**GeneralIndex**

**package** com.index;

**import** org.testng.Assert;

**import** org.testng.annotations.Test;

**import** com.init.Common;

**import** com.init.SeleniumInit;

**import** com.verification.GeneralVerification;

**public** **class** GeneralIndex **extends** SeleniumInit {

//To verify that all mandatory fields get the blank field validation message

@Test(priority=0)

**public** **void** blankfields() {

**int** step=1;

Common.*logcase*("To verify that all mandatory fields get the blank field validation message");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.blankfieldsvalidationmsg()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//To verify phone field get minimum length validation message

@Test(priority=0)

**public** **void** phonefield\_minimum\_length() {

**int** step=1;

Common.*logcase*("To verify phone field get minimum length validation message");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Enter Invalid Phone number(Less than 10 digits)");

generalVerification = generalIndexpage.enterinvalidphonenumber();

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.minimulengthvalidationmsg()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//To verify password field get minimum length validation message

@Test(priority=0)

**public** **void** password\_minimum\_length() {

**int** step=1;

Common.*logcase*("To verify password field get minimum length validation message");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Enter Invalid password(Less than 8 characters)");

generalVerification = generalIndexpage.enterinvalidpassword();

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.passwordminimulengthvalidationmsg()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//To verify that user gets the validation message while confirm password and password fields do not match

@Test(priority=0)

**public** **void** passwordnotmatch(){

**int** step=1;

Common.*logcase*("To verify that user gets the validation message while confirm password and password fields do not match");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Enter password(More than 7 characters)");

generalVerification = generalIndexpage.enterpasswordnotmatch();

Common.*logstep*("Step "+ (step++) +". Enter confirm password(not match with password)");

generalVerification = generalIndexpage.enterconfirmpasswordnotmatch();

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.passwordonotmatch()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//To verify that user gets the validation message for Invalid emailid

@Test(priority=0)

**public** **void** invalidemailid(){

**int** step=1;

Common.*logcase*("To verify that user gets the validation message for Invalid emailid");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Enter Invalid EmailId");

generalVerification = generalIndexpage.enterinvalidemailid();

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.invalidemailid()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//To verify that user is able to register with all valid details

@Test(priority=0)

**public** **void** allvaliddata(){

**int** step=1;

Common.*logcase*("To verify that user is able to register with all valid details");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Enter Valid First Name");

generalVerification = generalIndexpage.enterfirstname();

Common.*logstep*("Step "+ (step++) +". Enter Valid Last Name");

generalVerification = generalIndexpage.enterlastname();

Common.*logstep*("Step "+ (step++) +". Select Marital Status");

generalVerification = generalIndexpage.selectmaritalstatus();

Common.*logstep*("Step "+ (step++) +". Select Hobby");

generalVerification = generalIndexpage.selecthobby();

Common.*logstep*("Step "+ (step++) +". Select Country");

generalVerification = generalIndexpage.selectcountry();

Common.*logstep*("Step "+ (step++) +". Select Date of Birth");

generalVerification = generalIndexpage.selectdateofbirth();

Common.*logstep*("Step "+ (step++) +". Enter Valid Phone Number");

generalVerification = generalIndexpage.enterphonenumber();

Common.*logstep*("Step "+ (step++) +". Enter Valid User Name");

generalVerification = generalIndexpage.enterusername();

Common.*logstep*("Step "+ (step++) +". Enter Valid Emailid");

generalVerification = generalIndexpage.enteremailid();

Common.*logstep*("Step "+ (step++) +". Select Your Profile Picture");

generalVerification = generalIndexpage.selectyourprofilepic();

Common.*logstep*("Step "+ (step++) +". Enter About YourSelf");

generalVerification = generalIndexpage.enteraboutyourself();

Common.*logstep*("Step "+ (step++) +". Enter Valid Password");

generalVerification = generalIndexpage.enterpassword();

Common.*logstep*("Step "+ (step++) +". Enter Valid Confirm Password");

generalVerification = generalIndexpage.enterconfirmpassword();

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.successfullregistration()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//To verify that user get the validation message for exist username

@Test(priority=0)

**public** **void** usernamealreadyexist(){

**int** step=1;

Common.*logcase*("To verify that user get the validation message for exist username");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Enter Valid First Name");

generalVerification = generalIndexpage.enterfirstname();

Common.*logstep*("Step "+ (step++) +". Enter Valid Last Name");

generalVerification = generalIndexpage.enterlastname();

Common.*logstep*("Step "+ (step++) +". Select Marital Status");

generalVerification = generalIndexpage.selectmaritalstatus();

Common.*logstep*("Step "+ (step++) +". Select Hobby");

generalVerification = generalIndexpage.selecthobby();

Common.*logstep*("Step "+ (step++) +". Select Country");

generalVerification = generalIndexpage.selectcountry();

Common.*logstep*("Step "+ (step++) +". Select Date of Birth");

generalVerification = generalIndexpage.selectdateofbirth();

Common.*logstep*("Step "+ (step++) +". Enter Valid Phone Number");

generalVerification = generalIndexpage.enterphonenumber();

Common.*logstep*("Step "+ (step++) +". Enter Valid User Name");

generalVerification = generalIndexpage.enterexistusername();

Common.*logstep*("Step "+ (step++) +". Enter Valid Emailid");

generalVerification = generalIndexpage.enternotregisteremailid();

Common.*logstep*("Step "+ (step++) +". Select Your Profile Picture");

generalVerification = generalIndexpage.selectyourprofilepic();

Common.*logstep*("Step "+ (step++) +". Enter About YourSelf");

generalVerification = generalIndexpage.enteraboutyourself();

Common.*logstep*("Step "+ (step++) +". Enter Valid Password");

generalVerification = generalIndexpage.enterpassword();

Common.*logstep*("Step "+ (step++) +". Enter Valid Confirm Password");

generalVerification = generalIndexpage.enterconfirmpassword();

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.usernamealreadyexist()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//To verify that user get the validation message for already registered emailid

@Test(priority=0)

**public** **void** emailidalreadyexist(){

**int** step=1;

Common.*logcase*("To verify that user get the validation message for already registered emailid");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://demoqa.com/registration/");

**boolean** tf = generalVerification.verifyhomepage();

**if**(tf){

Common.*logstep*("Step "+ (step++) +". Enter Valid First Name");

generalVerification = generalIndexpage.enterfirstname();

Common.*logstep*("Step "+ (step++) +". Enter Valid Last Name");

generalVerification = generalIndexpage.enterlastname();

Common.*logstep*("Step "+ (step++) +". Select Marital Status");

generalVerification = generalIndexpage.selectmaritalstatus();

Common.*logstep*("Step "+ (step++) +". Select Hobby");

generalVerification = generalIndexpage.selecthobby();

Common.*logstep*("Step "+ (step++) +". Select Country");

generalVerification = generalIndexpage.selectcountry();

Common.*logstep*("Step "+ (step++) +". Select Date of Birth");

generalVerification = generalIndexpage.selectdateofbirth();

Common.*logstep*("Step "+ (step++) +". Enter Valid Phone Number");

generalVerification = generalIndexpage.enterphonenumber();

Common.*logstep*("Step "+ (step++) +". Enter Valid User Name");

generalVerification = generalIndexpage.enternotexistusername();

Common.*logstep*("Step "+ (step++) +". Enter Valid Emailid");

generalVerification = generalIndexpage.enterregisteremailid();

Common.*logstep*("Step "+ (step++) +". Select Your Profile Picture");

generalVerification = generalIndexpage.selectyourprofilepic();

Common.*logstep*("Step "+ (step++) +". Enter About YourSelf");

generalVerification = generalIndexpage.enteraboutyourself();

Common.*logstep*("Step "+ (step++) +". Enter Valid Password");

generalVerification = generalIndexpage.enterpassword();

Common.*logstep*("Step "+ (step++) +". Enter Valid Confirm Password");

generalVerification = generalIndexpage.enterconfirmpassword();

Common.*logstep*("Step "+ (step++) +". Click on Submit button");

generalVerification = generalIndexpage.submit\_click();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.emailidalreadyregistered()){

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//Toolsqa.com /Demosite/ Add product and checkout

//To verify that user is able to checkout with all valid fields

@Test

**public** **void** purchaseorcheckout(){

**int** step=1;

Common.*logcase*("To verify that user is able to checkout with all valid fields ");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : http://toolsqa.com");

Common.*logstep*("Step "+ (step++) +". click on Demo Sites Menu");

generalVerification = generalIndexpage.click\_on\_demosites();

Common.*logstep*("Step "+ (step++) +". click on E Commerce Demo site link");

generalVerification = generalIndexpage.click\_on\_e\_commerce\_demosite();

**boolean** tf = generalVerification.verifyhomepage\_purchase();

**if**(tf){

Common.*log*("---> E Commerce Demo site Home page is opened <---");

}

**else**

Common.*log*("---> E Commerce Demo site Home page is NOT opened <---");

Common.*logstep*("Step "+ (step++) +". Mouse Hover on Product Category");

generalVerification = generalIndexpage.mousehoveron\_product\_category();

Common.*logstep*("Step "+ (step++) +". Click on ipads link");

generalVerification = generalIndexpage.clickonipads();

Common.*logstep*("Step "+ (step++) +". Click on any product and open the product detail page");

generalVerification = generalIndexpage.productdetails();

Common.*logstep*("Step "+ (step++) +". Add product to the CART");

generalVerification = generalIndexpage.addproducttocart();

Common.*logstep*("Step "+ (step++) +". Open the Cart");

generalVerification = generalIndexpage.openthecart();

Common.*logverification*("Verify that product added to the cart");

**boolean** tf1 = generalVerification.checkproductinthecart();

**if**(tf1)

Common.*log*("---> Product is added to the cart <---");

**else**

Common.*log*("---> Product is not added to the cart <---");

Common.*logstep*("Step "+ (step++) +". Update the quantity of the product from 1 to 2");

generalVerification = generalIndexpage.updateproductquantity();

Common.*logstep*("Step "+ (step++) +". Click on Update Button");

generalVerification = generalIndexpage.clickonupdatebutton();

Common.*logverification*("Verify that the product quality has been updated in the cart");

**boolean** tf11 = generalVerification.checkproductquanity();

**if**(tf11)

Common.*log*("---> Product quantity is upadted into the cart <---");

**else**

Common.*log*("---> Product quantity is NOT upadted into the cart <---");

Common.*logstep*("Step "+ (step++) +". Click on Continue Button");

generalVerification = generalIndexpage.clickoncomtinuebutton();

Common.*logstep*("Step "+ (step++) +". Enter Email Address in Billing Information");

generalVerification = generalIndexpage.enteremailaddress();

Common.*logstep*("Step "+ (step++) +". Enter Your billing/contact details");

generalVerification = generalIndexpage.demo\_enterfirstname();

generalVerification = generalIndexpage.demo\_enterelastname();

generalVerification = generalIndexpage.demo\_enteraddress();

generalVerification = generalIndexpage.demo\_enterecity();

generalVerification = generalIndexpage.demo\_enterestate\_province();

generalVerification = generalIndexpage.demo\_entercountry();

generalVerification = generalIndexpage.demo\_enterpostalcode();

generalVerification = generalIndexpage.demo\_enterphonenumber();

Common.*logstep*("Step "+ (step++) +". Enter Shipping Address Information/Details");

generalVerification = generalIndexpage.entershippingaddress();

Common.*logstep*("Step "+ (step++) +". Click on Purchase Button");

generalVerification = generalIndexpage.clickonpurchasebutton();

**if**(generalVerification.purchaseverification()){

Common.*logstep*("Order placed successfully");

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**{

Common.*logstep*("Order is NOT placed successfully");

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//Thread and Shirts

//To verify that user is able to add the product into the cart and do checkout with all valid fields

@Test(priority=0)

**public** **void** addproducttocartandcheckout(){

**int** step=1;

Common.*logcase*("To verify that user is able to add the product into the cart and do checkout with all valid fields ");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : https://www.threadsandshirts.com/");

**boolean** tf = generalVerification.verifyhomepage\_threadandshirts();

**if**(tf)

Common.*log*("--> home page is opened <--");

**else**

Common.*log*("--> home page is not opened <--");

Common.*logstep*("Step "+ (step++) +". Mouse hover over shop menu");

generalVerification = generalIndexpage.mousehoveronshop();

Common.*logstep*("Step "+ (step++) +". Click on accessories");

generalVerification = generalIndexpage.clickonaccessories();

Common.*logstep*("Step "+ (step++) +". Click on Product");

generalVerification = generalIndexpage.clickonproduct();

Common.*logstep*("Step "+ (step++) +". Click on Add to cart button");

generalVerification = generalIndexpage.clickonaddtocartbutton();

Common.*logstep*("Step "+ (step++) +". Click on continues shopping button");

generalVerification = generalIndexpage.clickoncontinuesshoppingbutton();

Common.*logstep*("Step "+ (step++) +". Click on Add to cart button Again");

generalVerification = generalIndexpage.clickonaddtocartbutton();

Common.*logstep*("Step "+ (step++) +". Click on CheckOut button");

generalVerification = generalIndexpage.clickoncheckoutbutton();

Common.*logstep*("Step "+ (step++) +". Enter Promotion Code");

generalVerification = generalIndexpage.enterpromotioncode();

Common.*logstep*("Step "+ (step++) +". Click on Apply button");

generalVerification = generalIndexpage.clickonapplybutton();

Common.*logstep*("Step "+ (step++) +". Close the warning message");

generalVerification = generalIndexpage.closethemessage();

Common.*logstep*("Step "+ (step++) +". Click on checkout as guest");

generalVerification = generalIndexpage.clickonchechoutasguest();

Common.*logstep*("Step "+ (step++) +". Click on checkoutasguest button");

generalVerification = generalIndexpage.clickonchechoutasguestbutton();

Common.*logstep*("Step "+ (step++) +". Enter First Name");

generalVerification = generalIndexpage.enterfirstnamets();

Common.*logstep*("Step "+ (step++) +". Enter Last Name");

generalVerification = generalIndexpage.enterLastnamets();

Common.*logstep*("Step "+ (step++) +". Enter Email Address");

generalVerification = generalIndexpage.enteremailaddressts();

Common.*logstep*("Step "+ (step++) +". Enter Phone Number");

generalVerification = generalIndexpage.enterphonenumberts();

Common.*logstep*("Step "+ (step++) +". Enter Address");

generalVerification = generalIndexpage.enteraddressts();

Common.*logstep*("Step "+ (step++) +". Enter City");

generalVerification = generalIndexpage.entercityts();

Common.*logstep*("Step "+ (step++) +". Select the Country");

generalVerification = generalIndexpage.entercountryts();

Common.*logstep*("Step "+ (step++) +". Enter State");

generalVerification = generalIndexpage.enterstatets();

Common.*logstep*("Step "+ (step++) +". Enter Postal Code");

generalVerification = generalIndexpage.enterpostalcodets();

Common.*logstep*("Step "+ (step++) +". Shipping Address is same as Billing Address");

generalVerification = generalIndexpage.billingandshippingsameclickts();

Common.*logstep*("Step "+ (step++) +". Click on continue Button");

generalVerification = generalIndexpage.clickoncontinuebuttonts();

Common.*logstep*("Step "+ (step++) +". Click on continue Button of Shipping Options");

generalVerification = generalIndexpage.clickoncontinuebuttonshippingoptionsts();

Common.*logstep*("Step "+ (step++) +". Click on Cash On Delivery Options");

generalVerification = generalIndexpage.clickoncashondeliveryts();

Common.*logstep*("Step "+ (step++) +". Click on Submit My Order Button");

generalVerification = generalIndexpage.clicksubmitmyorderbuttonts();

**if**(generalVerification.orderplacedsuccessfully()){

Common.*logstep*("Order placed successfully");

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**{

Common.*logstep*("Order is NOT placed successfully");

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

//E-Signature

//Login and Home page Verification

@Test(priority=0)

**public** **void** esign\_loginwithvalidcredentials(){

**int** step=1;

Common.*logcase*("To verify that user is able to login with valid credentials");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : https://go-env-syd1.clickandsign.net:11443/dashboard/");

**boolean** tf = generalVerification.verifyesignaturehomepage();

**if**(tf){

Common.*log*("---> Home page is open <---");

Common.*logstep*("Step "+ (step++) +". Enter Username");

generalVerification = generalIndexpage.esign\_enterusername();

Common.*logstep*("Step "+ (step++) +". Enter Password");

generalVerification = generalIndexpage.esign\_password();

Common.*logstep*("Step "+ (step++) +". Click on Login Button");

generalVerification = generalIndexpage.esign\_clickonloginbutton();

}

**else**

Common.*log*("---> Home page is not open <---");

**if**(generalVerification.esign\_loginpageverification()){

Common.*logstep*("Logged in successfully!!");

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{ Common.*logstep*("Invalid password or Username!!");

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

@Test(priority=0)

**public** **void** esign\_createpackage\_borroweronly(){

**int** step=1;

Common.*logcase*("To verify that user is able to login with valid credentials");

Common.*logcase*("--------------------------------------------------------------------------");

Common.*logstep*("Step "+ (step++) +". open : https://go-env-syd1.clickandsign.net:11443/dashboard/");

**boolean** tf = generalVerification.verifyesignaturehomepage();

**if**(tf){

Common.*log*("---> Home page is open <---");

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**,"---> Home page is open <---");

}

**else**

{ Common.*log*("---> Home page is not open <---");

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**,"---> Home page is not open <---");

}

Common.*logstep*("Step "+ (step++) +". Enter Username");

generalVerification = generalIndexpage.esign\_enterusername();

Common.*logstep*("Step "+ (step++) +". Enter Password");

generalVerification = generalIndexpage.esign\_password();

Common.*logstep*("Step "+ (step++) +". Click on Login Button");

generalVerification = generalIndexpage.esign\_clickonloginbutton();

**if**(generalVerification.esign\_loginpageverification()){

Common.*log*("Logged in successfully!!");

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**,"Logged in successfully!!");

}

**else**

{ Common.*log*("Invalid password or Username!!");

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**,"Invalid password or Username!!");

}

Common.*logstep*("Step "+ (step++) +". Select the Workflow");

generalVerification=generalIndexpage.esign\_selectworkflow();

Common.*logstep*("Step "+ (step++) +". Click on Create button");

generalVerification=generalIndexpage.esign\_clickoncreatebutton();

Common.*logstep*("Step "+ (step++) +". Enter Package Reference");

generalVerification=generalIndexpage.esign\_enterpackagereference();

Common.*logstep*("Step "+ (step++) +". Enter First Name");

generalVerification=generalIndexpage.esign\_enterfirstname();

Common.*logstep*("Step "+ (step++) +". Enter Last Name");

generalVerification=generalIndexpage.esign\_enterlastname();

Common.*logstep*("Step "+ (step++) +". Enter Email Address");

generalVerification=generalIndexpage.esign\_enteremailaddress();

Common.*logstep*("Step "+ (step++) +". Enter Phone Number");

generalVerification=generalIndexpage.esign\_enterphonenumber();

Common.*logstep*("Step "+ (step++) +". Select WAFR Attachments");

generalVerification=generalIndexpage.esign\_numberofducuments();

Common.*logstep*("Step "+ (step++) +". Click on Create Button");

generalVerification=generalIndexpage.esign\_packagecreatebutton();

Common.*logstep*("Step "+ (step++) +". Click on Cloud Service Agreement");

generalVerification=generalIndexpage.esign\_clickoncloudserviceaggrement();

Common.*logstep*("Step "+ (step++) +". Upload the Cloud Service Agreement");

generalVerification=generalIndexpage.esign\_uploadagreement();

Common.*logstep*("Step "+ (step++) +". Click on SendNow Button");

generalVerification=generalIndexpage.esign\_clickonsendnowbutton();

Common.*logstep*("Step "+ (step++) +". Confirm the package with valid details");

**boolean** referencetf=generalVerification.esign\_packagenameverification();

**if**(referencetf){

Common.*log*("--> Package Created Successfully!! <--");

Common.*logStatus*("Pass");

Assert.*assertTrue*(**true**);

}

**else**

{ Common.*log*("--> package is not created!! <--");

Common.*logStatus*("Fail");

Assert.*assertTrue*(**false**);

}

}

}//ends of class

GeneralIndexPage

**package** com.indexpage;

**import** java.io.File;

**import** java.util.List;

**import** java.util.Set;

**import** org.openqa.selenium.By;

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

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

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

**import** org.openqa.selenium.support.FindBy;

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

**import** com.init.AbstractPage;

**import** com.init.Common;

**import** com.utility.TestData;

**import** com.verification.GeneralVerification;

**public** **class** GeneralIndexpage **extends** AbstractPage {

**public** GeneralIndexpage(WebDriver driver) {

**super**(driver);

}

//Variables

String root=**null**;

**public** **static** String *productname*= **null**;

**public** **static** String *packagereference*=**null**;

**public** **static** String *firstname*=**null**;

**public** **static** String *lastname*=**null**;

**public** **static** **int** *numberofdocuments*;

//profile picture path

//XPATH for all Registration Form Fields

@FindBy (xpath = "//input[@name='first\_name']") **private** WebElement first\_name\_txt;

@FindBy (xpath ="//input[@name='last\_name']") **private** WebElement last\_name\_txt;

@FindBy (xpath="//ul/li[2]/div/div[@class='radio\_wrap']/input" ) **private** List<WebElement> marital\_status;

@FindBy (xpath = "//ul/li[3]/div/div[@class='radio\_wrap']/input") **private** List<WebElement> hobby;

@FindBy (xpath="//select[@id='dropdown\_7']") **private** WebElement country;

@FindBy (xpath="//select[@id='dropdown\_7']/option") **private** List<WebElement> option\_country;

@FindBy (xpath = "//select[@id='dd\_date\_8']") **private** WebElement birth\_day;

@FindBy (xpath = "//select[@id='dd\_date\_8']/option") **private** List<WebElement> option\_birth\_day;

@FindBy (xpath="//select[@id='mm\_date\_8']") **private** WebElement birth\_month;

@FindBy (xpath="//select[@id='mm\_date\_8']/option") **private** List<WebElement> option\_birth\_month;

@FindBy (xpath="//select[@id='yy\_date\_8']") **private** WebElement birth\_year;

@FindBy (xpath="//select[@id='yy\_date\_8']/option") **private** List<WebElement> option\_birth\_year;

@FindBy(xpath=".//input[@id='phone\_9']") **private** WebElement phone\_no;

@FindBy (xpath="//input[@id='username']") **private** WebElement username;

@FindBy (xpath="//input[@id='email\_1']")**private** WebElement email;

@FindBy (xpath="//input[@id='profile\_pic\_10']") **private** WebElement yourprofile\_picture;

@FindBy(xpath = "//textarea[@name='description']") **private** WebElement aboutyourself;

@FindBy (xpath=".//input[@id='password\_2']") **private** WebElement password\_txt;

@FindBy (xpath=".//input[@id='confirm\_password\_password\_2']") **private** WebElement confirm\_password\_txt;

@FindBy(xpath = ".//input[@name='pie\_submit']") **private** WebElement submit\_btn;

//Demo sites

@FindBy (xpath="//span/span/span[contains(text(),'DEMO SITES')]") **private** WebElement demosites;

@FindBy(xpath="//span/span/span[contains(text(),'E-Commerce Demo Site')]") **private** WebElement e\_commerce\_demo\_site;

@FindBy (xpath="//a[contains(text(),'Product Category')]") **private** WebElement mousehoveron\_product\_category;

@FindBy (xpath="//a[contains(text(),'iPads')]") **private** WebElement ipads;

@FindBy (xpath="//a[contains(text(),'Apple iPad 2 16GB, Wi-Fi, 9.7in – Black')]") **private** WebElement clickon\_apple\_product;

@FindBy (xpath="//input[@value='Add To Cart']") **private** WebElement addproducttocart;

@FindBy (xpath="//em[contains(text(),'item')]") **private** WebElement opencart;

@FindBy (xpath="//\*[@id='checkout\_page\_container']/div[1]/table/tbody/tr[2]/td[3]/form/input[1]") **private** WebElement updatequantity;

@FindBy (xpath="//span[contains(text(),'Continue')]") **private** WebElement clickoncontinue;

@FindBy(xpath="//input[@title='billingemail']") **private** WebElement emailaddress;

@FindBy (xpath="//input[@title='billingfirstname']") **private** WebElement demo\_firstname;

@FindBy(xpath="//input[@title='billinglastname']")**private** WebElement demo\_lastname;

@FindBy (xpath="//textarea[@title='billingaddress']")**private** WebElement demo\_address;

@FindBy (xpath="//input[@title='billingcity']")**private** WebElement demo\_city;

@FindBy (xpath="//input[@title='billingstate']")**private** WebElement demo\_state\_province;

@FindBy(xpath="//select[@title='billingcountry']")**private** WebElement demo\_country;

@FindBy (xpath="//input[@title='billingpostcode']")**private** WebElement demo\_postalcode;

@FindBy (xpath="//input[@title='billingphone']")**private** WebElement demo\_phonenumber;

@FindBy (xpath="//input[@id='shippingSameBilling']")**private** WebElement demo\_shipping\_address;

@FindBy(xpath="//input[@class='make\_purchase wpsc\_buy\_button']")**private** WebElement demo\_clickonpurchase\_button;

@FindBy(xpath="//input[@value='Update']")**private** WebElement click\_on\_update;

//Thread and Shirts

@FindBy(xpath="//a[contains(text(),'SHOP')]")**private** WebElement shop;

@FindBy (xpath="//a/strong[contains(text(),'Accessories')]")**private** WebElement accessories;

@FindBy(xpath="//a[contains(text(),'Black Dobby')]")**private** WebElement product;

@FindBy(xpath="//div[1]/div/button[contains(text(),'Add to cart')]")**private** WebElement addtocartbutton;

@FindBy(xpath="//\*[@id='top']/div[8]/div/div[2]/div[1]/button")**private** WebElement continueshoppingbutton;

@FindBy(xpath="//\*[@id='top']/div[8]/div/div[2]/div[2]/a")**private** WebElement CheckOutbutton;

@FindBy(xpath="//input[@id='coupon\_field']")**private** WebElement promotioncode;

@FindBy(xpath="//td/form/div/span/button")**private** WebElement clickonapply;

@FindBy(xpath="//div//button[text()='×']")**private** WebElement closemessage;

@FindBy(xpath="//label//input[@id='checkout\_type\_guest']")**private** WebElement clickoncheckoutasguest;

@FindBy(xpath="//button[contains(text(),'Checkout as guest')]")**private** WebElement clickoncheckoutasguestbutton;

@FindBy(xpath="//input[@name='user\_data[b\_firstname]']")**private** WebElement firstname\_ts;

@FindBy(xpath="//input[@name='user\_data[b\_lastname]']")**private** WebElement lastname\_ts;

@FindBy(xpath="//div//div[3]//input[@name='user\_data[email]']")**private** WebElement emailaddress\_ts;

@FindBy(xpath="//input[@name='user\_data[b\_phone]']")**private** WebElement phonenumber\_ts;

@FindBy(xpath="//input[@name='user\_data[b\_address]']")**private** WebElement address\_ts;

@FindBy(xpath="//input[@name='user\_data[b\_city]']")**private** WebElement city\_ts;

@FindBy(xpath="//select[@name='user\_data[b\_country]']")**private** WebElement country\_ts;

@FindBy(xpath="//input[@name='user\_data[b\_state]']")**private** WebElement state\_ts;

@FindBy(xpath="//input[@name='user\_data[b\_zipcode]']")**private** WebElement postalcodets;

@FindBy(xpath="//input[@id='sw\_sa\_suffix\_yes']")**private** WebElement billingandshippingsameclickts;

@FindBy(xpath="//button[contains(text(),'Continue')]")**private** WebElement clickoncontinuebuttonts;

@FindBy(xpath="//strong[contains(text(),'Cash On Delivery')]/..//input")**private** WebElement cashondeliveryts;

@FindBy(xpath="//button[contains(text(),'Submit my order')]")**private** WebElement submitmyorderbuttonts;

//E-Signature

//Login

@FindBy(xpath="//input[@name='username']")**private** WebElement username\_esign;

@FindBy(xpath="//input[@name='password']")**private** WebElement password\_esign;

@FindBy(xpath="//button[@title='login']")**private** WebElement login\_esign;

//Select the workflow and create the package

@FindBy(xpath="//select[@id='workflows']")**private** WebElement selectworkflow\_esign;

//IBM Digital Contracts Demo v. 1

@FindBy(xpath="//button[@title='create']")**private** WebElement createbutton\_esign;

@FindBy(xpath="//input[@id='packageReference']")**private** WebElement packagereference\_esign;

@FindBy(xpath="//label[contains(text(),'First Name\*')]/../input[position()<last()-2]")**private** WebElement firstname\_esign;

@FindBy(xpath="//label[contains(text(),'Last Name\*')]/../input[position()=last()-2]")**private** WebElement lastname\_esign;

@FindBy(xpath="//label[contains(text(),'Last Name\*')]/../input[position()=last()-1]")**private** WebElement emailaddress\_esign;

@FindBy(xpath="//label[contains(text(),'Last Name\*')]/../input[position()=last()]")**private** WebElement mobile\_esign;

@FindBy(xpath="//span[contains(text(),'Transfer of Land')]/..//input")**private** List<WebElement> numberofdocuments\_esign;

@FindBy(xpath="//span/button[@title='create']")**private** WebElement packagecreatebutton;

@FindBy(xpath="//div//span[contains(text(),'Cloud Services Agreement')]/../input")**private** WebElement cloudserviceaggrement\_esign;

@FindBy(xpath="//input[@type='file']")**private** WebElement uploadtheaggreement\_esign;

@FindBy(xpath="//button[@id='sendPackage']")**private** WebElement clickonsendnowbutton\_esign;

**public** GeneralVerification submit\_click() {

// **TODO** Auto-generated method stub

submit\_btn.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterinvalidphonenumber() {

// **TODO** Auto-generated method stub

phone\_no.click();

String invalidphonenumber = TestData.*invalidphonenumber*();

Common.*log*("--->Enter Invalid Phone Number = "+invalidphonenumber+ " <---");

phone\_no.sendKeys(invalidphonenumber);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterinvalidpassword() {

// **TODO** Auto-generated method stub

password\_txt.click();

String invalidpassword = TestData.*invalidpassword*();

Common.*log*("--->Enter Invalid Password = "+invalidpassword+ " <---");

password\_txt.sendKeys(invalidpassword);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterpasswordnotmatch() {

// **TODO** Auto-generated method stub

password\_txt.click();

String password = TestData.*donotmatchpassword*();

Common.*log*("--->Enter Password = "+password+ " <---");

password\_txt.sendKeys(password);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterconfirmpasswordnotmatch() {

// **TODO** Auto-generated method stub

confirm\_password\_txt.click();

String password = TestData.*donotmatchconfirmpassword*();

Common.*log*("--->Enter Confrim Password(do not match with password) = "+password+ " <---");

confirm\_password\_txt.sendKeys(password);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterinvalidemailid() {

// **TODO** Auto-generated method stub

email.click();

String invalidemailid = TestData.*invalidemailid*();

Common.*log*("--->Enter Invalid Emailid = "+invalidemailid+" <---");

email.sendKeys(invalidemailid);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterfirstname() {

// **TODO** Auto-generated method stub

first\_name\_txt.click();

String firstname = TestData.*firstname*();

Common.*log*("--->Enter Invalid Emailid = "+firstname+" <---");

first\_name\_txt.sendKeys(firstname);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterlastname() {

// **TODO** Auto-generated method stub

last\_name\_txt.click();

String lastname = TestData.*lastname*();

last\_name\_txt.sendKeys(lastname);

Common.*log*("--->Enter Invalid Emailid = "+lastname+" <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification selectmaritalstatus() {

// **TODO** Auto-generated method stub

String value="value";

**for**(WebElement status : marital\_status){

**if**(status.getAttribute(value).trim().equals("married"))

status.click();

Common.*log*("--->Selected Maritalstatus = "+status.getAttribute(value).trim()+" <---");

Common.*pause*(2);

}

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification selecthobby() {

// **TODO** Auto-generated method stub

String value="value";

**for**(WebElement hobby\_status : hobby){

hobby\_status.click();

Common.*log*("--->Selected Hobby = "+hobby\_status.getAttribute(value).trim()+" <---");

Common.*pause*(2);

}

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification selectcountry() {

// **TODO** Auto-generated method stub

Select country\_dropdown= **new** Select(country);

**for**(**int** i=0;i<option\_country.size();i++)

{

**if**(i<3){

country\_dropdown.selectByIndex(i);

Common.*pause*(2);

}

**else**

**break**;

}

country\_dropdown.selectByVisibleText("India");

Common.*log*("---> Selected Country = Indian <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification selectdateofbirth() {

// **TODO** Auto-generated method stub

Select day\_dropdown= **new** Select(birth\_day);

**for**(**int** i=0;i<option\_birth\_day.size();i++)

{

**if**(i<3){

day\_dropdown.selectByIndex(i);

Common.*pause*(2);

}

**else**

**break**;

}

day\_dropdown.selectByVisibleText("30");

Select month\_dropdown= **new** Select(birth\_month);

**for**(**int** i=0;i<option\_birth\_month.size();i++)

{

**if**(i<3){

month\_dropdown.selectByIndex(i);

Common.*pause*(2);

}

**else**

**break**;

}

month\_dropdown.selectByVisibleText("12");

Select year\_dropdown= **new** Select(birth\_year);

**for**(**int** i=0;i<option\_birth\_year.size();i++)

{

**if**(i<3){

year\_dropdown.selectByIndex(i);

Common.*pause*(2);

}

**else**

**break**;

}

year\_dropdown.selectByVisibleText("1982");

Common.*log*("--->Selected Date Of Birth = 12/30/1982 <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterphonenumber() {

// **TODO** Auto-generated method stub

phone\_no.click();

String validphonenumber = TestData.*phonenumber*();

Common.*log*("--->Enter Valid Phone Number = "+validphonenumber+ " <---");

phone\_no.sendKeys(validphonenumber);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterusername() {

// **TODO** Auto-generated method stub

username.click();

String validusername = TestData.*username*();

Common.*log*("--->Enter Valid User Name = "+validusername+ " <---");

username.sendKeys(validusername);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enteremailid() {

// **TODO** Auto-generated method stub

email.click();

String validemailid = TestData.*emailid*();

Common.*log*("--->Enter Valid Emailid = "+validemailid+ " <---");

email.sendKeys(validemailid);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification selectyourprofilepic() {

// **TODO** Auto-generated method stub

String profile\_picture\_path = "C:/Users/Public/Pictures/Sample Pictures/Tulips.jpg";

yourprofile\_picture.sendKeys(profile\_picture\_path);

Common.*log*("--->Selected Profile Picture From = "+profile\_picture\_path+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enteraboutyourself() {

// **TODO** Auto-generated method stub

aboutyourself.click();

aboutyourself.sendKeys(TestData.*aboutyourself*());

Common.*log*("--->About YourSelf = "+TestData.*aboutyourself*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterpassword() {

// **TODO** Auto-generated method stub

password\_txt.click();

String validpassword = TestData.*password*();

Common.*log*("--->Enter Invalid Password = "+validpassword+ " <---");

password\_txt.sendKeys(validpassword);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterconfirmpassword() {

// **TODO** Auto-generated method stub

confirm\_password\_txt.click();

String password = TestData.*confirmpassword*();

Common.*log*("--->Enter Confrim Password = "+password+ " <---");

confirm\_password\_txt.sendKeys(password);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterexistusername() {

// **TODO** Auto-generated method stub

username.click();

String existvalidusername = TestData.*existusername*();

Common.*log*("--->Enter Valid User Name = "+existvalidusername+ " <---");

username.sendKeys(existvalidusername);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enternotregisteremailid() {

// **TODO** Auto-generated method stub

email.click();

String validemailid = TestData.*notregisteremailid*();

Common.*log*("--->Enter Valid Emailid = "+validemailid+ " <---");

email.sendKeys(validemailid);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enternotexistusername() {

// **TODO** Auto-generated method stub

username.click();

String notexistvalidusername = TestData.*notexistusername*();

Common.*log*("--->Enter Valid User Name = "+notexistvalidusername+ " <---");

username.sendKeys(notexistvalidusername);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterregisteremailid() {

// **TODO** Auto-generated method stub

email.click();

String validemailid = TestData.*registeremailid*();

Common.*log*("--->Enter Valid Emailid = "+validemailid+ " <---");

email.sendKeys(validemailid);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification click\_on\_demosites() {

// **TODO** Auto-generated method stub

Common.*pause*(8);

root = *driver*.getWindowHandle();

demosites.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification click\_on\_e\_commerce\_demosite() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

e\_commerce\_demo\_site.click();

Common.*pause*(8);

Set<String> handles = *driver*.getWindowHandles();

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

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

**for**(String handle : handles){

**if**(!root.equals(handle))

*driver*.switchTo().window(handle);

}

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification mousehoveron\_product\_category() {

// **TODO** Auto-generated method stub

Common.*pause*(10);

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

hover.moveToElement(mousehoveron\_product\_category).build().perform();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonipads() {

// **TODO** Auto-generated method stub

Common.*pause*(8);

ipads.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification productdetails() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

clickon\_apple\_product.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification addproducttocart() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

addproducttocart.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification openthecart() {

// **TODO** Auto-generated method stub

Common.*pause*(8);

opencart.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification updateproductquantity() {

// **TODO** Auto-generated method stub

Common.*pause*(8);

updatequantity.click();

updatequantity.clear();

updatequantity.sendKeys("2");

Common.*pause*(5);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickoncomtinuebutton() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

clickoncontinue.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enteremailaddress() {

// **TODO** Auto-generated method stub

Common.*pause*(10);

emailaddress.click();

String validemailid = TestData.*emailid*();

Common.*log*("--->Enter Valid Emailid = "+validemailid+ " <---");

emailaddress.sendKeys(validemailid);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_enterfirstname() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

demo\_firstname.click();

Common.*pause*(1);

demo\_firstname.sendKeys(TestData.*demo\_firstname*());

Common.*log*("--->Enter Valid FirstName = "+TestData.*demo\_firstname*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_enterelastname() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

demo\_lastname.click();

Common.*pause*(1);

demo\_lastname.sendKeys(TestData.*demo\_lastname*());

Common.*log*("--->Enter Valid LastName = "+TestData.*demo\_lastname*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_enteraddress() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

demo\_address.click();

Common.*pause*(1);

demo\_address.sendKeys(TestData.*demo\_address*());

Common.*log*("--->Enter Valid Address = "+TestData.*demo\_address*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_enterecity() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

demo\_city.click();

Common.*pause*(1);

demo\_city.sendKeys(TestData.*demo\_city*());

Common.*log*("--->Enter Valid City = "+TestData.*demo\_city*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_enterestate\_province() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

demo\_state\_province.click();

Common.*pause*(1);

demo\_state\_province.sendKeys(TestData.*demo\_state\_province*());

Common.*log*("--->Enter Valid state/Province = "+TestData.*demo\_state\_province*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_entercountry() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

Select country = **new** Select(demo\_country);

Common.*pause*(1);

country.selectByVisibleText(TestData.*demo\_country*());

Common.*log*("--->Enter Valid state/Province = "+TestData.*demo\_country*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_enterpostalcode() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

demo\_postalcode.click();

Common.*pause*(1);

demo\_postalcode.sendKeys(TestData.*demo\_postalcode*());

Common.*log*("--->Enter Valid state/Province = "+TestData.*demo\_postalcode*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification demo\_enterphonenumber() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

demo\_phonenumber.click();

Common.*pause*(1);

demo\_phonenumber.sendKeys(TestData.*demo\_phonenumber*());

Common.*log*("--->Enter Valid state/Province = "+TestData.*demo\_phonenumber*()+ " <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification entershippingaddress() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

demo\_shipping\_address.click();

Common.*log*("---> Chose Shipping Address same as Billing Address <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonpurchasebutton() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

demo\_clickonpurchase\_button.click();

Common.*log*("---> Click on purchase button <---");

Common.*pause*(8);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonupdatebutton() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

click\_on\_update.click();

Common.*log*("---> Click on Update button <---");

**return** **new** GeneralVerification(*driver*);

}

//Threads and Shirts

**public** GeneralVerification mousehoveronshop() {

// **TODO** Auto-generated method stub

Common.*pause*(10);

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

**if**(shop.isDisplayed()){

act.moveToElement(shop).build().perform();

Common.*log*("--> mouse hover on shop menu <--");

Common.*pause*(10);

}

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonaccessories() {

// **TODO** Auto-generated method stub

Common.*pause*(10);

**if**(accessories.isDisplayed()){

accessories.click();

Common.*log*("--> click on accessories link <--");

}

**else**

Common.*log*("--> accessories link is not displayed <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonproduct() {

// **TODO** Auto-generated method stub

Common.*pause*(8);

**if**(product.isDisplayed()){

*productname* = product.getText();

product.click();

Common.*log*("--> Click on the product <--");

}

**else**

Common.*log*("--> didn't click on the product <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonaddtocartbutton() {

// **TODO** Auto-generated method stub

Common.*pause*(8);

**if**(addtocartbutton.isDisplayed()){

Common.*log*("--> Click on Add To Cart button <--");

addtocartbutton.click();

}

**else**

Common.*log*("--> didn't Click on Add To Cart button <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickoncontinuesshoppingbutton() {

// **TODO** Auto-generated method stub

Common.*waitForElement*(*driver*, "//\*[@id='top']/div[8]/div/div[2]/div[1]/button");

Common.*pause*(3);

Common.*log*("--> click on : " + continueshoppingbutton.getText() + " <--");

continueshoppingbutton.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickoncheckoutbutton() {

// **TODO** Auto-generated method stub

Common.*waitForElement*(*driver*, "//\*[@id='top']/div[8]/div/div[2]/div[2]/a");

Common.*pause*(3);

Common.*log*("--> click on : " + CheckOutbutton.getText() + " <--");

CheckOutbutton.click();

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterpromotioncode() {

// **TODO** Auto-generated method stub

Common.*pause*(7);

**if**(promotioncode.isDisplayed()){

promotioncode.sendKeys("BESTDAD15");

Common.*log*("--> Enter Promotion Code : BESTDAD15 <--");

}

**else**

Common.*log*("--> Not Enter Promotion Code :---- <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonapplybutton() {

// **TODO** Auto-generated method stub

Common.*pause*(4);

**if**(clickonapply.isDisplayed()){

clickonapply.click();

Common.*log*("--> Click on Apply Button <--");

}

**else**

Common.*log*("--> Didn't Click on Apply Button <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification closethemessage() {

// **TODO** Auto-generated method stub

Common.*pause*(5);

**if**(closemessage.isDisplayed()){

closemessage.click();

Common.*log*("--> Warning Message closed <--");

}

**else**

Common.*log*("--> Warning Message closed <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonchechoutasguest() {

// **TODO** Auto-generated method stub

Common.*pause*(4);

**if**(clickoncheckoutasguest.isDisplayed()){

clickoncheckoutasguest.click();

Common.*log*("--> Click on checkout as guest user <--");

}

**else**

Common.*log*("--> Didn't Click on checkout as guest user <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickonchechoutasguestbutton() {

// **TODO** Auto-generated method stub

Common.*pause*(4);

**if**(clickoncheckoutasguestbutton.isDisplayed()){

clickoncheckoutasguestbutton.click();

Common.*log*("--> Click on checkout as guest user button <--");

}

**else**

Common.*log*("--> Didn't Click on checkout as guest user button <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterfirstnamets() {

// **TODO** Auto-generated method stub

Common.*pause*(4);

**if**(firstname\_ts.isDisplayed()){

firstname\_ts.sendKeys(TestData.*ts\_firstname*());

Common.*log*("--> Enter First Name : " + TestData.*ts\_firstname*()+" <--");

}

**else**

Common.*log*("--> First Name is not Entered <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterLastnamets() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(lastname\_ts.isDisplayed()){

lastname\_ts.sendKeys(TestData.*ts\_lastname*());

Common.*log*("--> Enter Last Name : " + TestData.*ts\_lastname*()+" <--");

}

**else**

Common.*log*("--> Last Name is not Entered <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enteremailaddressts() {

// **TODO** Auto-generated method stub

Common.*pause*(4);

**if**(emailaddress\_ts.isDisplayed()){

emailaddress\_ts.sendKeys(TestData.*ts\_emailaddress*());

Common.*log*("--> Enter Email Address :" +TestData.*ts\_emailaddress*() + " <--");

}

**else**

Common.*log*("--> Email Address is not entered <--");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterphonenumberts() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(phonenumber\_ts.isDisplayed()){

phonenumber\_ts.sendKeys(TestData.*ts\_phone*());

Common.*log*("--> Enter Phone Number :"+TestData.*ts\_phone*()+" <---");

}

**else**

Common.*log*("--> Phone Number is not entered <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enteraddressts() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(address\_ts.isDisplayed()){

address\_ts.sendKeys(TestData.*ts\_address*());

Common.*log*("--> Enter Address :"+TestData.*ts\_address*()+" <---");

}

**else**

Common.*log*("--> Address is not entered <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification entercityts() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(city\_ts.isDisplayed()){

city\_ts.sendKeys(TestData.*ts\_city*());

Common.*log*("--> Enter City :"+TestData.*ts\_city*()+" <---");

}

**else**

Common.*log*("--> City is not entered <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification entercountryts() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(country\_ts.isDisplayed()){

Select country = **new** Select(country\_ts);

country.selectByVisibleText(TestData.*ts\_country*());

//country\_ts.sendKeys(TestData.ts\_city());

Common.*log*("--> Select Country :"+TestData.*ts\_country*()+" <---");

}

**else**

Common.*log*("--> Country is not Selected <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterstatets() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(state\_ts.isDisplayed()){

state\_ts.sendKeys(TestData.*ts\_state*());

Common.*log*("--> Enter State :"+TestData.*ts\_state*()+" <---");

}

**else**

Common.*log*("--> State is not Entered <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification enterpostalcodets() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(postalcodets.isDisplayed()){

postalcodets.sendKeys(TestData.*ts\_postalcode*());

Common.*log*("--> Enter Postal Code :"+TestData.*ts\_postalcode*()+" <---");

}

**else**

Common.*log*("--> Postal Code is not entered <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification billingandshippingsameclickts() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(billingandshippingsameclickts.isDisplayed()){

billingandshippingsameclickts.click();

Common.*log*("--> Shipping address is same as billing address <---");

}

**else**

Common.*log*("--> Shipping address is NOT same as billing address <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickoncontinuebuttonts() {

// **TODO** Auto-generated method stub

Common.*pause*(7);

**if**(clickoncontinuebuttonts.isDisplayed()){

clickoncontinuebuttonts.click();

Common.*log*("--> Click on Continue Button <---");

}

**else**

Common.*log*("--> Didn't Click on continue button <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickoncashondeliveryts() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(cashondeliveryts.isDisplayed()){

cashondeliveryts.click();

Common.*log*("--> Click on Cash On Delivery Option <---");

}

**else**

Common.*log*("--> Didn't Click on Cash On Delivery Option <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clicksubmitmyorderbuttonts() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(submitmyorderbuttonts.isDisplayed()){

submitmyorderbuttonts.click();

Common.*log*("--> Click on Submit My Order button <---");

}

**else**

Common.*log*("--> Didn't Click on Submit My Order button <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification clickoncontinuebuttonshippingoptionsts() {

// **TODO** Auto-generated method stub

WebElement move = *driver*.findElement(By.*xpath*("//a[contains(text(),'Billing and Shipping Address')]"));

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

act.moveToElement(move).build().perform();

**if**(clickoncontinuebuttonts.isDisplayed()){

clickoncontinuebuttonts.click();

Common.*log*("--> Click on Continue Button <---");

}

**else**

Common.*log*("--> Didn't Click on continue button <---");

**return** **new** GeneralVerification(*driver*);

}

//E-Signature

**public** GeneralVerification esign\_enterusername() {

// **TODO** Auto-generated method stub

Common.*pause*(4);

**if**(username\_esign.isDisplayed()){

username\_esign.sendKeys(TestData.*esign\_username*());

Common.*log*("--> Enter Username :" +TestData.*esign\_username*()+"<---");

}

**else**

Common.*log*("--> Username textfield is not available <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_password() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

**if**(password\_esign.isDisplayed()){

password\_esign.sendKeys(TestData.*esign\_password*());

Common.*log*("--> Enter password :" +TestData.*esign\_password*()+"<---");

}

**else**

Common.*log*("--> Password textfield is not available <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_clickonloginbutton() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

**if**(login\_esign.isDisplayed()){

login\_esign.click();

Common.*log*("--> clicked on Login Button <---");

}

**else**

Common.*log*("--> Not Able to click on Login Button <---");

Common.*pause*(8);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_selectworkflow() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(selectworkflow\_esign.isDisplayed()){

Select workflow = **new** Select(selectworkflow\_esign);

workflow.selectByVisibleText("IBM Digital Contracts Demo v. 1");

Common.*log*("--> Selected the work Flow:" + "IBM Digital Contracts Demo v. 1" + " <---");

}

**else**

Common.*log*("--> Workflow is not selected <---");

Common.*pause*(3);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_clickoncreatebutton() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(createbutton\_esign.isDisplayed()){

createbutton\_esign.click();

Common.*log*("--> clicked on Create Button <---");

}

**else**

Common.*log*("--> Not Able to click on Create Button <---");

Common.*pause*(5);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_enterpackagereference() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(packagereference\_esign.isDisplayed()){

*packagereference* = "Packagereference\_"+TestData.*randomAlphaNumeric*(4);

packagereference\_esign.sendKeys(*packagereference*);

Common.*log*("--> Enter Packagereference :" +*packagereference*+" <---");

}

**else**

Common.*log*("--> Package reference textfield is not available <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_enterfirstname() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

**if**(firstname\_esign.isDisplayed()){

*firstname* = "Firstname"+TestData.*randomAlpha*(4);

firstname\_esign.sendKeys(*firstname*);

Common.*log*("--> Enter First Name :" +*firstname*+" <---");

}

**else**

Common.*log*("--> First Name textfield is not available <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_enterlastname() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

**if**(lastname\_esign.isDisplayed()){

*lastname* = "Lastname"+TestData.*randomAlpha*(4);

lastname\_esign.sendKeys(*lastname*);

Common.*log*("--> Enter Last Name :" +*lastname*+" <---");

}

**else**

Common.*log*("--> Last Name textfield is not available <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_enteremailaddress() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

**if**(emailaddress\_esign.isDisplayed()){

String emailaddress = TestData.*randomAlphaNumeric*(4)+"@mailinator.com";

emailaddress\_esign.sendKeys(emailaddress);

Common.*log*("--> Enter Email Address :" +emailaddress+" <---");

}

**else**

Common.*log*("--> Email Address textfield is not available <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_enterphonenumber() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

**if**(mobile\_esign.isDisplayed()){

String phonenumber ="+61444" +TestData.*randomNumeric*(6);

mobile\_esign.sendKeys(phonenumber);

Common.*log*("--> Enter Phone Number :" +phonenumber+" <---");

}

**else**

Common.*log*("--> Phone Number textfield is not available <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_numberofducuments() {

// **TODO** Auto-generated method stub

Common.*pause*(2);

**int** i = Integer.*parseInt*(TestData.*numberofducuments*());

**int** count = i/2;

**for**(WebElement document: numberofdocuments\_esign){

**if**(i>0){

document.click();

}

**else**

**break**;

i--;

}

*numberofdocuments*=count;

Common.*log*("Number of Documents Attached :"+count);

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_packagecreatebutton() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

**if**(packagecreatebutton.isDisplayed()){

packagecreatebutton.click();

Common.*log*("--> clicked on Create Button <---");

}

**else**

Common.*log*("--> Not Able to click on Create Button <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_clickoncloudserviceaggrement() {

// **TODO** Auto-generated method stub

Common.*pause*(15);

**if**(cloudserviceaggrement\_esign.isDisplayed()){

cloudserviceaggrement\_esign.click();

Common.*log*("--> clicked on Cloud Service Aggrement <---");

}

**else**

Common.*log*("--> Not Able to clicked on Cloud Service Aggrement <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_uploadagreement() {

// **TODO** Auto-generated method stub

Common.*pause*(5);

**if**(uploadtheaggreement\_esign.isDisplayed()){

File file = **new** File("Cloud Hosting Agreement.pdf");

//file = file.getAbsoluteFile();

String absolutePath = file.getAbsolutePath();

uploadtheaggreement\_esign.sendKeys(absolutePath);

//uploadtheaggreement\_esign.sendKeys("resources/Cloud Hosting Agreement.pdf");

Common.*log*("--> Uploaded the Aggrement <---");

}

**else**

Common.*log*("--> Not Able to Uploaded the Aggrement <---");

**return** **new** GeneralVerification(*driver*);

}

**public** GeneralVerification esign\_clickonsendnowbutton() {

// **TODO** Auto-generated method stub

Common.*pause*(18);

**if**(clickonsendnowbutton\_esign.isDisplayed()){

clickonsendnowbutton\_esign.click();

Common.*log*("--> Clicked on SendNow button <---");

}

**else**

Common.*log*("--> Not Able to Click on SendNow button <---");

**return** **new** GeneralVerification(*driver*);

}

}//end of class

**GeneralVerificationPage**

**package** com.verification;

**import** java.util.List;

**import** org.openqa.selenium.By;

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

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

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

**import** org.openqa.selenium.support.FindBy;

**import** com.init.AbstractPage;

**import** com.init.Common;

**public** **class** GeneralVerification **extends** AbstractPage {

**public** GeneralVerification(WebDriver driver) {

**super**(driver);

}

**public** **boolean** verifyhomepage() {

Common.*pause*(10);

WebElement regi = *driver*.findElement(By.*xpath*("//a[@title='About us']"));

Common.*log*("--->Home page open");

**return** Common.*isElementDisplayed*(regi);

}

**public** **boolean** blankfieldsvalidationmsg() {

// **TODO** Auto-generated method stub

List<WebElement> validation\_msg = *driver*.findElements(By.*xpath*("//span[contains(text(),'\* This field is required')]"));

**int** count = validation\_msg.size();

String msg = validation\_msg.get(0).getText();

//SeleniumInit.log(msg);

**if**(count==7 && msg.equalsIgnoreCase("\* This field is required")){

String msg1 = Integer.*toString*(count);

Common.*log*("---> There are " + msg1 +" All validation messages dispalyed::"+msg+" <---");

**return** **true**;

}

**else**{

Common.*log*("---> None validation message display <---");

**return** **false**;

}

}

**public** **boolean** minimulengthvalidationmsg() {

// **TODO** Auto-generated method stub

WebElement phone\_mini\_validation\_msg = *driver*.findElement(By.*xpath*(".//span[contains(text(),'\* Minimum 10 Digits starting with Country Code')]"));

String str = phone\_mini\_validation\_msg.getText();

Common.*pause*(5);

**if**(str.equals("\* Minimum 10 Digits starting with Country Code")){

Common.*log*("---> Minimum phone field length validation message displayed <---");

**return** **true**;

}

**else**{

Common.*log*("---> Minimum phone field length validation message is not displayed <---");

**return** **false**;

}

}

**public** **boolean** passwordminimulengthvalidationmsg() {

// **TODO** Auto-generated method stub

WebElement password\_mini\_validation\_msg = *driver*.findElement(By.*xpath*(".//span[contains(text(),'\* Minimum 8 characters required')]"));

String str = password\_mini\_validation\_msg.getText().trim();

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

Common.*pause*(5);

**if**(password\_mini\_validation\_msg.isDisplayed()){

Common.*log*("---> Minimum password field length validation message displayed <---");

**return** **true**;

}

**else**{

Common.*log*("---> Minimum password field length validation message is not displayed <---");

**return** **false**;

}

}

**public** **boolean** passwordonotmatch() {

// **TODO** Auto-generated method stub

WebElement confirm\_password\_validation\_msg = *driver*.findElement(By.*xpath*(".//span[contains(text(),'\* Fields do not match')]"));

Common.*pause*(3);

String str = confirm\_password\_validation\_msg.getText();

**boolean** tf = str.equals("\* Fields do not match");

**if**(tf){

Common.*log*("---> Confirm password do not match with passwor d<---");

**return** **true**;

}

**else**{

Common.*log*("---> Confirm password do not match with password validation message is not displayed <---");

**return** **false**;

}

}

**public** **boolean** invalidemailid() {

// **TODO** Auto-generated method stub

WebElement invalid\_emailid\_validation\_msg = *driver*.findElement(By.*xpath*("//span[contains(text(),'\* Invalid email address')]"));

**if**(invalid\_emailid\_validation\_msg.isDisplayed()){

Common.*log*("---> Enter email Id is invalid!!! <---");

**return** **true**;

}

**else**

Common.*log*("---> Validation message is not displayed <---");

**return** **false**;

}

**public** **boolean** successfullregistration() {

// **TODO** Auto-generated method stub

WebElement registration\_msg = *driver*.findElement(By.*xpath*("//p[contains(text(),'Thank you for your registration')]"));

**if**(Common.*isElementDisplayed*(registration\_msg)){

Common.*log*("---> Thank you for your registration!!!You have registered successfully!!! <---");

**return** **true**;

}

**else**{

Common.*log*("---> Unsuccessfull!!Check the error please!!! <---");

**return** **false**;

}

}

**public** **boolean** usernamealreadyexist() {

// **TODO** Auto-generated method stub

WebElement error\_registration\_msg = *driver*.findElement(By.*xpath*("//p[@class='piereg\_login\_error']"));

**if**(error\_registration\_msg.isDisplayed()){

Common.*log*("---> UserName Already Exist validation message is displayed <---");

**return** **true**;

}

**else**{

Common.*log*("---> UserName Already Exist validation message is NOT displayed <---");

**return** **false**;

}

}

**public** **boolean** emailidalreadyregistered() {

// **TODO** Auto-generated method stub

WebElement error\_registration\_msg = *driver*.findElement(By.*xpath*("//p[@class='piereg\_login\_error']"));

**if**(error\_registration\_msg.isDisplayed()){

Common.*log*("---> EmailId is already registered validation message is displayed <---");

**return** **true**;

}

**else**{

Common.*log*("---> EmailId is already registered validation message is NOT displayed <---");

**return** **false**;

}

}

**public** **boolean** verifyhomepage\_purchase() {

// **TODO** Auto-generated method stub

Common.*pause*(10);

WebElement account\_icon = *driver*.findElement(By.*xpath*("//span[contains(text(),'Account')]"));

**if**(account\_icon.isDisplayed())

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** checkproductinthecart() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

WebElement productincart = *driver*.findElement(By.*xpath*("//a[contains(text(),'Apple iPad 2 16GB, Wi-Fi, 9.7in - Black')]"));

**if**(productincart.isDisplayed())

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** checkproductquanity() {

// **TODO** Auto-generated method stub

Common.*pause*(3);

WebElement productquantity = *driver*.findElement(By.*xpath*("//\*[@id='checkout\_page\_container']/div[1]/table/tbody/tr[2]/td[3]/form/input[1]"));

System.***out***.println("Number of product in the CART :" +productquantity.getAttribute("value"));

**if**(productquantity.getAttribute("value").equals("2"))

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** purchaseverification() {

// **TODO** Auto-generated method stub

Common.*pause*(10);

WebElement order = *driver*.findElement(By.*xpath*("//p[contains(text(),'Thank you, your purchase is pending. You will be sent an email once the order clears.')]"));

**if**(order.isDisplayed())

**return** **true**;

**else**

**return** **false**;

}

// Threads and Shirts

**public** **boolean** verifyhomepage\_threadandshirts() {

// **TODO** Auto-generated method stub

String url = *driver*.getCurrentUrl();

**if**(url.equals("https://www.threadsandshirts.com/"))

**return** **true**;

**else**

**return** **false**;

}

@FindBy(xpath="//p[contains(text(),'Congratulations! Your order has been successfully placed.')]") **private** WebElement orderplacedsuccessfullyts;

**public** **boolean** orderplacedsuccessfully() {

// **TODO** Auto-generated method stub

Common.*pause*(8);

String str = orderplacedsuccessfullyts.getText();

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

**if**(str.equals("Congratulations! Your order has been successfully placed. Order details.")){

WebElement orderdetailsbutton = *driver*.findElement(By.*xpath*("//div/a[contains(text(),'Order details')]"));

orderdetailsbutton.click();

Common.*pause*(10);

WebElement productnameinbill = *driver*.findElement(By.*xpath*("//td/a[contains(text(),'Black Dobby')]"));

**if**(generalIndexpage.*productname*.equals(productnameinbill.getText())){

Common.*log*("--> Product name verify with the bill <--");

**return** **true**;

}

**else**{

Common.*log*("--> Product name is not verify with the bill <--");

**return** **false**;

}

}

**else**

**return** **false**;

}

//E-Signature

**public** **boolean** verifyesignaturehomepage() {

// **TODO** Auto-generated method stub

Common.*pause*(5);

String url = *driver*.getCurrentUrl();

**if**(url.equals("https://go-env-syd1.clickandsign.net:11443/dashboard/"))

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** esign\_loginpageverification() {

// **TODO** Auto-generated method stub

Common.*pause*(6);

WebElement welcomemsg = *driver*.findElement(By.*xpath*("//span[contains(text(),'Welcome Sydney Test')]"));

String msg = welcomemsg.getText();

**if**(msg.equals("Welcome Sydney Test"))

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** esign\_packagenameverification() {

// **TODO** Auto-generated method stub

Common.*pause*(25);

String firstandlastname = generalIndexpage.*firstname*+" "+generalIndexpage.*lastname*;

WebElement caretedpackagename\_esign = *driver*.findElement(By.*xpath*("//td[contains(text(),'"+generalIndexpage.*packagereference*+"')]"));

**if**(caretedpackagename\_esign.isDisplayed()){

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

act.moveToElement(caretedpackagename\_esign).build().perform();

caretedpackagename\_esign.click();

Common.*logverification*("--> package name is confirmed <--");

}

**else**

Common.*logverification*("--> package name is NOT confirmed <--");

Common.*pause*(5);

**int** check=0;

List<WebElement> numberofdocuments = *driver*.findElements(By.*xpath*("//span[@class='fancytree-title']"));

System.***out***.println(generalIndexpage.*numberofdocuments*);

**for**(WebElement element :numberofdocuments){

/\*System.out.println(element.getText());

System.out.println("check : " + check);\*/

**if**(element.getText().contains("Transfer of Land \*") || element.getText().contains("Certificate of Insurance 1 \*")||element.getText().contains( "Certificate of Insurance 3 \*")||element.getText().contains( "DCC \*" )||element.getText().contains("Borrowers Letter of Authority \*"))

{

check++;

Common.*log*("--> Attached Documents : "+element.getText()+ " <--");

**if**(generalIndexpage.*numberofdocuments*==check){

Common.*logverification*("--> Verified number of documents Attached:"+ check +" Verify the above documents list <--");

}

/\*else

Common.logverification("--> Number of documents Attached is not matched <--");\*/

}

}

WebElement firstandlastname\_esign =*driver*.findElement(By.*xpath*("//span[contains(text(),'"+firstandlastname+"')]"));

**if**(firstandlastname\_esign.isDisplayed()){

Common.*logverification*("--> First Name and Last Name is confirmed <--");

**return** **true**;

}

**else**

Common.*logverification*("--> First Name and Last Name is NOT confirmed <--");

**return** **false**;

}

}//class ends

**AbstractPage**

**package** com.init;

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

**import** org.openqa.selenium.support.PageFactory;

**import** org.openqa.selenium.support.pagefactory.AjaxElementLocatorFactory;

**import** org.openqa.selenium.support.pagefactory.ElementLocatorFactory;

**public** **abstract** **class** AbstractPage **extends** SeleniumInit {

**public** **int** DRIVER\_WAIT = 15;

**public** **static** WebDriver *driver*;

/\*\*

\* Initialize UserAbstractPage.

\*

\* **@param** Driver

\* .

\*/

**public** AbstractPage(WebDriver driver) {

**this**.*driver* = driver;

ElementLocatorFactory finder = **new** AjaxElementLocatorFactory(driver,

DRIVER\_WAIT);

PageFactory.*initElements*(finder, **this**);

System.***out***.println("======Abstract Page======");

}

}

**Common**

**package** com.init;

**import** java.awt.Toolkit;

**import** java.awt.datatransfer.StringSelection;

**import** java.io.File;

**import** java.io.IOException;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Random;

**import** java.util.Set;

**import** java.util.SimpleTimeZone;

**import** java.util.TimeZone;

**import** org.apache.commons.io.FileUtils;

//import org.apache.commons.lang.RandomStringUtils;

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

**import** org.openqa.selenium.By;

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

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

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.StaleElementReferenceException;

**import** org.openqa.selenium.TakesScreenshot;

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

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

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

**import** org.openqa.selenium.remote.Augmenter;

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

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

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

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

**import** org.testng.Reporter;

/\*\*

\* Define Common Webdriver

\*/

**public** **class** Common {

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

**protected** **static** Wait<WebDriver> *wait*;

**public** **static** String *alerttext*;

/\*\*

\* Find web-element for given locator.

\*

\* **@param** elementName

\* **@return**

\*/

**public** **static** WebElement findElement(WebDriver driver, String elementName) {

String locator;

locator = elementName;

**int** count = 0;

**while** (count < 4) {

**try** {

**if** (locator.startsWith("link=") || locator.startsWith("LINK=")) {

locator = locator.substring(5); // remove "link=" from

// locator

**try** {

**if** (locator.contains(" "))

**return** driver.findElement(By.*partialLinkText*(locator));

**return** driver.findElement(By.*linkText*(locator));

} **catch** (Exception e) {

**return** **null**;

}

}

**if** (locator.startsWith("id=")) {

locator = locator.substring(3); // remove "id=" from locator

**try** {

**return** driver.findElement(By.*id*(locator));

} **catch** (Exception e) {

**return** **null**;

}

} **else** **if** (locator.startsWith("//")) {

**try** {

**return** driver.findElement(By.*xpath*(locator));

} **catch** (Exception e) {

**return** **null**;

}

} **else** **if** (locator.startsWith("css=")) {

locator = locator.substring(4); // remove "css=" from

// locator

**try** {

**return** driver.findElement(By.*cssSelector*(locator));

} **catch** (Exception e) {

**return** **null**;

}

} **else** **if** (locator.startsWith("name=")) {

locator = locator.substring(5); // remove "name=" from

// locator

**try** {

**return** driver.findElement(By.*name*(locator));

} **catch** (Exception e) {

**return** **null**;

}

} **else** {

**try** {

**return** driver.findElement(By.*id*(locator));

} **catch** (Exception e) {

**return** **null**;

}

}

} **catch** (StaleElementReferenceException e) {

e.toString();

count = count + 1;

// System.out.println("Trying["+

// count+"] to recover from a stale element :" +

// e.getMessage());

}

count = count + 4;

}

**return** **null**;

}

**public** **static** **void** moveToObjectelement(WebDriver driver, String xpath) {

driver.switchTo().frame(driver.findElement(By.*xpath*(xpath)));

}

/\*\*

\* Perform horizontal scrolling

\*

\* **@param** driver

\* **@param** element

\*/

**public** **static** **void** scrollToHorizontal(WebDriver driver, WebElement element) {

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

WebElement draggablePartOfScrollbar = element;

// drag downwards

**int** numberOfPixelsToDragTheScrollbarDown = 50;

**for** (**int** i = 10; i < 500; i = i + numberOfPixelsToDragTheScrollbarDown) {

**try** {

// this causes a gradual drag of the scroll bar, 10 units at a

// time

dragger.moveToElement(draggablePartOfScrollbar).clickAndHold()

.moveByOffset(numberOfPixelsToDragTheScrollbarDown, 0).release().perform();

Thread.*sleep*(1000L);

} **catch** (Exception e1) {

}

}

}

/\*\*

\* Perform vertical scrolling

\*

\* **@param** driver

\* **@param** element

\*/

**public** **static** **void** scrollToVertical(WebDriver driver, WebElement element) {

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

WebElement draggablePartOfScrollbar = element;

// drag downwards

**int** numberOfPixelsToDragTheScrollbarDown = 50;

**for** (**int** i = 10; i < 500; i = i + numberOfPixelsToDragTheScrollbarDown) {

System.***out***.println("1");

**try** {

// this causes a gradual drag of the scroll bar, 10 units at a

// time

dragger.moveToElement(draggablePartOfScrollbar).clickAndHold()

.moveByOffset(0, numberOfPixelsToDragTheScrollbarDown).release().perform();

Thread.*sleep*(1000L);

} **catch** (Exception e1) {

}

}

}

/\*\*

\* Checks checkbox or toggle element.

\*

\* **@param** element

\* Checkbox element.

\*/

**public** **static** **void** checkChkBox(WebElement element) {

**boolean** isCheckBoxSet;

isCheckBoxSet = element.isSelected();

**if** (!isCheckBoxSet) {

element.click();

}

}

**public** **static** **void** movetoalert(WebDriver webDriver) {

**try** {

Alert alert = webDriver.switchTo().alert();

*alerttext* = alert.getText();

} **catch** (Exception e) {

}

}

/\*\*

\* Open Mailinator

\*

\* **@param** emailAddress

\*/

**public** **static** **void** openMailinator(WebDriver driver, String emailAddress) {

String url = "https://www.mailinator.com/";

*goToUrl*(driver, url);

WebElement eleInbox = driver.findElement(By.*xpath*(".//\*[@id='inboxfield']"));

eleInbox.sendKeys(emailAddress);

driver.findElement(By.*xpath*("//button[contains(@class,'btn-dark')]")).click();

*pause*(2);

}

**public** **static** **void** openEmail(WebDriver driver) {

WebElement eleFrame = driver.findElement(By.*xpath*("//iframe[@id='msg\_body']"));

driver.switchTo().frame(eleFrame);

*pause*(1);

WebElement eleInbox = driver.findElement(By.*xpath*("//div//div[contains(.,'Verify your email')]"));

eleInbox.click();

*pause*(1);

WebElement eleBtnVerify = driver.findElement(By.*xpath*("//td/a[contains(.,'Verify my email')]"));

eleBtnVerify.click();

*pause*(1);

}

**public** **static** String getCurrentUrl(WebDriver driver) {

**return** driver.getCurrentUrl();

}

**public** **static** String replaceString(String baseString, String targetString, String replaceString) {

String newString = baseString.replace(targetString, replaceString);

**return** newString;

}

**public** **static** **void** openNewTab(WebDriver driver) {

String selectLinkOpeninNewTab = Keys.*chord*(Keys.***COMMAND***, "t");

driver.findElement(By.*tagName*("body")).sendKeys(selectLinkOpeninNewTab);

// driver.findElement(By.xpath("//body")).sendKeys(Keys.COMMAND +"t");

*pause*(3);

}

**public** **static** **void** JsopnNewtab(WebDriver driver) {

*pause*(5);

System.***out***.println("===========" + System.*getProperty*("os.name"));

((JavascriptExecutor) driver).executeScript("window.open();");

*pause*(3);

ArrayList<String> tabs = **new** ArrayList<String>(driver.getWindowHandles());

driver.switchTo().window(tabs.get(1));

*pause*(3);

}

**public** **static** **void** SwitchtoTab(WebDriver driver, **int** tabNumber) {

ArrayList<String> tabs = **new** ArrayList<String>(driver.getWindowHandles());

driver.switchTo().window(tabs.get(tabNumber));

}

**public** **static** **void** openPrivareWindow(WebDriver driver) {

driver.findElement(By.*xpath*("//body")).sendKeys(Keys.***CONTROL*** + "t" + "n");

*pause*(3);

}

**public** **static** **void** log(String msg) {

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

Reporter.*log*("</br>" + msg);

//Reporter.log(msg);

}

**public** **static** **void** logStatus(String Status) {

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

**if** (Status.equalsIgnoreCase("Pass")) {

*log*("<br><Strong><font color=#008000>Pass</font></strong></br>");

} **else** **if** (Status.equalsIgnoreCase("Fail")) {

*log*("<br><Strong><font color=#FF0000>Fail</font></strong></br>");

}

}

**public** **static** **void** logstep(String msg) {

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

Reporter.*log*("<br><strong>" + msg + "</strong></br>");

}

**public** **static** **void** logcase(String msg) {

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

Reporter.*log*("<strong> <h3 style=\"color:DarkViolet\"> " + msg

+ "</h3> </strong>");

}

**public** **static** **void** logverification(String msg) {

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

Reporter.*log*("<br><Strong><font color=#009000>"+ msg+"</font></strong></br>");

}

/\*\*

\* Gets current time in the following format Month, Date, Hours, Minutes,

\* Seconds, Millisecond.

\*

\* **@return** Current date.

\*/

**public** **static** String getCurrentTimeStampString() {

java.util.Date date = **new** java.util.Date();

SimpleDateFormat sd = **new** SimpleDateFormat("MMddHHmmssSS");

TimeZone timeZone = TimeZone.*getDefault*();

Calendar cal = Calendar.*getInstance*(**new** SimpleTimeZone(timeZone.getOffset(date.getTime()), "GMT"));

sd.setCalendar(cal);

**return** sd.format(date);

}

/\*\*

\* Takes screenshot and adds it to TestNG report.

\*

\* **@param** driver

\* WebDriver instance.

\*/

**public** **static** **void** makeScreenshot(WebDriver driver, String screenshotName) {

WebDriver augmentedDriver = **new** Augmenter().augment(driver);

/\* Take a screenshot \*/

File screenshot = ((TakesScreenshot) augmentedDriver).getScreenshotAs(OutputType.***FILE***);

String nameWithExtention = screenshotName + ".png";

/\* Copy screenshot to specific folder \*/

**try** {

/\*

\* String reportFolder = "target" + File.separator + "failsafe-reports" +

\* File.separator + "firefox" + File.separator;

\*/

String reportFolder = "test-output" + File.***separator***;

String screenshotsFolder = "screenshots";

File screenshotFolder = **new** File(reportFolder + screenshotsFolder);

**if** (!screenshotFolder.getAbsoluteFile().exists()) {

screenshotFolder.mkdir();

}

FileUtils.*copyFile*(screenshot,

**new** File(screenshotFolder + File.***separator*** + nameWithExtention).getAbsoluteFile());

} **catch** (IOException e) {

*log*("Failed to capture screenshot: " + e.getMessage());

}

*log*(*getScreenshotLink*(nameWithExtention, nameWithExtention)); // add

// screenshot

// link

// to

// the

// report

}

/\*\*

\* Log given message to Reporter output.

\*

\* **@param** msg

\* Message/Log to be reported.

\*/

/\*\*

\* Generates link for TestNG report.

\*

\* **@param** screenshot\_name

\* Screenshot name.

\* **@param** link\_text

\* Link text.

\* **@return** Formatted link for TestNG report.

\*/

**public** **static** String getScreenshotLink(String screenshot\_name, String link\_text) {

*log*("<br><Strong><font color=#FF0000>--Failed</font></strong>");

**return** "<a href='../test-output/screenshots/" + screenshot\_name + "' target='\_new'>" + link\_text + "</a>";

}

/\*\*

\* Checks whether the needed WebElement is displayed or not.

\*

\* **@param** element

\* Needed element

\*

\* **@return** true or false.

\*/

**public** **static** **boolean** isElementDisplayed(WebElement element) {

**try** {

**return** element.isDisplayed();

} **catch** (Exception e) {

**return** **false**;

}

}

**public** **static** **boolean** isElementNotDisplayed(WebElement element) {

**try** {

**return** !element.isDisplayed();

} **catch** (Exception e) {

**return** **false**;

}

}

/\*\*

\* Wait(max. 1 minute) till given element does not disappear from page.

\*

\* **@param** by

\* Locator of element.

\* **@return**

\* **@throws** InterruptedException

\*/

**public** **static** **boolean** waitForElementIsDisplayed(WebElement by) **throws** InterruptedException {

**for** (**int** second = 0;; second++) {

**if** (second >= 60) {

**break**;

}

**try** {

**if** (*isElementDisplayed*(by))

**break**;

} **catch** (Exception e) {

}

*pause*(1);

}

**return** **false**;

}

/\*\*

\* Checks if given elements is checked.

\*

\* **@param** locator

\* Locator of element.

\*

\* **@return** true if checked else false.

\*/

**public** **static** **boolean** isChecked(WebDriver driver, String locator) {

**return** *findElement*(driver, locator).isSelected();

}

/\*\*

\* Checks whether the needed WebElement is displayed or not.

\*

\* **@param** elementLocator

\* **@return**

\*/

**public** **static** **boolean** isElementDisplayed(WebDriver driver, By elementLocator) {

**try** {

**return** driver.findElement(elementLocator).isDisplayed();

} **catch** (Exception e) {

**return** **false**;

}

}

/\*\*

\* Set data in to clipboard

\*

\* **@param** string

\*/

**public** **static** **void** setClipboardData(String string) {

StringSelection stringSelection = **new** StringSelection(string);

Toolkit.*getDefaultToolkit*().getSystemClipboard().setContents(stringSelection, **null**);

}

/\*\*

\* Checks whether the visibility of Element Located

\*

\* **@param** by

\* **@return**

\*/

**public** **static** ExpectedCondition<WebElement> visibilityOfElementLocated(**final** By by) {

**return** **new** ExpectedCondition<WebElement>() {

**public** WebElement apply(WebDriver driver) {

WebElement element = driver.findElement(by);

**return** element.isDisplayed() ? element : **null**;

}

};

}

/\*\*

\* Wait up to String locator present

\*

\* **@param** selector

\*/

**public** **static** **void** waitForElement(WebDriver driver, String xpath) {

*wait* = **new** WebDriverWait(driver, 600);

**try** {

*wait*.until(*visibilityOfElementLocated*(By.*xpath*(xpath)));

} **catch** (Exception e) {

}

}

/\*\*

\* Finds handle to second window other than given handle to current window and

\* switches to as well.

\*

\* **@param** handleCurrentWindow

\* **@return** handleSecondWindow

\*/

**public** **static** String findAndSwitchToSecondWindow(WebDriver driver, String handleCurrentWindow) {

*pause*(1000);

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

String handleSecondWindow = **null**;

**for** (String window : windows) {

**if** (!window.contains(handleCurrentWindow)) {

handleSecondWindow = window;

}

}

// Switch to the second window.

**try** {

*pause*(2000);

driver.switchTo().window(handleSecondWindow);

} **catch** (Throwable failure) { // If there is problem in switching

// window, then re-try.

*pause*(1000);

driver.switchTo().window(handleSecondWindow);

}

**return** handleSecondWindow;

}

/\*\*

\* Select data from dropwon or combobox by Value.

\*

\* **@param** element

\* **@param** value

\*/

**public** **static** **void** selectFromCombo(WebElement element, String value) {

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

select.selectByValue(value);

}

/\*\*

\* Select data form dropdown or combobox by visible element

\*

\* **@param** element

\* **@param** value

\*/

**public** **static** **void** selectFromComboByVisibleElement(WebElement element, String value) {

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

select.selectByVisibleText(value);

}

/\*\*

\* Wait up to By element present

\*

\* **@param** element

\*/

**public** **static** **void** waitForElement(WebDriver driver, By element) {

**try** {

*wait* = **new** WebDriverWait(driver, 750);

// wait.until(visibilityOfElementLocated(element));

} **catch** (Exception e) {

}

}

/\*\*

\* Clicks on visible or not visible element.

\*

\* **@param** element

\* Web element.

\*/

**public** **static** **void** jsClick(WebDriver driver, WebElement element) {

((JavascriptExecutor) driver).executeScript("return arguments[0].click();", element);

// this.waitForAjax("0");

}

/\*\*

\* Highlight the element and click on same

\*

\* **@param** driver

\* **@param** element

\*/

**public** **static** **void** clickOn(WebDriver driver, WebElement element) {

*highlightElement*(driver, element);

element.click();

}

/\*\*

\* Generates random symbols;

\*

\* **@param** length

\* Length of the generated symbols.

\*

\* **@return** StringBuffer object.

\*/

/\*public static String generateRandomChars(int length) {

String random = RandomStringUtils.random(length);

return random;

}\*/

/\*\*

\* Generate Random Number in Length

\*

\* **@param** length

\* **@return**

\*/

**public** **static** **int** generateRandomNumber(**int** length) {

Random rand = **new** Random();

**int** numNoRange = rand.nextInt();

**return** numNoRange;

}

/\*\*

\* Mouse Hover in Web element

\*

\* **@param** element

\*/

**public** **static** **void** mouseOver(WebDriver driver, WebElement element) {

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

builder.moveToElement(element).build().perform();

}

/\*\*

\* Get text in a given element.

\*

\* **@param** elementName

\* Locator of element.

\*

\* **@return** text in given element.

\*/

**public** **static** String getText(WebDriver driver, String elementName) {

String text;

**try** {

text = *findElement*(driver, elementName).getText();

} **catch** (Exception e) {

text = "Element was not found";

}

**return** text;

}

/\*\*

\*

\* Get text in of given Element using JavaScript

\*

\* **@param** driver

\* **@param** element

\* webElement

\* **@return**

\*/

**public** **static** String getTextJS(WebDriver driver, WebElement element) {

**return** (String) ((JavascriptExecutor) driver).executeScript("return jQuery(arguments[0]).text();", element);

}

/\*\*

\* Get value of given element dynamically.

\*

\* **@param** locator

\* Locator of element.

\*

\* **@return** Dynamic value.

\*/

**public** String getValue(WebDriver driver, String locator) {

**return** *findElement*(driver, locator).getAttribute("value");

}

/\*\*

\* Checks if given element is being displayed on page.

\*

\* **@param** elementName

\* Locator of element.

\*

\* **@return** true if displayed else false.

\*/

**public** **static** **boolean** isElementDisplayed(WebDriver driver, String elementName) {

WebElement webElement;

**try** {

webElement = *findElement*(driver, elementName);

**return** webElement.isDisplayed();

} **catch** (Exception e) {

**return** **false**;

}

}

/\*\*

\* Wait till given element is present.

\*

\* **@param** locator

\* Locator of element.

\*/

**public** **static** **void** waitForConditionIsElementPresent(WebDriver driver, String locator) {

**for** (**int** second = 0;; second++) {

**if** (second >= 10) {

**break**;

}

**try** {

**if** (*isElementPresent*(driver, locator))

**break**;

} **catch** (Throwable failure) {

}

*pause*(1000);

}

}

/\*\*

\* Checks if element loaded in browser memory.

\*

\* **@param** locator

\* Locator of element.

\* **@return** true if loaded else false.

\*/

**public** **static** **boolean** isElementPresent(WebDriver driver, String locator) {

WebElement webElement = *findElement*(driver, locator);

**if** (webElement != **null**) {

**return** **true**;

} **else** {

**return** **false**;

}

}

/\*\*

\* Pauses for given seconds.

\*

\* **@param** secs

\*/

**public** **static** **void** pause(**int** secs) {

**try** {

Thread.*sleep*(secs \* 1000);

} **catch** (InterruptedException interruptedException) {

}

}

/\*\*

\* Get random numeric of given lenth.

\*

\* **@param** length

\* desired length.

\* **@return**

\*/

**public** **static** **int** randomNumericValueGenerate(**int** length) {

Random randomGenerator = **new** Random();

**int** randomInt = randomGenerator.nextInt(length);

**return** randomInt;

}

/\*\*

\* Clears and type new value into given text-box.

\*

\* **@param** Web

\* Element Locator of element.

\*

\* **@param** value

\* New text/value.

\*/

**public** **static** **void** type(WebElement webElement, String value) {

webElement.clear();

webElement.sendKeys(value);

}

/\*\*

\* Wait till all ajax calls finish.

\*

\* **@param** num

\* Number of ajax calls to finish.

\*/

**public** **static** **void** waitForAjax(WebDriver driver, String num) {

String ajax;

ajax = *ajaxFinised*(driver, num);

**for** (**int** second = 0;; second++) {

**if** (second >= 20) {

**break**;

} **else** **if** (ajax.equals("true")) {

**break**;

}

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

/\*\*

\* Wait till ajax call finish.

\*

\* **@throws** InterruptedException

\*/

**public** **void** waitForAjax(WebDriver driver) **throws** InterruptedException {

String ajax;

ajax = *ajaxFinised*(driver, "1");

**for** (**int** second = 0;; second++) {

**if** (second >= 15) {

**break**;

} **else** **if** (ajax.equals("true")) {

**break**;

}

Thread.*sleep*(1000);

}

}

/\*\*

\* Checks that all ajax calls are completed on page.

\*

\* **@param** num

\* Number of ajax calls to wait for completion.

\*

\* **@return** "true" if completed else "false".

\*/

**public** **static** String ajaxFinised(WebDriver driver, String num) {

Object isAjaxFinished;

JavascriptExecutor js = (JavascriptExecutor) driver;

isAjaxFinished = js.executeScript("return jQuery.active == " + num);

**return** isAjaxFinished.toString();

}

/\*\*

\* Select Random String From Combobox.

\*

\* **@param** by

\* **@param** driver

\* **@return** selected random string

\* **@throws** InterruptedException

\*/

**public** **static** String selectRandomOptionFromCombo(WebElement eleDropDown, WebDriver driver)

**throws** InterruptedException {

String selectedOption = "";

// WebElement selectCombo = driver.findElement(by);

Thread.*sleep*(2);

List<WebElement> getAllOption = eleDropDown.findElements(By.*xpath*("option"));

ArrayList<String> arrayOfAllOption = **new** ArrayList<String>();

**for** (WebElement ele : getAllOption) {

**if** (!ele.getText().startsWith("Select")) {

arrayOfAllOption.add(ele.getText());

}

}

**int** index = **new** Random().nextInt(arrayOfAllOption.size());

**if** (Integer.*signum*(index) == -1) {

index = -index;

// index=Math.abs(index);

}

selectedOption = arrayOfAllOption.get(index);

System.***out***.println("Selected Option Is----====>" + selectedOption);

**return** selectedOption;

}

/\*\*

\* Get Total Number Of Elements

\*

\* **@param** driver

\* **@param** by

\* **@return** interger number of total elements

\*/

**public** **static** **int** getNumOfElements(WebDriver driver, By by) {

**int** i = 0;

List<WebElement> ele = driver.findElements(by);

i = ele.size();

System.***out***.println("Total Number Of Elements Are >>> " + i);

**return** i;

}

/\*\*

\* Refresh Current Page

\*

\* **@param** driver

\*/

**public** **static** **void** refresh(WebDriver driver) {

driver.navigate().refresh();

}

/\*\*

\* Open URL in New Window

\*

\* **@param** driver

\* **@param** url

\*/

**public** **static** **void** openUrlInNewTab(WebDriver driver, String url) {

System.***out***.println("--------->" + System.*getProperty*("os.name"));

**if** (System.*getProperty*("os.name").equalsIgnoreCase("Mac OS X")) {

driver.findElement(By.*tagName*("body")).sendKeys(Keys.***COMMAND*** + "t");

} **else** {

driver.findElement(By.*tagName*("body")).sendKeys(Keys.***CONTROL*** + "t");

}

driver.get(url);

}

/\*\*

\* Close Current Tab In Web Browser

\*

\* **@param** driver

\*/

**public** **static** **void** closeCurrentTab(WebDriver driver) {

**if** (System.*getProperty*("os.name").equalsIgnoreCase("Mac OS X")) {

driver.findElement(By.*tagName*("body")).sendKeys(Keys.***COMMAND*** + "w");

} **else** {

driver.findElement(By.*tagName*("body")).sendKeys(Keys.***CONTROL*** + "w");

}

}

/\*\*

\* Perform Mouse Hover on element

\*

\* **@param** driver

\* **@param** ele

\*/

**public** **static** **void** mouseHover(WebDriver driver, WebElement ele) {

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

action.moveToElement(ele).build().perform();

}

/\*\*

\* Perform Mouse Hover using java sript executer

\*

\* **@param** driver

\* **@param** ele

\*/

**public** **static** **void** mouseHoverUsingJS(WebDriver driver, WebElement ele) {

String mouseOverScript = "if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('mouseover',true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObject) { arguments[0].fireEvent('onmouseover');}";

((JavascriptExecutor) driver).executeScript(mouseOverScript, ele);

}

/\*\*

\* Go to URL.

\*

\* **@param** driver

\* **@param** url

\*/

**public** **static** **void** goToUrl(WebDriver driver, String url) {

driver.get(url);

}

/\*\*

\* Go to previous page

\*

\* **@param** driver

\*/

**public** **static** **void** goToPreviuosPage(WebDriver driver) {

driver.navigate().back();

}

/\*\*

\* Highlight Element

\*

\* **@param** driver

\* **@param** element

\*/

**public** **static** **void** highlightElement(WebDriver driver, WebElement element) {

/\*

\* for (int i = 0; i < 2; i++) { JavascriptExecutor js = (JavascriptExecutor)

\* driver; js.executeScript("arguments[0].setAttribute('style', arguments[1]);",

\* element, "color: yellow; border: 2px solid yellow;");

\* js.executeScript("arguments[0].setAttribute('style', arguments[1]);",

\* element, ""); }

\*/

// draw a border around the found element

((JavascriptExecutor) driver).executeScript("arguments[0].style.border = '3px solid yellow'", element);

*pause*(2);

}

/\*\*

\* Stop page loading

\*

\* **@param** driver

\*/

**public** **static** **void** stopPageLoading(WebDriver driver) {

driver.findElement(By.*tagName*("body")).sendKeys(Keys.***ESCAPE***);

}

**public** **static** **void** jsClickNew(WebDriver driver, WebElement element) {

// ((JavascriptExecutor) driver).executeScript(

// "return

// ((document.getElementsByTagName('object')[0]).contentDocument).arguments[0].click();",

// element);

// this.waitForAjax("0");

}

}

/\*public class Common {

}\*/

/\*public class Common {

}

\*/

**ITestStatus**

**package** com.init;

/\*\*

\* Definitions of test statuses

\*

\* **@author** DELL

\*

\*/

**public** **interface** ITestStatus {

**int** ***PASSED*** = 1;

**int** ***FAILED*** = 2;

**int** ***SKIPPED*** = 3;

}

/\*public class ITestStatus {

}

\*/

**SeleniumInit**

**package** com.init;

**import** java.io.File;

**import** java.io.IOException;

**import** java.lang.reflect.Method;

**import** java.net.URL;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

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

**import** java.util.logging.Logger;

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

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

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.firefox.FirefoxProfile;

**import** org.openqa.selenium.ie.InternetExplorerDriver;

**import** org.openqa.selenium.opera.OperaDriver;

**import** org.openqa.selenium.remote.CapabilityType;

**import** org.openqa.selenium.remote.DesiredCapabilities;

**import** org.openqa.selenium.remote.RemoteWebDriver;

**import** org.openqa.selenium.safari.SafariDriver;

**import** org.testng.IResultMap;

**import** org.testng.ITestContext;

**import** org.testng.ITestResult;

**import** org.testng.Reporter;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.internal.Utils;

**import** com.indexpage.GeneralIndexpage;

**import** com.verification.GeneralVerification;

**public** **class** SeleniumInit {

**public** String suiteName = "";

**public** String testName = "";

/\* Minimum requirement for test configur ation \*/

**protected** String testUrl; // Test url

**protected** String seleniumHub; // Selenium hub IP

**protected** String seleniumHubPort; // Selenium hub port

**protected** String targetBrowser; // Target browser

**protected** **static** String *test\_data\_folder\_path* = **null**;

**public** **static** String *currentWindowHandle* = ""; // Get Current Window handle

**public** **static** String *browserName* = "";

**public** **static** String *osName* = "";

**public** **static** String *browserVersion* = "";

**public** **static** String *browsernm* = "";

**public** GeneralIndexpage generalIndexpage;

**public** GeneralVerification generalVerification;

**public** String email = "FrameWork@demo.com";

**public** String password = "FrameWork";

**protected** **static** String *screenshot\_folder\_path* = **null**;

**public** **static** String *currentTest*; // current running test

**protected** **static** Logger *logger* = Logger.*getLogger*("testing");

**protected** WebDriver driver;

// Common Common = new Common(driver);

/\* Page's declaration \*/

/\*\*

\* Fetches suite-configuration from XML suite file.

\*

\* **@param** testContext

\*/

@BeforeTest(alwaysRun = **true**)

**public** **void** fetchSuiteConfiguration(ITestContext testContext) {

testUrl = testContext.getCurrentXmlTest().getParameter("selenium.url");

// testUrl = TestData.getURL();

/\* System.out.println("======" + testUrl + "========="); \*/

seleniumHub = testContext.getCurrentXmlTest().getParameter("selenium.host");

seleniumHubPort = testContext.getCurrentXmlTest().getParameter("selenium.port");

targetBrowser = testContext.getCurrentXmlTest().getParameter("selenium.browser");

*browsernm* = targetBrowser;

}

/\*\*

\* WebDriver initialization

\*

\* **@return** WebDriver object

\* **@throws** IOException

\* **@throws** InterruptedException

\*/

@BeforeMethod(alwaysRun = **true**)

**public** **void** setUp(Method method, ITestContext testContext) **throws** IOException, InterruptedException {

*currentTest* = method.getName(); // get Name of current test.

URL remote\_grid = **new** URL("http://" + seleniumHub + ":" + seleniumHubPort + "/wd/hub");

String SCREENSHOT\_FOLDER\_NAME = "screenshots";

String TESTDATA\_FOLDER\_NAME = "test\_data";

*test\_data\_folder\_path* = **new** File(TESTDATA\_FOLDER\_NAME).getAbsolutePath();

*screenshot\_folder\_path* = **new** File(SCREENSHOT\_FOLDER\_NAME).getAbsolutePath();

DesiredCapabilities capability = **null**;

**if** (targetBrowser == **null** || targetBrowser.contains("firefox")) {

FirefoxProfile profile = **new** FirefoxProfile();

**if** (System.*getProperty*("os.name").equalsIgnoreCase("Mac OS X")) {

// path = "/Users/Jignesh/developer/test-automation";

} **else** {

// path = "c:\\Downloads\_new";

}

profile.setPreference("dom.max\_chrome\_script\_run\_time", "999");

profile.setPreference("dom.max\_script\_run\_time", "999");

profile.setPreference("browser.download.folderList", 2);

// profile.setPreference("browser.download.dir", path);

profile.setPreference("browser.helperApps.neverAsk.openFile",

"text/csv,application/x-msexcel,application/excel,application/x-excel,application/vnd.ms-excel,image/png,image/jpeg,text/html,text/plain,application/msword,application/xml");

profile.setPreference("browser.helperApps.neverAsk.saveToDisk",

"text/csv,application/x-msexcel,application/excel,application/x-excel,application/vnd.ms-excel,image/png,image/jpeg,text/html,text/plain,application/msword,application/xml");

profile.setPreference("browser.download.manager.showWhenStarting", **false**);

profile.setPreference("browser.download,manager.focusWhenStarting", **false**);

// profile.setPreference("browser.download.useDownloadDir",true);

profile.setPreference("browser.helperApps.alwaysAsk.force", **false**);

profile.setPreference("browser.download.manager.alertOnEXEOpen", **false**);

profile.setPreference("browser.download.manager.closeWhenDone", **false**);

profile.setPreference("browser.download.manager.showAlertOnComplete", **false**);

profile.setPreference("browser.download.manager.useWindow", **false**);

profile.setPreference("browser.download.manager.showWhenStarting", **false**);

profile.setPreference("services.sync.prefs.sync.browser.download.manager.showWhenStarting", **false**);

profile.setPreference("pdfjs.disabled", **true**);

profile.setAcceptUntrustedCertificates(**true**);

profile.setPreference("security.OCSP.enabled", 0);

//profile.setEnableNativeEvents(false);

profile.setPreference("network.http.use-cache", **false**);

// added Dependancy to disable hardware acceleration.

/\*

\* profile.setPreference("gfx.direct2d.disabled",true);

\* profile.setPreference("layers.acceleration.disabled", true);

\*/

profile.setPreference("gfx.direct2d.disabled", **true**);

profile.setPreference("layers.acceleration.disabled", **true**);

// profile.setPreference("webgl.force-enabled", true);

// Proxy proxy = new Proxy().setHttpProxy("localhost:3129");

// cap.setCapability(CapabilityType.PROXY, proxy);

capability = DesiredCapabilities.*firefox*();

// proxy code

// capability.setCapability(CapabilityType.PROXY,proxy);

capability.setJavascriptEnabled(**true**);

capability.setCapability(FirefoxDriver.***PROFILE***, profile);

*browserName* = capability.getBrowserName();

*osName* = System.*getProperty*("os.name");

*browserVersion* = capability.getVersion().toString();

System.***out***.println("=========" + "firefox Driver " + "==========");

driver = **new** RemoteWebDriver(remote\_grid, capability);

} **else** **if** (targetBrowser.contains("ie11")) {

capability = DesiredCapabilities.*internetExplorer*();

System.*setProperty*("webdriver.ie.driver",

"/IEDriverServer.exe");

capability.setBrowserName("internet explorer");

capability.setCapability(InternetExplorerDriver.***INTRODUCE\_FLAKINESS\_BY\_IGNORING\_SECURITY\_DOMAINS***, **true**);

capability.setCapability(CapabilityType.ForSeleniumServer.***ENSURING\_CLEAN\_SESSION***, **true**);

capability.setJavascriptEnabled(**true**);

*browserName* = capability.getVersion();

*osName* = capability.getPlatform().*getCurrent*().name();

*browserVersion* = capability.getVersion();

driver = **new** RemoteWebDriver(remote\_grid, capability);

} **else** **if** (targetBrowser.contains("opera")) {

capability = DesiredCapabilities.~~opera~~();

System.*setProperty*("webdriver.opera.driver",

"/operadriver.exe");

capability.setJavascriptEnabled(**true**);

*browserName* = capability.getVersion();

*osName* = capability.getPlatform().*getCurrent*().name();

*browserVersion* = capability.getVersion();

driver = **new** ~~OperaDriver~~(capability);

} **else** **if** (targetBrowser.contains("chrome")) {

ChromeOptions options = **new** ChromeOptions();

capability = DesiredCapabilities.*chrome*();

System.*setProperty*("webdriver.chrome.driver", "lib/chromedriver.exe");

File chromeDriver;

**if** (System.*getProperty*("os.name").equalsIgnoreCase("Mac OS X")) {

chromeDriver = **new** File("/lib/chromedriver");

} **else** {

chromeDriver = **new** File("/lib/chromedriver.exe");

}

/\* System.setProperty("webdriver.chrome.driver",

chromeDriver.getAbsolutePath());\*/

capability.setCapability(ChromeOptions.***CAPABILITY***, options);

capability.setBrowserName("chrome");

capability.setJavascriptEnabled(**true**);

*browserName* = capability.getVersion();

*osName* = capability.getPlatform().name();

*browserVersion* = capability.getVersion();

options.addArguments("disable-geolocation");

driver = **new** ChromeDriver();

//driver = new RemoteWebDriver(remote\_grid, capability);

} **else** **if** (targetBrowser.contains("safari")) {

// System.setProperty("webdriver.safari.driver","/Users/jesus/Desktop/SafariDriver.safariextz");

// driver = new SafariDriver();

SafariDriver profile = **new** SafariDriver();

capability = DesiredCapabilities.*safari*();

capability.setJavascriptEnabled(**true**);

capability.setBrowserName("safari");

*browserName* = capability.getBrowserName();

*osName* = capability.getPlatform().name();

*browserVersion* = capability.getVersion();

// capability.setCapability(SafariDriver.CLEAN\_SESSION\_CAPABILITY,

// profile);

**this**.driver = **new** ~~SafariDriver~~(capability);

}

generalIndexpage = **new** GeneralIndexpage(driver);

generalVerification = **new** GeneralVerification(driver);

suiteName = testContext.getSuite().getName();

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

driver.get(testUrl);

System.***out***.println("TestData URL :: " + testUrl);

driver.manage().window().maximize();

*currentWindowHandle* = driver.getWindowHandle();

System.***out***.println("Current Window Handle ID : --->>" + *currentWindowHandle*);

suiteName = testContext.getSuite().getName();

System.***out***.println("Current Xml Suite is:---->" + suiteName);

}

/\*\*

\* Log For Failure Test Exception.

\*

\* **@param** tests

\*/

**private** **void** getShortException(IResultMap tests) {

**for** (ITestResult result : tests.getAllResults()) {

Throwable exception = result.getThrowable();

List<String> msgs = Reporter.*getOutput*(result);

**boolean** hasReporterOutput = msgs.size() > 0;

**boolean** hasThrowable = exception != **null**;

**if** (hasThrowable) {

**boolean** wantsMinimalOutput = result.getStatus() == ITestResult.***SUCCESS***;

**if** (hasReporterOutput) {

*log*("<h3>" + (wantsMinimalOutput ? "Expected Exception" : "Failure Reason:") + "</h3>");

}

// Getting first line of the stack trace

String str = Utils.*stackTrace*(exception, **true**)[0];

Scanner scanner = **new** Scanner(str);

String firstLine = scanner.nextLine();

*log*(firstLine);

}

}

}

/\*\*

\* After Method

\*

\* **@param** testResult

\*/

@AfterMethod(alwaysRun = **true**)

**public** **void** tearDown(ITestResult testResult) {

ITestContext ex = testResult.getTestContext();

**try** {

testName = testResult.getName();

**if** (!testResult.isSuccess()) {

/\* Print test result to Jenkins Console \*/

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

System.***out***.println("TEST FAILED - " + testName);

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

System.***out***.println("ERROR MESSAGE: " + testResult.getThrowable());

System.***out***.println("\n");

Reporter.*setCurrentTestResult*(testResult);

/\* Make a screenshot for test that failed \*/

String screenshotName = Common.*getCurrentTimeStampString*() + testName;

Reporter.*log*("<br> <b>Please look to the screenshot - </b>");

Common.*makeScreenshot*(driver, screenshotName);

// Reporter.log(testResult.getThrowable().getMessage());

getShortException(ex.getFailedTests());

} **else** {

**try** {

Common.*pause*(5);

/\*

\* driver.findElement(

\* By.xpath("//div[@class='container']//a[contains(.,'Logout')]" )) .click();

\*/

Common.*pause*(5);

} **catch** (Exception e) {

*log*("<br></br> Not able to perform logout");

}

System.***out***.println("TEST PASSED - " + testName + "\n"); // Print

// test

// resule

// to

// Jenkins

// Console

}

/\*

\* final File folder = new File("C:/Downloads\_new"); File files[] =

\* folder.listFiles();

\*

\* if (files.length > 0) { for (File f : files) { if (f.delete()) { System.out

\* .println("file deleted From Downloads\_new folder"); } }

\*

\* }

\*/

System.***out***.println("here is test status--------------------" + testResult.getStatus());

//driver.manage().deleteAllCookies();

driver.close();

driver.quit();

} **catch** (Throwable throwable) {

System.***out***.println("message from tear down" + throwable.getMessage());

}

}

/\*\*

\* Log given message to Reporter output.

\*

\* **@param** msg

\* Message/Log to be reported.

\*/

**public** **static** **void** log(String msg) {

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

Reporter.*log*("<br></br>" + msg);

}

**public** **static** **void** logList(ArrayList<String> msg) {

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

Reporter.*log*("<br></br>" + msg);

}

**public** **static** **void** testDescription(String msg) {

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

Reporter.*log*("<strong> <h4 style=\"color:DarkViolet\"> " + "Testcase Description: " + msg + "</h4> </strong>");

}

**public** **static** **void** testcaseId(String msg) {

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

Reporter.*log*("<strong> <h4 style=\"color:DarkViolet\"> " + "Test Case ID: " + msg + "</h4> </strong>");

}

**public** **static** **void** logverification(**int** i, String msg) {

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

Reporter.*log*("<br></br><b style=\"color:OrangeRed \"> Expected Result-" + i + ": </b><b>" + msg + "</b>");

}

}

/\*public class SeleniumInit {

}\*/

/\*public class SeleniumInit {

}\*/

**Utility**

**package** com.utility;

**import** java.io.File;

**import** java.io.FileInputStream;

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

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

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

**import** org.apache.poi.xssf.usermodel.XSSFWorkbook;

**public** **class** TestData {

**public** **static** Sheet getExcelSheet(**int** sheetIndex) {

String dataFilePath = "resources/Workbook.xlsx";

File datafile = **new** File(dataFilePath);

String fullpath = datafile.getAbsolutePath();

Sheet firstSheet = **null**;

**try** {

System.***out***.println("full path " + datafile.getAbsolutePath() + " con " + datafile.getCanonicalPath());

FileInputStream inputStream = **new** FileInputStream(**new** File(fullpath));

Workbook workbook = **new** XSSFWorkbook(inputStream);

firstSheet = workbook.getSheetAt(sheetIndex);

workbook.close();

inputStream.close();

} **catch** (Exception e) {

}

**return** firstSheet;

}

**public** **static** String firstname() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(0).getCell(0).getStringCellValue();

}

**public** **static** String lastname() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(1).getCell(0).getStringCellValue();

}

**public** **static** String phonenumber() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(2).getCell(0).getStringCellValue();

}

**public** **static** String username() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(3).getCell(0).getStringCellValue();

}

**public** **static** String emailid() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(4).getCell(0).getStringCellValue();

}

**public** **static** String aboutyourself() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(5).getCell(0).getStringCellValue();

}

**public** **static** String password() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(6).getCell(0).getStringCellValue();

}

**public** **static** String confirmpassword() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(7).getCell(0).getStringCellValue();

}

//Invalid data

**public** **static** String invalidphonenumber() {

System.***out***.println("Call getURL.......");

*getExcelSheet*(0).getRow(2).getCell(1).setCellType(Cell.***CELL\_TYPE\_STRING***);

**return** *getExcelSheet*(0).getRow(2).getCell(1).getStringCellValue();

}

**public** **static** String invalidpassword() {

// **TODO** Auto-generated method stub

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(6).getCell(1).getStringCellValue();

}

**public** **static** String donotmatchpassword() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(6).getCell(2).getStringCellValue();

}

**public** **static** String donotmatchconfirmpassword() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(7).getCell(2).getStringCellValue();

}

**public** **static** String invalidemailid() {

// **TODO** Auto-generated method stub

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(4).getCell(2).getStringCellValue();

}

**public** **static** String existusername() {

// **TODO** Auto-generated method stub

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(3).getCell(3).getStringCellValue();

}

**public** **static** String notregisteremailid() {

// **TODO** Auto-generated method stub

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(4).getCell(3).getStringCellValue();

}

**public** **static** String notexistusername() {

// **TODO** Auto-generated method stub

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(3).getCell(4).getStringCellValue();

}

**public** **static** String registeremailid() {

// **TODO** Auto-generated method stub

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(4).getCell(4).getStringCellValue();

}

//Demo Sites

**public** **static** String demo\_firstname() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(0).getCell(5).getStringCellValue();

}

**public** **static** String demo\_lastname() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(1).getCell(5).getStringCellValue();

}

**public** **static** String demo\_address() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(2).getCell(5).getStringCellValue();

}

**public** **static** String demo\_city() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(3).getCell(5).getStringCellValue();

}

**public** **static** String demo\_state\_province() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(4).getCell(5).getStringCellValue();

}

**public** **static** String demo\_country() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(0).getRow(5).getCell(5).getStringCellValue();

}

**public** **static** String demo\_postalcode() {

System.***out***.println("Call getURL.......");

*getExcelSheet*(0).getRow(6).getCell(5).setCellType(Cell.***CELL\_TYPE\_STRING***);

**return** *getExcelSheet*(0).getRow(6).getCell(5).getStringCellValue();

}

**public** **static** String demo\_phonenumber() {

System.***out***.println("Call getURL.......");

*getExcelSheet*(0).getRow(7).getCell(5).setCellType(Cell.***CELL\_TYPE\_STRING***);

**return** *getExcelSheet*(0).getRow(7).getCell(5).getStringCellValue();

}

//Threads and Shirts

**public** **static** String ts\_firstname() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(1).getRow(0).getCell(1).getStringCellValue();

}

**public** **static** String ts\_lastname() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(1).getRow(1).getCell(1).getStringCellValue();

}

**public** **static** String ts\_emailaddress() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(1).getRow(2).getCell(1).getStringCellValue();

}

**public** **static** String ts\_phone() {

System.***out***.println("Call getURL.......");

*getExcelSheet*(1).getRow(3).getCell(1).setCellType(Cell.***CELL\_TYPE\_STRING***);

**return** *getExcelSheet*(1).getRow(3).getCell(1).getStringCellValue();

}

**public** **static** String ts\_address() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(1).getRow(4).getCell(1).getStringCellValue();

}

**public** **static** String ts\_city() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(1).getRow(5).getCell(1).getStringCellValue();

}

**public** **static** String ts\_country() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(1).getRow(6).getCell(1).getStringCellValue();

}

**public** **static** String ts\_state() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(1).getRow(7).getCell(1).getStringCellValue();

}

**public** **static** String ts\_postalcode() {

System.***out***.println("Call getURL.......");

*getExcelSheet*(1).getRow(8).getCell(1).setCellType(Cell.***CELL\_TYPE\_STRING***);

**return** *getExcelSheet*(1).getRow(8).getCell(1).getStringCellValue();

}

//E-Signature

**public** **static** String esign\_username() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(2).getRow(0).getCell(1).getStringCellValue();

}

**public** **static** String esign\_password() {

System.***out***.println("Call getURL.......");

**return** *getExcelSheet*(2).getRow(1).getCell(1).getStringCellValue();

}

/\*public static String esign\_firstname() {

System.out.println("Call getURL.......");

return getExcelSheet(2).getRow(2).getCell(1).getStringCellValue();

}

public static String esign\_lastname() {

System.out.println("Call getURL.......");

return getExcelSheet(2).getRow(3).getCell(1).getStringCellValue();

}

public static String esign\_emailaddress() {

System.out.println("Call getURL.......");

return getExcelSheet(2).getRow(4).getCell(1).getStringCellValue();

}

public static String esign\_phonenumber() {

System.out.println("Call getURL.......");

return getExcelSheet(2).getRow(5).getCell(1).getStringCellValue();

}\*/

**private** **static** **final** String ***ALPHA\_NUMERIC\_STRING*** = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

**public** **static** String randomAlphaNumeric(**int** count) {

StringBuilder builder = **new** StringBuilder();

**while** (count-- != 0) {

**int** character = (**int**)(Math.*random*()\****ALPHA\_NUMERIC\_STRING***.length());

builder.append(***ALPHA\_NUMERIC\_STRING***.charAt(character));

}

**return** builder.toString();

}

**private** **static** **final** String ***ALPHA\_STRING*** = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

**public** **static** String randomAlpha(**int** count) {

StringBuilder builder = **new** StringBuilder();

**while** (count-- != 0) {

**int** character = (**int**)(Math.*random*()\****ALPHA\_STRING***.length());

builder.append(***ALPHA\_STRING***.charAt(character));

}

**return** builder.toString();

}

**private** **static** **final** String ***NUMERIC\_STRING*** = "01234567890123456789";

**public** **static** String randomNumeric(**int** count) {

StringBuilder builder = **new** StringBuilder();

**while** (count-- != 0) {

**int** character = (**int**)(Math.*random*()\****NUMERIC\_STRING***.length());

builder.append(***NUMERIC\_STRING***.charAt(character));

}

**return** builder.toString();

}

**private** **static** **final** String ***numberofdocuments*** = "2468";

**public** **static** String numberofducuments() {

**int** count=1;

StringBuilder builder = **new** StringBuilder();

**while** (count-- != 0) {

**int** character = (**int**)(Math.*random*()\****numberofdocuments***.length());

builder.append(***numberofdocuments***.charAt(character));

}

**return** builder.toString();

}

}

**Index Suite File**

**Esign**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<suite name=*"E-Signature - Test Suite 1"* verbose=*"10"* parallel=*"tests"*

thread-count=*"1"*>

<!-- Put IP of localhost/Grid Hub machine -->

<parameter name=*"selenium.host"* value=*"localhost"* />

<!-- Put Port of localhost RC/Grid Hub machine -->

<parameter name=*"selenium.port"* value=*"4444"* />

<!-- Put target browser name as below: 1) Firefox : firefox 2) chrome :

chrome 3) IExplorer : ie8 / ie9 -->

<parameter name=*"selenium.browser"* value=*"chrome"* />

<!-- <parameter name="selenium.url"

value="http://dashboard.staging.clearfunds.io/login" /> -->

<parameter name=*"selenium.url"*

value=*" https://go-env-syd1.clickandsign.net:11443/dashboard/"* />

<!-- <test name="Login Functionality with Valid Credentials :: 01">

<classes>

<class name="com.index.GeneralIndex">

<methods>

<include name="esign\_loginwithvalidcredentials" />

</methods>

</class>

</classes>

</test> -->

<test name=*"Create a package using IBM Digital Contracts Demo v. 1 workflow :: 01"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"esign\_createpackage\_borroweronly"* />

</methods>

</class>

</classes>

</test>

</suite>

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<suite name=*"Registration Form Fill up"* verbose=*"10"* parallel=*"tests"*

thread-count=*"1"*>

<!-- Put IP of localhost/Grid Hub machine -->

<parameter name=*"selenium.host"* value=*"localhost"* />

<!-- Put Port of localhost RC/Grid Hub machine -->

<parameter name=*"selenium.port"* value=*"4444"* />

<!-- Put target browser name as below: 1) Firefox : firefox 2) chrome :

chrome 3) IExplorer : ie8 / ie9 -->

<parameter name=*"selenium.browser"* value=*"chrome"* />

<!-- <parameter name="selenium.url"

value="http://dashboard.staging.clearfunds.io/login" /> -->

<parameter name=*"selenium.url"*

value=*"http://demoqa.com/registration/"* />

<test name=*"BlankFields Validation Messages:: 01"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"blankfields"* />

</methods>

</class>

</classes>

</test>

<test name=*"Minimum Length Validation Messages for phone field :: 02"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"phonefield\_minimum\_length"* />

</methods>

</class>

</classes>

</test>

<test name=*"Minimum Length Validation Messages for phone field :: 03"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"password\_minimum\_length"* />

</methods>

</class>

</classes>

</test>

<test name=*"Password Do not Match :: 04"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"passwordnotmatch"* />

</methods>

</class>

</classes>

</test>

<test name=*"Invalid Email Id validation :: 05"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"invalidemailid"* />

</methods>

</class>

</classes>

</test>

<test name=*"All Valid Data :: 06"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"allvaliddata"* />

</methods>

</class>

</classes>

</test>

<test name=*"All Valid Data Username Exist :: 07"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"usernamealreadyexist"* />

</methods>

</class>

</classes>

</test>

<test name=*"All Valid Data Username Exist :: 08"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"emailidalreadyexist"* />

</methods>

</class>

</classes>

</test>

</suite>

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<suite name=*"Demo Frame Work- Test Suite 1"* verbose=*"10"* parallel=*"tests"*

thread-count=*"1"*>

<!-- Put IP of localhost/Grid Hub machine -->

<parameter name=*"selenium.host"* value=*"localhost"* />

<!-- Put Port of localhost RC/Grid Hub machine -->

<parameter name=*"selenium.port"* value=*"4444"* />

<!-- Put target browser name as below: 1) Firefox : firefox 2) chrome :

chrome 3) IExplorer : ie8 / ie9 -->

<parameter name=*"selenium.browser"* value=*"chrome"* />

<!-- <parameter name="selenium.url"

value="http://dashboard.staging.clearfunds.io/login" /> -->

<parameter name=*"selenium.url"*

value=*" http://toolsqa.com"* />

<test name=*"Test Script :: 01"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"purchaseorcheckout"* />

</methods>

</class>

</classes>

</test>

</suite>

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<suite name=*"Demo Frame Work- Test Suite 1"* verbose=*"10"* parallel=*"tests"*

thread-count=*"1"*>

<!-- Put IP of localhost/Grid Hub machine -->

<parameter name=*"selenium.host"* value=*"localhost"* />

<!-- Put Port of localhost RC/Grid Hub machine -->

<parameter name=*"selenium.port"* value=*"4444"* />

<!-- Put target browser name as below: 1) Firefox : firefox 2) chrome :

chrome 3) IExplorer : ie8 / ie9 -->

<parameter name=*"selenium.browser"* value=*"chrome"* />

<!-- <parameter name="selenium.url"

value="http://dashboard.staging.clearfunds.io/login" /> -->

<parameter name=*"selenium.url"*

value=*" https://www.threadsandshirts.com/"* />

<test name=*"Test Script :: 01"*>

<classes>

<class name=*"com.index.GeneralIndex"*>

<methods>

<include name=*"addproducttocartandcheckout"* />

</methods>

</class>

</classes>

</test>

</suite>

**POM file**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>esignature</groupId>

<artifactId>esignature</artifactId>

<version>0.0.1-SNAPSHOT</version>

<build>

<!-- <sourceDirectory>src</sourceDirectory> -->

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.3</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.19.1</version>

<configuration>

<suiteXmlFiles>

<!-- TestNG suite XML files -->

<suiteXmlFile>Build/esignindex.xml</suiteXmlFile>

<!-- <suiteXmlFile>Build/index.xml</suiteXmlFile>

<suiteXmlFile>Build/index1.xml</suiteXmlFile>

<suiteXmlFile>Build/index2.xml</suiteXmlFile> -->

</suiteXmlFiles>

<reportsDirectory>test-output</reportsDirectory>

<properties>

<property>

<name>usedefaultlisteners</name>

<value>true</value>

</property>

<property>

<name>reporter</name>

<value>listenReport.Reporter</value>

</property>

</properties>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.13.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.8.8</version>

<!-- <scope>test</scope> -->

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-io -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-io</artifactId>

<version>1.3.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.7</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.13</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.13</version>

</dependency>

</dependencies>

</project>

**POM file**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>projectvp</groupId>

<artifactId>projectvp</artifactId>

<version>0.0.1-SNAPSHOT</version>

<build>

<!-- <sourceDirectory>src</sourceDirectory> -->

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.3</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.19.1</version>

<configuration>

<suiteXmlFiles>

<!-- TestNG suite XML files -->

<!-- <suiteXmlFile>Build/index.xml</suiteXmlFile>

<suiteXmlFile>Build/index1.xml</suiteXmlFile> -->

<suiteXmlFile>Build/index2.xml</suiteXmlFile>

</suiteXmlFiles>

<reportsDirectory>test-output</reportsDirectory>

<properties>

<property>

<name>usedefaultlisteners</name>

<value>true</value>

</property>

<property>

<name>reporter</name>

<value>listenReport.Reporter</value>

</property>

</properties>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.13.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.8.8</version>

<!-- <scope>test</scope> -->

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-io -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-io</artifactId>

<version>1.3.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.7</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.13</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.13</version>

</dependency>

</dependencies>

</project>