Day17

POM

-> Page object Model
-> obj repository . for storing webelements
-> design pattern
-> Reduce code duplication
-> test case maintanence

① Non-POM layout
-> class
  -> webelement
    -> Action
    -> Methods
-> Testcase
  -> Methods

② POM layout
pages — Package 1
class
webelement
Action method

Test — package2
test case
method

eg: Facebook.com

Firstclass Obj Repository Fblogin.

Inspect email/pass/Login store

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

value Pass/Method()

```
public ~~Fblog~~, void setvalue( string email,
{                                    pw)

driver.findElement (fbemail). sendkey(.email);
driver. findElement (fbPasswor). sendke (pw);
```

```
public void login()
 {
  driver.findEle(fblogin).click();
```

constructor

```
public Fbloginpage
  (WebDriver dri)
{
this.driver=driver
}.
```

next class (Fblogintest)

```
@Test

Fbloginpage ob= new Fbloginpage(driver);
  ob.setvalue("abc@gmail.com", "asun");
  ob.login();

}.
```

Object Repository
constructor
method's

Test
Test method's

Page Factory:

→ Page obj design pattern supports class

① → @Find by
② → initelements (for initializing)
   static method
   @Find by annotation value
   initialize

@FindBy (id ="elementID") webelement element;

Why use POM?

- Elements changes dynamically
- Complex UI
- Simple UI

[use when you want full control over element handling]

Why PageFactory?

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

[use when you want clean code of lazy initialization]