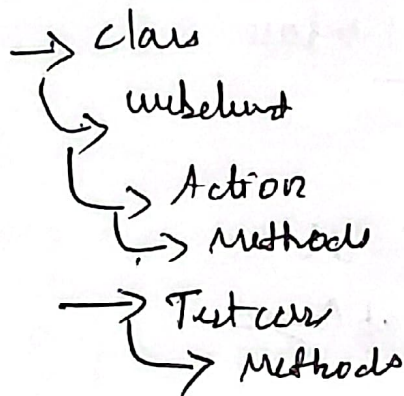


Day 17

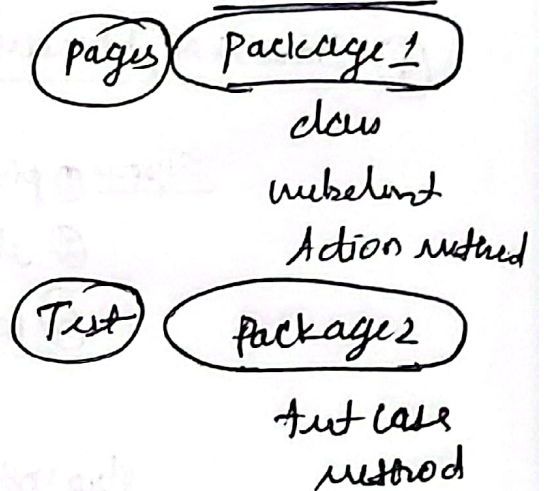
## POM

- Page object Model
- obj repository. for storing web elements
- design pattern
- Reduce code duplication
- test case maintenance

### ① Non-POM layout



### ② POM layout



eg: Facebook.com

Find user Obj Repository Fblogin

1. spec email/pass/login store

{  
By fbemail = By.id("email");  
By fbpass = By.id("pass");  
By fblogin = By.name("login");  
}

value Pass/Method

```
public Page void setvalue (String email,  
pass)
```

```
driver.findElement(fbemail).sendKeys(email);  
driver.findElement(fbpass).sendKeys(pass);
```

public void login()

{  
    login.findElm(fblogin).click();  
}

run test class (fbloginTest)

① Test

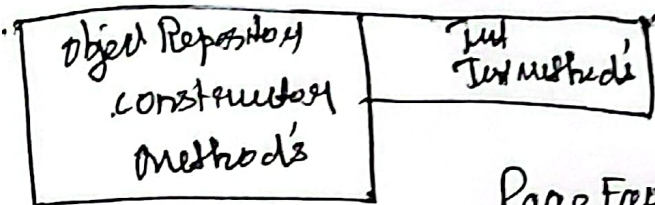
FbloginPage ob = new FbloginPage(driver);

ob.setValue("@gmail.com", "aswini");

ob.login();

}

constructor  
public FbloginPage  
(WebDriver driver)  
{  
    // constructor code  
}



Page Factory :

→ Page obj design pattern supports driver

① → @FindBy

② → initialize (for initializing)

static method

@FindBy annotation value  
initialize

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

Why WIPOM?

- Elements changes dynamically
- Complex UI
- Simple UI

use when you want  
full control over  
element handling

Why Page Factory?

- code readability is priority
- performance optimization is needed.
- Simple UI

use when you want  
clean code of  
lazy initialization