**Generic Libraries / Utility**

**What is Generic components in Automation Framework?**

🡺it’s one of the automation framework components which is common for all the application

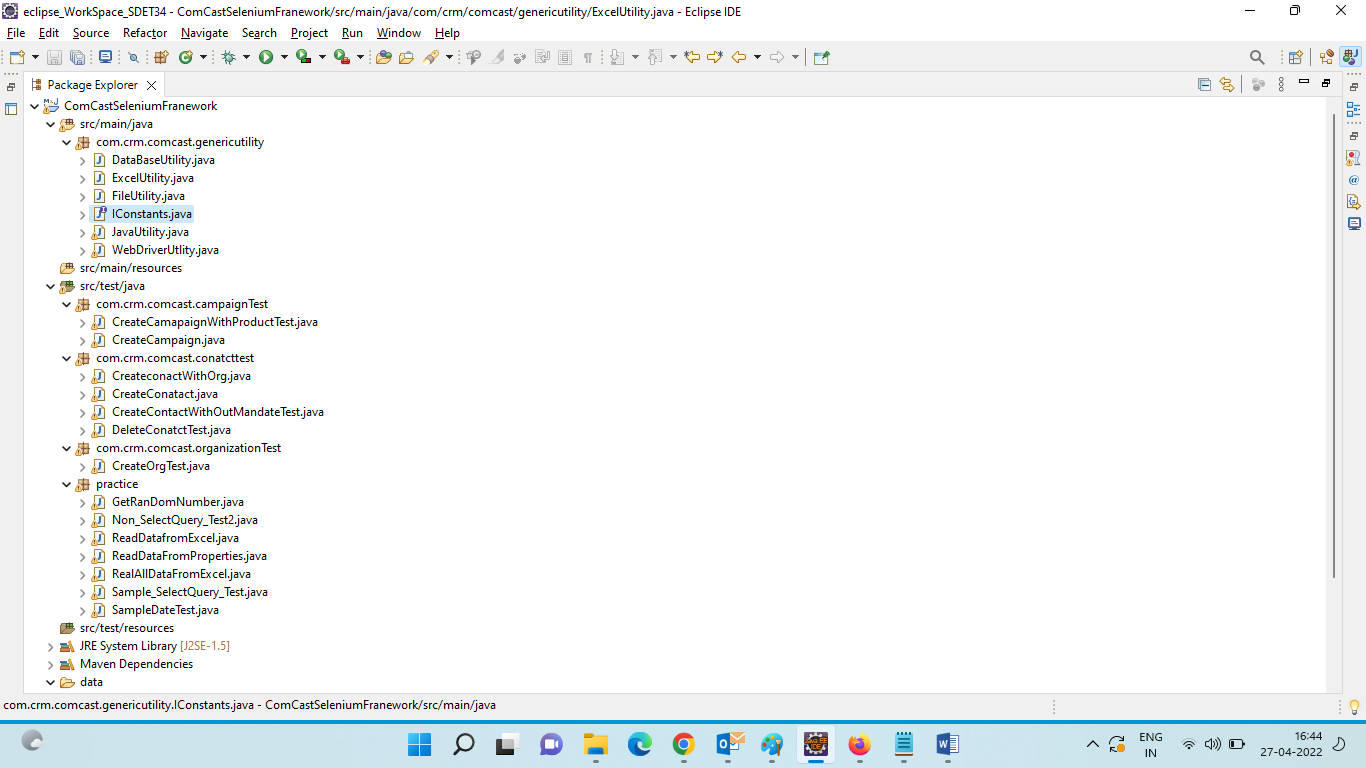
🡺its collection of generic class contains reusable methods / libraries

🡺The methods which can be used to any application is called Generic/common methods

**What is the advantages of Generic components?**

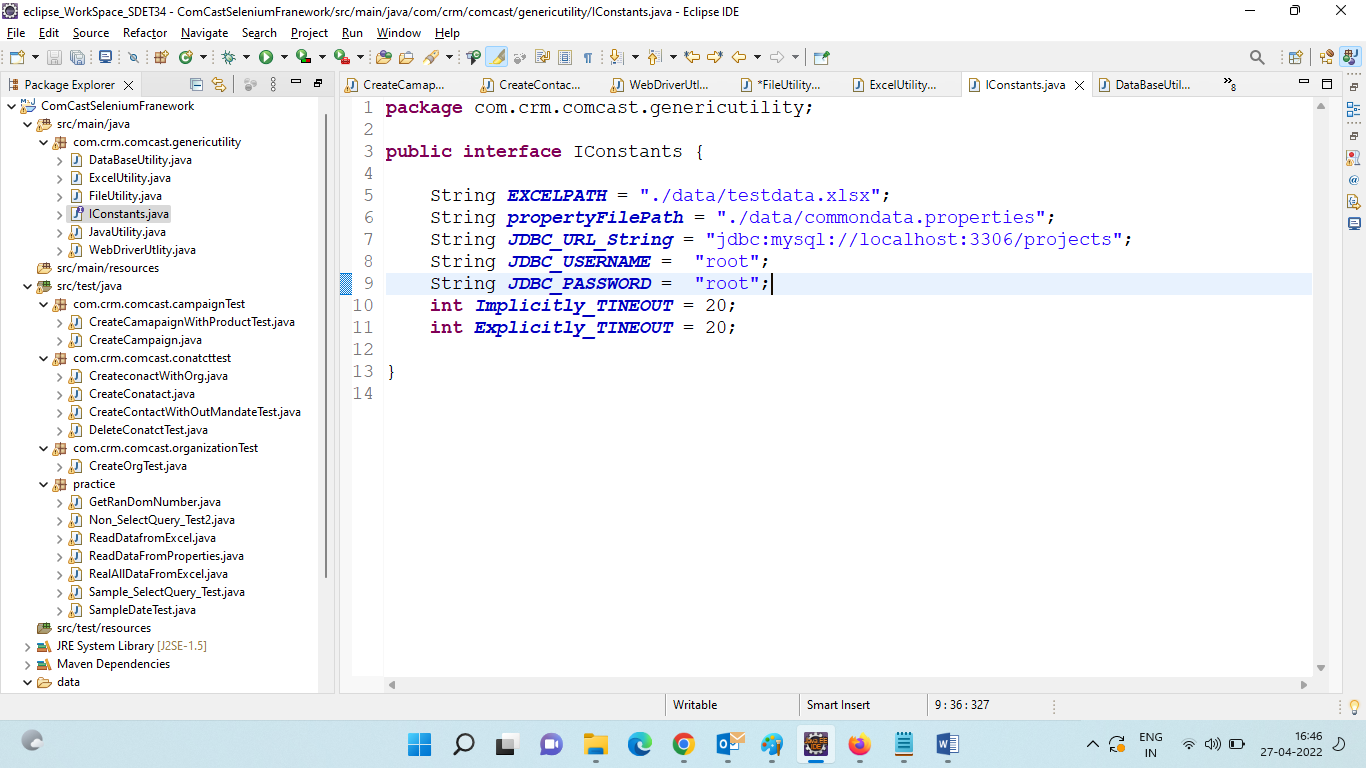
* Reusability of code
* Code Optimization
* Test script development is faster
* Test Code readability
* Generic libraries are common to all automation projects
* Avoid duplicate Code
* no need to remember the syntax every time , just create once & use multiple times

**Generic Utility Structure in Automation Project**

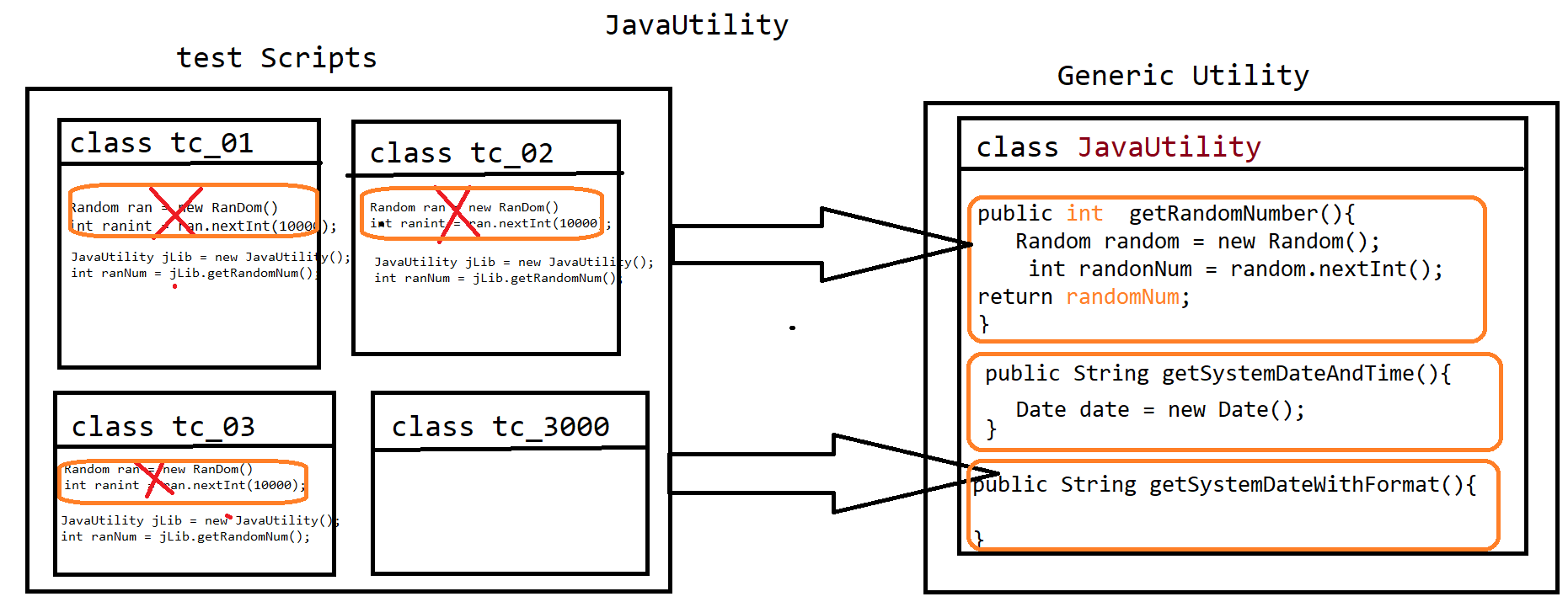


=====================Iconstants========================

It’s an Interface where contains all the common data which can be used across the Framework



1. Java Utility Libraries



🡺 Java Utility is one class in generic component, which contain java specific methods which can be used across the test Scripts / Application

🡺 its contains several generic reusable methods like

* getRandomNum() : it’s used to generate random number for every invocation
* getSystemDate() : it’s used to generate system date and time

=====================Code===============================

**package** com.crm.comcast.genericutility;

**import** java.util.Date;

**import** java.util.Random;

/\*\*

\* it's Contains java specific libraries like getRandomdata & getsystemDate etc

\* **@author** Deepak

\*

\*/

**public** **class** JavaUtility {

/\*\*

\* its used to generate the random number with in the range of 1000

\* **@return**

\*/

**public** **static** **int** getRanDomNumber() {

Random ranDom = **new** Random();

**int** ranDomNum = ranDom.nextInt(1000);

**return** ranDomNum;

}

/\*\*

\* its used to get the current system date based on YYYY-MM-DD formate

\* **@return**

\*/

**public** String getSystemDate() {

Date date = **new** Date();

String currentDate = date.toString();

System.***out***.println(currentDate);

String[] arr = currentDate.split(" ");

String yyyy = arr[5];

String dd = arr[2];

**int** mm = date.~~getMonth~~()+1;

String formate = yyyy+"-"+mm+"-"+dd;

**return** formate;

}

/\*\*

\* its used to get the current system date based on DD-MM-YYYY format

\* **@return**

\*/

**public** String getSystemDateInIST() {

Date date = **new** Date();

String currentDate = date.toString();

System.***out***.println(currentDate);

String[] arr = currentDate.split(" ");

String yyyy = arr[5];

String dd = arr[2];

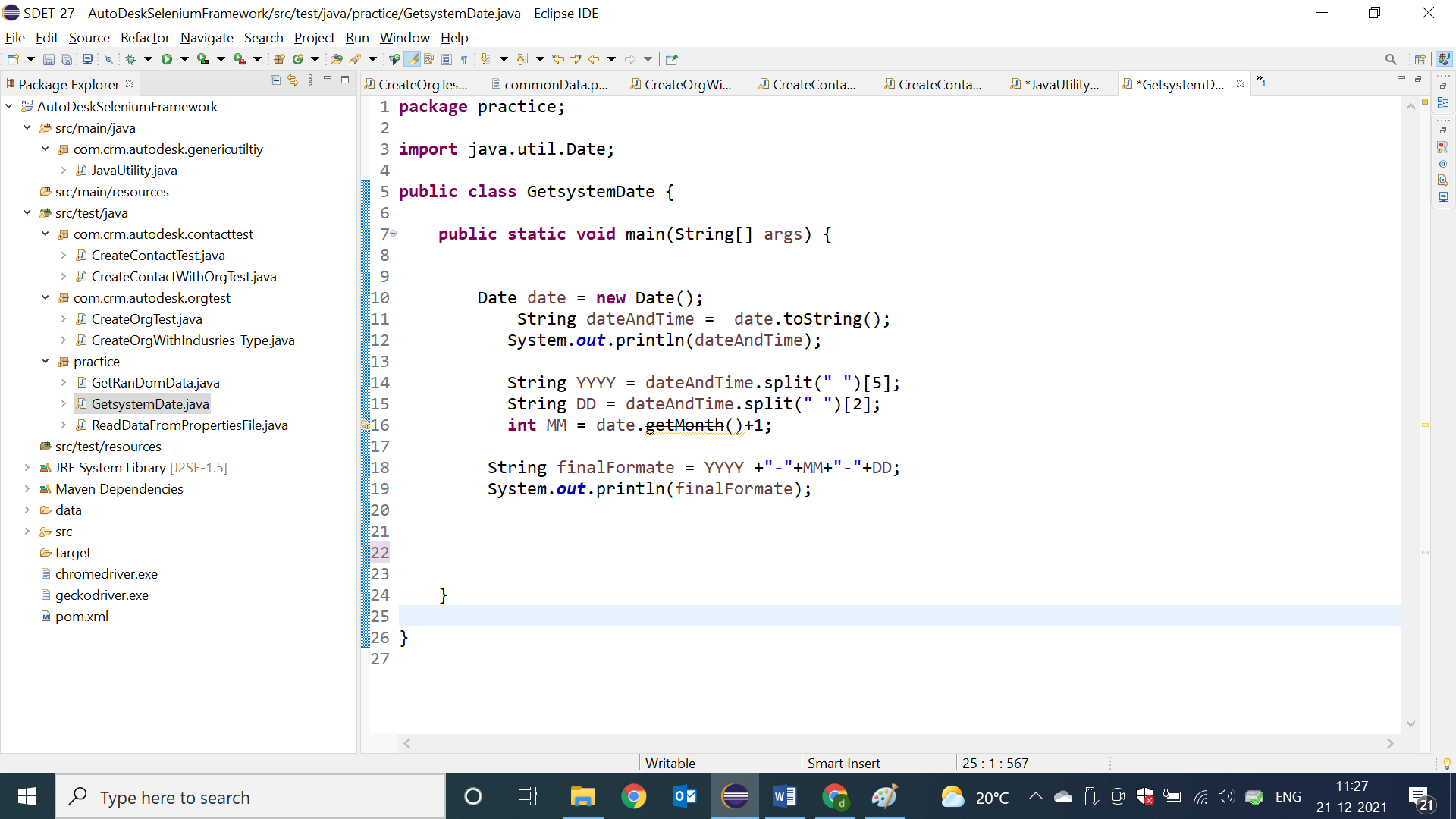
**int** mm = date.~~getMonth~~()+1;

String formate = dd+"-"+mm+"-"+yyyy;

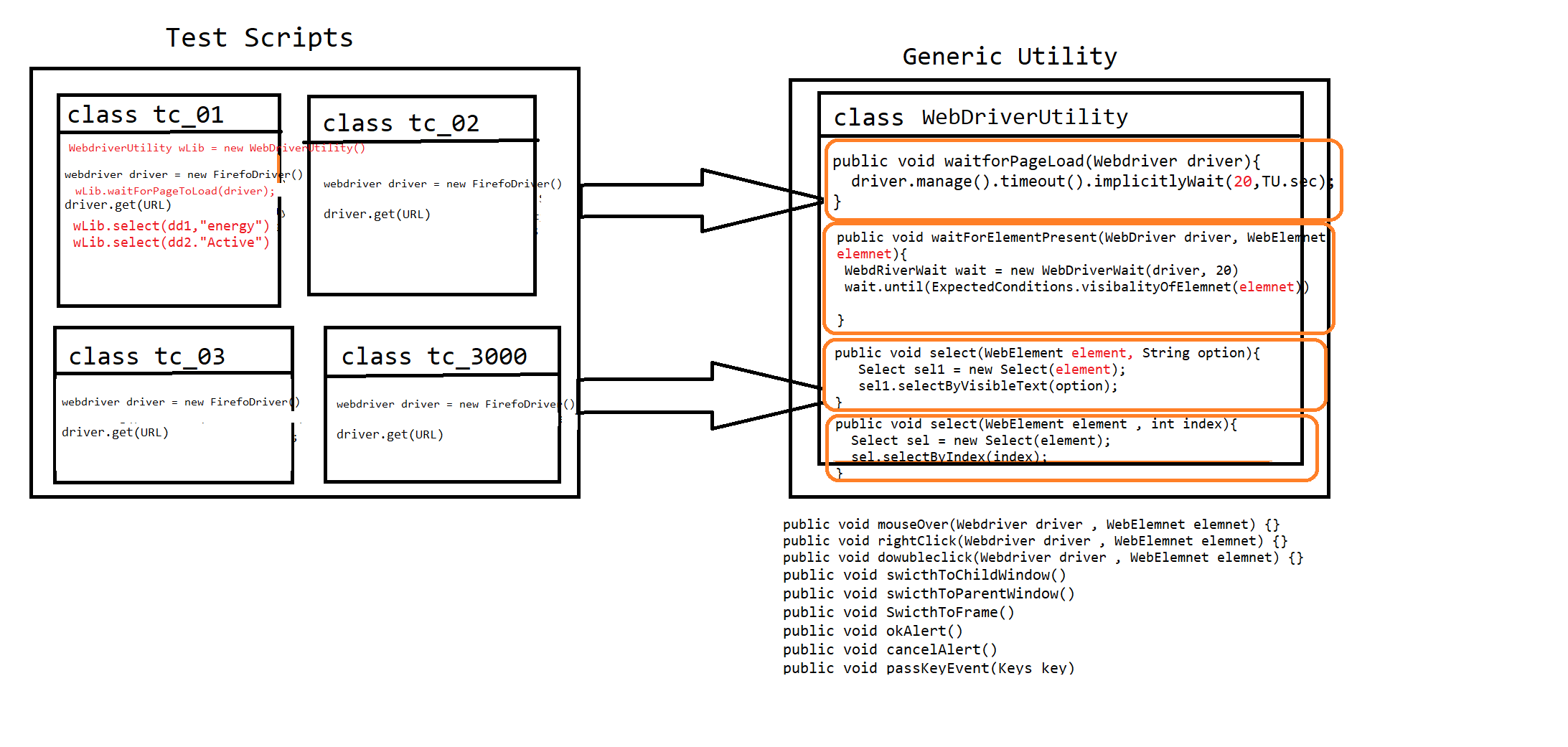
**return** formate;

}

}



1. WebDriver Utility Libraries



* WebdriverUtility is a Generic class , which contains webdriver specific reusable actions like
* waitForPageToLoad()
* waitForElement()
* select()
* accpertAlert()
* cancelAlert() .Etc

==========Code======================

**package** com.crm.comcast.genericutility;

**import** java.util.Iterator;

**import** java.util.Set;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.Alert;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.Keys;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.Select;

**import** org.openqa.selenium.support.ui.WebDriverWait;

/\*\*

\* contains all reusable Actions of Webdriver

\* **@author** Deepak

\*

\*/

**public** **class** WebDriverUtlity {

/\*\*

\* it's an implicitly wait Always wait for Element in DOM document & release the control if element available

\* **@param** driver

\*/

**public** **void** waitForElementInDOM(WebDriver driver) {

driver.manage().timeouts().implicitlyWait(IConstants.***Implicitly\_TINEOUT***, TimeUnit.***SECONDS***);

}

/\*\*

\* it's an Explicitly wait Always wait for Page to be loaded & available in GUI

\* **@param** driver

\* **@param** partailPageURL

\*/

**public** **void** waitForPage(WebDriver driver , String partailPageURL) {

WebDriverWait wait = **new** WebDriverWait(driver, IConstants.***Explicitly\_TIMEOUT***);

wait.until(ExpectedConditions.*urlContains*(partailPageURL));

}

/\*\*

\* it's an Explicitly wait Always wait for Page to be loaded & available in GUI

\* **@param** driver

\* **@param** partailPageURL

\*/

**public** **void** waitForElement(WebDriver driver , WebElement element) {

WebDriverWait wait = **new** WebDriverWait(driver, IConstants.***Explicitly\_TINEOUT***);

wait.until(ExpectedConditions.*visibilityOf*(element));

}

/\*\*

\* used to Switch to Any Window based on Window Title

\* **@param** driver

\* **@param** partialWindowTitle

\*/

**public** **void** swithToWindow(WebDriver driver , String partialWindowTitle) {

Set<String> set = driver.getWindowHandles();

Iterator<String> it = set.iterator();

**while** (it.hasNext()) {

String wID = it.next();

driver.switchTo().window(wID);

String currentWindowTitle = driver.getTitle();

**if**(currentWindowTitle.contains(partialWindowTitle)) {

**break**;

}

}

}

/\*\*

\* used to Switch to Alert Window & click on OK button

\* **@param** driver

\*/

**public** **void** swithToAlertWindowAndAccpect(WebDriver driver ,String expectedMsg) {

Alert alt = driver.switchTo().alert();

**if**(alt.getText().trim().equalsIgnoreCase(expectedMsg.trim())) {

System.***out***.println("Alert Message is verified");

}**else** {

System.***out***.println("Alert Message is not verified");

}

alt.accept();

}

/\*\*

\* used to Switch to Alert Window & click on Cancel button

\* **@param** driver

\*/

**public** **void** swithToAlertWindowAndCancel(WebDriver driver, String expectedMsg) {

Alert alt = driver.switchTo().alert();

**if**(alt.getText().equals(expectedMsg)) {

System.***out***.println("Alert Message is verified");

}**else** {

System.***out***.println("Alert Message is not verified");

}

alt.dismiss();

}

/\*\*

\* used to Switch to Frame Window based on index

\* **@param** driver

\* **@param** index

\*/

**public** **void** swithToFrame(WebDriver driver , **int** index) {

driver.switchTo().frame(index);

}

/\*\*

\* used to Switch to Frame Window based on id or name attribute

\* **@param** driver

\* **@param** id\_name\_attribute

\*/

**public** **void** swithToFrame(WebDriver driver , String id\_name\_attribute) {

driver.switchTo().frame(id\_name\_attribute);

}

/\*\*

\* used to select the value from the dropDwon based on index

\* **@param** driver

\* **@param** index

\*/

**public** **void** select(WebElement element , **int** index) {

Select sel = **new** Select(element);

sel.selectByIndex(index);

}

/\*\*

\* used to select the value from the dropDwon based on value / option avlaible in GUI

\* **@param** element

\* **@param** value

\*/

**public** **void** select(WebElement element , String text) {

Select sel = **new** Select(element);

sel.selectByVisibleText(text);

}

/\*\*

\* used to place mouse cursor on specified element

\* **@param** driver

\* **@param** elemnet

\*/

**public** **void** mouseOverOnElement(WebDriver driver , WebElement elemnet)

{

Actions act = **new** Actions(driver);

act.moveToElement(elemnet).perform();

}

/\*\*

\* used to right click on specified element

\* **@param** driver

\* **@param** elemnet

\*/

**public** **void** rightClickOnElement(WebDriver driver , WebElement elemnet)

{

Actions act = **new** Actions(driver);

act.contextClick(elemnet).perform();

}

/\*\*

\*

\* **@param** driver

\* **@param** javaScript

\*/

**public** **void** executeJavaScript(WebDriver driver , String javaScript) {

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeAsyncScript(javaScript, **null**);

}

**public** **void** waitAndClick(WebElement element) **throws** InterruptedException

{

**int** count = 0;

**while**(count<20) {

**try** {

element.click();

**break**;

}**catch**(Throwable e){

Thread.*sleep*(1000);

count++;

}

}

}

/\*\*

\* pass enter Key appertain in to Browser

\* **@param** driver

\*/

**public** **void** passEnterKey(WebDriver driver) {

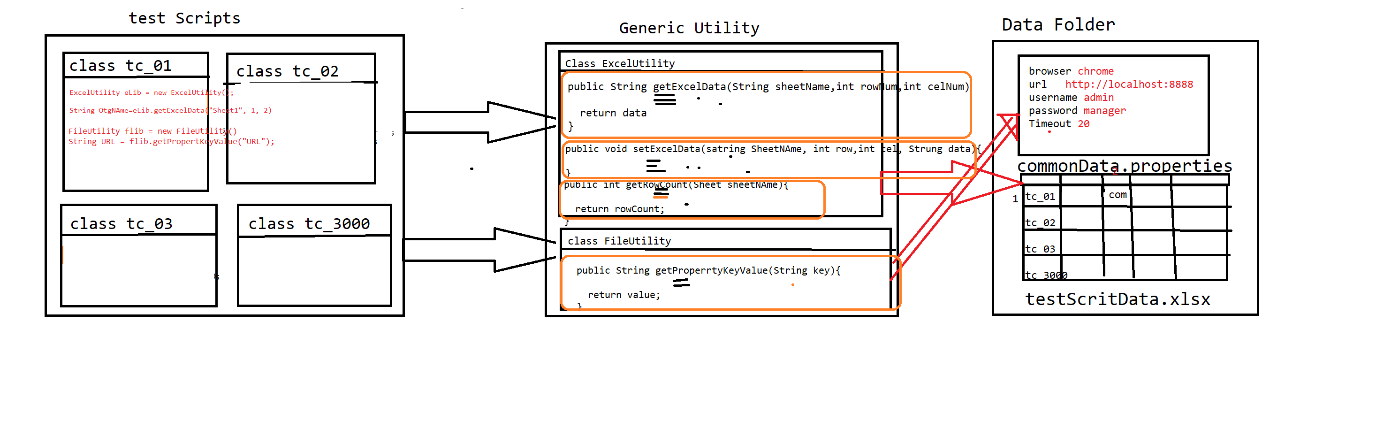
Actions act = **new** Actions(driver);

act.sendKeys(Keys.***ENTER***).perform();

}

}

1. Excel Utility libraries



* As per the rule of automation, data should not be hardcoded with in the test scripts, so that to get the data from external file like Excel & .propertes file

We go for ExcelUtility & FileUtlity

* Excel Utility class is developed using apache Poi libraries, which is used to read the data from Excel
* FileUtility is used to get the data from .properties file

==========Code=======================

**package** com.crm.comcast.genericutility;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

/\*\*

\* its developed using Apache POi libraries , which used to handle Microsoft Excel sheet

\* **@author** Deepak

\*

\*/

**public** **class** ExcelUtility {

/\*\*

\* its used to read the data from Excel-Workbook based on below

\* **@param** sheetName

\* **@param** rowNum

\* **@param** celNum

\* **@return** String data

\* **@throws** Throwable

\*/

**public** String getExcelData(String sheetName , **int** rowNum , **int** celNum) **throws** Throwable {

FileInputStream fis = **new** FileInputStream(IConstants.***EXCELPATH***);

Workbook wb = WorkbookFactory.*create*(fis);

Sheet sh = wb.getSheet(sheetName);

Row row = sh.getRow(rowNum);

String data = row.getCell(celNum).getStringCellValue();

**return** data;

}

/\*\*

\* used to get the last used row number on specified Sheet

\* **@param** sheetName

\* **@return**

\* **@throws** Throwable

\*/

**public** **int** getRowCount(String sheetName) **throws** Throwable {

FileInputStream fis = **new** FileInputStream(IConstants.***EXCELPATH***);

Workbook wb = WorkbookFactory.*create*(fis);

Sheet sh = wb.getSheet(sheetName);

wb.close();

**return** sh.getLastRowNum();

}

/\*\*

\* used to write data back to Excel based on below parameter

\* **@param** sheetName

\* **@param** rowNum

\* **@param** celNum

\* **@param** data

\* **@throws** Throwable

\*/

**public** **void** setDataExcel(String sheetName , **int** rowNum, **int** celNum ,String data) **throws** Throwable {

FileInputStream fis = **new** FileInputStream(IConstants.***EXCELPATH***);

Workbook wb = WorkbookFactory.*create*(fis);

Sheet sh = wb.getSheet(sheetName);

Row row = sh.getRow(rowNum);

Cell cel = row.createCell(celNum);

cel.setCellValue(data);

FileOutputStream fos = **new** FileOutputStream(IConstants.***EXCELPATH***);

wb.write(fos);

wb.close();

}

}

1. **File Utility=====================**

**package** com.crm.comcast.genericutility;

**import** java.io.FileInputStream;

**import** java.util.Properties;

**import** javax.swing.Icon;

/\*\*

\* it;s contains External File specific libraries

\* **@author** Deepak

\*

\*/

**public** **class** FileUtility {

/\*\*

\* its used return the value from the property file based on key

\* **@param** key

\* **@return** value

\* **@throws** Throwable

\*/

**public** String getPropertyKeyValue(String key) **throws** Throwable {

FileInputStream fis = **new** FileInputStream(IConstants.***propertyFilePath***);

Properties pObj = **new** Properties();

pObj.load(fis);

String value = pObj.getProperty(key);

**return** value;

}

}

================================Data base Utility==================

Data base uitilty is implemented using JDBC , which is used to connect to any database using java code,

Below utility contains 4 methods used for below operations

1. connectDB: used for db connection
2. closeDB : used for close db connection
3. execyteQuery : used to execute select query
4. execyteUpdate : used to execute non select query
5. executeQuerryAndVerify : used to execute query & verify the data in db in specified column

package com.crm.comcast.genericutility;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import com.mysql.jdbc.Driver;

public class DataBaseUtility {

static Driver driver;

static Connection connection;

static ResultSet result;

/\*\*

\* This method will perform the mysql database connection

\* @throws SQLException

\*/

public void connectDB() throws SQLException {

try {

driver=new Driver();

DriverManager.registerDriver(driver);

connection=DriverManager.getConnection(IConstants.JDBC\_URL\_String,IConstants.JDBC\_USERNAME,IConstants.JDBC\_PASSWORD);

} catch (SQLException e) {

e.printStackTrace();

}

}

/\*\*

\* This method will close the mysql database

\* @throws SQLException

\*/

public void closeDB() throws SQLException {

try {

connection.close();

}catch (Exception e) {

}

}

/\*\*

\* This method will execute the querry

\* @param query

\* @return

\* @throws Throwable

\*/

public ResultSet execyteQuery(String query) throws Throwable {

result = connection.createStatement().executeQuery(query);

return result;

}

/\*\*

\* This method will execute the querry

\* @param query

\* @return

\* @throws Throwable

\*/

public int execyteUpdate(String query) throws Throwable {

int result = connection.createStatement().executeUpdate(query);

return result;

}

/\*\*

\* This method will execute querry based on querry and it will verify the data.

\* @param querry

\* @param columnName

\* @param expectedData

\* @return

\* @throws Throwable

\*/

public boolean executeQuerryAndVerify(String querry,int cloumnIndex,String expectedData) throws Throwable {

boolean flag=false;

result=connection.createStatement().executeQuery(querry);

while(result.next()) {

if(result.getString(cloumnIndex).equals(expectedData)) {

flag=true;

break;

}

}

if(flag) {

System.out.println(expectedData+"==>Data is verified in the data base table");

return flag;

}else {

System.out.println(expectedData+"==>data is not verified in the database");

return flag;

}

}

}

}====================================Same Program using Generic utility ===================

============test Name : CreateCampaignWith Product=================

