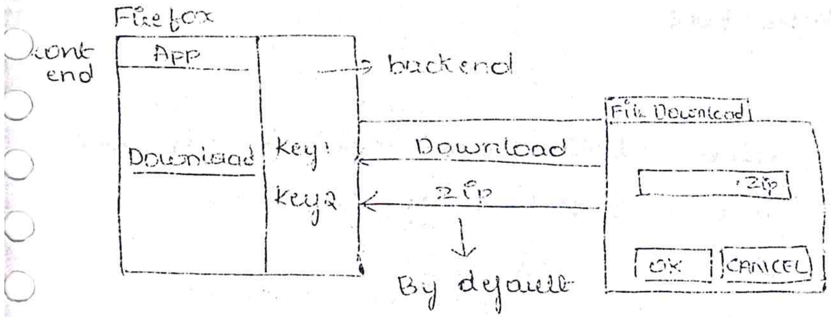


Browser stores as this way

① where to save
② what type of file

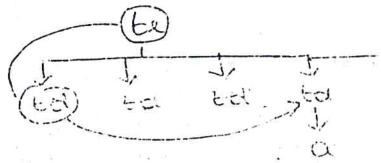
Instead of performing action, webdriver handles the pop up → by first we configure the key values for the back end such as where to save, what type of file and pass this configuration to driver before launching a browser.

To find key & its value → on firefox url type about : config, this shows the back end of browser.

Program

```
public class Download File
{
 public static void main(String[] args).
 {
 // using setpreference(mtd) change the browser
 default setting
 Firefox Profile p = new Firefox profile( );
 // Download the file in Desktop
 p. setPreference (" browser.download folderList", 0);
 // Provide the type of file which is downloading
 ("browser.helperApps.neverAsk.saveToDisk", "application/zip");
 // pass profile object to enable setting for the browser
 WebDriver driver = new FirefoxDriver(p);
 driver.get(" http //docs.seleniumhq.org /download/");
 String xDown = "//td[text() = 'Java']/following-sibling :: td[3]/a";
 driver.findElement (By.xpath(xDown)).click();
 }
}
```

→ save in folder




```
public class DownloadFile
{
 public static void main(String[] args)
 {
 Firefox Profile p = new FirefoxProfile (),
 p. setPreference (" browser. download. folderList",
 2);
 p. setPreference (" browser. helper Apps. never Ask.
 save ToDisk", "application/zip");
 p. setPreference (" browser. download. dir" "C:\\
 users \\ DEEPU \\ Desktop \\ new \\ chromefiles .exe");
 WebDriver driver = new FirefoxDriver(P);
 driver. get (" http // docs. seleniumhq. org /
 download /");
```

String xpath = "//td[text()='Java']/following-sibling::td[3]/a";

driver.findElement(By.xpath(xpath)).click();

}

}

→ write a code to handle SSL (Secured socket layer)

```
public class HandleSSL
{
    public static void main(String[] args)
    {
        FirefoxProfile p = new FirefoxProfile();
        p.setAssumeUntrustedCertificateIssuer(false);
        WebDriver driver = new FirefoxDriver(p);
        driver.get("https://199.184.60.79/");
    }
}
```

NOTE:

* FirefoxProfile is a advanced webdriver class which is used to handle browser native popup in firefox browser only
* In order to change the default configuration of the firefox browser, go for setpreference method available in firefox profile
* To handle SSL certification popup in Firefox browser go for setAssumeUntrustedCertificateIssuer()
* In order to Enable plug-in for webdriver launched browser go for addExtension()
* To view browser backend keys in Firefox browser, go to firefox url navigation path, type about:config & press Enter

Keys	Values
browser.download.folderList	1 - Download 0 - Desktop 2 - local drive path
browser.helperApps.neverAsk.saveToDisk	Mime Type .zip = application/zip .png = image/png .xml = application/xml .csv = application/vnd.ms
browser.download.dir	local drive path

Test NG [unit testing framework tool]

NG → Next generation

unit testing framework tool like (according to specific language)

- → Junit → Java
- → Nunit → .net
- → testNG → Java, .net (most popular tool)
- → RSpec → Ruby
- → pydac → python

u these unit testing framework tool work as a plug-in for Eclipse (leaving .net)

Visual Studio

↳ Selenium + .net + Nunit

Test Next Generation.

* It is a unit testing frame work tool which is implemented as Plug-In for the Eclipse. The Test NG is a open source plugin available in test NG community.

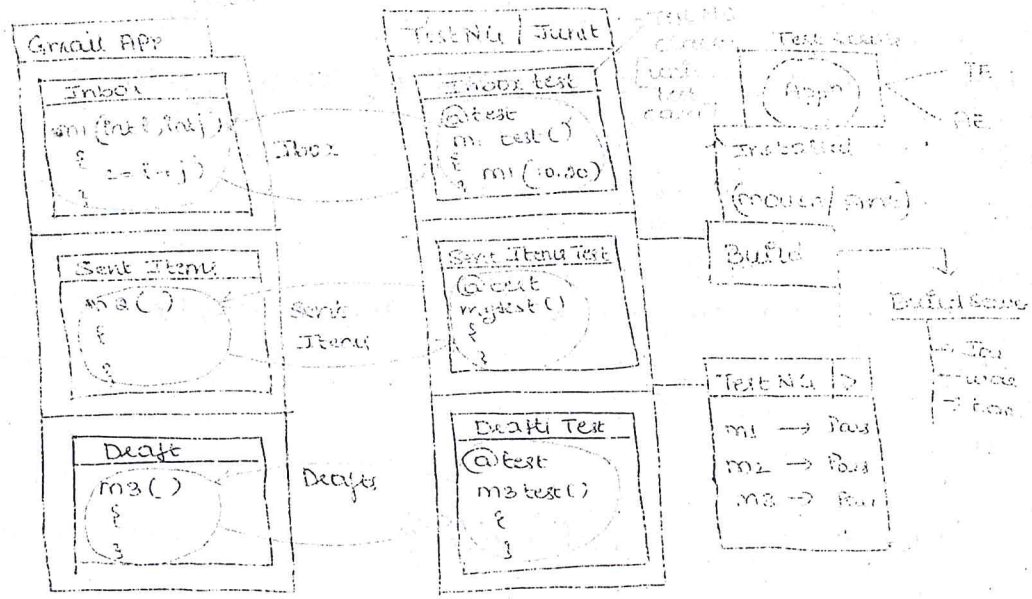
Junit is the default Plug-in for the Eclipse, no need to install if we want to use Junit feature

Developer Usage of Test NG

In testNG there is no main method like Java, so in testNG we use @test

Once coding is done, white box testing has to be done so they write unit test cases for every line in TestNG

Unit test case can be written by Java Developer White box Testing, API test Engg



* Incase of development Test NG will be used to implement unit cases using Test NG classes, each unit test cases will go & test the business logic of the source code

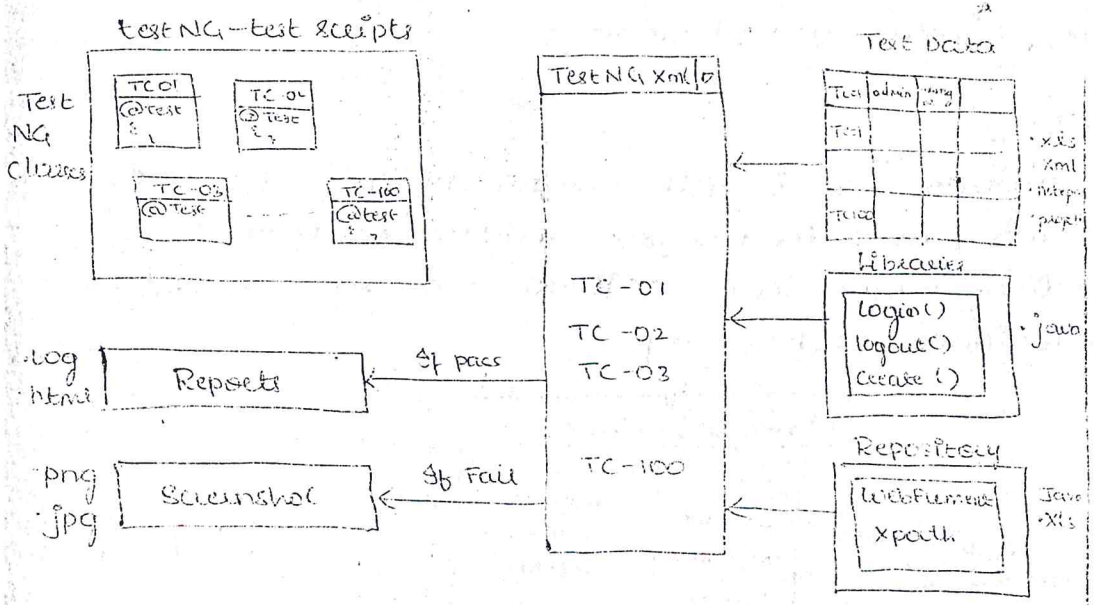
* Unit case are always written by Java developer/ white box test Engg (SE) API test Engg.

* whenever entire development is done, developer will write unit test cases to check / test the issues in the source code

* whenever unit testing is done, Build engg will convert source code to executable build using Ant / maven tool (Build testing tool)

Test NG usage in Automation

① using java classes we cannot do batch execution so automation Engg write test cases using Test NG to do batch execution



NOTE

* In case of Automation, Test NG will be used to handle frame work component & achieve batch execution without any manual interaction
* In order to achieve batch execution all the test case should be implemented using TestNG classes
* Batch Execution can be done Xml file
* Test NG is inspired from Junit & Nunit but introducing some new functionalist that makes Test NG is more powerful tool.
* New functionality are → parallel Execution
  → Group Execution
  → parameterization
  → html reports
  → Batch Execution is easy

Installation Steps of Test NG.

Step1 :- click on Help option in Eclipse window

Step2 :- Click on Eclipse market place.

Step3 :- Write testng in Find editbox and click on Go button

Step4 :- Click on Install button for TestNG for Eclipse option

Step5 :- Click on Next

Step6 : click on first radio button (accept license agreement) and click on finish.

Step7 : If we get warning msg click on Yes & then restart Eclipse

* In order to cross verify TestNG installation, in Eclipse click on window
  ↓
  Show view
  ↓
  Others
  ↓
  Expand Java folder
  ↓
  we should able to see TestNG option

NOTE

* writing Test NG cases typically uses 5 steps

Step1 : create test NG class

Step2 create testNG test within a class

Step3 : Execute testNG's test

Step4 : Verify testNG report

Verify Eclipse console report.

Step1 :

Create test NG class

Select package right click & select testNG & click on create test NG class

Step2 :

Create testNG test within a class

```
class SampleTest
{
  @Test
  public void CustomNameTest()
  {
      Java code
      WebDriver code
  }
}
```

• Java

Instead of main method as in java we can write only custom(prefix) Name (for main) in TestNG @Test

Step3 : Execute TestNG test

Select the testNG class right click and, select Run as and click on TestNG class test

Only one main method execute in java but TestNG can execute more

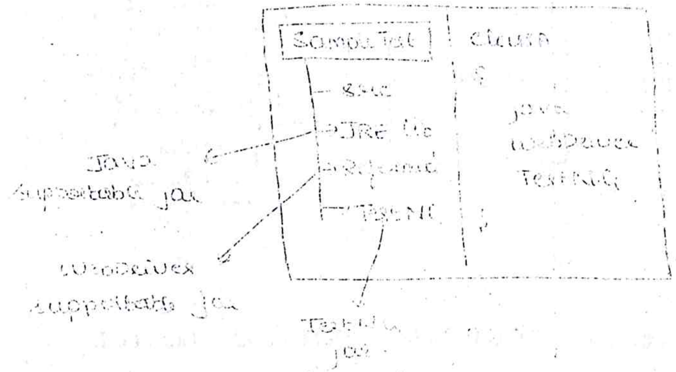
Step4 : Verify html report

→ After the test execution select project & Press F5 (refresh), so after the refresh automatically we get test-output in the same project folder.

→ Select emailable-report.html → right click & ... open with web browser.

NOTE

* whenever we install TestNG plug-in for the eclipse, in order to use TestNG feature in Eclipse make sure TestNG libraries should be available for the project



→ How to add TestNG jar?

In configuration Build path window click on Add Library instead of Add External jar button

Select TestNG & click on Next & finish & then click on ok

Available Annotation in TestNG →

Annotation is a metadata which provides instruction to compiler in run time execution.

Annotation → it is interface.

@Test
@BeforeMethod
@AfterMethod
@BeforeClass
@AfterClass
@BeforeGroup
@AfterGroup
@BeforeSuite
@AfterSuite

→ TestNG always execute based on alphabetical order.

@Test

* Whenever we execute TestNG class, testNG compiler always look for @Test annotation method to start the execution.
* In one TestNG class, multiple @Test annotation methods / cases are allowed but each method should have @Test annotation before the method signature.
* TestNG class Name & Method Name should be end with test string as per the coding standard.
* In TestNG class we go for @Test method instead of public static main method.
* TestNG compiler always execute the test based on method name alphabetical order.
* @Test method is used to implement actual automation test case, each annotation dedicated for single test.

@BeforeMethod & @AfterMethod

Syntax:

```
@BeforeMethod
public void configBeforeMethod()
{
    ≡
}
@Test
@AfterMethod
public void configAfterMethod()
{
    ≡
}
```

Eg:

```
public class NewTest
{
    @Test
    public void createCustomerTest()
    {
      System.out.println("execute CreateCustomerTest");
    }
```

```
@BeforeMethod
public void configBeforeMethod()
{
System.out.println("execute config before
                    method");
}

@AfterMethod
public void configAfterMethod()
{
System.out.println("execute config After
                    method");
}
```

O/P

execute config before method
execute create customer Test
execute config After method

Realtime Example

For AccuTime,

project & customer

- Create customer
  1. Launch Browser
  2. Login to app
  3. Create customer
  4. Verify customer
  5. Logout
  6. Close browser
- delete
- modify customer
  1. Launch Browser
  2. Login to App
  3. Create customer
  4. modify & verify customer
  5. Logout
  6. close browser

Program

```
class projectAndCustomerTest{

@BeforeMethod
public void configBeforeMtd()
{
System.out.println("Launch Browser & Login");
}

@Test
public void createCustomerTest()
{
System.out.println("create customer & verify
                    customer");
}

@Test
public void modifyCustomerTest()
{
System.out.println("create, modify customer &
                    verify customer");
}

@AfterMethod
```

```
public void configAfterMtd()
{
  System.out.println("Logout & close browser");
}
```

Output

Launch browser & login

Create customer & verify customer

Logout & close browser

Launch browser & login

Create, Modify customer & verify customer

Logout & close browser

90-95 ⟶ Selenium + java 90%

↓

open source

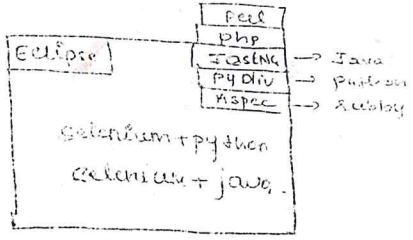
platform independent

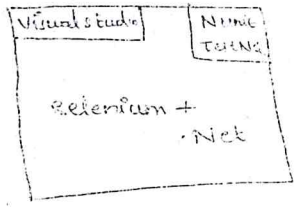
Selenium + .net 5%

python ⟶ 10 times faster than Java.

e.g. YouTube [Online Application]

Selenium + python 5%





Eclipse does not support .net.

NOTE

* Before Annotation method will be executed before executing each test method in a class
* After Annotation method will be executed after executing each test method in a class.
* In order to insert similar pre-condition & post-condition for multiple test cases we go for before & after Annotation method.
* Before and After annotation method will be used only for test configuration, which cannot be executed without @Test method.

@Before class and @After class

This will executed only once in a entire class execution.

Uses ⟶ ① Launching & closing browser

② Initialize object / variable.

```
e.g. class Project
{
  @Test
  {
    int i=10;
local variable. WebDriver d = new FirefoxDriver()
  }
```

once PD
local variable
in First Test
we cannot
use in
other Test

```
@Test
{
    Sop(i);   X
}
}
```

So by using before class, we can initialize an object / variable.

```
Class Project
{
  int i;
  WebDriver d;
  @BeforeClass
  {
    i=10;
    d= new FirefoxDriver();
  }
  @Test
  {
    Sop(i);
  }
  @Test
  {
    Sop(i);
  }
}
```

don't create dependency between Test cases
(don't create one login create, modify & delete)
always freshly login navigate perform & logout

② DB related Test cases.

Project for Customer

Create Customer	Modify Customer
① Launch Browser	① Launch Browser
② Login	② Login
③ Create customer	③ Create customer
④ Verify customer	④ modify & verify
⑤ Logout	⑤ Logout
⑥ Close browser	⑥ Close browser

Program

```
public class ProjectAndCustomerTest
{
  @BeforeClass
  public void configBeforeClass()
  {
    System.out.println("=== LaunchBrowser, object/
      variable initialization, connect to db
      ===");
  }
  @BeforeMethod
  public void configBeforeMtd()
  {
    System.out.println("login");
  }
  @Test
```

```
public void createCustomerTest()
{
  System.out.println("create customer & verify
                      customer");
}

@Test
public void modifyCustomerTest()
{
  System.out.println("create, modify customer
      & verify customer");
}

@AfterMethod
public void configAfterMtd()
{
  System.out.println("logout");
}

@AfterClass
public void configAfterClass()
{
  System.out.println("=== close browser, disconnect
                      DB ===");
}
}
```

Output

===Launch Browser, object/variable intialization, connect to db ===

Login

Create customer & verify customer

Logout

Login

create, modify customer & verify customer

Logout

=== close browser, disconnect DB ===

---> There can be multiple BeforeMethod & AfterMethod but the code looks complex Hence we can use one BeforeMethod.

Eg

```
@BM
{
  Enter username
  Enter password
}
```

```
@BM
{
  Enter username
  Enter password
}
```

---> Each & every test case should be unique (dependency shld nt exist)

```
class Sample {
  @BC
  {
    Launch & Login
    create HDFC Customer
```

```
@Test
{
  Verify HDFC customer
}
@Test
{
  modify to Airtel, Verify
}
@Test
{
  delete
}
@AC
{
  Logout
}
```

NOTE

Before class annotation method will be executed before execution the first @Test method in a class

After class annotation method will be executed after executing all the @Test method in a class.

BeforeClass & AfterClass Method will be used for Global configuration (Launch, object intialization, DB) which will be executed only once in a entire class execution.

As per the automation rule each test case should be unique (test case should not have dependancy between one to other test) → whereas

Verify mail in Sent Item

→ login to Gmail
→ compose mail
→ Send mail
→ navigate sent item page
→ Verify sent msg
→ Logout

Verify Send msg

→ Login Gmail
→ Send mail
→ Verify mail
→ Logout

End to End Scenario

① Login with A
② compose
③ Send mail
④ Logout
⑤ Login with B
⑥ check my Inbox
⑦ delete msg
⑧ Logout

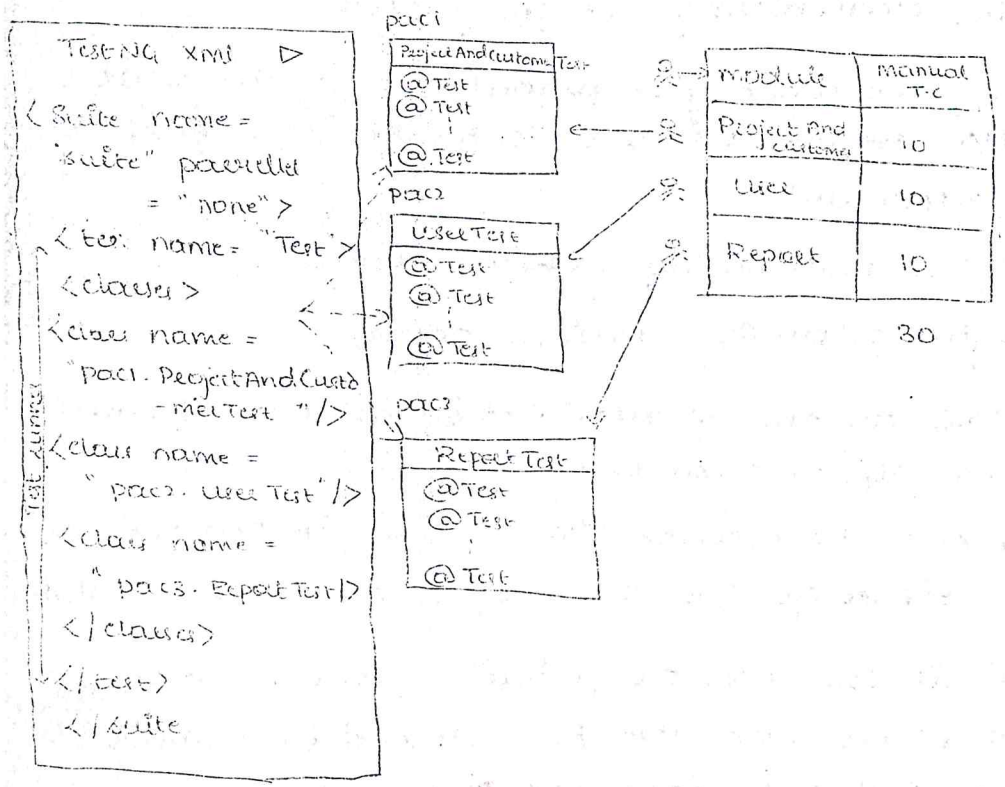
Batch Execution

How to create TestNG xml file automatically through Eclipse ?

→ Select all the ~~package~~ package, right click and select TestNG option & click on convert to TestNG & then click on finish

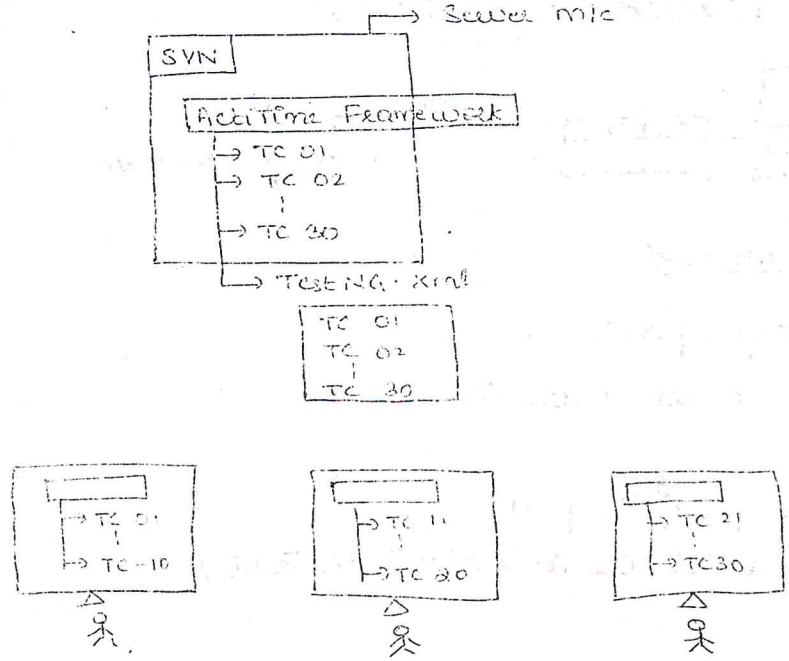
→ Automatically then xml file will be created in the name project

→ whenever we double click / open xml file we might see xml in table format, to open source code click on source button.



Note

* Collection of multiple testcase is called Test suite, execute multiple testcase through TestNG xml file is called Batch Execution.
* In order to achieve batch execution, all the testcase should be implemented using TestNG classes and TestNG Annotations.
* In TestNG Batch execution can be done by xml file & xml file always starts with suite test & classes xml tags
* One TestNG suite can have multiple Test Runners.
* In order to execute n - no of TestNG classes, specify the TestNG class Name along with package name in class xml Tag.
* In one xml file we can invoke n no of testNG classes but class should be within a project.



* If we want to preserve order of packages in TestNG xml file.

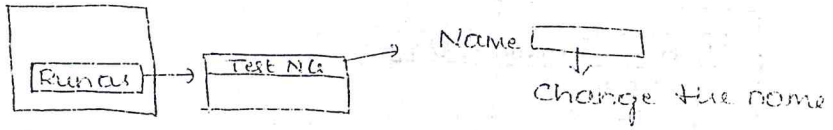
<Suite name = "suite" parallel = "none"
preserve order = "true" >

Suite xml tag contains name & parallel mandatory attributes and few additional/ optional attributes like preserve Order and thread count

preserve order :- using preserve order attribute we can execute all the testcases in order which choose our specified way in xml file

* In order to achieve the Batch execution, Select TestNG file → right click → Run as → TestNG suite.

* We can create more than one TestNG class but name should be different.



Dependent Methods

```
:- package pac;
   public class OrderTest
   {
      @Test (priority = 1)
      public void createSalesOrderTest()
      {
         System.out.println("login, create so, verify, logout");
      }
      @Test (priority = 2)
      public void createBillingTest()
      {
         System.out.println("login, create billing, verify, logout");
      }
   }
```

Output

login, create so, verify; logout
login, creat billing, verify, logout

→ Suppose we want give priority as @Test methods TestNG executes based on alphabetical order. so output will be,

login, create billing, verify, logout
login, create so, verify, logout

* TestNG always executes the @Test method based on alphabetical order
* In order to execute the test case in our specified order either go for priority (or) Dependon Methods
* "0" is the topmost priority for the Test
* -ve values can also be allowed in priority then that will have the highest priority

-20 ⟶ Highest priority.
-10
0
+10
+20

→ By dependsOnMethods used,

@Test(dependsOnMethods = "createSalesOrderTest")
first it executes createSalesOrderTest after that it executes createBillingTest methods

→
```
@Test
public void createSalesOrderTest(){
    Sop("login, createSO, verify, logout");
}
@Test(dependsOnMethods = "createSalesOrderTest")
public void createBillingTest(){
    Sop("login, createbilling, verify, logout");
}
```

* How to disable the test case in TestNG class?

→
```
@Test(enabled = "false")
public void createSalesOrderTest(){
    Sop("login, createSO, verify, logout");
}
@Test
public void createBillingTest(){
    Sop("login, createbilling, verify, logout");
}
```

* How to execute same Testcase multiple times

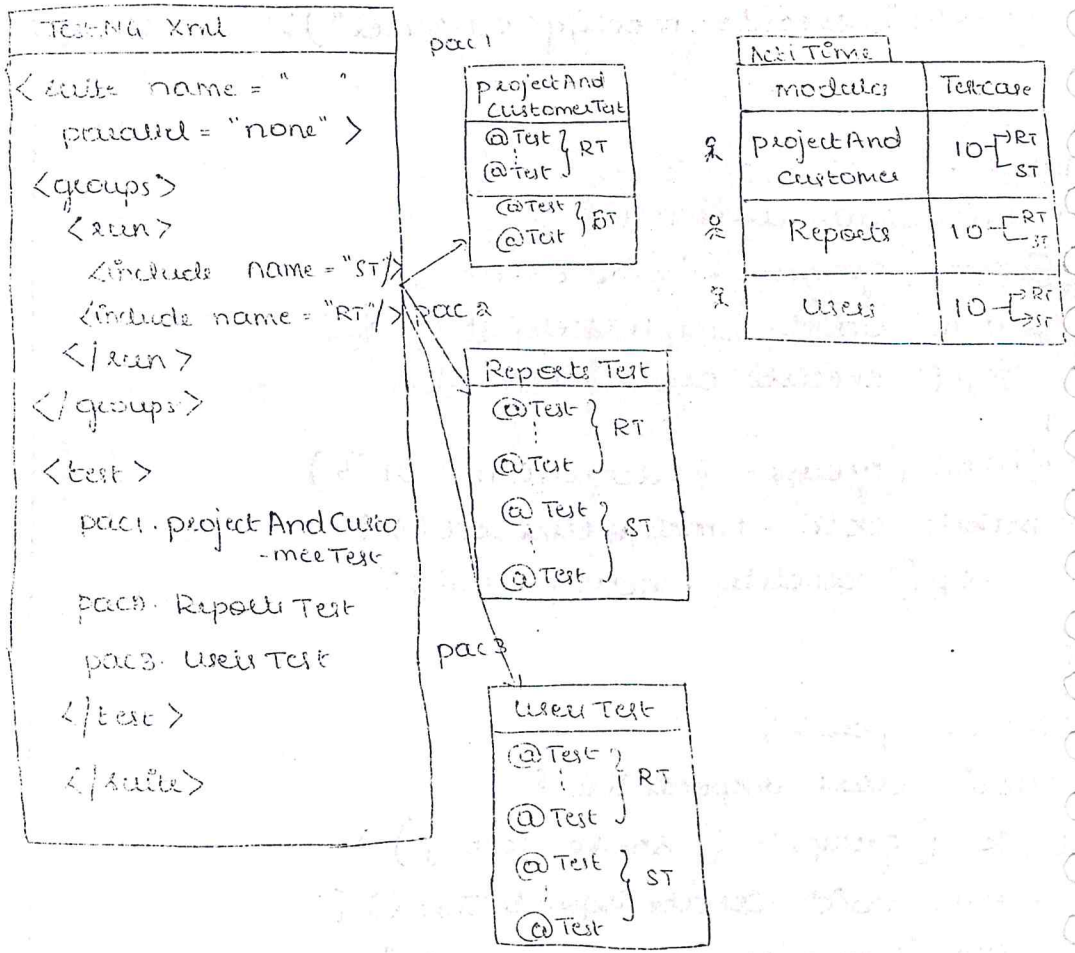
→
```
@Test(invocationCount = 10)
public void createSalesOrderTest(){
    Sop("login, createSO, verify, logout");
}
```

2

```
@Test
public void createBillingTest(){
    Sop("login, createBilling, verify, logout");
}
```

Grouping Execution

+ collection of similar Test cases
+ One Test case can have multiple groupname.



RT ⟶ Regression Test

ST ⟶ Smoke Test

Program

```
package pac1;
public class projectAndCustomerTest {
@Test ( groups = {" smoke Test "} )
public void createCustomerTest () {
    Sop (" execute create customer");
}
@Test ( groups = {" Regression Test "} )
public void modifyCustomerTest () {
    Sop (" execute modify customer");
}
```

```
package pac2;
public class UserTest {
@Test ( groups = {" smoke Test "} )
public void createUserTest () {
  Sop (" execute create user");
}
@Test (groups = {" regression Test "} )
public void modifyUserTest () {
  Sop (" execute modify User");
}
}
```

```
package pac3;
public class RepeatsTest {
@Test ( groups = {" smoke Test "} )
public void createRepeatsTest () {
  Sop (" execute create repeats");
}
@Test ( groups = {" regression Test "} )
public void modifyRepeatsTest () {
  Sop (" execute modify repeats");
}
}
```

TESTNG xml

```
<suite name = " suite " parallel = " none " >
  <groups>
    <run>
      <include name = smoke Test >
    </run>
  </groups>
  <test name = 'test'>
    <classes>
      <class Name = pac1.projectAndCustomerTest' />
      <class Name = 'pac2. UserTest' />
      <class Name = 'pac3. RepeatsTest' />
    </classes>
  </test>
</suite>
```

o/p :- execute create customer
execute create User
execute create repeats

```
<groups>
  <run> <include name = ' regression Test' />
  </run>
</groups>
```

o/p :- execute modify customer
execute modify User

NOTE

* Execute collection of similar test cases is called grouping execution, in order to invoke grouping execution insert group name in xml file, group name should be inserted before test and after suite tag.

Syntax:

```
<suite>
    <groups>
        <run>
            <include name = 'groupname' />
        </run>
    </groups>
<test>
```

* In one xml file multiple group invocation is allowed.

```
<suite>
    <groups>
        <run>
            <include name = 'smoketest' />
            <include name = 'regressiontest' />
        </run>
    </groups>
<test>
```

* In order to skip similar group test case (no execution) go for exclude html tag.

```
<suite>
    <groups>
        <run>
            <exclude name = 'smoketest' />
        </run>
    </groups>
<test>
```

* In order to achieve group execution each & every @Test method should have groupname along with annotation (@Test), one testcase can have multiple group name

Eg

```
@Test(groups = {"regressionTest", "smokeTest"})
public void modifyCustomerTest()
{
    System.out.println("execute modify customer");
}
```

→ While grouping, no both include & exclude html tag is included while executing.

→ If Before method is provided it wont execute for grouping. Here we have to provide group name

* In grouping execution configuration method will not invoked in execution without group name

Eg

```
public class
{
  @BeforeMethod (groups {"SmokeTest", "RegressionTe
                          -st"})
    public void configMtd ()
    {
      System.out.println ("before");
    }

@Test (groups {"SmokeTest"})
 public void createCustomerTest()
 {
   System.out.println ("execute create customer")
 }

@Test (groups {"RegressionTest"})
 public void modifyCustomerTest ()
 {
   System.out.println ("execute modify customer");
 }
```

Output

```
before
execute create customer
before
execute modify customer
```

→ If Before / After → class / method to be execute in grouping execution we should provide group name

```
<groups>
  <run>
    <include name='SmokeTest' />
    <include name='RegressionTest' />
  </run>
</groups>
```

→ whenever Before method fails → all Test Method skip

Assertions

Eg

Login page verification

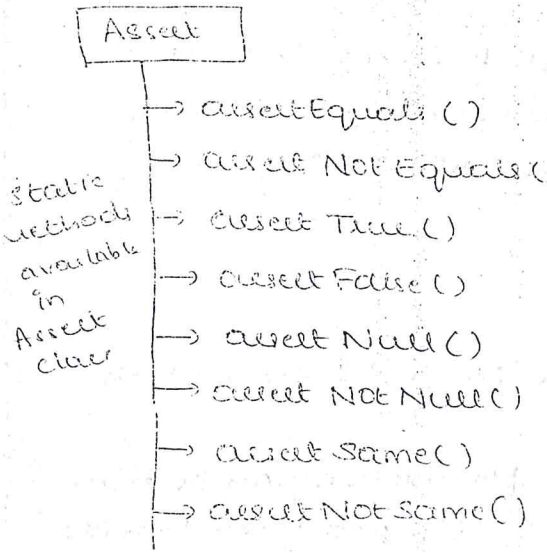
① Verify invalid msg
- → login to actiTime with invalid data
- → verify the error msg

ER → "invalid msg should be displayed"

② Verify logo
- → navigate to actiTime
- → Verify Login page logo

ER → "Logo Image should be displayed"

→ In test NG, if we use if else statement as in java then there will be no failures of test case so we go for Assertions (means failure count can be evaluated)



all are static methods.

Asseet Equals()

Syntax :- Asseet.asseetEquals("Expected Val", "act val", "Error Msg")

mandatory

optional

Asseet True()

Syntax :- Asseet.asseetTrue(condition, "Error msg")

Program

```
public class ProjectAndCustomerTest {

    @Test
    public void verifyInvalidMsgTest()
    {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://deepu-pc/login.do");
        String expectedMsg = "Username or Password is invalid";
        driver.findElement(By.id("loginButton")).click();
        String actMsg = driver.findElement(By.xpath("//
            Span[contains(text(),'Pas')]")).getText();
        Asseet.asseetEquals(actMsg, ExpectedMsg, "
            Expected msg is not displayed ==FAIL");
        System.out.println("msg is displayed ==
            PASS");
        driver.quit();
    }

    @Test
    public void verifyLogoTest()
    {
        System.out.println("execute verifyLogoTest");
```

```
WebDriver driver = new FirefoxDriver();
driver.get("http://demo-pc/login.do");
boolean flag = driver.findElement(By.xpath("//
  img[contains(@src,'time.gif')]")).isDisplayed
                                    ();
Assert.assertTrue(flag, "Image is not displayed
                        == FAIL");
System.out.println("Image is displayed ==
                        PASS");
driver.quit();
}
```

Output :-

Run : 2 Failure : 1

AssertionFailedException :- actual with expected result

at what line is error.

NOTE

* Assert is a TestNG class which is used to verify the expected result of the test case.
* All the methods available in Assert class are static methods, those method can be used to verify toString variable, Array, Collection Object Array etc.
* As per the Automation rule expected result should be verified with Assert methods.
* whenever Assert method fail TestNG generate Assert Error exception along with custom msg and stop the current test execution.
* Whenever assert method fails in test suite execution, TestNG stop the current test and continue execution with remaining test scripts.

Advantages of Assert

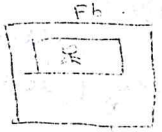
* Assert methods are used to fail the testNG test
* Assert methods are used for debugging because (To find which testcase got failed)
  whenever assert fails testNG fail the test along with line Number
* Using Assert methods we can compare any primitive datatype (or) collection (or) Arrays.
  → Assert.equals() → compare primitive type
  assert.same() → compare object type

Note :-

Assert.same() method is used to compare two object (or) two object Array only

Scenario

① Login to fb
② upload Image
③ Verify Image is displayed
④ Remove Image
⑤ Verify Image should not be available in



boolean flag1 = drive.findElement(By.xpath("//img")).isDisplayed();

Assert.assertTrue(flag1)

boolean flag2 = driver.findElement(By.xpath("//img")).isDisplayed();

Assert.assertTrue(flag2); ✗ ⟶ it fails the test

Assert.assertFalse(flag2); ✓

NOTE

* assertFalse() method is always look for false boolean value. If the argument is false pass the test or else fail the test.

* assertTrue() method is always look for true argument. If the argument is true pass the test or else fail the test.

⟶ SoftAssert ⟶ is similar to verify checkpoint in Selenium IDE

In java if-else acts as verify
verify ⟶ is like if-else does not fails the testNG test.

but SoftAssert fails the testNG test throws exception but continues further execution on the same test.

⟶ SoftAssert 2 methods are non-static

↓

assertAll() ⟶ collect all the error & display in console. it is mandatory

Eg

```
public class UserTest
{
    @Test
    public void createUserTest()
    {
        SoftAssert sa = new SoftAssert();
        System.out.println("step : 1");
        sa.assertEquals("A", "B", "customer name is not same");
        System.out.println("step : 2");
        sa.assertEquals("100", "101", "customer amount is not same");
        System.out.println("step : 3");
        sa.assertEquals("1 crore", "1 crore", "acc is not same");
        System.out.println("step : 4");
        sa.assertAll(); sop("execute create user");
    }
    @Test
    public void modifyUserTest()
```

}
System.out.println("execute modify user");
}

Output

test run : 2 failure : 1 Skip : 0

Step:1
Step:2
Step:3
Step:4
execute create user
execute modify user

Assert Error Exception :
Customer name is not same
amount is not same

NOTE

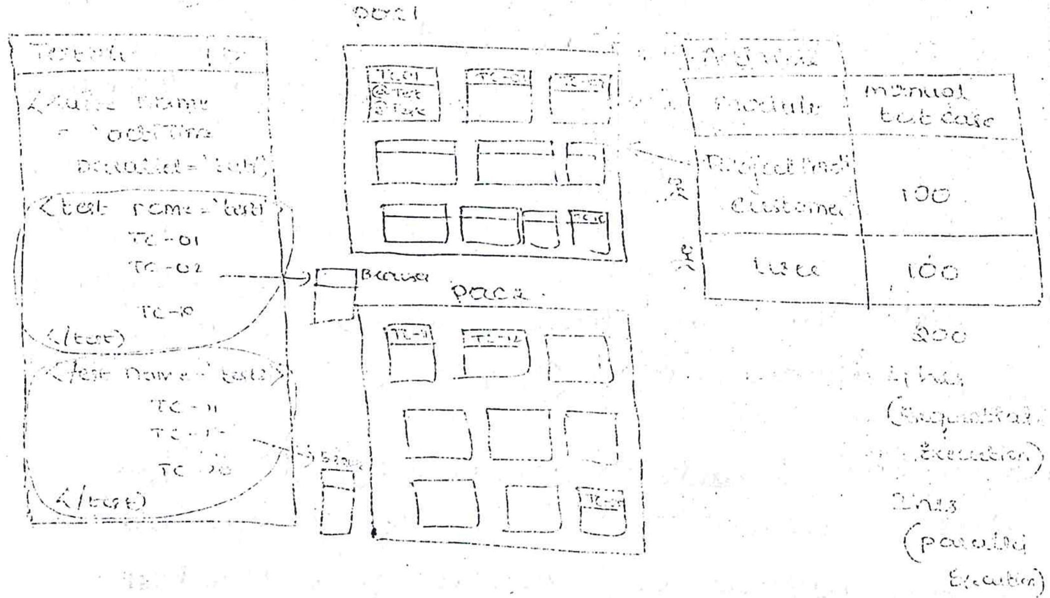
Diff between Soft Assert & Hard Assert

Soft Assert	Hard Assert
all the methods available in soft Assert are non static	* all the methods available in Hard are static
whenever soft Assert fails TestNG fail the testNG test & continue remaining Steps execution	* whenever Hard Assert fails TestNG test & generate error msg & stop the current test Execution
* assertAll() method should be inserted at the end of the test, to collect all the assert error and along	* assertAll () method is not required (not available only)

Based on product knowledge we decide which assert to use
(mandatory fields → Hard assert)

Parallel Execution

Distributing & execute the test case in multiple browser is called parallel Execution.



thread count → (No of thread used to launch browser for parallel execution)
↓
maximum of 5 threads

TestNG xml file

Parallel Execution E.g

<suite name="suite" parallel="tests" thread-count="2">

<test name="Test1">

<classes>

```
        <class name = "pac1.UserTest"/>
      </classes>
    </test>
    <test name = "Test2">
      <classes>
        <class name = "pac2.RepeatTest"/>
      </classes>
    </test>
</suite>
```

Eg: thread-count = Sequential execution will happen

→ So for parallel Execution, first enable the parallel field in xml tag of suite by assigning a value "tests" & also thread-count should be assigned (default is 5 → maximum).

NOTE

* In order to execute multiple testcase with multiple browsers parallely (concurrently) we go for parallel execution.
* In order to get the result in short period of time we go for parallel execution.
* Execute same test case with different browser is called cross-browser testing (or) also called as compatability testing.

* To achieve parallel execution, enable parallel attribute in suite xml tag & distribute test case with multiple test runner.
* Thread count should be depending on No of test runner and maximum test runner we can go till 5 (... thread ... crash)

Interview Question

1. What is TestNG?
→ Unit testing framework tool
2. Why TestNG? Why not JUnit?
→ To achieve Batch execution we go for TestNG. JUnit does not support parallel execution, Grouping Execution, html reports. So we go for TestNG. (JUnit ... but no such ... also to generate report we have to ...)
3. What is Annotation and explain all the annotation with eg?
4. How to achieve batch execution in TestNG?
→ create xml file & include all the classes & execute
5. What is grouping execution & How to achieve
→ Execute similar test case

What is parallel Execution & How to achieve?

What is Assert and explain all the assert methods with real time example?

Diff b/w soft Assert & hard Assert?

What is listeners annotation and where to use?

Keep on monitoring Execution events at run time & if bug exists take a screenshot

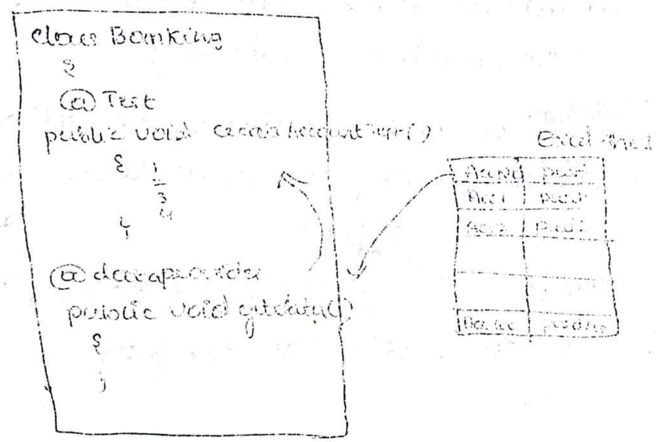
Listener is a annotation in TestNG which is used to monitor the execution events happening in run time. If any test case fails using listeners we can take a screenshot

What is parameterization and How to achieve parameterization?

Execute same test case with different test data is called parameterization

Scenario : Create Account

① Login to Bank
② Create 100 Account
③ Verify all Account
④ Logout



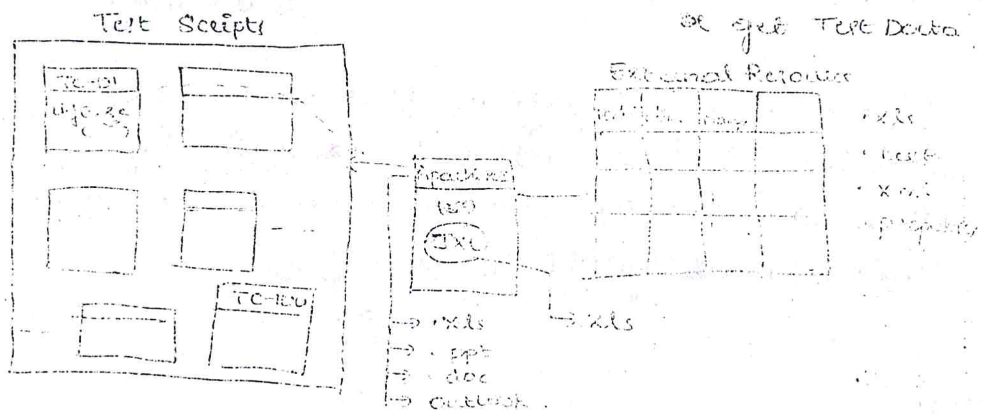
In order to achieve parameterization in TestNG we use @dataprovider annotation

File Handling

Data fixed in Test case → Hardcoded data

If 1000 test cases we have to change data, it will be tedious process. So we do not hard code test data in test cases.

So we use External Resources to provide Test Data or get Test Data



WebDriver does not support Excel Sheet (since it is a Stand alone Application)

So we take help of "Apache POI" to support Microsoft document to WebDriver

Note

- As per the Automation rule, test data should not be hard coded within the test scripts, instead we should get data from external resource & run the test

* External Resource might be
  - .xml
  - .xls
  - .ppt
  - .db
  - .txt

* WebDriver does not support to read any other documents like .properties, .txt etc so in order to get data from .xls file we should take help of third party tools like Apache POI / JXL.

* Apache POI is a open source tool available in Apache community which supports all microsoft documents.

* Apache POI is not a UI based tool, it is just a collection of jars.

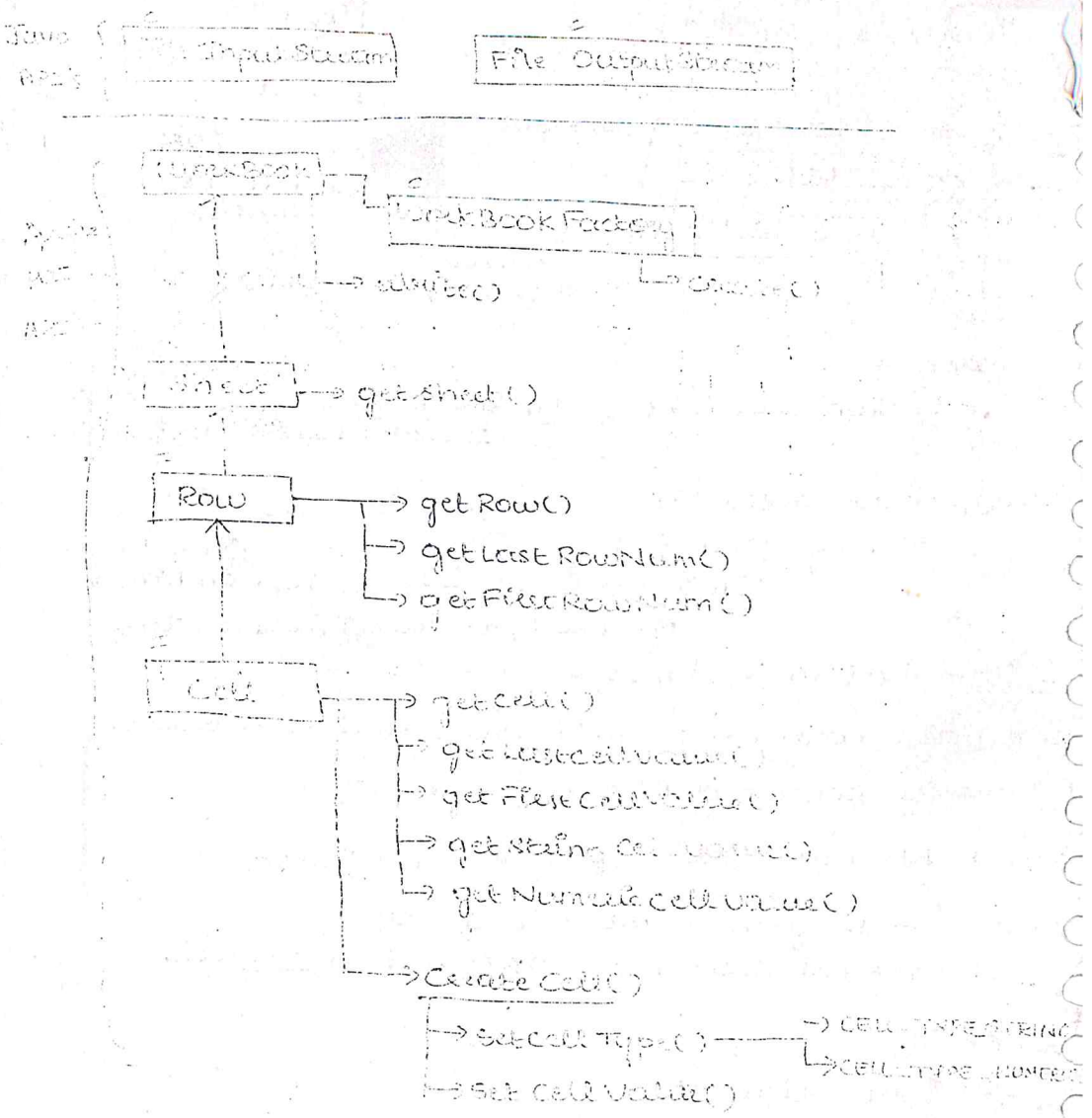
Installation steps of Apache POI

① Go to Google → type → download apache POI
click on first link & navigate to apache community
https://poi.apache.org

② Below the Binary distribution section click on poi-bin-3.13-20150929.zip file.

③ In next page click on first link.

Apache POI Architecture



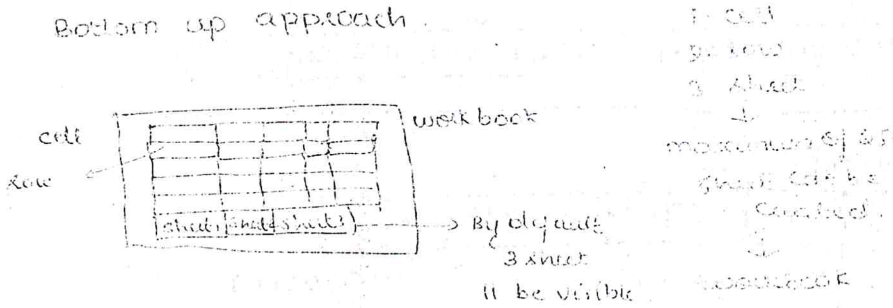
→ POI is old api and only support xls

→ Apache POI is a very powerful tool which supports all microsoft documents

→ All are static final methods

Excel ——→ microsoft

Bottom up approach.



→ window #

extension ——→ .xlsx

① get location of the file using java API's (to read/write)

File Inputstream ——→ It supports Notepad

For Excel, we use Apache POI Interface

② open Excel in Read mode

③ Get the control of 'sheet' [which sheet is Data written]

④ get the control of 1st Row (which row).

⑤ Get the cell value from 2, 3, 4 column

→ Index start from 0 for Row & column

→ All Apache POI should be imported from ss model

Program

```
public class UserTest
{
  @Test
  public void createUserTest () throws Encrypted
        -DocumentException, InterruptedException
  {
   // write a code to get the test data from Excel

   // step1 : get the workbook file location
   FileInputStream fis = new FileInputStream("C:\\
       Users\\Deepa\\Desktop\\TestData.xlsx");

   // step2 . open workbook in read mode
   Workbook wb = WorkbookFactory.create(fis);

   // step3 : get the control of the "Sheet1"
   Sheet sh = wb.getSheet("Sheet1");

   // step4 : get the control of 1st Row
   Row row = sh.getRow(1);

   // step5 : get the cell data from 2, 3 & 4 column
   String userName = row.getCell(2).getStringCell
                         Value();

   String password = row.getCell(3).getStringCell
                         Value();

   String [illegible] = row.getCell(4).getStringCellValue();
   System.out.println(userName);
   System.out.println(password);
   System.out.println([illegible]);

   WebDriver driver = new FirefoxDriver();
   driver.get("http://deepa-pc/login.do");
   driver.findElement(By.name("username"))
            .sendKeys(userName);

   driver.findElement(By.name("pwd")).sendKeys
            (password);
```

```
driver.findElement(By.id("LoginButton")).click();
driver.quit();
}
}
```

Output

```
admin
manager
Rest - RAM
Passed : CreateUserTest
Test run: 1  Failure:0  Skip:0
```

TC_01	create user	admin	manager	RAM, SAM

```
String use = row.getCell(a).getStringCellValue();
String[] arr = use.split(",");
String user1 = arr[0];
String newUser = arr[1];
Sop(arr[0]);
Sop(arr[1]);
```

Output

```
RAM
SAM
```

→ Write a sample code to write data back to Excel.

→ To write data back to excel

① we should close the excel if already opened

② First to write
open excel sheet in read mode
& then open Excel sheet in write mode
Save & close the sheet
(write)

Program

```
public class CreateTest
{
@Test
public void CreateUserTest() throws EncryptedDocumentException, InvalidFormatException, IOException
{
String filePath = "C:\\Users\\DEEPU\\Desktop\\TestData.xlsx";
FileInputStream fis = new FileInputStream(filePath);
Workbook wb = WorkbookFactory.create(fis);
Sheet sh = wb.getSheet("Sheet1");
Row row = sh.getRow(1);
Cell cell = row.createCell(5);
cell.setCellType(cell.CELL_TYPE_STRING);
FileOutputStream fos = new FileOutputStream(filePath);
cell.setCellValue("PASS");
wb.write(fos); → save
```