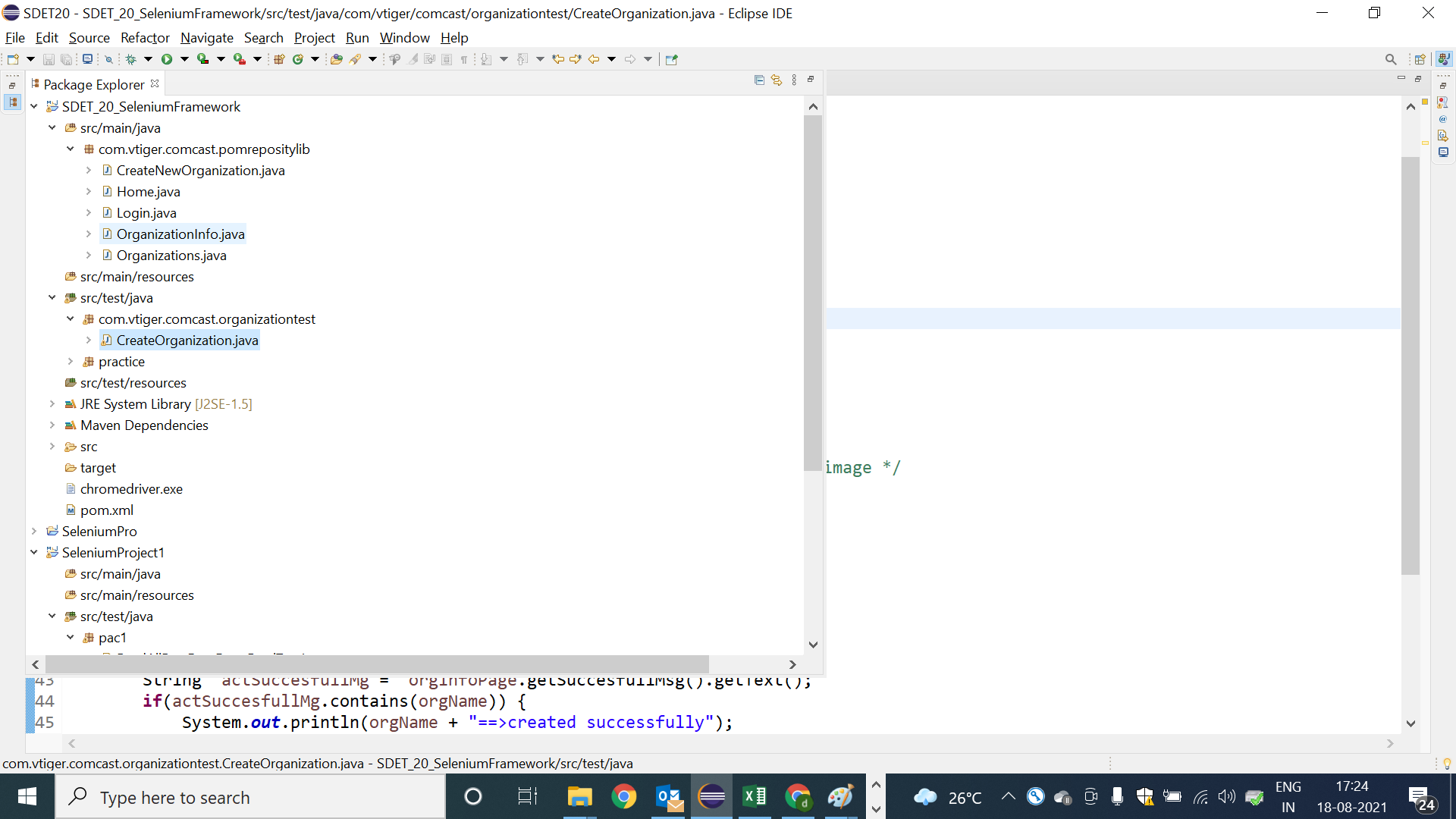
**AutoHealing** : during execution , if one locator fails to identify the element , it will retry to identify the same element using another locator

@FindBys : it contains multiple @findBy annotation , it mean we can identify the elements using multiple locator (multiple conditions) , it will use AND condition to during execution of locator

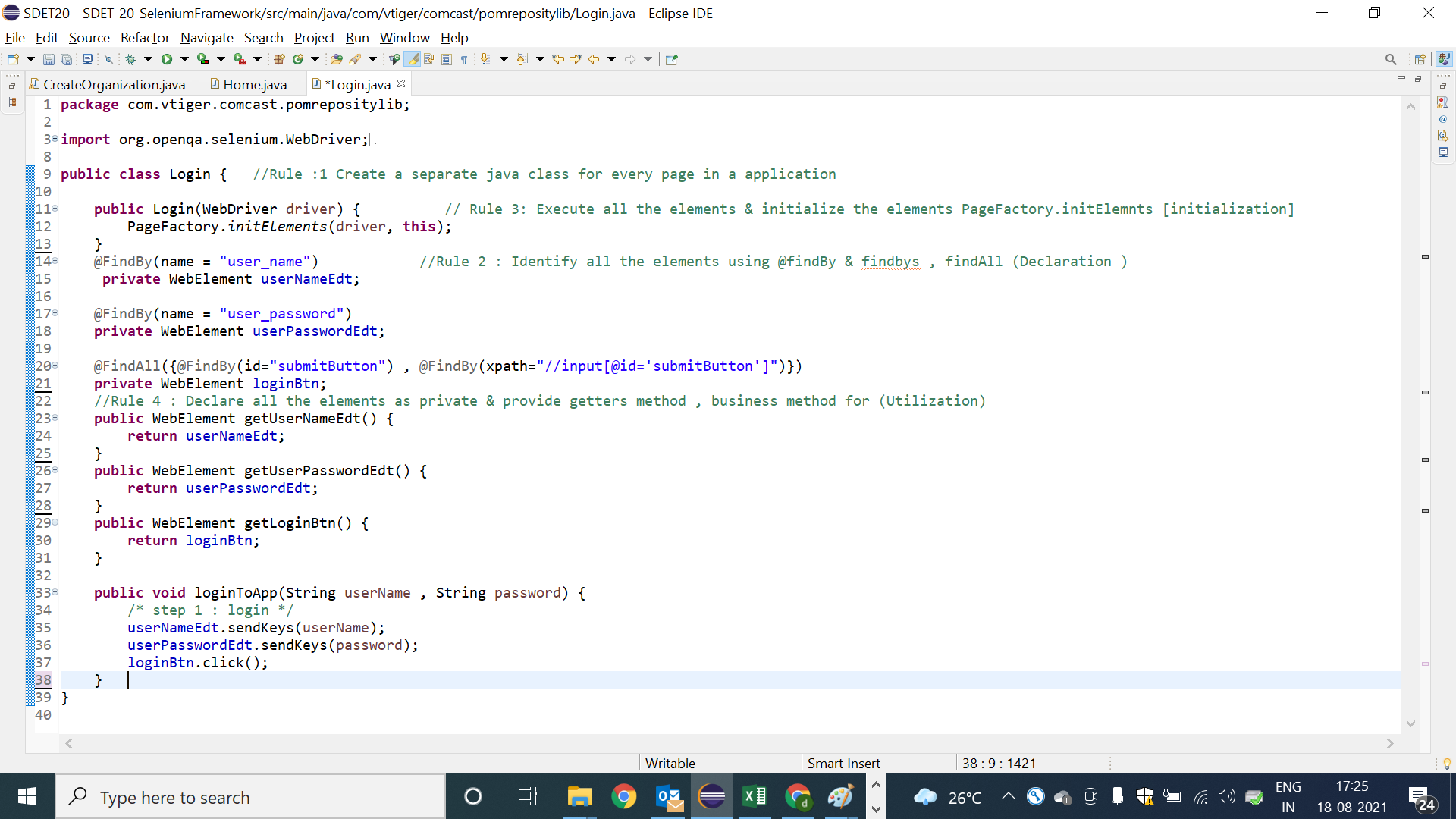
@findBys({ @findBy(@id=’username’) , @findBy(name=’user’)})

Private Webelements userNAmeEdt;

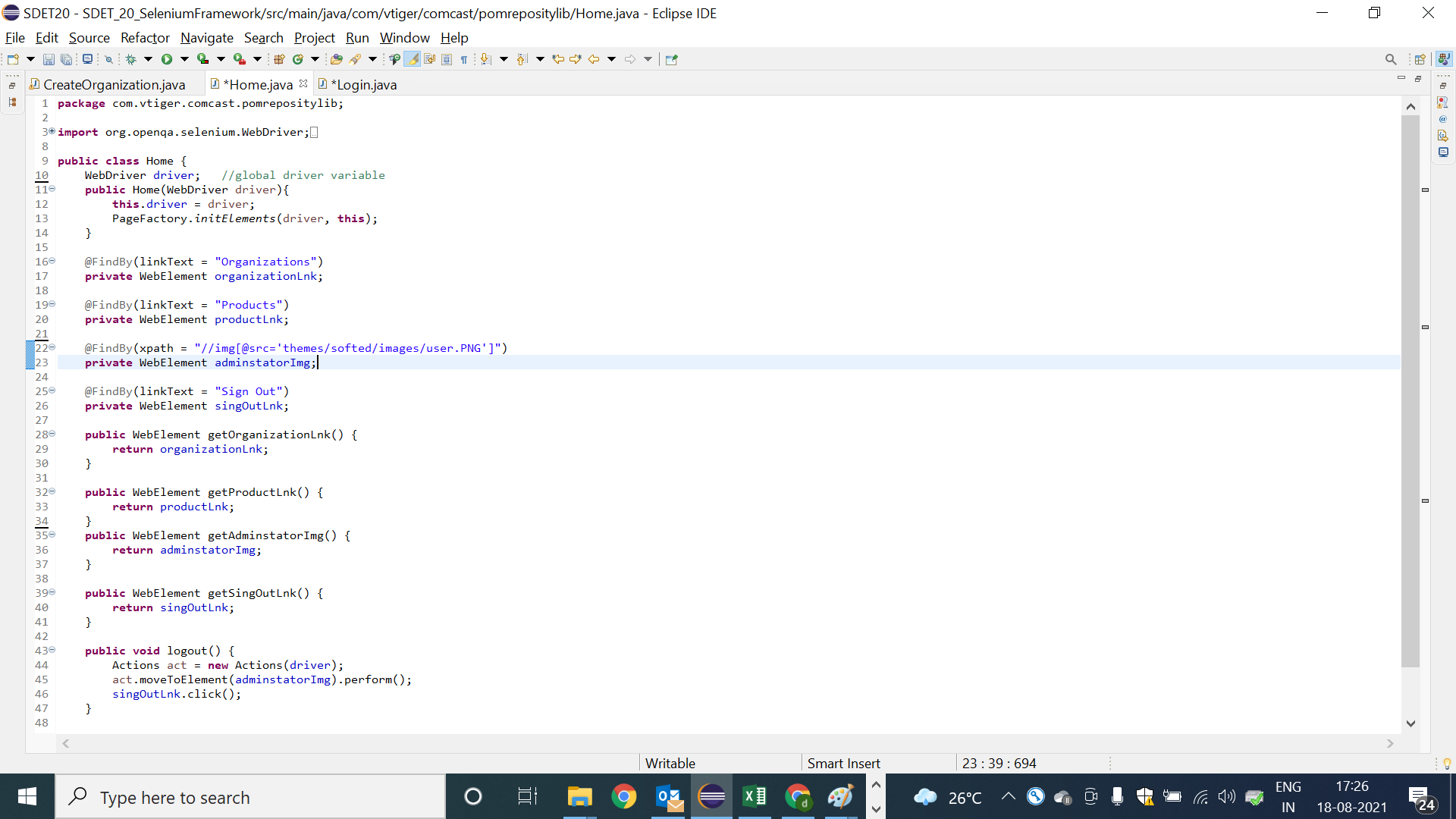
Project Structure



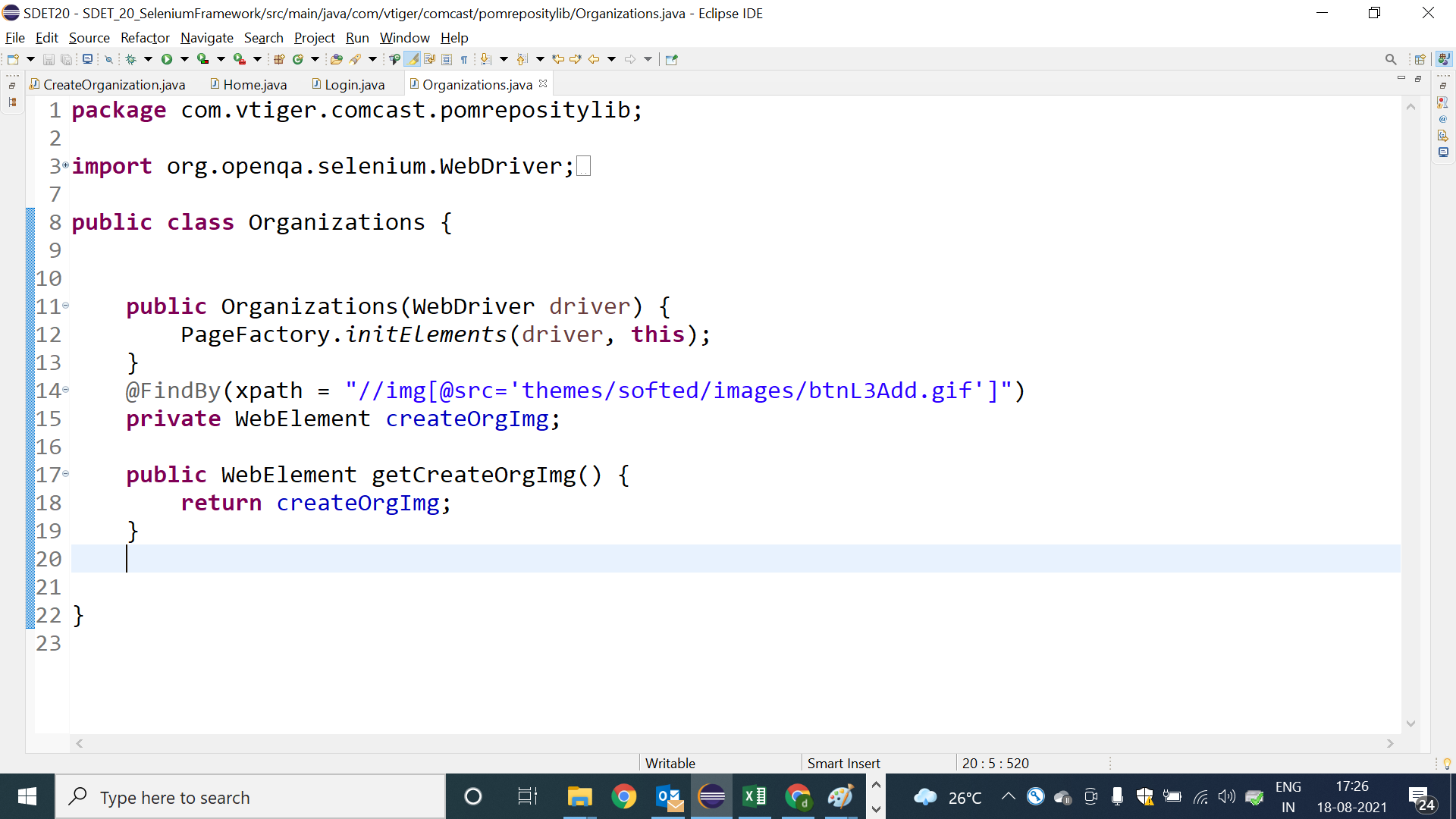
Pom Classes : Login



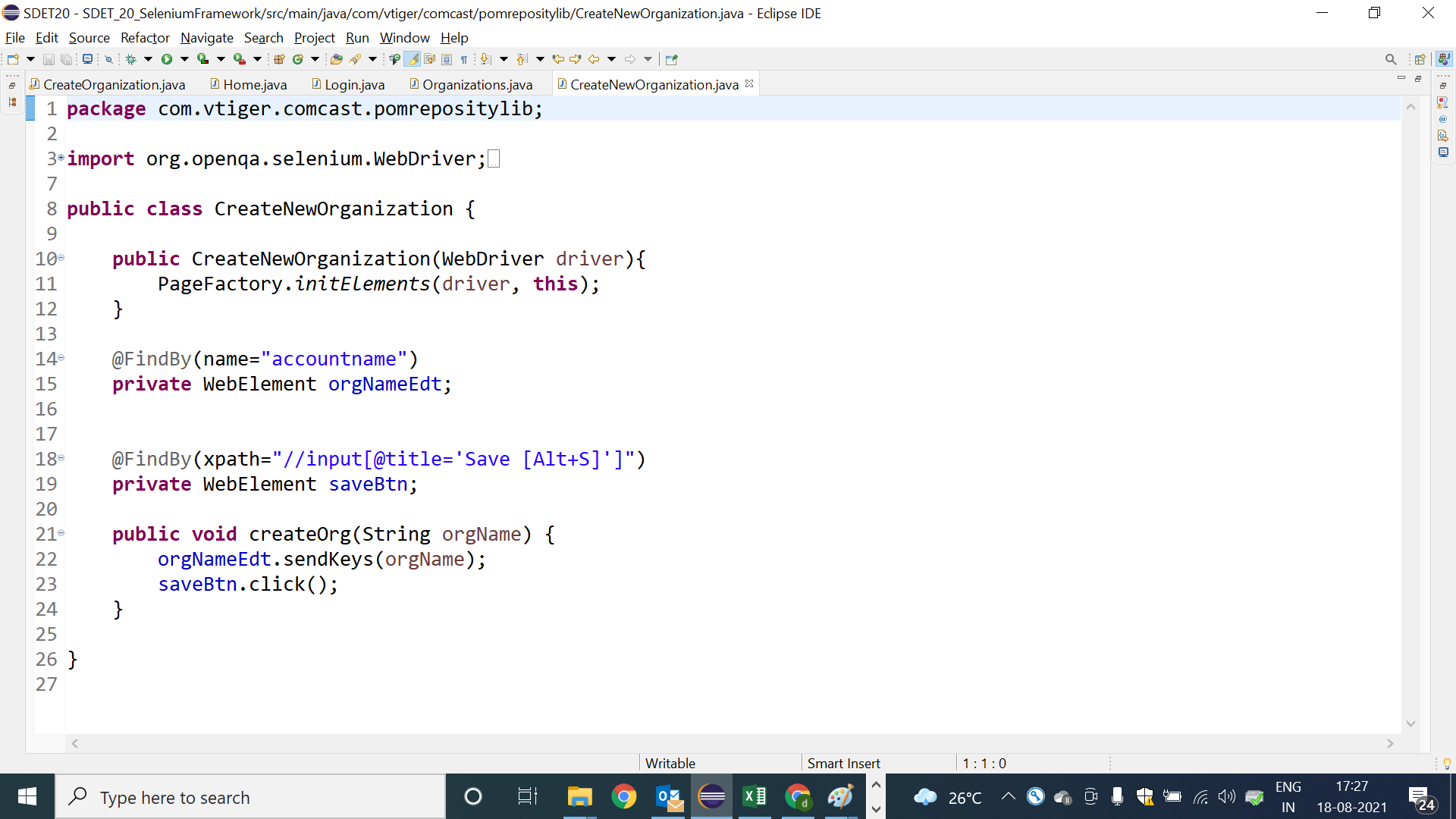
Pom Classes : Home



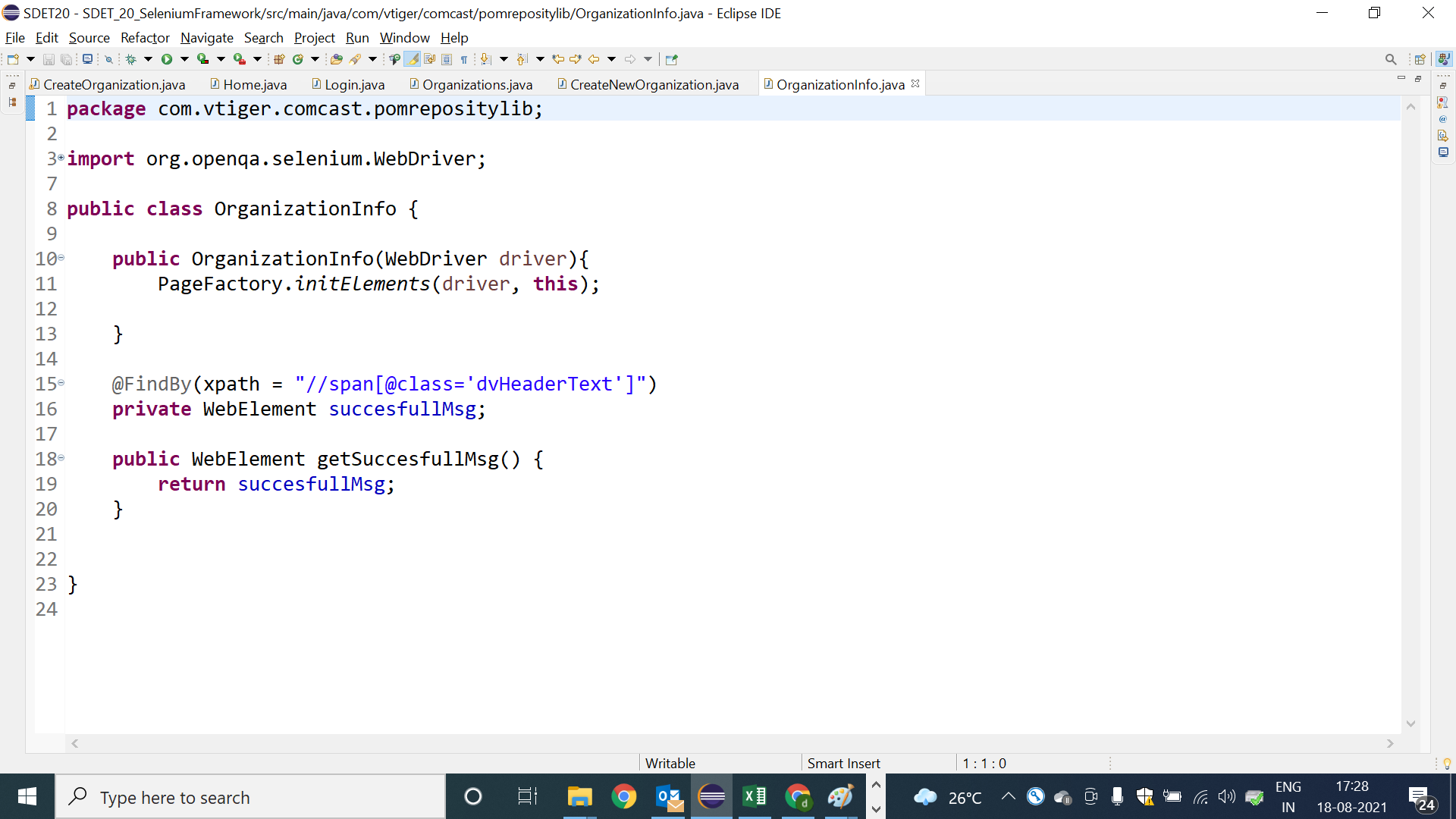
POM class : Organization



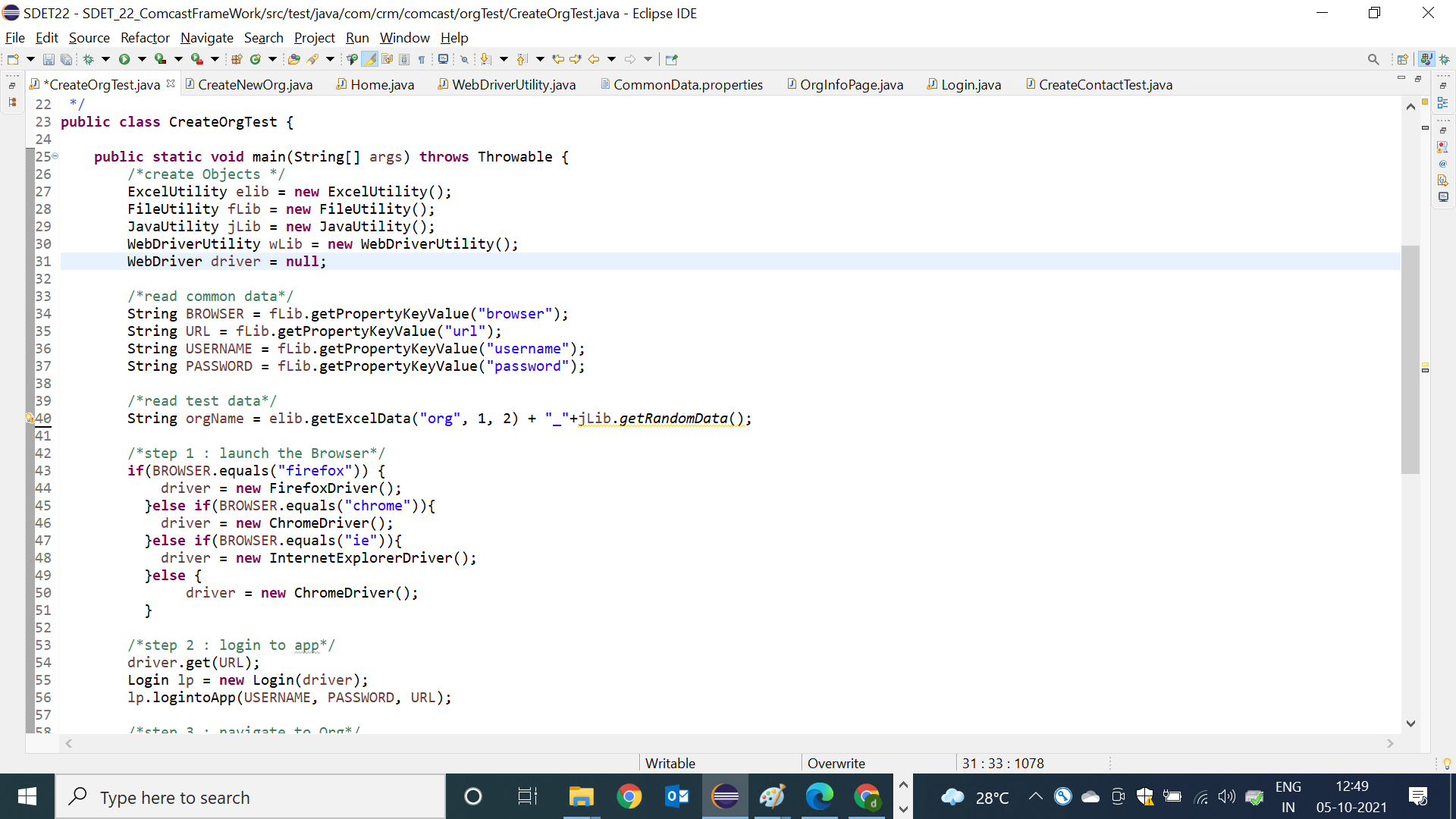
POM class : Create new Organization Page

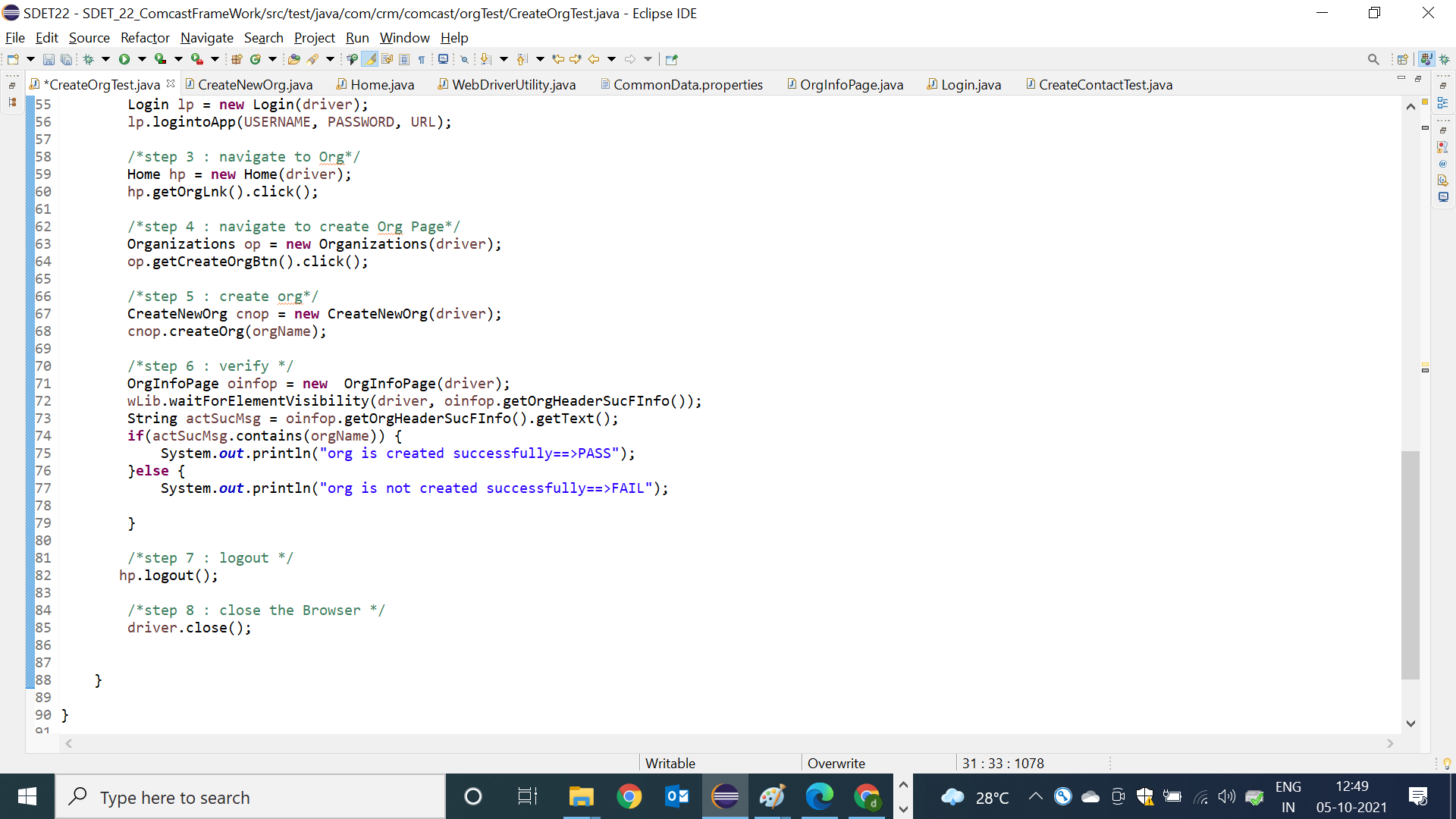


POm class : OrganizationInfo

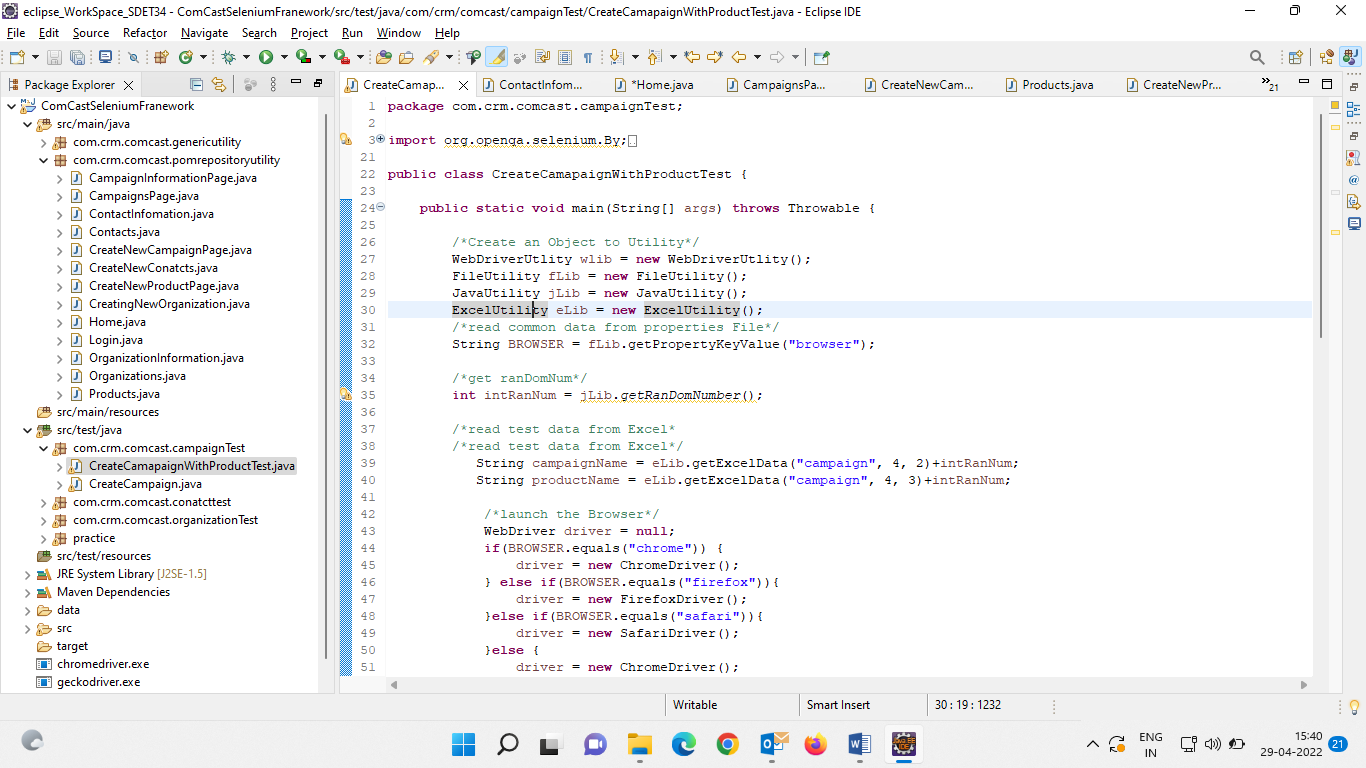


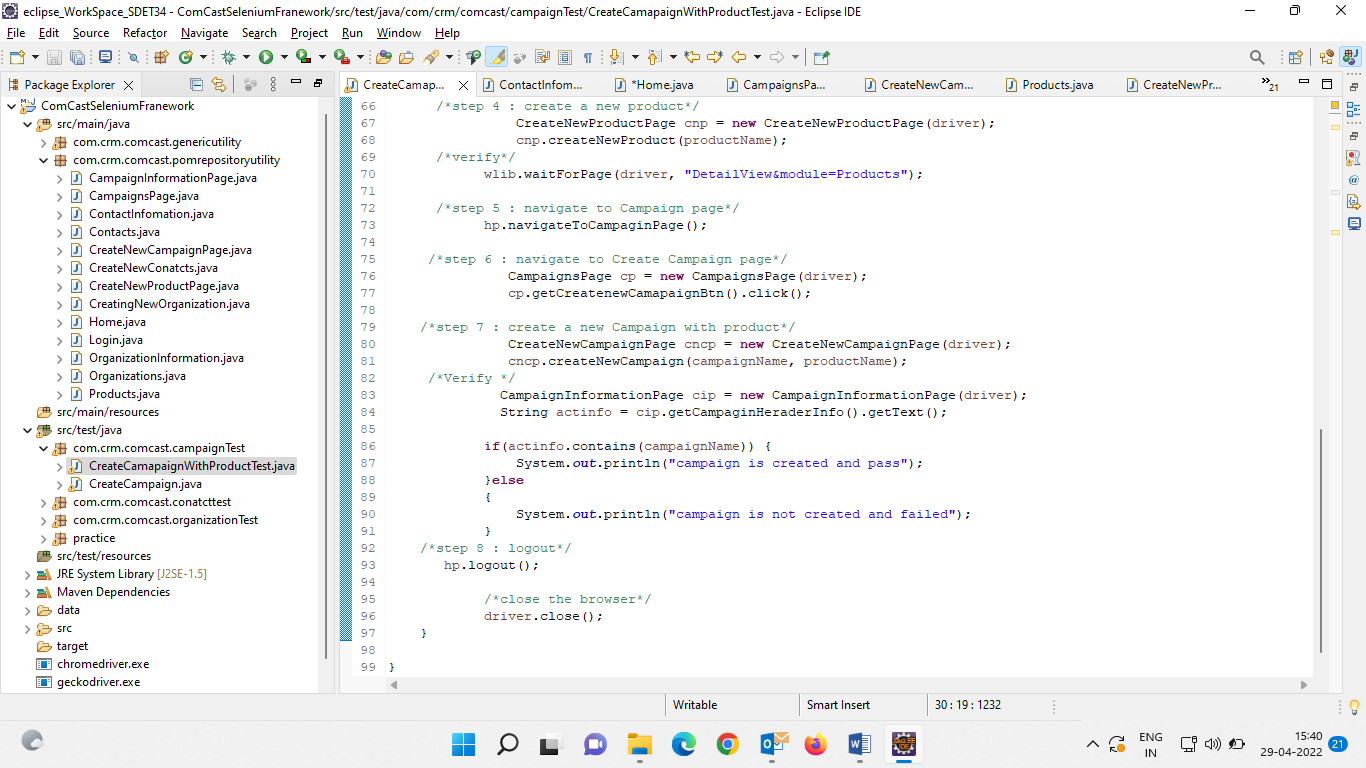
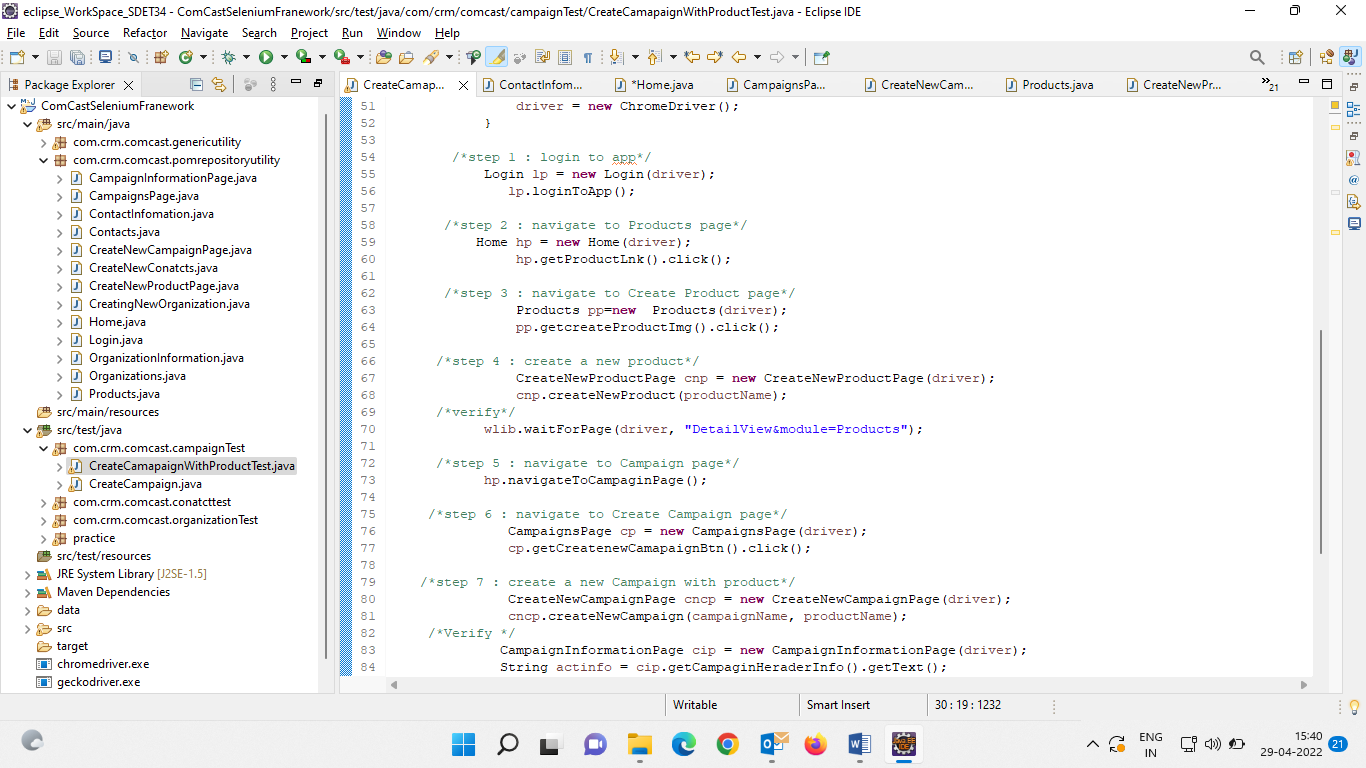
Test Scripts using POM class





===============createCampagin with Product [integration test case]========





================================Pom class==================

package com.crm.comcast.pomrepositoryutility;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

import com.crm.comcast.genericutility.WebDriverUtlity;

/\*\*

\*

\* @author Deepak

\*

\*/

public class Home extends WebDriverUtlity{

WebDriver driver;

public Home(WebDriver driver) { //R3

this.driver = driver;

PageFactory.initElements(driver, this);

}

public WebElement getProductLnk() {

return productLnk;

}

@FindBy(linkText = "Contacts")

private WebElement contactLnk;

@FindBy(linkText = "Campaigns")

private WebElement campaginLnk;

@FindBy(xpath = "//img[@src='themes/softed/images/user.PNG']")

private WebElement adminImg;

@FindBy(linkText = "Sign Out")

private WebElement signOutLnk;

@FindBy(linkText = "Organizations")

private WebElement orgLink;

@FindBy(linkText = "More")

private WebElement moreDropDownMenu;

@FindBy(linkText = "Vendors")

private WebElement vendorsLnk;

@FindBy(linkText = "Products")

private WebElement productLnk;

public WebDriver getDriver() {

return driver;

}

public WebElement getOrgLink() {

return orgLink;

}

public WebElement getContactLnk() {

return contactLnk;

}

public WebElement getMoreDropDownMenu() {

return moreDropDownMenu;

}

public WebElement getCampaginLnk() {

return campaginLnk;

}

public WebElement getAdminImg() {

return adminImg;

}

public WebElement getSignOutLnk() {

return signOutLnk;

}

/\*\*

\* used for app logout

\*/

public void logout() {

mouseOverOnElement(driver, adminImg);

signOutLnk.click();

}

public void navigateToCampaginPage() {

mouseOverOnElement(driver, moreDropDownMenu);

waitForElement(driver, campaginLnk);

campaginLnk.click();

}

/\*\*

\*

\*/

public void navigateToVenderPage() {

mouseOverOnElement(driver, moreDropDownMenu);

waitForElement(driver, vendorsLnk);

vendorsLnk.click();

}

}

========================Products=============

package com.crm.comcast.pomrepositoryutility;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class Products {

WebDriver driver;

public Products(WebDriver driver) { //R3

this.driver = driver;

PageFactory.initElements(driver, this);

}

@FindBy(id="search\_txt")

private WebElement searchProductsEdt;

@FindBy(name = "search")

private WebElement searchBtn;

public WebElement getSearchProductsEdt() {

return searchProductsEdt;

}

public WebElement getSearchBtn() {

return searchBtn;

}

@FindBy(xpath = "//img[@title='Create Product...']")

private WebElement createProductImg;

public WebElement getcreateProductImg() {

return createProductImg;

}

}

======================create new Products=================

package com.crm.comcast.pomrepositoryutility;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class CreateNewProductPage {

WebDriver driver;

public CreateNewProductPage(WebDriver driver) { //R3

this.driver = driver;

PageFactory.initElements(driver, this);

}

@FindBy(name = "productname")

private WebElement productNameEdt;

@FindBy(xpath = "//input[@title='Save [Alt+S]']")

private WebElement saveBtn;

public void createNewProduct(String productName) {

productNameEdt.sendKeys(productName);

saveBtn.click();

}

}

=======================Camapgins===================

package com.crm.comcast.pomrepositoryutility;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class CampaignsPage {

WebDriver driver;

public CampaignsPage(WebDriver driver) { //R3

this.driver = driver;

PageFactory.initElements(driver, this);

}

@FindBy(xpath = "//img[@alt='Create Campaign...']")

private WebElement createnewCamapaignBtn;

public WebElement getCreatenewCamapaignBtn() {

return createnewCamapaignBtn;

}

}

========================Create new Camapagin==========

package com.crm.comcast.pomrepositoryutility;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

import com.crm.comcast.genericutility.WebDriverUtlity;

public class CreateNewCampaignPage extends WebDriverUtlity{

WebDriver driver;

public CreateNewCampaignPage(WebDriver driver) { //R3

this.driver = driver;

PageFactory.initElements(driver, this);

}

@FindBy(name = "campaignname")

private WebElement campaignEdt;

@FindBy(xpath = "//input[@title='Save [Alt+S]']")

private WebElement saveBtn;

@FindBy(xpath = "//input[@name='product\_name']/following-sibling::img")

private WebElement productLookUpImg;

public WebElement getCampaignEdt() {

return campaignEdt;

}

public WebElement getSaveBtn() {

return saveBtn;

}

public WebElement getProductLookUpImg() {

return productLookUpImg;

}

/\*\*

\* create a campaign with mandatory field

\* @param campaginName

\*/

public void createNewCampaign(String campaginName) {

campaignEdt.sendKeys(campaginName);

saveBtn.click();

}

/\*\*

\* create a campaign with mandatory field

\* @param campaginName

\*/

public void createNewCampaign(String campaginName, String productName) {

campaignEdt.sendKeys(campaginName);

productLookUpImg.click();

swithToWindow(driver, "Products&action");

Products pp = new Products(driver);

pp.getSearchProductsEdt().sendKeys(productName);

pp.getSearchBtn().click();

driver.findElement(By.xpath("//a[text()='"+productName+"']")).click();

swithToWindow(driver, "Campaigns&action");

saveBtn.click();

}

}

==================================Campagain Information===============

package com.crm.comcast.pomrepositoryutility;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class ContactInfomation {

WebDriver driver;

public ContactInfomation(WebDriver driver) { //R3

this.driver = driver;

PageFactory.initElements(driver, this);

}

@FindBy(className = "dvHeaderText")

private WebElement orgHeaderSucMsg;

@FindBy(id = "mouseArea\_Organization Name")

private WebElement orgNAmeInfo;

public WebElement getOrgNAmeInfo() {

return orgNAmeInfo;

}

public WebElement getOrgHeaderSucMsg() {

return orgHeaderSucMsg;

}

}