Selenium POM (Page Object Model)

- It is a design pattern, used in test automation to create an Object Repository for web page UI elements
- Operations & flows in a particular UI shall be separated from verification

How does it work?

- For each web page in the application, there needs a corresponding Page Class
- Page Class will identify the WebElements of that web page and also contains Page methods which perform operations on those web elements.
- Name of these methods should be given as per the task they perform
- All web elements of the application and the method that operate on these web elements are maintained inside a class file
- A task like verification should be seperated as part of test methods (may be in another class file)

Advantages:

- ✓ Reduces code duplication and improves test maintenance
- ✓ Code will be cleaner & easy to understand
- ✓ Object repository is independent of test cases, so we can use the same object repository for different purposes with different test scenarios.
- ✓ Code becomes less & optimized because of reusable page methods in POM classes

Example:

PageClass 1

```
package selenium_Week3_POM;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

public class POMClass {

    WebDriver driver;

    By email = By.id("email");
    By pwd = By.id("pass");
    By login = By.name("login");
```

```
    public POMClass(WebDriver driver){
        this.driver = driver;
    }

    public void setEmail(String emailId){

    driver.findElement(email).sendKeys(emailId);
    }

    public void setPwd(String password){
        driver.findElement(pwd).sendKeys(password);
    }

    public void clickLogin(){
        driver.findElement(login).click();
    }

    public void userLogin(String emailId, String
pass){
        this.setEmail(emailId);
        this.setPwd(pass);
        this.clickLogin();
    }
}
```

PageClass 2

```
package selenium_Week3_POM;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

public class RespPage {

    WebDriver driver;
```

```
    By forgotPwd = By.linkText("Forgotten
password?");

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

    public void clickForgotPwd(){
        driver.findElement(forgotPwd).click();
    }
}
```

Main Class:

```
package selenium_Week3_POM;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class FacebookValidation {

    public static void main(String[] args) throws
InterruptedException {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.facebook.com/");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
```

```
        POMClass pc = new POMClass(driver);
        pc.userLogin("abc@gmail.com", "abc1");

        Thread.sleep(3000);
        RespPage rp = new RespPage(driver);
        rp.clickForgotPwd();
    }

}
```

Page Factory:

- ✓ It is an inbuilt Page Object Model framework in Selenium WebDriver
- ✓ It is much optimized
- ✓ It is used for initialization of Page objects to instantiate the Page Object itself
- ✓ It is also used to initialize Page class elements without using "FindElement/s"
- ✓ With the help of class PageFactory, we use annotations @FindBy to find a web element
- ✓ We use intiElements method to initialize web elements

Example:

PageClass 1

```
package selenium_Week3_PageFactory;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class LoginPage {

    WebDriver driver;

    @FindBy(id="email")
    WebElement email;

    @FindBy(id="pass")
    WebElement pwd;
```

```
    @FindBy(name="login")
    WebElement login;

    public LoginPage(WebDriver driver){
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public void setEmail(String emailId){
        email.sendKeys(emailId);
    }

    public void setPwd(String password){
        pwd.sendKeys(password);
    }

    public void clickLogin(){
        login.click();
    }

    public void userLogin(String emailId, String
pass){
        this.setEmail(emailId);
        this.setPwd(pass);
        this.clickLogin();
    }
}
```

PageClass 2

```
package selenium_Week3_PageFactory;

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;

public class RespPage {

    WebDriver driver;

    @FindBy(linkText="Forgotten password?")
    WebElement forgotPwd;

    public RespPage(WebDriver driver){
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public void clickForgotPwd(){
        forgotPwd.click();
    }
}
```

Main Class:

```
package selenium_Week3_PageFactory;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

import selenium_Week3_POM.RespPage;

public class PageFactoryClass {

    public static void main(String[] args) throws
InterruptedException {


    System.setProperty("webdriver.chrome.driver",
```

```
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\Drivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.facebook.com/");

    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

        LoginPage lp = new LoginPage(driver);
        lp.userLogin("abcd@gmail.com", "abcd1");

        Thread.sleep(3000);
        RespPage rp = new RespPage(driver);
        rp.clickForgotPwd();
    }

}
```

Exercise:

1. Create an Object Repository for RoboForm website; Access few elements in a method, few other in another methods, etc.
2. Apply PageFactory in DemoQA website to create different Object Repository for different pages & action on the same.