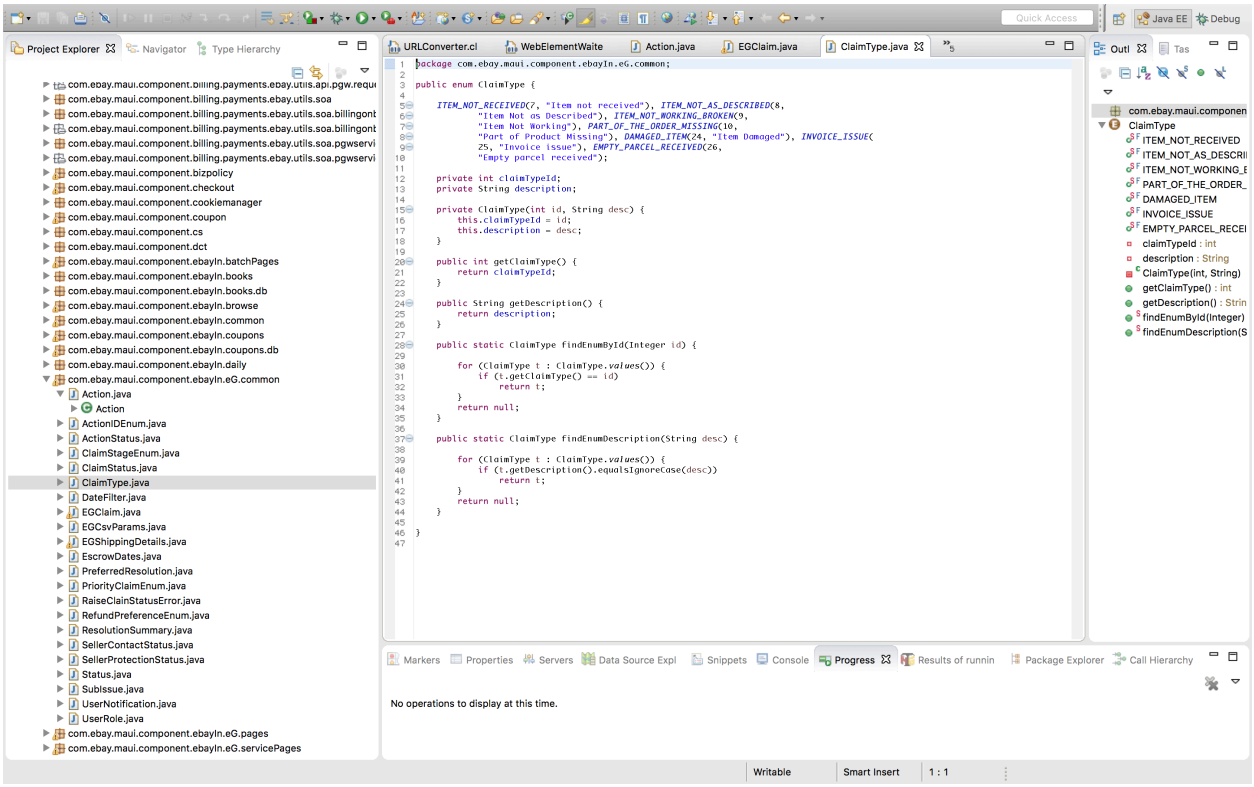
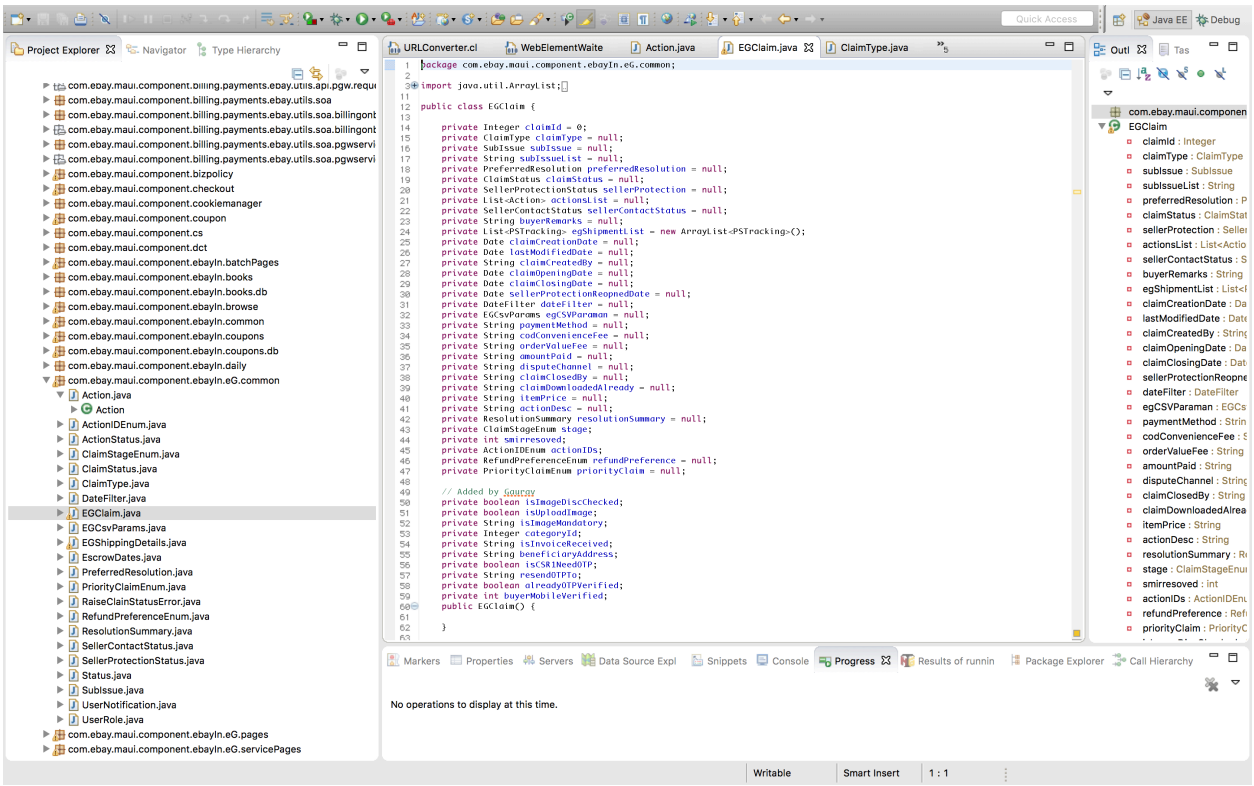
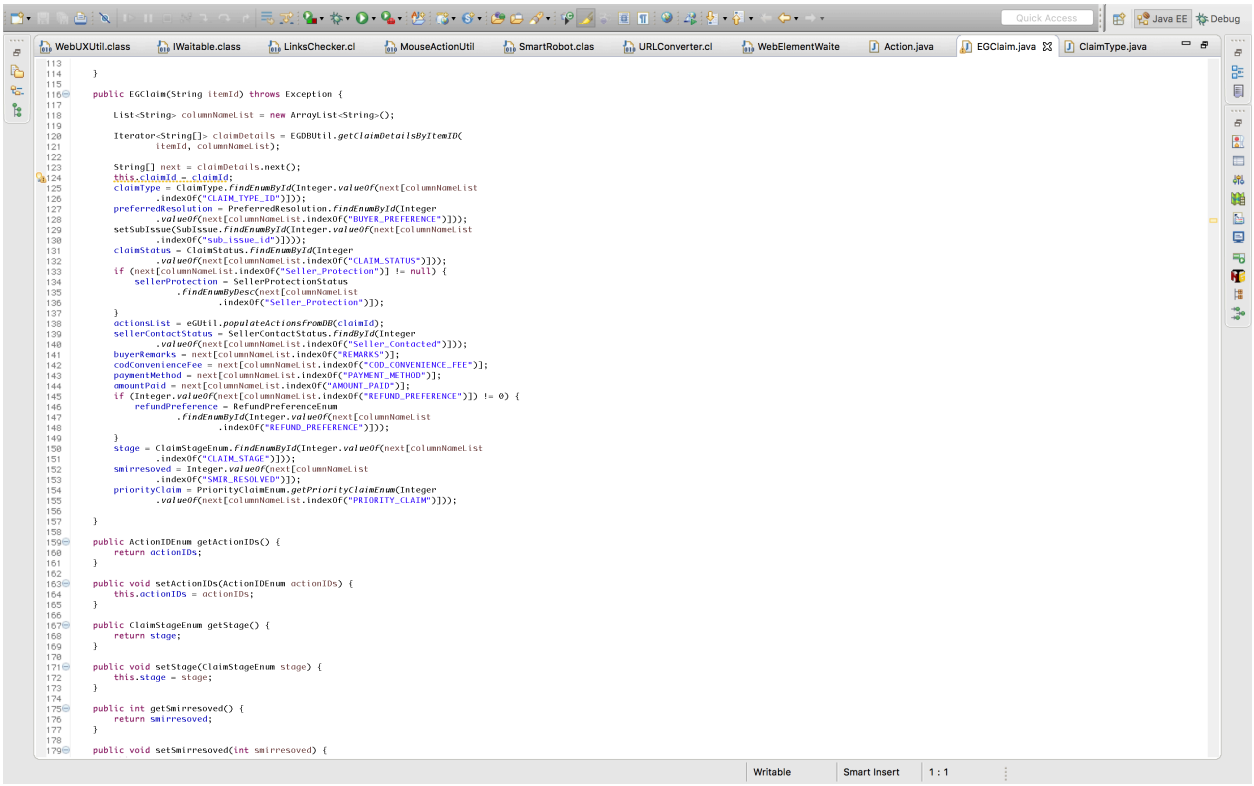
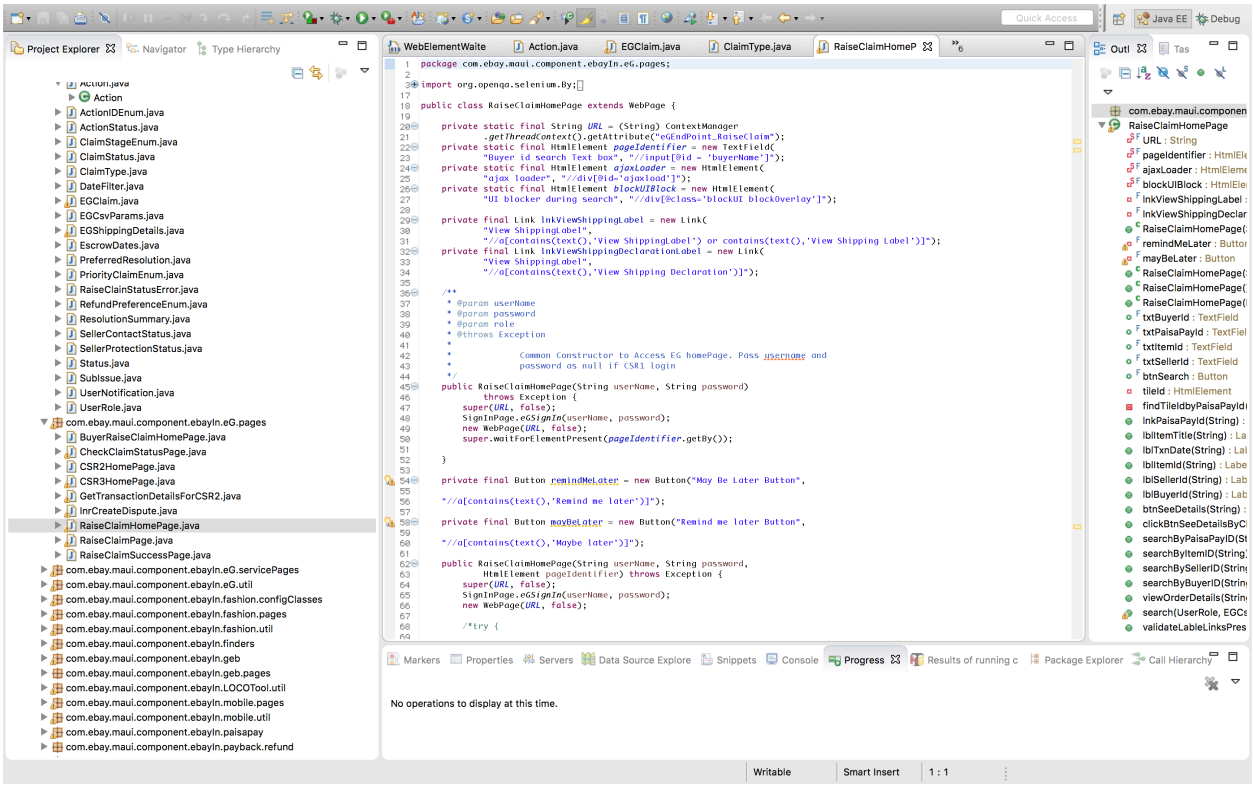



```
package com.ebay.maui.component.ebayIn.eG.common;

public enum ClaimType {

    ITEM_NOT_RECEIVED(7, "Item not received"), ITEM_NOT_AS_DESCRIBED(8,
            "Item Not as Described"), ITEM_NOT_WORKING_BROKEN(9,
            "Item Not Working"), PART_OF_THE_ORDER_MISSING(10,
            "Part of Product Missing"), DAMAGED_ITEM(24, "Item Damaged"), INVOICE_ISSUE(
            25, "Invoice issue"), EMPTY_PARCEL_RECEIVED(26,
            "Empty parcel received");

    private int claimTypeId;
    private String description;

    private ClaimType(int id, String desc) {
        this.claimTypeId = id;
        this.description = desc;
    }

    public int getClaimType() {
        return claimTypeId;
    }

    public String getDescription() {
        return description;
    }

    public static ClaimType findEnumById(Integer id) {

        for (ClaimType t : ClaimType.values()) {
            if (t.getClaimType() == id)
                return t;
        }
        return null;
    }

    public static ClaimType findEnumDescription(String desc) {

        for (ClaimType t : ClaimType.values()) {
            if (t.getDescription().equalsIgnoreCase(desc))
                return t;
        }
        return null;
    }
}
```

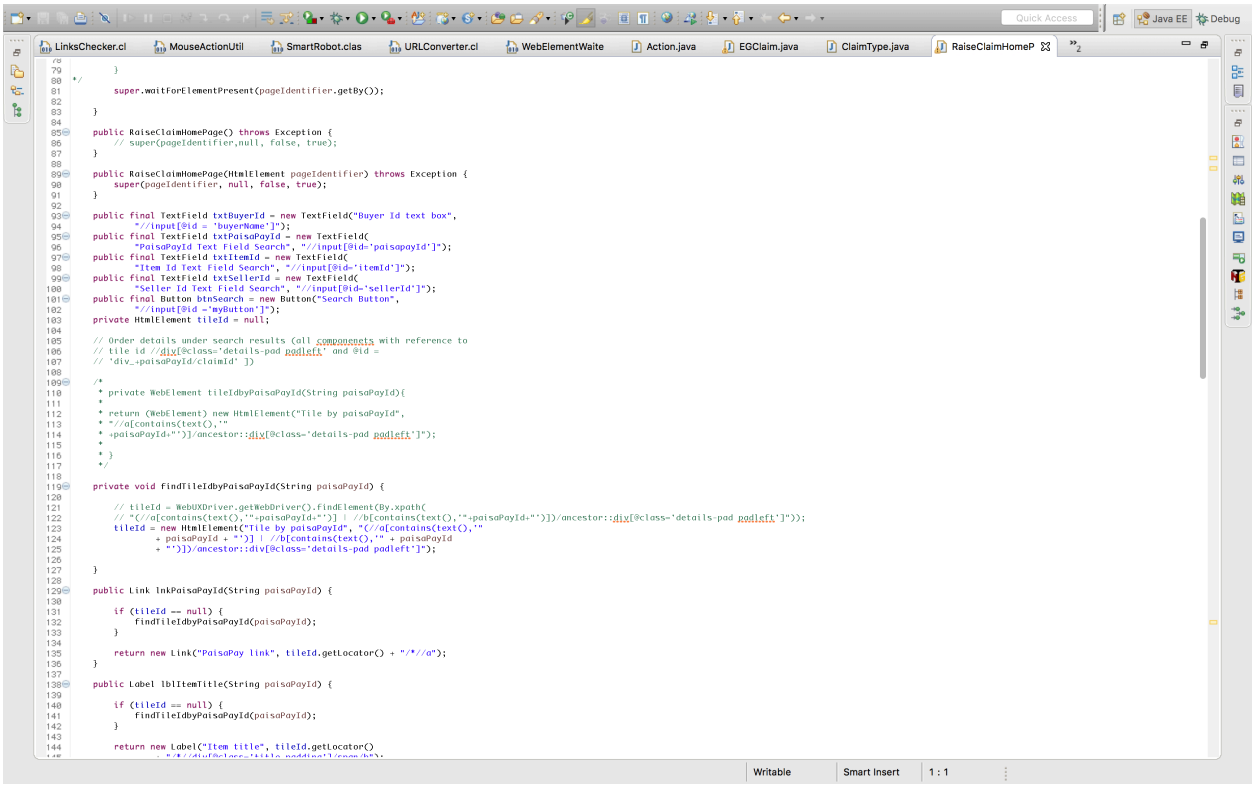



```
package com.ebay.maui.component.ebayIn.eG.common;

import java.util.ArrayList;

public class EGClaim {

    private Integer claimId = 0;
    private ClaimType claimType = null;
    private SubIssue subIssue = null;
    private String subIssueList = null;
    private PreferredResolution preferredResolution = null;
    private ClaimStatus claimStatus = null;
    private SellerProtectionStatus sellerProtection = null;
    private List<Action> actionsList = null;
    private SellerContactStatus sellerContactStatus = null;
    private String buyerRemarks = null;
    private List<PSTracking> egShipmentList = new ArrayList<PSTracking>();
    private Date claimCreationDate = null;
    private Date lastModifiedDate = null;
    private String claimCreatedBy = null;
    private Date claimOpeningDate = null;
    private Date claimClosingDate = null;
    private Date sellerProtectionReopnedDate = null;
    private DateFilter dateFilter = null;
    private EGCsvParams egCSVParaman = null;
    private String paymentMethod = null;
    private String codConvenienceFee = null;
    private String orderValueFee = null;
    private String amountPaid = null;
    private String disputeChannel = null;
    private String claimClosedBy = null;
    private String claimDownloadedAlready = null;
    private String itemPrice = null;
    private String actionDesc = null;
    private ResolutionSummary resolutionSummary = null;
    private ClaimStageEnum stage;
    private int smirresoved;
    private ActionIDEnum actionIDs;
    private RefundPreferenceEnum refundPreference = null;
    private PriorityClaimEnum priorityClaim = null;

    // Added by Gourav
    private boolean isImageDiscChecked;
    private boolean isUploadImage;
    private String isImageMandatory;
    private Integer categoryId;
    private String isInvoiceReceived;
    private String beneficiaryAddress;
    private boolean isCSRINeedOTP;
    private String resendOTPTo;
    private boolean alreadyOTPVerified;
    private int buyerMobileVerified;
    public EGClaim() {

    }
```

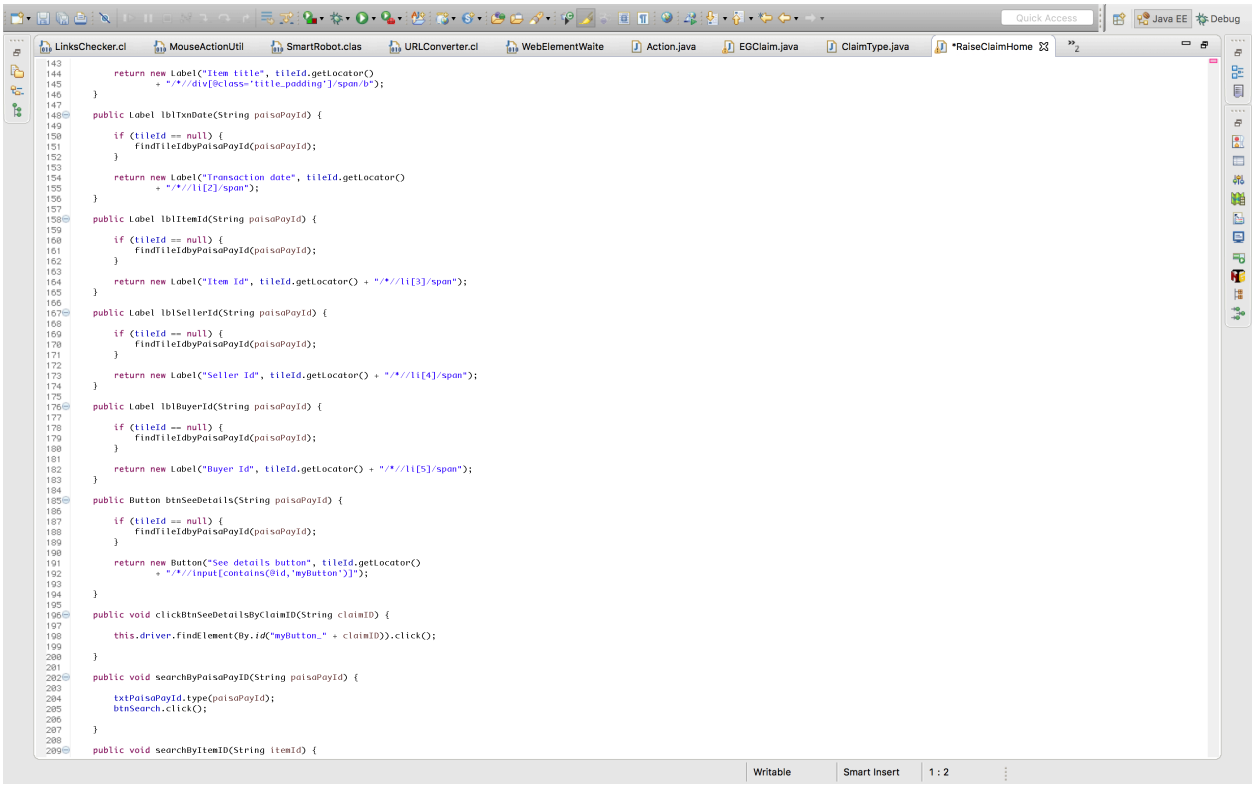

Quick Access
Java EE
Debug
WebUXUtil.class
IWaitable.class
LinksChecker.cl
MouseActionUtil
SmartRobot.clas
URLConverter.cl
WebElementWaite
Action.java
EGClaim.java
ClaimType.java
}
public EGClaim(String itemId) throws Exception {
List<String> columnNameList = new ArrayList<String>();
Iterator<String[]> claimDetails = EGDBUtil.getClaimDetailsByItemID(
itemId, columnNameList);
String[] next = claimDetails.next();
this.claimId = claimId;
claimType = ClaimType.findEnumById(Integer.valueOf(next[columnNameList
.indexOf("CLAIM_TYPE_ID")]));
preferredResolution = PreferredResolution.findEnumById(Integer
.valueOf(next[columnNameList.indexOf("BUYER_PREFERENCE")]));
setSubIssue(SubIssue.findEnumById(Integer.valueOf(next[columnNameList
.indexOf("sub_issue_id")])));
claimStatus = ClaimStatus.findEnumById(Integer
.valueOf(next[columnNameList.indexOf("CLAIM_STATUS")]));
if (next[columnNameList.indexOf("Seller_Protection")] != null) {
sellerProtection = SellerProtectionStatus
.findEnumByDesc(next[columnNameList
.indexOf("Seller_Protection")]);
}
actionsList = eGUtil.populateActionsfromDB(claimId);
sellerContactStatus = SellerContactStatus.findById(Integer
.valueOf(next[columnNameList.indexOf("Seller_Contacted")]));
buyerRemarks = next[columnNameList.indexOf("REMARKS")];
codConvenienceFee = next[columnNameList.indexOf("COD_CONVENIENCE_FEE")];
paymentMethod = next[columnNameList.indexOf("PAYMENT_METHOD")];
amountPaid = next[columnNameList.indexOf("AMOUNT_PAID")];
if (Integer.valueOf(next[columnNameList.indexOf("REFUND_PREFERENCE")]) != 0) {
refundPreference = RefundPreferenceEnum
.findEnumById(Integer.valueOf(next[columnNameList
.indexOf("REFUND_PREFERENCE")]));
}
stage = ClaimStageEnum.findEnumById(Integer.valueOf(next[columnNameList
.indexOf("CLAIM_STAGE")]));
smirresoved = Integer.valueOf(next[columnNameList
.indexOf("SMIR_RESOLVED")]);
priorityClaim = PriorityClaimEnum.getPriorityClaimEnum(Integer
.valueOf(next[columnNameList.indexOf("PRIORITY_CLAIM")]));
}
public ActionIDEnum getActionIDs() {
return actionIDs;
}
public void setActionIDs(ActionIDEnum actionIDs) {
this.actionIDs = actionIDs;
}
public ClaimStageEnum getStage() {
return stage;
}
public void setStage(ClaimStageEnum stage) {
this.stage = stage;
}
public int getSmirresoved() {
return smirresoved;
}
public void setSmirresoved(int smirresoved) {
Writable
Smart Insert
1 : 1



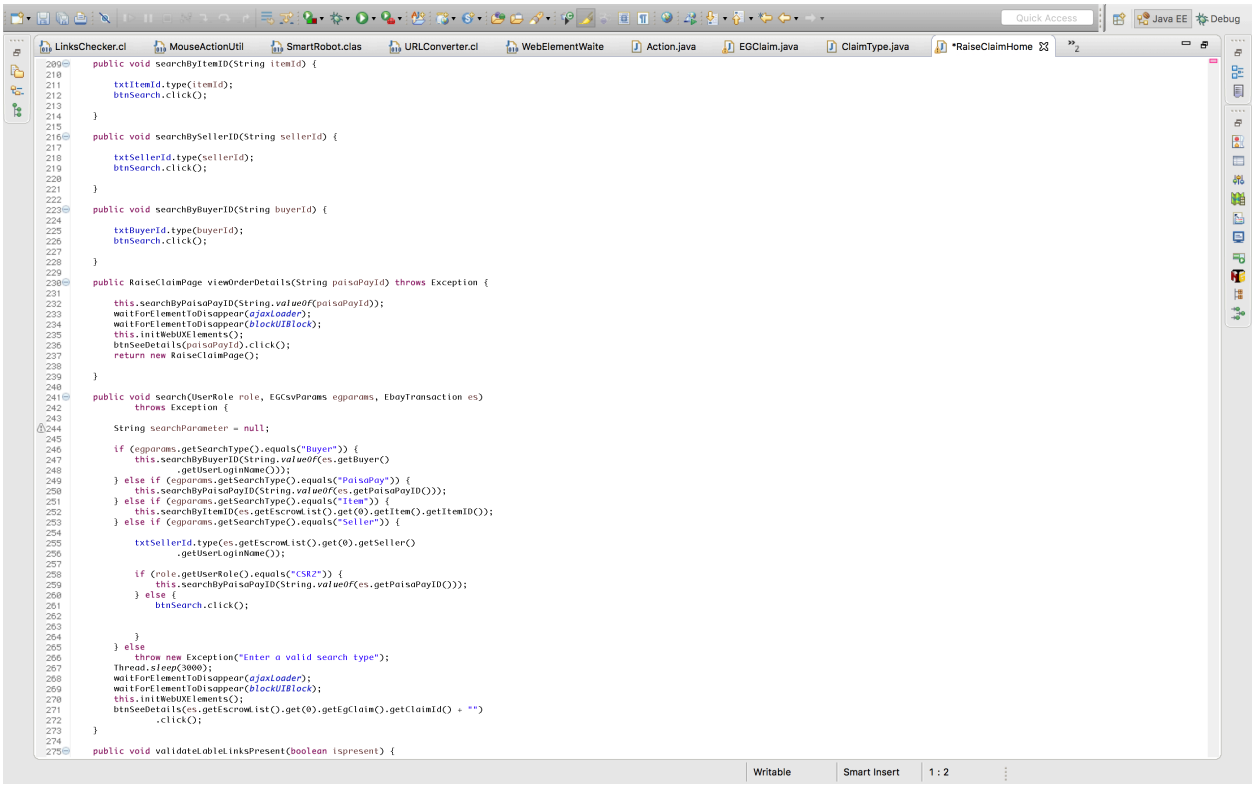
```
package com.ebay.maui.component.ebayIn.eG.pages;

import org.openqa.selenium.By;

public class RaiseClaimHomePage extends WebPage {

    private static final String URL = (String) ContextManager
            .getThreadContext().getAttribute("eGEndPoint_RaiseClaim");
    private static final HtmlElement pageIdentifier = new TextField(
            "Buyer id search Text box", "//input[@id = 'buyerName']");
    private static final HtmlElement ajaxLoader = new HtmlElement(
            "ajax loader", "//div[@id='ajaxload']");
    private static final HtmlElement blockUIBlock = new HtmlElement(
            "UI blocker during search", "//div[@class='blockUI blockOverlay']");

    private final Link lnkViewShippingLabel = new Link(
            "View ShippingLabel",
            "//a[contains(text(),'View ShippingLabel') or contains(text(),'View Shipping Label')]");
    private final Link lnkViewShippingDeclarationLabel = new Link(
            "View ShippingLabel",
            "//a[contains(text(),'View Shipping Declaration')]");

    /**
     * @param userName
     * @param password
     * @param role
     * @throws Exception
     *
     *             Common Constructor to Access EG homePage. Pass username and
     *             password as null if CSR1 login
     */
    public RaiseClaimHomePage(String userName, String password)
            throws Exception {
        super(URL, false);
        SignInPage.eGSignIn(userName, password);
        new WebPage(URL, false);
        super.waitForElementPresent(pageIdentifier.getBy());

    }

    private final Button remindMeLater = new Button("May Be Later Button",

    "//a[contains(text(),'Remind me later')]");

    private final Button mayBeLater = new Button("Remind me later Button",

    "//a[contains(text(),'Maybe later')]");

    public RaiseClaimHomePage(String userName, String password,
            HtmlElement pageIdentifier) throws Exception {
        super(URL, false);
        SignInPage.eGSignIn(userName, password);
        new WebPage(URL, false);

        /*try {
```



```
        }
*/
        super.waitForElementPresent(pageIdentifier.getBy());

    }

    public RaiseClaimHomePage() throws Exception {
        // super(pageIdentifier,null, false, true);
    }

    public RaiseClaimHomePage(HtmlElement pageIdentifier) throws Exception {
        super(pageIdentifier, null, false, true);
    }

    public final TextField txtBuyerId = new TextField("Buyer Id text box",
            "//input[@id = 'buyerName']");
    public final TextField txtPaisaPayId = new TextField(
            "PaisaPayId Text Field Search", "//input[@id='paisapayId']");
    public final TextField txtItemId = new TextField(
            "Item Id Text Field Search", "//input[@id='itemId']");
    public final TextField txtSellerId = new TextField(
            "Seller Id Text Field Search", "//input[@id='sellerId']");
    public final Button btnSearch = new Button("Search Button",
            "//input[@id ='myButton']");
    private HtmlElement tileId = null;

    // Order details under search results (all components with reference to
    // tile id //div[@class='details-pad padleft' and @id =
    // 'div_+paisaPayId/claimId' ])

    /*
     * private WebElement tileIdbyPaisaPayId(String paisaPayId){
     *
     * return (WebElement) new HtmlElement("Tile by paisaPayId",
     * "//a[contains(text(),'"
     * +paisaPayId+"')]/ancestor::div[@class='details-pad padleft']");
     *
     * }
     */

    private void findTileIdbyPaisaPayId(String paisaPayId) {

        // tileId = WebUXDriver.getWebDriver().findElement(By.xpath(
        // "(//a[contains(text(),'"+paisaPayId+"')] | //b[contains(text(),'"+paisaPayId+"')])/ancestor::div[@class='details-pad padleft']"));
        tileId = new HtmlElement("Tile by paisaPayId", "(//a[contains(text(),'"
                + paisaPayId + "')] | //b[contains(text(),'" + paisaPayId
                + "')])/ancestor::div[@class='details-pad padleft']");

    }

    public Link lnkPaisaPayId(String paisaPayId) {

        if (tileId == null) {
            findTileIdbyPaisaPayId(paisaPayId);
        }

        return new Link("PaisaPay link", tileId.getLocator() + "/*//a");
    }

    public Label lblItemTitle(String paisaPayId) {

        if (tileId == null) {
            findTileIdbyPaisaPayId(paisaPayId);
        }

        return new Label("Item title", tileId.getLocator()
```



```
        return new Label("Item title", tileId.getLocator()
                + "/*//div[@class='title_padding']/span/b");
    }

    public Label lblTxnDate(String paisaPayId) {

        if (tileId == null) {
            findTileIdbyPaisaPayId(paisaPayId);
        }

        return new Label("Transaction date", tileId.getLocator()
                + "/*//li[2]/span");
    }

    public Label lblItemId(String paisaPayId) {

        if (tileId == null) {
            findTileIdbyPaisaPayId(paisaPayId);
        }

        return new Label("Item Id", tileId.getLocator() + "/*//li[3]/span");
    }

    public Label lblSellerId(String paisaPayId) {

        if (tileId == null) {
            findTileIdbyPaisaPayId(paisaPayId);
        }

        return new Label("Seller Id", tileId.getLocator() + "/*//li[4]/span");
    }

    public Label lblBuyerId(String paisaPayId) {

        if (tileId == null) {
            findTileIdbyPaisaPayId(paisaPayId);
        }

        return new Label("Buyer Id", tileId.getLocator() + "/*//li[5]/span");
    }

    public Button btnSeeDetails(String paisaPayId) {

        if (tileId == null) {
            findTileIdbyPaisaPayId(paisaPayId);
        }

        return new Button("See details button", tileId.getLocator()
                + "/*//input[contains(@id,'myButton')]");

    }

    public void clickBtnSeeDetailsByClaimID(String claimID) {

        this.driver.findElement(By.id("myButton_" + claimID)).click();

    }

    public void searchByPaisaPayID(String paisaPayId) {

        txtPaisaPayId.type(paisaPayId);
        btnSearch.click();

    }

    public void searchByItemID(String itemId) {
```

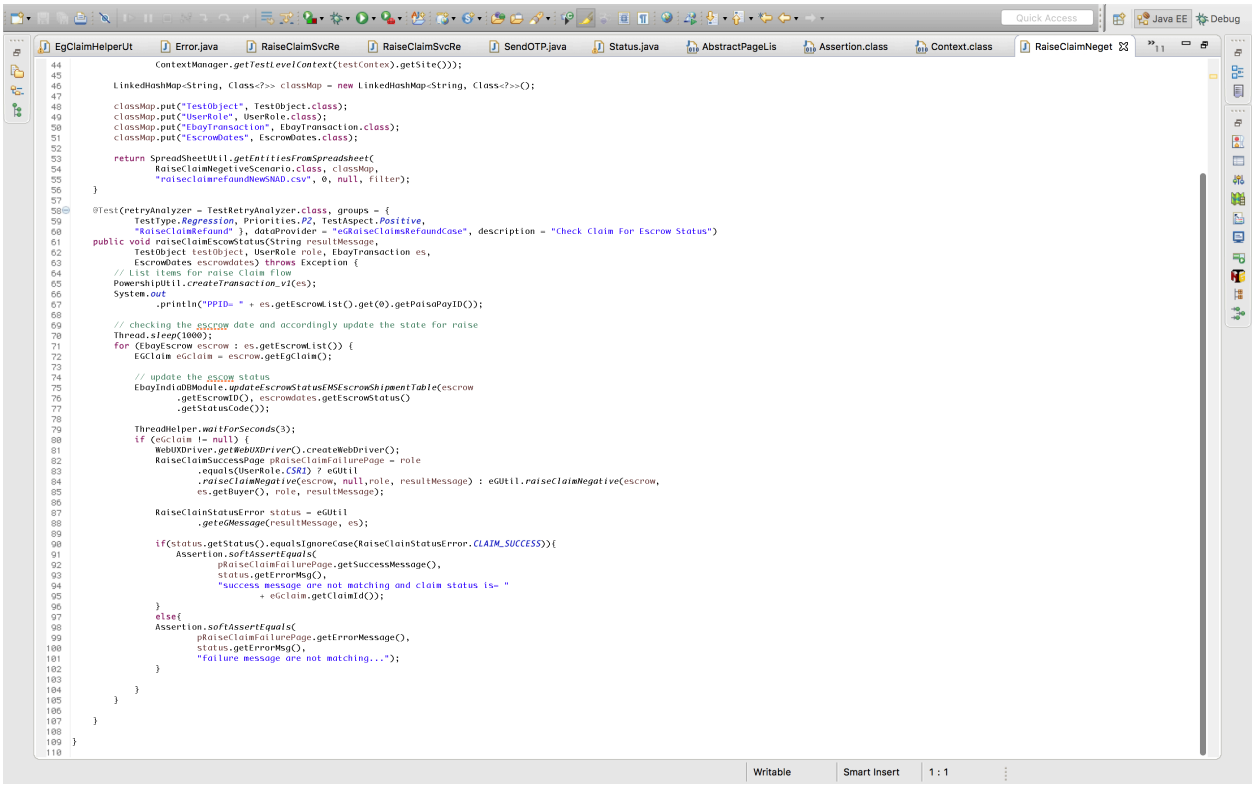


```
    public void searchByItemID(String itemId) {

        txtItemId.type(itemId);
        btnSearch.click();

    }

    public void searchBySellerID(String sellerId) {

        txtSellerId.type(sellerId);
        btnSearch.click();

    }

    public void searchByBuyerID(String buyerId) {

        txtBuyerId.type(buyerId);
        btnSearch.click();

    }

    public RaiseClaimPage viewOrderDetails(String paisaPayId) throws Exception {

        this.searchByPaisaPayID(String.valueOf(paisaPayId));
        waitForElementToDisappear(ajaxLoader);
        waitForElementToDisappear(blockUIBlock);
        this.initWebUXElements();
        btnSeeDetails(paisaPayId).click();
        return new RaiseClaimPage();

    }

    public void search(UserRole role, EGCsvParams egparams, EbayTransaction es)
            throws Exception {

        String searchParameter = null;

        if (egparams.getSearchType().equals("Buyer")) {
            this.searchByBuyerID(String.valueOf(es.getBuyer()
                    .getUserLoginName()));
        } else if (egparams.getSearchType().equals("PaisaPay")) {
            this.searchByPaisaPayID(String.valueOf(es.getPaisaPayID()));
        } else if (egparams.getSearchType().equals("Item")) {
            this.searchByItemID(es.getEscrowList().get(0).getItem().getItemID());
        } else if (egparams.getSearchType().equals("Seller")) {

            txtSellerId.type(es.getEscrowList().get(0).getSeller()
                    .getUserLoginName());

            if (role.getUserRole().equals("CSR2")) {
                this.searchByPaisaPayID(String.valueOf(es.getPaisaPayID()));
            } else {
                btnSearch.click();


            }
        } else
            throw new Exception("Enter a valid search type");
        Thread.sleep(3000);
        waitForElementToDisappear(ajaxLoader);
        waitForElementToDisappear(blockUIBlock);
        this.initWebUXElements();
        btnSeeDetails(es.getEscrowList().get(0).getEgClaim().getClaimId() + "")
                .click();
    }

    public void validateLableLinksPresent(boolean ispresent) {
```

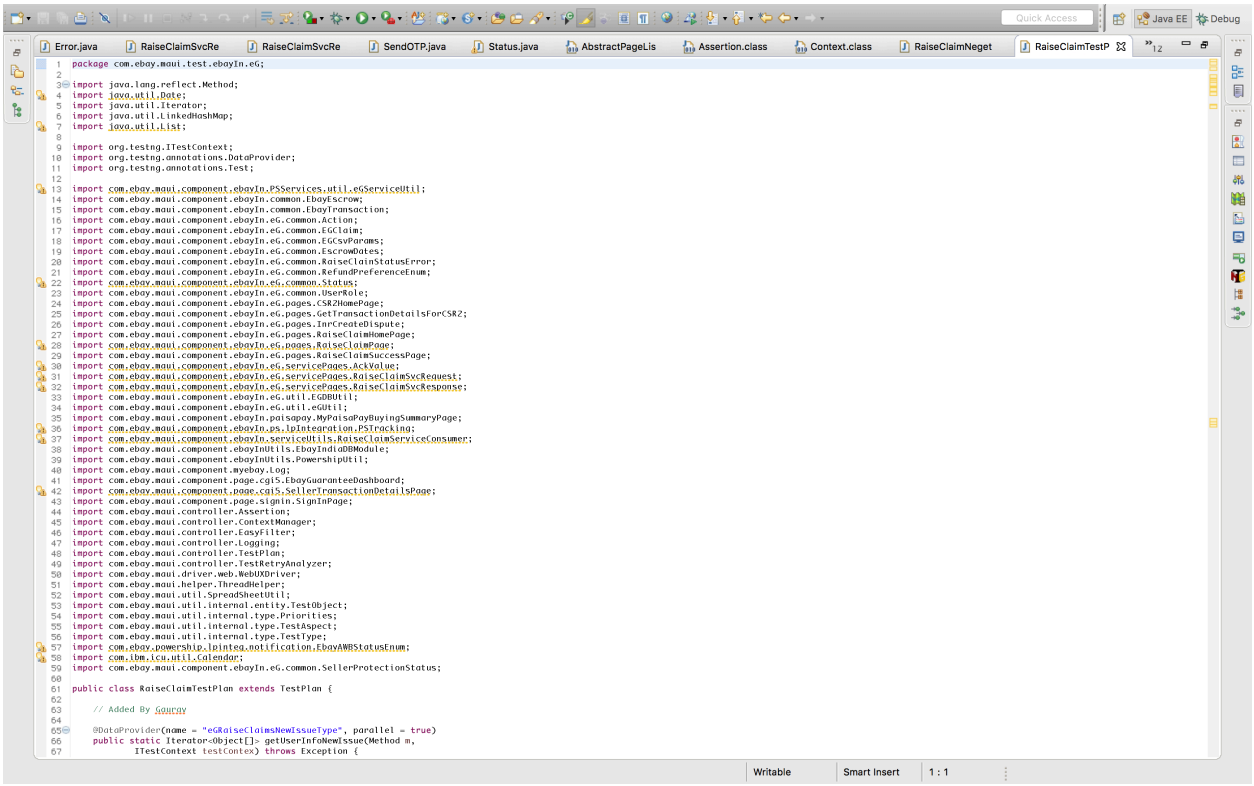
```
        waitForElementToDisappear(ajaxLoader);
        waitForElementToDisappear(blockUIBlock);
        this.initWebUXElements();
        btnSeeDetails(paisaPayId).click();
        return new RaiseClaimPage();

    }

    public void search(UserRole role, EGCsvParams egparams, EbayTransaction es)
            throws Exception {

        String searchParameter = null;

        if (egparams.getSearchType().equals("Buyer")) {
            this.searchByBuyerID(String.valueOf(es.getBuyer()
                    .getUserLoginName()));
        } else if (egparams.getSearchType().equals("PaisaPay")) {
            this.searchByPaisaPayID(String.valueOf(es.getPaisaPayID()));
        } else if (egparams.getSearchType().equals("Item")) {
            this.searchByItemID(es.getEscrowList().get(0).getItem().getItemID());
        } else if (egparams.getSearchType().equals("Seller")) {

            txtSellerId.type(es.getEscrowList().get(0).getSeller()
                    .getUserLoginName());

            if (role.getUserRole().equals("CSR2")) {
                this.searchByPaisaPayID(String.valueOf(es.getPaisaPayID()));
            } else {
                btnSearch.click();


            }
        } else
            throw new Exception("Enter a valid search type");
        Thread.sleep(3000);
        waitForElementToDisappear(ajaxLoader);
        waitForElementToDisappear(blockUIBlock);
        this.initWebUXElements();
        btnSeeDetails(es.getEscrowList().get(0).getEgClaim().getClaimId() + "")
                .click();
    }

    public void validateLableLinksPresent(boolean ispresent) {

        if (ispresent) {
            Assertion.assertTrue(isElementPresent(By.xpath(lnkViewShippingLabel
                    .getLocator())),
                    "View Shipping Label does not present in History");
            Assertion
                    .assertTrue(
                            isElementPresent(By
                                    .xpath(lnkViewShippingDeclarationLabel
                                            .getLocator())),
                            "view shippin Declaration label does not present in History");
        } else {
            Assertion.assertFalse(isElementPresent(By
                    .xpath(lnkViewShippingLabel.getLocator())),
                    "View Shipping Label present in History");
            Assertion.assertFalse(isElementPresent(By
                    .xpath(lnkViewShippingDeclarationLabel.getLocator())),
                    "view shippin Declaration label present in History");
        }

    }
}
```

Plan

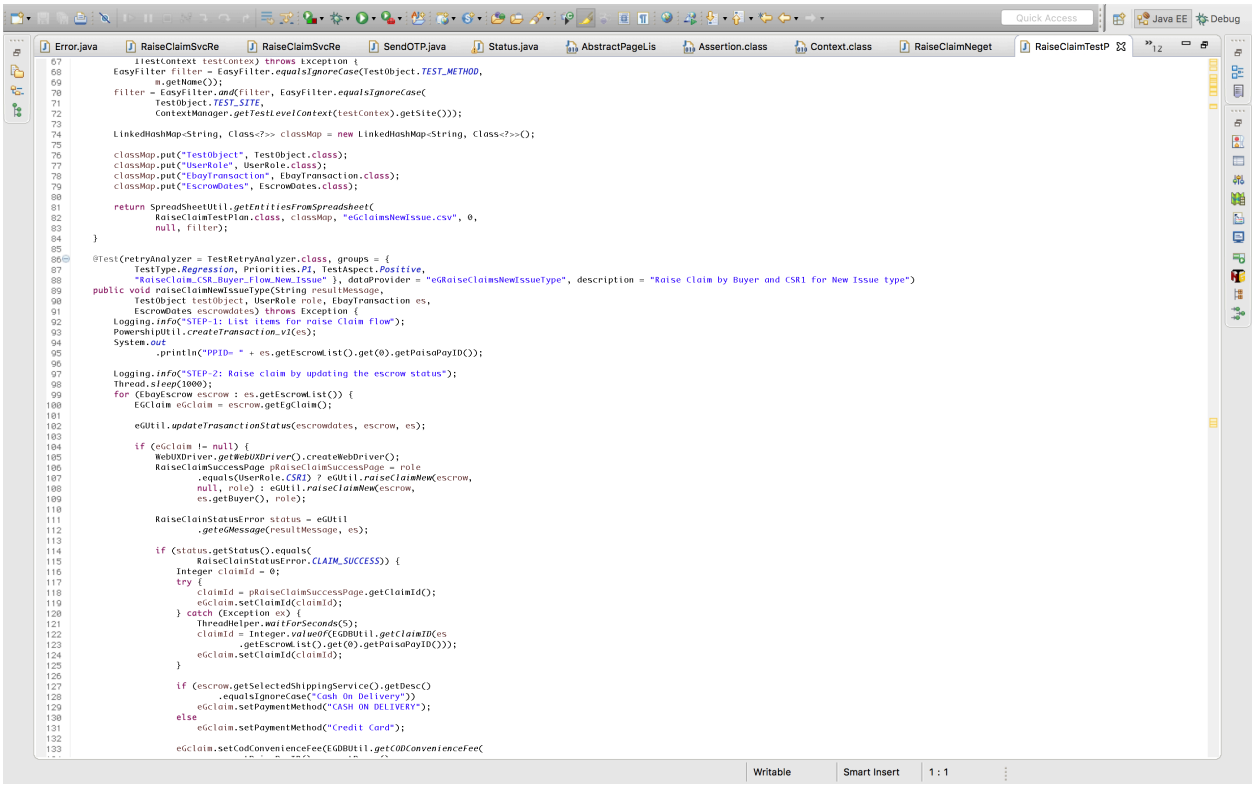
```
package com.ebay.maui.test.ebayIn.eG;

import java.lang.reflect.Method;

public class RaiseClaimNegetiveScenario extends TestPlan {

    @DataProvider(name = "eGRaiseClaimsRefaundCase", parallel = true)
    public static Iterator<Object[]> getUserInfo(Method m,
            ITestContext testContex) throws Exception {
        EasyFilter filter = EasyFilter.equalsIgnoreCase(TestObject.TEST_METHOD,
                m.getName());
        filter = EasyFilter.and(filter, EasyFilter.equalsIgnoreCase(
                TestObject.TEST_SITE,
                ContextManager.getTestLevelContext(testContex).getSite()));

        LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();

        classMap.put("TestObject", TestObject.class);
        classMap.put("UserRole", UserRole.class);
        classMap.put("EbayTransaction", EbayTransaction.class);
        classMap.put("EscrowDates", EscrowDates.class);

        return SpreadSheetUtil.getEntitiesFromSpreadsheet(
                RaiseClaimNegetiveScenario.class, classMap,
                "raiseclaimrefaundNewSNAD.csv", 0, null, filter);
    }

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
            TestType.Regression, Priorities.P2, TestAspect.Positive,
            "RaiseClaimRefaund" }, dataProvider = "eGRaiseClaimsRefaundCase", description = "Check Claim For Escrow Status")
    public void raiseClaimEscowStatus(String resultMessage,
            TestObject testObject, UserRole role, EbayTransaction es,
            EscrowDates escrowdates) throws Exception {
        // List items for raise Claim flow
        PowershipUtil.createTransaction_v1(es);
        System.out
                .println("PPID= " + es.getEscrowList().get(0).getPaisaPayID());

        // checking the escrow date and accordingly update the state for raise
        Thread.sleep(1000);
        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();

            // update the escow status
            EbayIndiaDBModule.updateEscrowStatusEMSEscrowShipmentTable(escrow
                    .getEscrowID(), escrowdates.getEscrowStatus()
                    .getStatusCode());

            ThreadHelper.waitForSeconds(3);
            if (eGclaim != null) {
                WebUXDriver.getWebUXDriver().createWebDriver();
                RaiseClaimSuccessPage pRaiseClaimFailurePage = role
                        .equals(UserRole.CSR1) ? eGUtil
                        .raiseClaimNegative(escrow, null,role, resultMessage) : eGUtil.raiseClaimNegative(escrow,
                        es.getBuyer(), role, resultMessage);

                RaiseClaimStatusError status = eGUtil
                        .geteGMessage(resultMessage, es);

                if(status.getStatus().equalsIgnoreCase(RaiseClaimStatusError.CLAIM_SUCCESS)){
                    Assertion.softAssertEquals(
                            pRaiseClaimFailurePage.getSuccessMessage(),
                            status.getErrorMsg(),
                            "Success message are not matching and claim status is= "
                                    + eGclaim.getClaimId());
                }
                else{
```



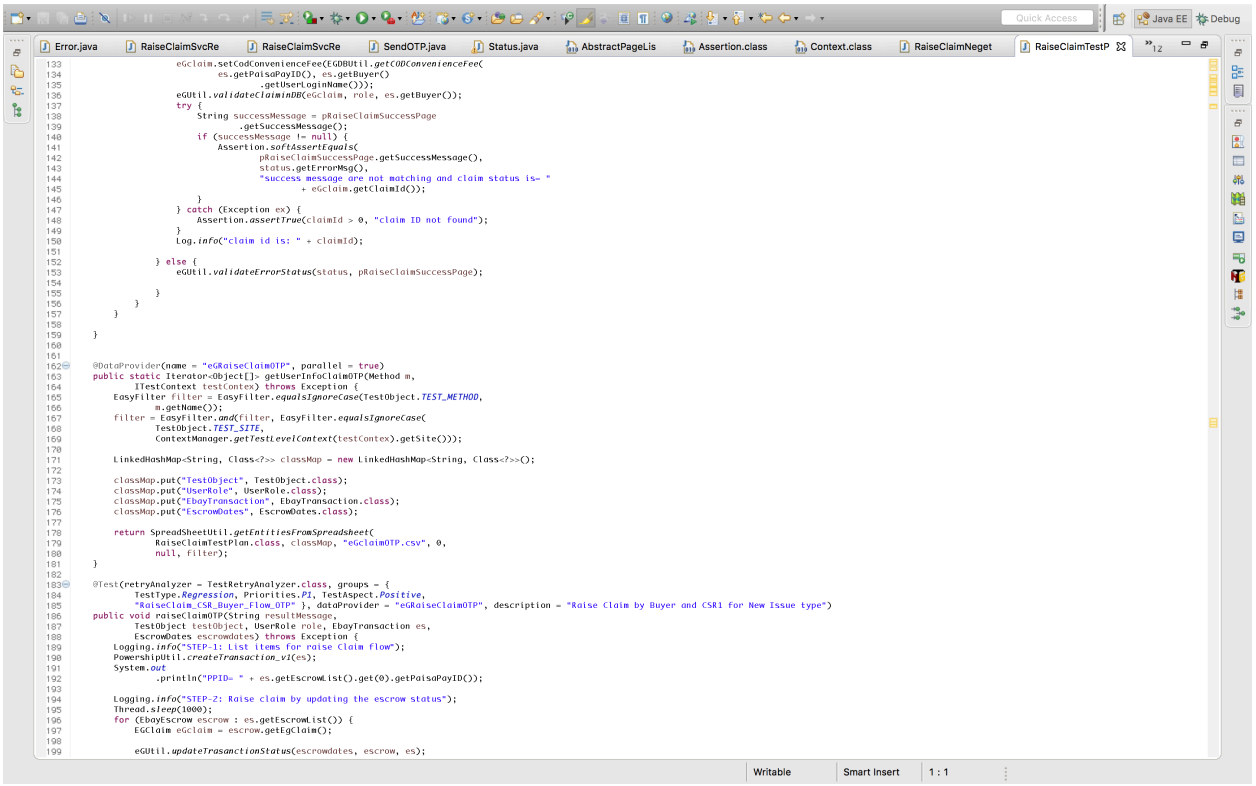
```
                ContextManager.getTestLevelContext(testContex).getSite()));

        LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();

        classMap.put("TestObject", TestObject.class);
        classMap.put("UserRole", UserRole.class);
        classMap.put("EbayTransaction", EbayTransaction.class);
        classMap.put("EscrowDates", EscrowDates.class);

        return SpreadSheetUtil.getEntitiesFromSpreadsheet(
                RaiseClaimNegetiveScenario.class, classMap,
                "raiseclaimrefaundNewSNAD.csv", 0, null, filter);
    }

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
            TestType.Regression, Priorities.P2, TestAspect.Positive,
            "RaiseClaimRefaund" }, dataProvider = "eGRaiseClaimsRefaundCase", description = "Check Claim For Escrow Status")
    public void raiseClaimEscrowStatus(String resultMessage,
            TestObject testObject, UserRole role, EbayTransaction es,
            EscrowDates escrowdates) throws Exception {
        // List items for raise Claim flow
        PowershipUtil.createTransaction_v1(es);
        System.out
                .println("PPID= " + es.getEscrowList().get(0).getPaisaPayID());

        // checking the escrow date and accordingly update the state for raise
        Thread.sleep(1000);
        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();

            // update the escow status
            EbayIndiaDBModule.updateEscrowStatusEMSEscrowShipmentTable(escrow
                    .getEscrowID(), escrowdates.getEscrowStatus()
                    .getStatusCode());

            ThreadHelper.waitForSeconds(3);
            if (eGclaim != null) {
                WebUXDriver.getWebUXDriver().createWebDriver();
                RaiseClaimSuccessPage pRaiseClaimFailurePage = role
                        .equals(UserRole.CSR1) ? eGUtil
                        .raiseClaimNegative(escrow, null,role, resultMessage) : eGUtil.raiseClaimNegative(escrow,
                        es.getBuyer(), role, resultMessage);

                RaiseClainStatusError status = eGUtil
                        .geteGMessage(resultMessage, es);

                if(status.getStatus().equalsIgnoreCase(RaiseClainStatusError.CLAIM_SUCCESS)){
                    Assertion.softAssertEquals(
                            pRaiseClaimFailurePage.getSuccessMessage(),
                            status.getErrorMsg(),
                            "success message are not matching and claim status is= "
                                    + eGclaim.getClaimId());
                }
                else{
                Assertion.softAssertEquals(
                        pRaiseClaimFailurePage.getErrorMessage(),
                        status.getErrorMsg(),
                        "failure message are not matching...");
                }

            }
        }

    }
}
```



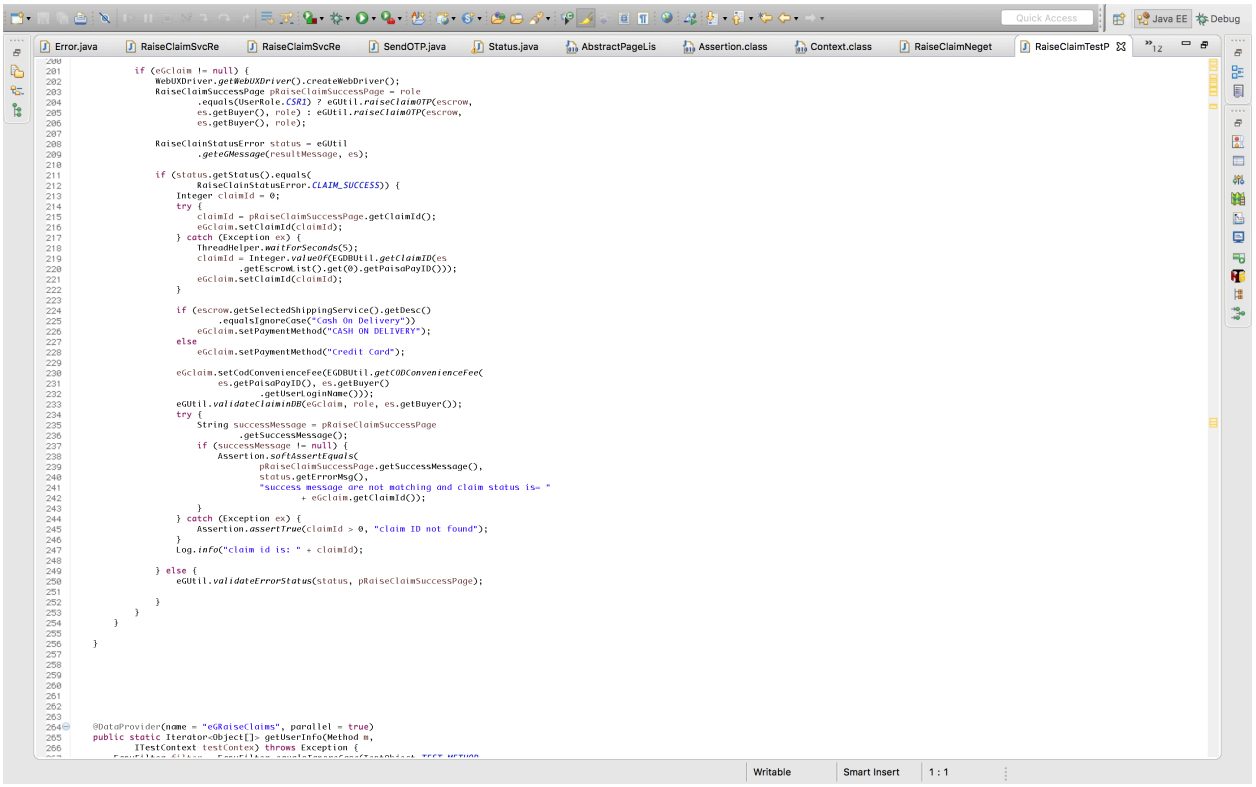
```
package com.ebay.maui.test.ebayIn.eG;

import java.lang.reflect.Method;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.List;

import org.testng.ITestContext;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

import com.ebay.maui.component.ebayIn.PSServices.util.eGServiceUtil;
import com.ebay.maui.component.ebayIn.common.EbayEscrow;
import com.ebay.maui.component.ebayIn.common.EbayTransaction;
import com.ebay.maui.component.ebayIn.eG.common.Action;
import com.ebay.maui.component.ebayIn.eG.common.EGClaim;
import com.ebay.maui.component.ebayIn.eG.common.EGCsvParams;
import com.ebay.maui.component.ebayIn.eG.common.EscrowDates;
import com.ebay.maui.component.ebayIn.eG.common.RaiseClainStatusError;
import com.ebay.maui.component.ebayIn.eG.common.RefundPreferenceEnum;
import com.ebay.maui.component.ebayIn.eG.common.Status;
import com.ebay.maui.component.ebayIn.eG.common.UserRole;
import com.ebay.maui.component.ebayIn.eG.pages.CSR2HomePage;
import com.ebay.maui.component.ebayIn.eG.pages.GetTransactionDetailsForCSR2;
import com.ebay.maui.component.ebayIn.eG.pages.InrCreateDispute;
import com.ebay.maui.component.ebayIn.eG.pages.RaiseClaimHomePage;
import com.ebay.maui.component.ebayIn.eG.pages.RaiseClaimPage;
import com.ebay.maui.component.ebayIn.eG.pages.RaiseClaimSuccessPage;
import com.ebay.maui.component.ebayIn.eG.servicePages.AckValue;
import com.ebay.maui.component.ebayIn.eG.servicePages.RaiseClaimSvcRequest;
import com.ebay.maui.component.ebayIn.eG.servicePages.RaiseClaimSvcResponse;
import com.ebay.maui.component.ebayIn.eG.util.EGDBUtil;
import com.ebay.maui.component.ebayIn.eG.util.eGUtil;
import com.ebay.maui.component.ebayIn.paisapay.MyPaisaPayBuyingSummaryPage;
import com.ebay.maui.component.ebayIn.ps.ipIntegration.PSTracking;
import com.ebay.maui.component.ebayIn.serviceUtils.RaiseClaimServiceConsumer;
import com.ebay.maui.component.ebayInUtils.EbayIndiaDBModule;
import com.ebay.maui.component.ebayInUtils.PowershipUtil;
import com.ebay.maui.component.myebay.Log;
import com.ebay.maui.component.page.cgi5.EbayGuaranteeDashboard;
import com.ebay.maui.component.page.cgi5.SellerTransactionDetailsPage;
import com.ebay.maui.component.page.signin.SignInPage;
import com.ebay.maui.controller.Assertion;
import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.EasyFilter;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.controller.TestPlan;
import com.ebay.maui.controller.TestRetryAnalyzer;
import com.ebay.maui.driver.web.WebUXDriver;
import com.ebay.maui.helper.ThreadHelper;
import com.ebay.maui.util.SpreadSheetUtil;
import com.ebay.maui.util.internal.entity.TestObject;
import com.ebay.maui.util.internal.type.Priorities;
import com.ebay.maui.util.internal.type.TestAspect;
import com.ebay.maui.util.internal.type.TestType;
import com.ebay.powership.ipinteg.notification.EbayAWBStatusEnum;
import com.ibm.icu.util.Calendar;
import com.ebay.maui.component.ebayIn.eG.common.SellerProtectionStatus;

public class RaiseClaimTestPlan extends TestPlan {

    // Added By Gaurav

    @DataProvider(name = "eGRaiseClaimsNewIssueType", parallel = true)
    public static Iterator<Object[]> getUserInfoNewIssue(Method m,
            ITestContext testContex) throws Exception {
```



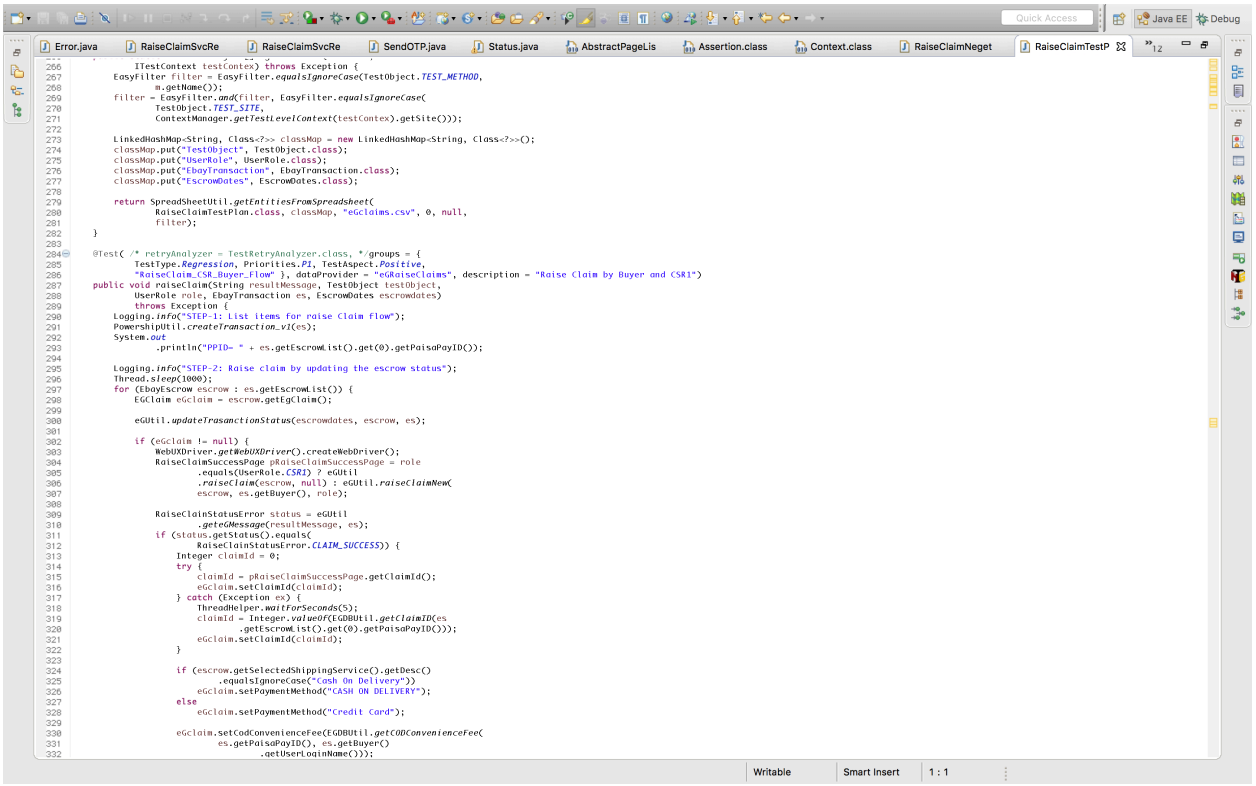
```
        ITestContext testContex) throws Exception {
    EasyFilter filter = EasyFilter.equalsIgnoreCase(TestObject.TEST_METHOD,
            m.getName());
    filter = EasyFilter.and(filter, EasyFilter.equalsIgnoreCase(
            TestObject.TEST_SITE,
            ContextManager.getTestLevelContext(testContex).getSite()));

    LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();

    classMap.put("TestObject", TestObject.class);
    classMap.put("UserRole", UserRole.class);
    classMap.put("EbayTransaction", EbayTransaction.class);
    classMap.put("EscrowDates", EscrowDates.class);

    return SpreadSheetUtil.getEntitiesFromSpreadsheet(
            RaiseClaimTestPlan.class, classMap, "eGclaimsNewIssue.csv", 0,
            null, filter);
}

@Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
        TestType.Regression, Priorities.P1, TestAspect.Positive,
        "RaiseClaim_CSR_Buyer_Flow_New_Issue" }, dataProvider = "eGRaiseClaimsNewIssueType", description = "Raise Claim by Buyer and CSR1 for New Issue type")
public void raiseClaimNewIssueType(String resultMessage,
        TestObject testObject, UserRole role, EbayTransaction es,
        EscrowDates escrowdates) throws Exception {
    Logging.info("STEP-1: List items for raise Claim flow");
    PowershipUtil.createTransaction_v1(es);
    System.out
            .println("PPID= " + es.getEscrowList().get(0).getPaisaPayID());

    Logging.info("STEP-2: Raise claim by updating the escrow status");
    Thread.sleep(1000);
    for (EbayEscrow escrow : es.getEscrowList()) {
        EGClaim eGclaim = escrow.getEgClaim();

        eGUtil.updateTrasanctionStatus(escrowdates, escrow, es);

        if (eGclaim != null) {
            WebUXDriver.getWebUXDriver().createWebDriver();
            RaiseClaimSuccessPage pRaiseClaimSuccessPage = role
                    .equals(UserRole.CSR1) ? eGUtil.raiseClaimNew(escrow,
                    null, role) : eGUtil.raiseClaimNew(escrow,
                    es.getBuyer(), role);

            RaiseClaimStatusError status = eGUtil
                    .geteGMessage(resultMessage, es);

            if (status.getStatus().equals(
                    RaiseClaimStatusError.CLAIM_SUCCESS)) {
                Integer claimId = 0;
                try {
                    claimId = pRaiseClaimSuccessPage.getClaimId();
                    eGclaim.setClaimId(claimId);
                } catch (Exception ex) {
                    ThreadHelper.waitForSeconds(5);
                    claimId = Integer.valueOf(EGDBUtil.getClaimID(es
                            .getEscrowList().get(0).getPaisaPayID()));
                    eGclaim.setClaimId(claimId);
                }

                if (escrow.getSelectedShippingService().getDesc()
                        .equalsIgnoreCase("Cash On Delivery"))
                    eGclaim.setPaymentMethod("CASH ON DELIVERY");
                else
                    eGclaim.setPaymentMethod("Credit Card");

                eGclaim.setCodConvenienceFee(EGDBUtil.getCODConvenienceFee(
```



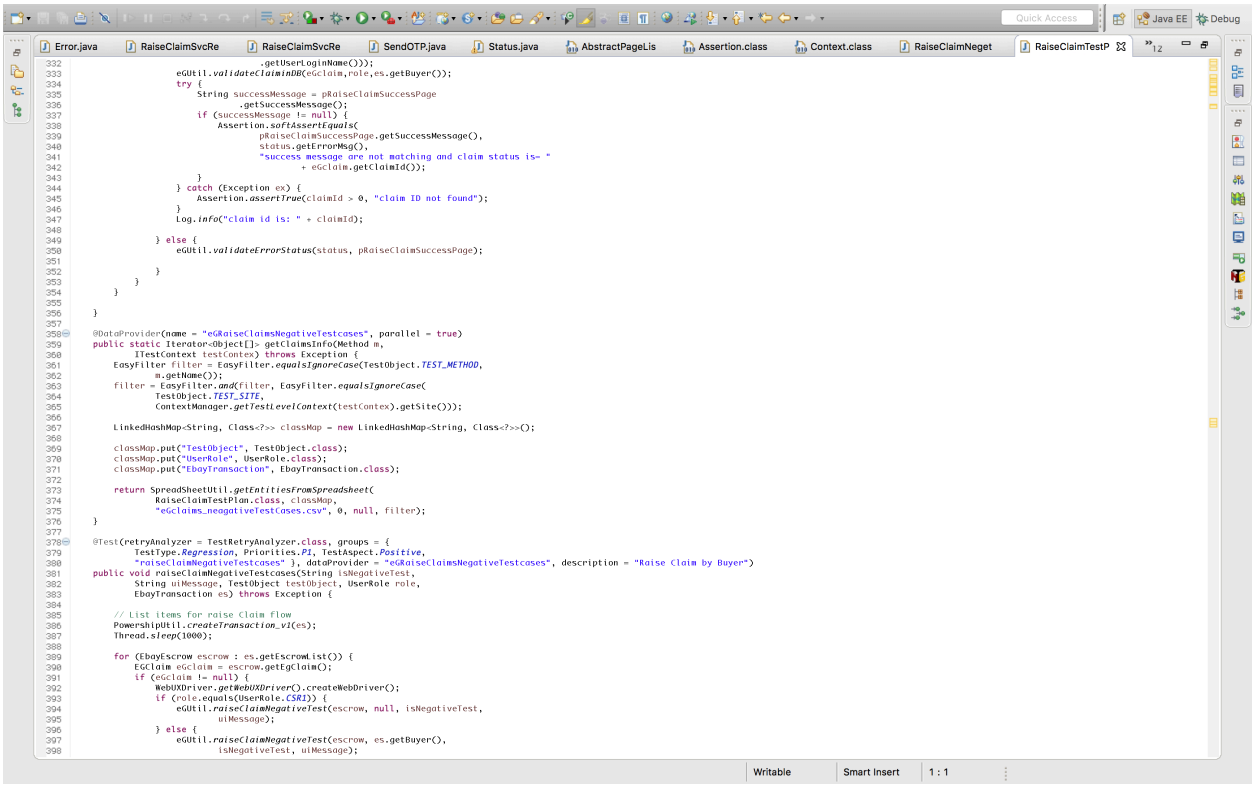
```
                eGclaim.setCodConvenienceFee(EGDBUtil.getCODConvenienceFee(
                        es.getPaisaPayID(), es.getBuyer()
                                .getUserLoginName()));
                eGUtil.validateClaiminDB(eGclaim, role, es.getBuyer());
                try {
                    String successMessage = pRaiseClaimSuccessPage
                            .getSuccessMessage();
                    if (successMessage != null) {
                        Assertion.softAssertEquals(
                                pRaiseClaimSuccessPage.getSuccessMessage(),
                                status.getErrorMsg(),
                                "success message are not matching and claim status is- "
                                        + eGclaim.getClaimId());
                    }
                } catch (Exception ex) {
                    Assertion.assertTrue(claimId > 0, "claim ID not found");
                }
                Log.info("claim id is: " + claimId);

            } else {
                eGUtil.validateErrorStatus(status, pRaiseClaimSuccessPage);

            }
        }
    }

}


@DataProvider(name = "eGRaiseClaimOTP", parallel = true)
public static Iterator<Object[]> getUserInfoClaimOTP(Method m,
        ITestContext testContex) throws Exception {
    EasyFilter filter = EasyFilter.equalsIgnoreCase(TestObject.TEST_METHOD,
            m.getName());
    filter = EasyFilter.and(filter, EasyFilter.equalsIgnoreCase(
            TestObject.TEST_SITE,
            ContextManager.getTestLevelContext(testContex).getSite()));

    LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();

    classMap.put("TestObject", TestObject.class);
    classMap.put("UserRole", UserRole.class);
    classMap.put("EbayTransaction", EbayTransaction.class);
    classMap.put("EscrowDates", EscrowDates.class);

    return SpreadSheetUtil.getEntitiesFromSpreadsheet(
            RaiseClaimTestPlan.class, classMap, "eGclaimOTP.csv", 0,
            null, filter);
}

@Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
        TestType.Regression, Priorities.P1, TestAspect.Positive,
        "RaiseClaim_CSR_Buyer_Flow_OTP" }, dataProvider = "eGRaiseClaimOTP", description = "Raise Claim by Buyer and CSR1 for New Issue type")
public void raiseClaimOTP(String resultMessage,
        TestObject testObject, UserRole role, EbayTransaction es,
        EscrowDates escrowdates) throws Exception {
    Logging.info("STEP-1: List items for raise Claim flow");
    PowershipUtil.createTransaction_v1(es);
    System.out
            .println("PPID= " + es.getEscrowList().get(0).getPaisaPayID());

    Logging.info("STEP-2: Raise claim by updating the escrow status");
    Thread.sleep(1000);
    for (EbayEscrow escrow : es.getEscrowList()) {
        EGClaim eGclaim = escrow.getEgClaim();

        eGUtil.updateTrasanctionStatus(escrowdates, escrow, es);
```



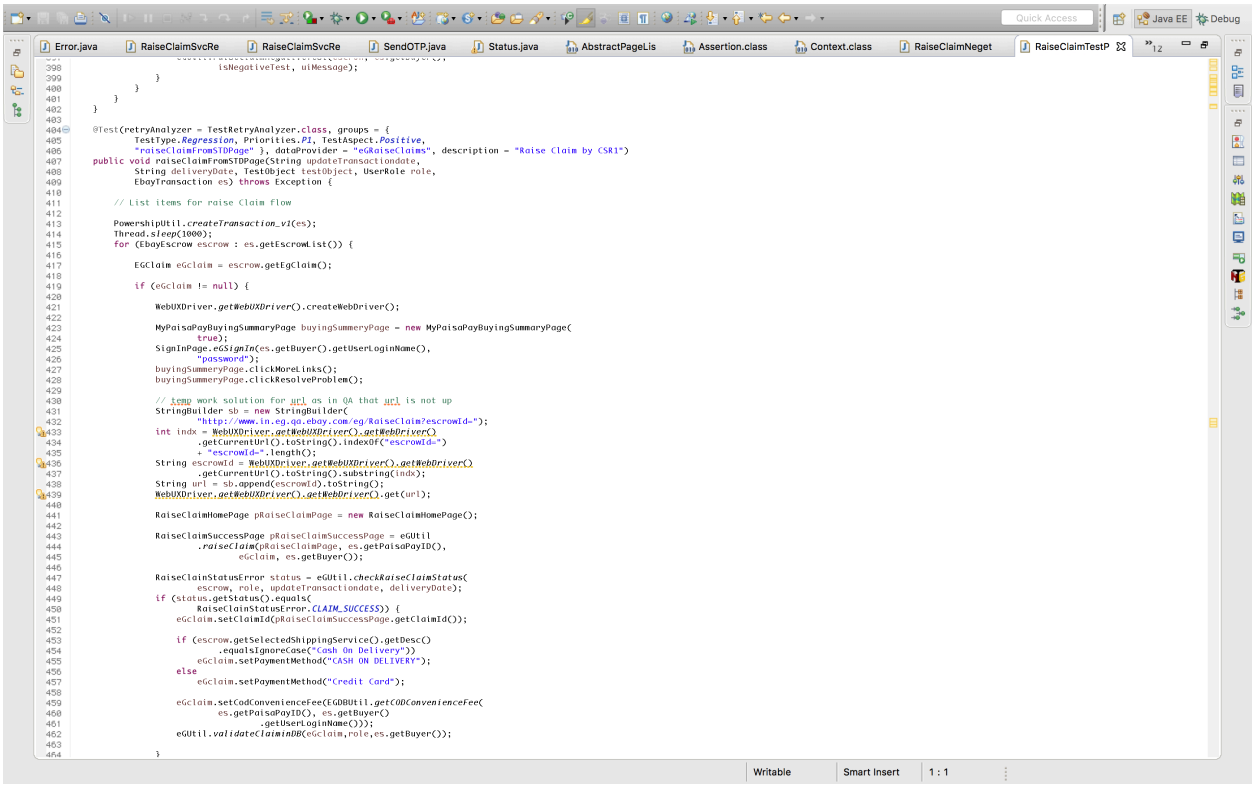
```
            if (eGclaim != null) {
                WebUXDriver.getWebUXDriver().createWebDriver();
                RaiseClaimSuccessPage pRaiseClaimSuccessPage = role
                        .equals(UserRole.CSR1) ? eGUtil.raiseClaimOTP(escrow,
                        es.getBuyer(), role) : eGUtil.raiseClaimOTP(escrow,
                        es.getBuyer(), role);

                RaiseClaimStatusError status = eGUtil
                        .geteGMessage(resultMessage, es);

                if (status.getStatus().equals(
                        RaiseClaimStatusError.CLAIM_SUCCESS)) {
                    Integer claimId = 0;
                    try {
                        claimId = pRaiseClaimSuccessPage.getClaimId();
                        eGclaim.setClaimId(claimId);
                    } catch (Exception ex) {
                        ThreadHelper.waitForSeconds(5);
                        claimId = Integer.valueOf(EGDBUtil.getClaimID(es
                                .getEscrowList().get(0).getPaisaPayID()));
                        eGclaim.setClaimId(claimId);
                    }

                    if (escrow.getSelectedShippingService().getDesc()
                            .equalsIgnoreCase("Cash On Delivery"))
                        eGclaim.setPaymentMethod("CASH ON DELIVERY");
                    else
                        eGclaim.setPaymentMethod("Credit Card");

                    eGclaim.setCodConvenienceFee(EGDBUtil.getCODConvenienceFee(
                            es.getPaisaPayID(), es.getBuyer()
                                    .getUserLoginName()));
                    eGUtil.validateClaiminDB(eGclaim, role, es.getBuyer());
                    try {
                        String successMessage = pRaiseClaimSuccessPage
                                .getSuccessMessage();
                        if (successMessage != null) {
                            Assertion.softAssertEquals(
                                    pRaiseClaimSuccessPage.getSuccessMessage(),
                                    status.getErrorMsg(),
                                    "success message are not matching and claim status is- "
                                            + eGclaim.getClaimId());

                        }
                    } catch (Exception ex) {
                        Assertion.assertTrue(claimId > 0, "claim ID not found");
                    }
                    Log.info("claim id is: " + claimId);

                } else {
                    eGUtil.validateErrorStatus(status, pRaiseClaimSuccessPage);

                }
            }
        }
    }




    @DataProvider(name = "eGRaiseClaims", parallel = true)
    public static Iterator<Object[]> getUserInfo(Method m,
            ITestContext testContex) throws Exception {
```



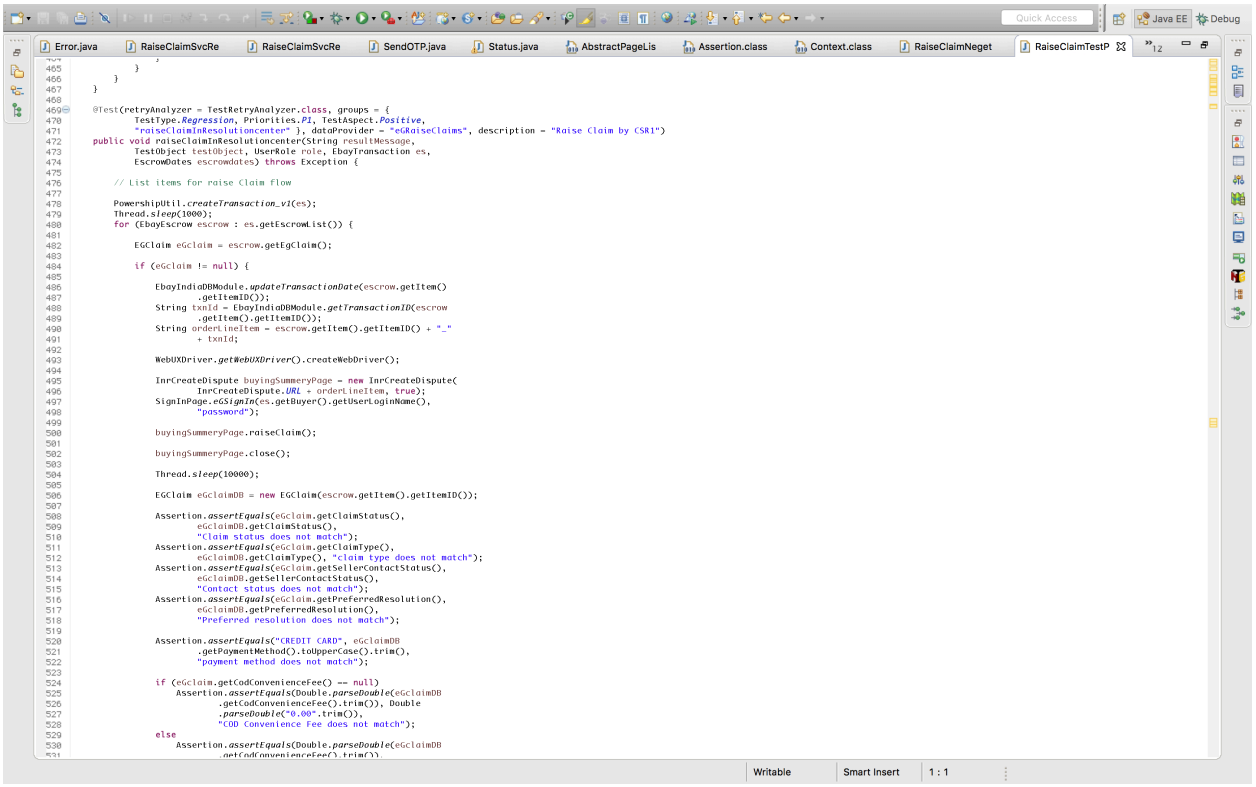
```
            ITestContext testContex) throws Exception {
        EasyFilter filter = EasyFilter.equalsIgnoreCase(TestObject.TEST_METHOD,
                m.getName());
        filter = EasyFilter.and(filter, EasyFilter.equalsIgnoreCase(
                TestObject.TEST_SITE,
                ContextManager.getTestLevelContext(testContex).getSite()));

        LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();
        classMap.put("TestObject", TestObject.class);
        classMap.put("UserRole", UserRole.class);
        classMap.put("EbayTransaction", EbayTransaction.class);
        classMap.put("EscrowDates", EscrowDates.class);

        return SpreadSheetUtil.getEntitiesFromSpreadsheet(
                RaiseClaimTestPlan.class, classMap, "eGclaims.csv", 0, null,
                filter);
    }

    @Test( /* retryAnalyzer = TestRetryAnalyzer.class, */groups = {
            TestType.Regression, Priorities.P1, TestAspect.Positive,
            "RaiseClaim_CSR_Buyer_Flow" }, dataProvider = "eGRaiseClaims", description = "Raise Claim by Buyer and CSR1")
    public void raiseClaim(String resultMessage, TestObject testObject,
            UserRole role, EbayTransaction es, EscrowDates escrowdates)
            throws Exception {
        Logging.info("STEP-1: List items for raise Claim flow");
        PowershipUtil.createTransaction_v1(es);
        System.out
                .println("PPID- " + es.getEscrowList().get(0).getPaisaPayID());

        Logging.info("STEP-2: Raise claim by updating the escrow status");
        Thread.sleep(1000);
        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();

            eGUtil.updateTrasanctionStatus(escrowdates, escrow, es);

            if (eGclaim != null) {
                WebUXDriver.getWebUXDriver().createWebDriver();
                RaiseClaimSuccessPage pRaiseClaimSuccessPage = role
                        .equals(UserRole.CSR1) ? eGUtil
                        .raiseClaim(escrow, null) : eGUtil.raiseClaimNew(
                        escrow, es.getBuyer(), role);

                RaiseClaimStatusError status = eGUtil
                        .geteGMessage(resultMessage, es);
                if (status.getStatus().equals(
                        RaiseClaimStatusError.CLAIM_SUCCESS)) {
                    Integer claimId = 0;
                    try {
                        claimId = pRaiseClaimSuccessPage.getClaimId();
                        eGclaim.setClaimId(claimId);
                    } catch (Exception ex) {
                        ThreadHelper.waitForSeconds(5);
                        claimId = Integer.valueOf(EGDBUtil.getClaimID(es
                                .getEscrowList().get(0).getPaisaPayID()));
                        eGclaim.setClaimId(claimId);
                    }

                    if (escrow.getSelectedShippingService().getDesc()
                            .equalsIgnoreCase("Cash On Delivery"))
                        eGclaim.setPaymentMethod("CASH ON DELIVERY");
                    else
                        eGclaim.setPaymentMethod("Credit Card");

                    eGclaim.setCodConvenienceFee(EGDBUtil.getCODConvenienceFee(
                            es.getPaisaPayID(), es.getBuyer()
                                    .getUserLoginName()));
```



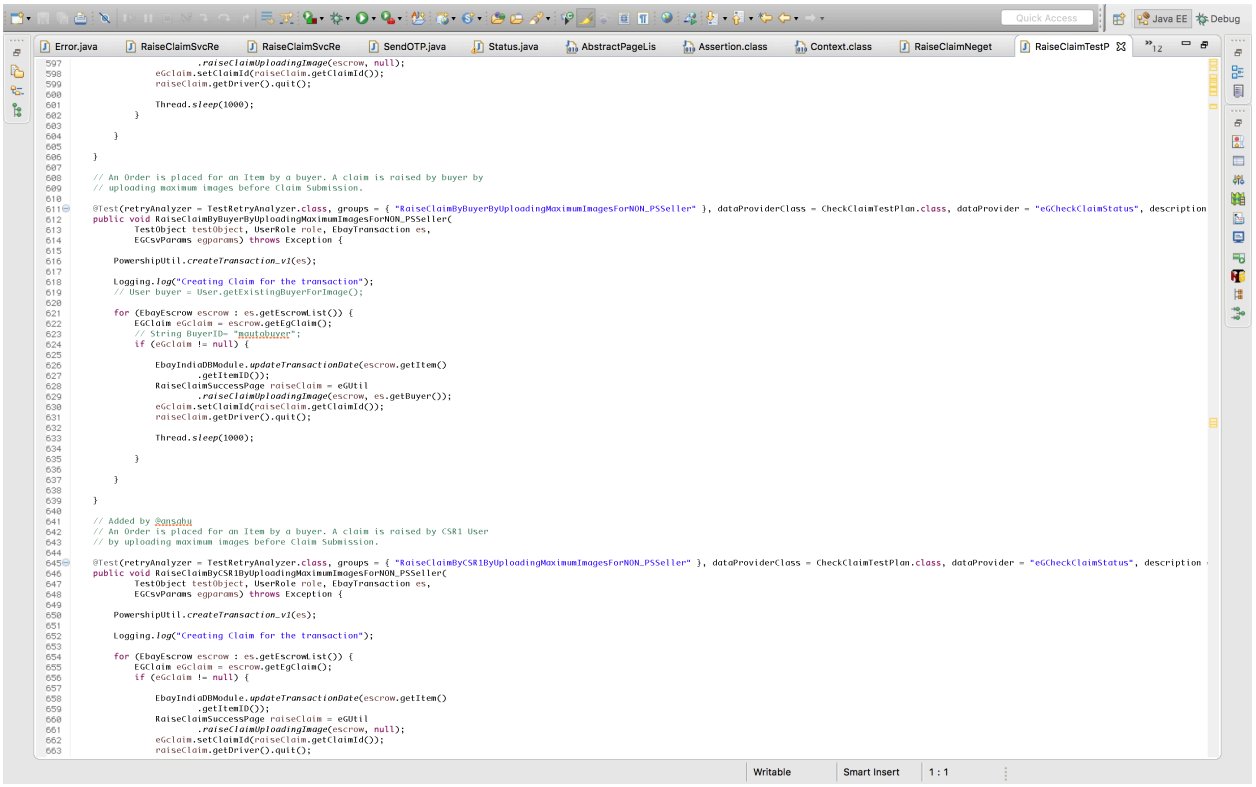
```
                                .getUserLoginName()));
                        eGUtil.validateClaiminDB(eGclaim,role,es.getBuyer());
                        try {
                            String successMessage = pRaiseClaimSuccessPage
                                    .getSuccessMessage();
                            if (successMessage != null) {
                                Assertion.softAssertEquals(
                                        pRaiseClaimSuccessPage.getSuccessMessage(),
                                        status.getErrorMsg(),
                                        "success message are not matching and claim status is~ "
                                                + eGclaim.getClaimId());
                            }
                        } catch (Exception ex) {
                            Assertion.assertTrue(claimId > 0, "claim ID not found");
                        }
                        Log.info("claim id is: " + claimId);

                    } else {
                        eGUtil.validateErrorStatus(status, pRaiseClaimSuccessPage);

                    }
                }
            }

        }

        @DataProvider(name = "eGRaiseClaimsNegativeTestcases", parallel = true)
        public static Iterator<Object[]> getClaimsInfo(Method m,
                ITestContext testContex) throws Exception {
            EasyFilter filter = EasyFilter.equalsIgnoreCase(TestObject.TEST_METHOD,
                    m.getName());
            filter = EasyFilter.and(filter, EasyFilter.equalsIgnoreCase(
                    TestObject.TEST_SITE,
                    ContextManager.getTestLevelContext(testContex).getSite()));

            LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();

            classMap.put("TestObject", TestObject.class);
            classMap.put("UserRole", UserRole.class);
            classMap.put("EbayTransaction", EbayTransaction.class);

            return SpreadSheetUtil.getEntitiesFromSpreadsheet(
                    RaiseClaimTestPlan.class, classMap,
                    "eGclaims_negativeTestCases.csv", 0, null, filter);
        }

        @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
                TestType.Regression, Priorities.P1, TestAspect.Positive,
                "raiseClaimNegativeTestcases" }, dataProvider = "eGRaiseClaimsNegativeTestcases", description = "Raise Claim by Buyer")
        public void raiseClaimNegativeTestcases(String isNegativeTest,
                String uiMessage, TestObject testObject, UserRole role,
                EbayTransaction es) throws Exception {

            // List items for raise Claim flow
            PowershipUtil.createTransaction_v1(es);
            Thread.sleep(1000);

            for (EbayEscrow escrow : es.getEscrowList()) {
                EGClaim eGclaim = escrow.getEgClaim();
                if (eGclaim != null) {
                    WebUXDriver.getWebUXDriver().createWebDriver();
                    if (role.equals(UserRole.CSR1)) {
                        eGUtil.raiseClaimNegativeTest(escrow, null, isNegativeTest,
                                uiMessage);
                    } else {
                        eGUtil.raiseClaimNegativeTest(escrow, es.getBuyer(),
                                isNegativeTest, uiMessage);
```



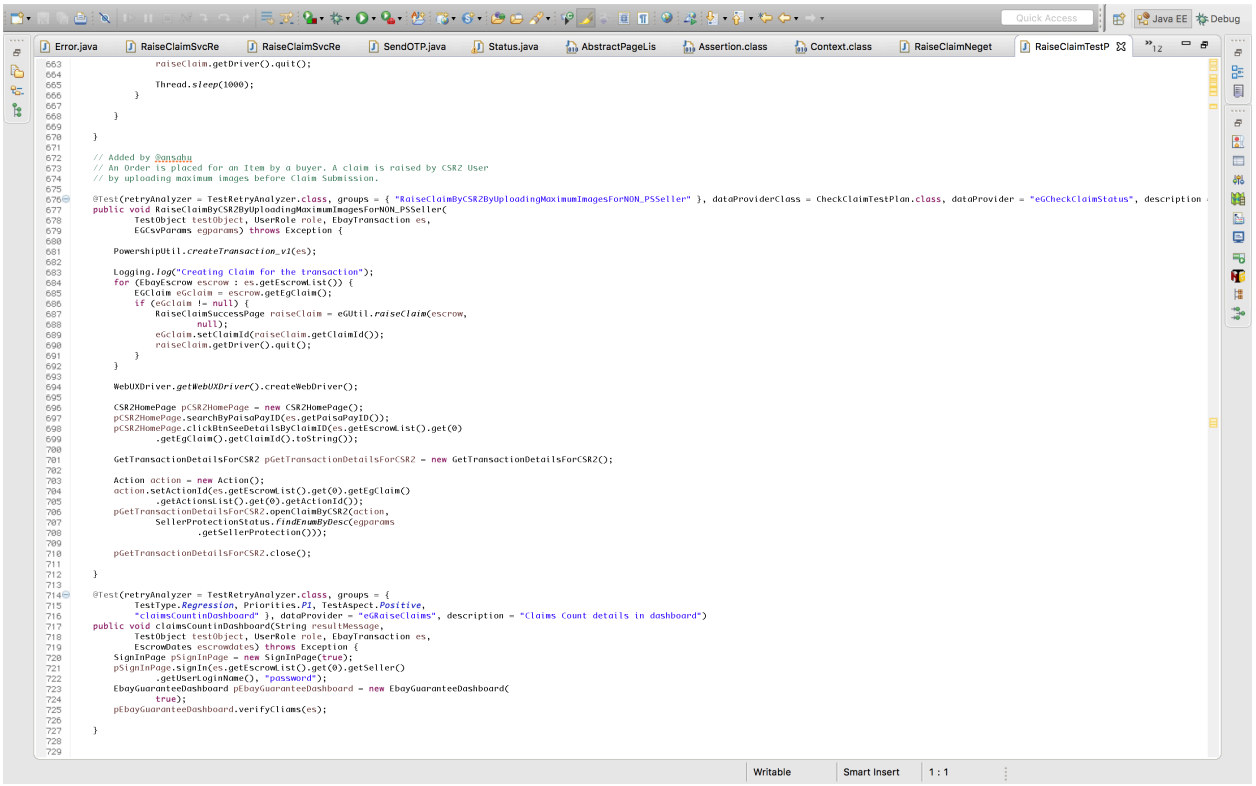
```
                                isNegativeTest, uiMessage);
                    }
                }
            }
        }

        @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
                TestType.Regression, Priorities.P1, TestAspect.Positive,
                "raiseClaimFromSTDPage" }, dataProvider = "eGRaiseClaims", description = "Raise Claim by CSR1")
        public void raiseClaimFromSTDPage(String updateTransactiondate,
                String deliveryDate, TestObject testObject, UserRole role,
                EbayTransaction es) throws Exception {

            // List items for raise Claim flow

            PowershipUtil.createTransaction_v1(es);
            Thread.sleep(1000);
            for (EbayEscrow escrow : es.getEscrowList()) {

                EGClaim eGclaim = escrow.getEgClaim();

                if (eGclaim != null) {

                    WebUXDriver.getWebUXDriver().createWebDriver();

                    MyPaisaPayBuyingSummaryPage buyingSummeryPage = new MyPaisaPayBuyingSummaryPage(
                            true);
                    SignInPage.eGSignIn(es.getBuyer().getUserLoginName(),
                            "password");
                    buyingSummeryPage.clickMoreLinks();
                    buyingSummeryPage.clickResolveProblem();

                    // temp work solution for url as in QA that url is not up
                    StringBuilder sb = new StringBuilder(
                            "http://www.in.eg.qa.ebay.com/eg/RaiseClaim?escrowId=");
                    int indx = WebUXDriver.getWebUXDriver().getWebDriver()
                            .getCurrentUrl().toString().indexOf("escrowId=")
                            + "escrowId=".length();
                    String escrowId = WebUXDriver.getWebUXDriver().getWebDriver()
                            .getCurrentUrl().toString().substring(indx);
                    String url = sb.append(escrowId).toString();
                    WebUXDriver.getWebUXDriver().getWebDriver().get(url);

                    RaiseClaimHomePage pRaiseClaimPage = new RaiseClaimHomePage();

                    RaiseClaimSuccessPage pRaiseClaimSuccessPage = eGUtil
                            .raiseClaim(pRaiseClaimPage, es.getPaisaPayID(),
                                    eGclaim, es.getBuyer());

                    RaiseClaimStatusError status = eGUtil.checkRaiseClaimStatus(
                            escrow, role, updateTransactiondate, deliveryDate);
                    if (status.getStatus().equals(
                            RaiseClaimStatusError.CLAIM_SUCCESS)) {
                        eGclaim.setClaimId(pRaiseClaimSuccessPage.getClaimId());

                        if (escrow.getSelectedShippingService().getDesc()
                                .equalsIgnoreCase("Cash On Delivery"))
                            eGclaim.setPaymentMethod("CASH ON DELIVERY");
                        else
                            eGclaim.setPaymentMethod("Credit Card");

                        eGclaim.setCodConvenienceFee(EGDBUtil.getCODConvenienceFee(
                                es.getPaisaPayID(), es.getBuyer()
                                        .getUserLoginName()));
                        eGUtil.validateClaiminDB(eGclaim,role,es.getBuyer());

                    }
```



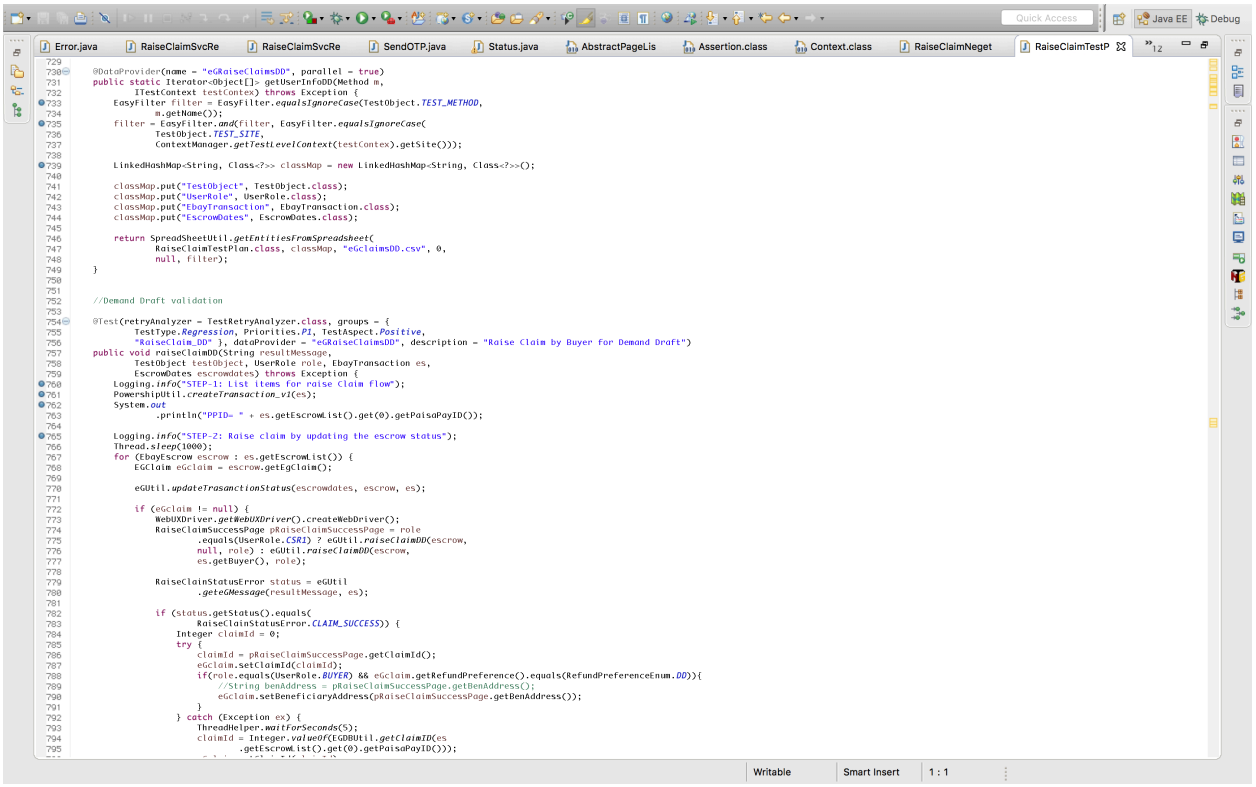
```
            }
        }
    }

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
            TestType.Regression, Priorities.P1, TestAspect.Positive,
            "raiseClaiminResolutioncenter" }, dataProvider = "eGRaiseClaims", description = "Raise Claim by CSR1")
    public void raiseClaiminResolutioncenter(String resultMessage,
            TestObject testObject, UserRole role, EbayTransaction es,
            EscrowDates escrowdates) throws Exception {

        // List items for raise Claim flow

        PowershipUtil.createTransaction_v1(es);
        Thread.sleep(1000);
        for (EbayEscrow escrow : es.getEscrowList()) {

            EGClaim eGclaim = escrow.getEgClaim();

            if (eGclaim != null) {

                EbayIndiaDBModule.updateTransactionDate(escrow.getItem()
                        .getItemID());
                String txnId = EbayIndiaDBModule.getTransactionID(escrow
                        .getItem().getItemID());
                String orderLineItem = escrow.getItem().getItemID() + "-"
                        + txnId;

                WebUXDriver.getWebUXDriver().createWebDriver();

                InrCreateDispute buyingSummeryPage = new InrCreateDispute(
                        InrCreateDispute.URL + orderLineItem, true);
                SignInPage.eGSignIn(es.getBuyer().getUserLoginName(),
                        "password");

                buyingSummeryPage.raiseClaim();

                buyingSummeryPage.close();

                Thread.sleep(10000);

                EGClaim eGclaimDB = new EGClaim(escrow.getItem().getItemID());

                Assertion.assertEquals(eGclaim.getClaimStatus(),
                        eGclaimDB.getClaimStatus(),
                        "Claim status does not match");
                Assertion.assertEquals(eGclaim.getClaimType(),
                        eGclaimDB.getClaimType(), "claim type does not match");
                Assertion.assertEquals(eGclaim.getSellerContactStatus(),
                        eGclaimDB.getSellerContactStatus(),
                        "Contact status does not match");
                Assertion.assertEquals(eGclaim.getPreferredResolution(),
                        eGclaimDB.getPreferredResolution(),
                        "Preferred resolution does not match");

                Assertion.assertEquals("CREDIT CARD", eGclaimDB
                        .getPaymentMethod().toUpperCase().trim(),
                        "payment method does not match");

                if (eGclaim.getCodConvenienceFee() == null)
                    Assertion.assertEquals(Double.parseDouble(eGclaimDB
                            .getCodConvenienceFee().trim()), Double
                            .parseDouble("0.00".trim()),
                            "COD Convenience Fee does not match");
                else
                    Assertion.assertEquals(Double.parseDouble(eGclaimDB
                            .getCodConvenienceFee().trim()),
                            Double.parseDouble(eGclaim.getCodConvenienceFee()
                                    .trim()),
                            "COD Convenience Fee does not match");

            }
        }

    }

    // Added by Soronya
    // An Order is placed for an Item by a buyer. A claim is raised by buyer by
    // uploading maximum images before Claim Submission.

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
            TestType.Regression, Priorities.P2, TestAspect.Positive,
            "RaiseClaimByBuyerByUploadingMaximumImages" }, dataProviderClass = CheckClaimTestPlan.class, dataProvider = "eGCheckClaimStatus", description = "RaiseClaimByBuyerByUploadingImages,RaiseClaimByBuyerByUploadi
    public void RaiseClaimByBuyerByUploadingMaximumImages(
            TestObject testObject, UserRole role, EbayTransaction es,
            EGCsvParams egparams) throws Exception {

        PowershipUtil.createTransaction_v1(es);

        Logging.log("Creating Claim for the transaction");
        // User buyer = User.getExistingBuyerForImage();

        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();
            // String BuyerID= "mgutobuyer";
            if (eGclaim != null) {

                EbayIndiaDBModule.updateTransactionDate(escrow.getItem()
                        .getItemID());
                RaiseClaimSuccessPage raiseClaim = eGUtil
                        .raiseClaimUploadingImage(escrow, es.getBuyer());
                eGclaim.setClaimId(raiseClaim.getClaimId());
                raiseClaim.getDriver().quit();

                Thread.sleep(1000);

            }

        }

    }

    // Added by Soronya
    // An Order is placed for an Item by buyer. A claim is raised by CSR1 User
    // by uploading maximum images before Claim Submission.

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = { "RaiseClaimByCSR1ByUploadingMaximumImages" }, dataProviderClass = CheckClaimTestPlan.class, dataProvider = "eGCheckClaimStatus", description = "RaiseClaimBy
    public void RaiseClaimByCSR1ByUploadingMaximumImages(TestObject testObject,
            UserRole role, EbayTransaction es, EGCsvParams egparams)
            throws Exception {

        PowershipUtil.createTransaction_v1(es);

        Logging.log("Creating Claim for the transaction");

        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();
            if (eGclaim != null) {

                EbayIndiaDBModule.updateTransactionDate(escrow.getItem()
                        .getItemID());
                RaiseClaimSuccessPage raiseClaim = eGUtil
                        .raiseClaimUploadingImage(escrow, null);
```



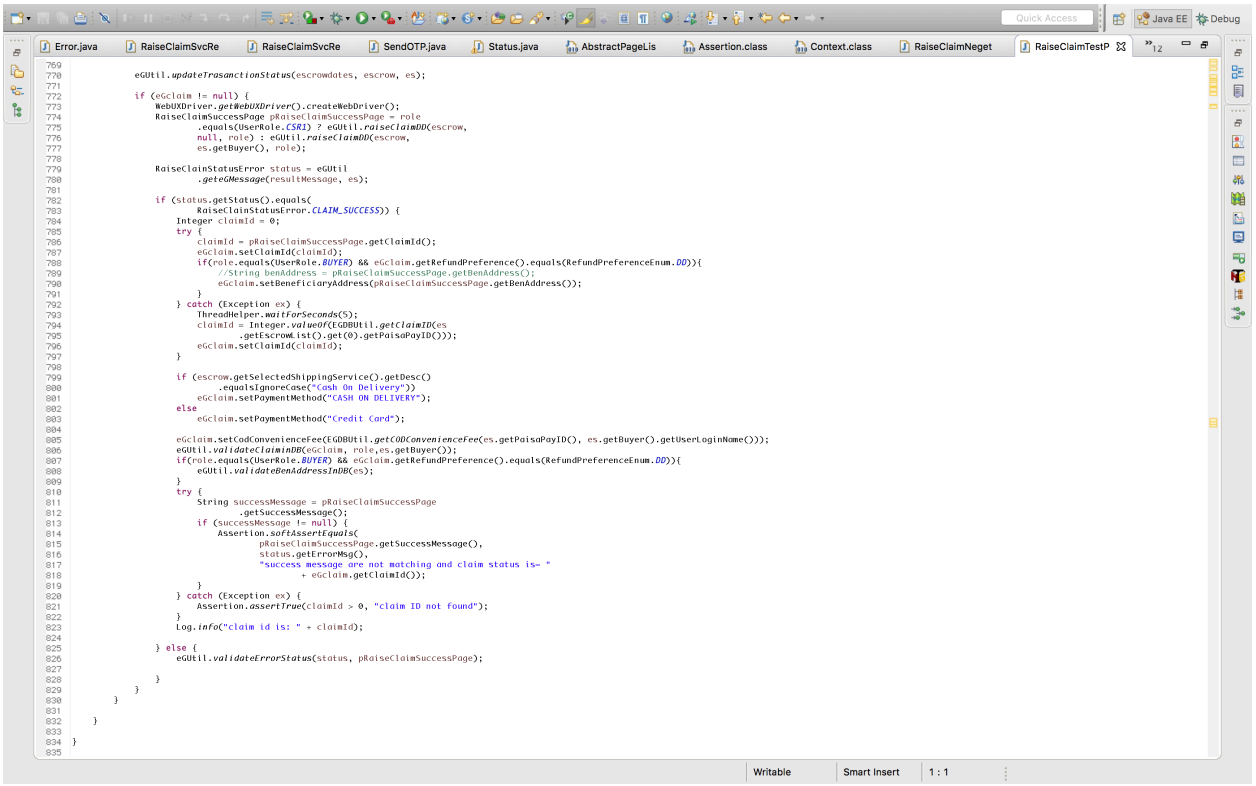
```
                        .raiseClaimUploadingImage(escrow, null);
                eGclaim.setClaimId(raiseClaim.getClaimId());
                raiseClaim.getDriver().quit();

                Thread.sleep(1000);
            }

        }
    }

    // An Order is placed for an Item by a buyer. A claim is raised by buyer by
    // uploading maximum images before Claim Submission.

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = { "RaiseClaimByBuyerByUploadingMaximumImagesForNON_PSSeller" }, dataProviderClass = CheckClaimTestPlan.class, dataProvider = "eGCheckClaimStatus", description
    public void RaiseClaimByBuyerByUploadingMaximumImagesForNON_PSSeller(
            TestObject testObject, UserRole role, EbayTransaction es,
            EGCsvParams egparams) throws Exception {

        PowershipUtil.createTransaction_v1(es);

        Logging.log("Creating Claim for the transaction");
        // User buyer = User.getExistingBuyerForImage();

        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();
            // String BuyerID= "moutobuyer";
            if (eGclaim != null) {

                EbayIndiaDBModule.updateTransactionDate(escrow.getItem()
                        .getItemID());
                RaiseClaimSuccessPage raiseClaim = eGUtil
                        .raiseClaimUploadingImage(escrow, es.getBuyer());
                eGclaim.setClaimId(raiseClaim.getClaimId());
                raiseClaim.getDriver().quit();

                Thread.sleep(1000);

            }

        }

    }

    // Added by @ansabu
    // An Order is placed for an Item by a buyer. A claim is raised by CSR1 User
    // by uploading maximum images before Claim Submission.

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = { "RaiseClaimByCSR1ByUploadingMaximumImagesForNON_PSSeller" }, dataProviderClass = CheckClaimTestPlan.class, dataProvider = "eGCheckClaimStatus", description
    public void RaiseClaimByCSR1ByUploadingMaximumImagesForNON_PSSeller(
            TestObject testObject, UserRole role, EbayTransaction es,
            EGCsvParams egparams) throws Exception {

        PowershipUtil.createTransaction_v1(es);

        Logging.log("Creating Claim for the transaction");

        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();
            if (eGclaim != null) {

                EbayIndiaDBModule.updateTransactionDate(escrow.getItem()
                        .getItemID());
                RaiseClaimSuccessPage raiseClaim = eGUtil
                        .raiseClaimUploadingImage(escrow, null);
                eGclaim.setClaimId(raiseClaim.getClaimId());
                raiseClaim.getDriver().quit();
```



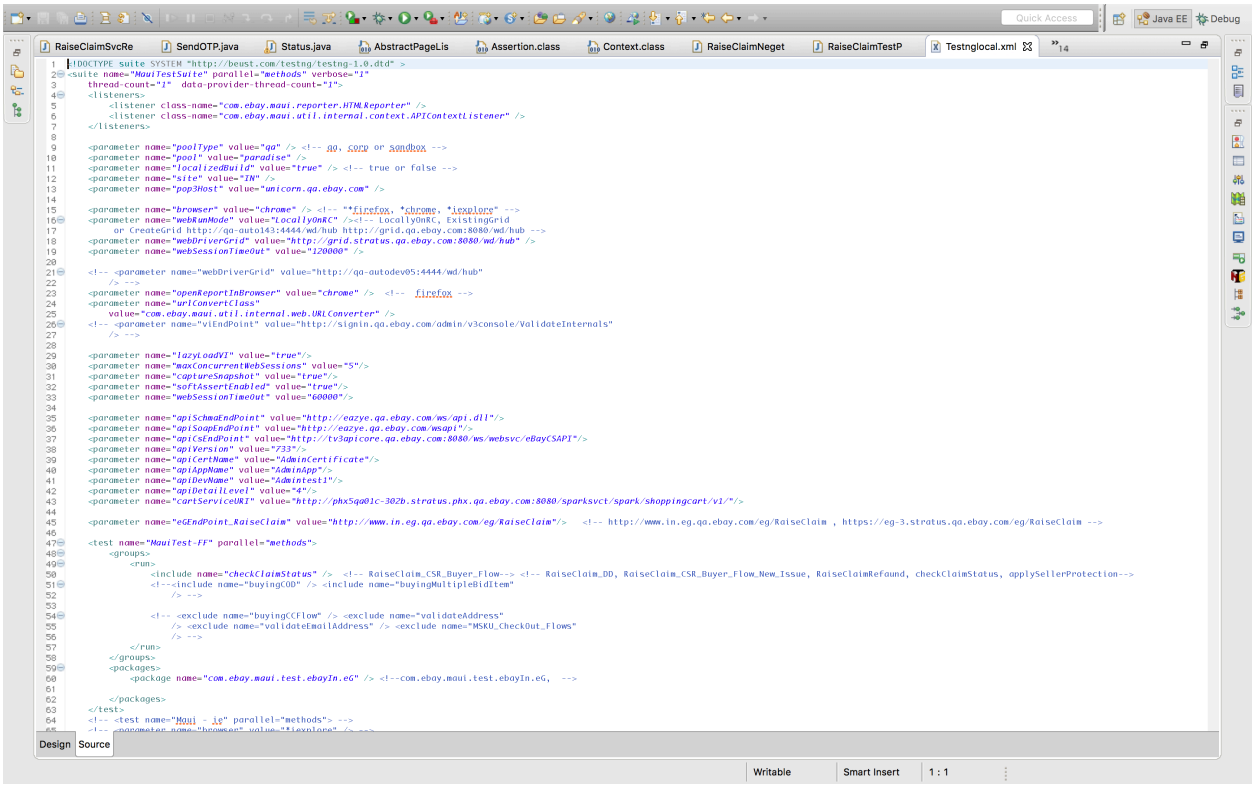
```
                raiseClaim.getDriver().quit();

                Thread.sleep(1000);
            }

        }
    }

    // Added by @ansabu
    // An Order is placed for an Item by a buyer. A claim is raised by CSR2 User
    // by uploading maximum images before Claim Submission.

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = { "RaiseClaimByCSR2ByUploadingMaximumImagesForNON_PSSeller" }, dataProviderClass = CheckClaimTestPlan.class, dataProvider = "eGCheckClaimStatus", description
    public void RaiseClaimByCSR2ByUploadingMaximumImagesForNON_PSSeller(
            TestObject testObject, UserRole role, EbayTransaction es,
            EGCsvParams egparams) throws Exception {

        PowershipUtil.createTransaction_v1(es);

        Logging.log("Creating Claim for the transaction");
        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();
            if (eGclaim != null) {
                RaiseClaimSuccessPage raiseClaim = eGUtil.raiseClaim(escrow,
                        null);
                eGclaim.setClaimId(raiseClaim.getClaimId());
                raiseClaim.getDriver().quit();
            }
        }

        WebUXDriver.getWebUXDriver().createWebDriver();

        CSR2HomePage pCSR2HomePage = new CSR2HomePage();
        pCSR2HomePage.searchByPaisaPayID(es.getPaisaPayID());
        pCSR2HomePage.clickBtnSeeDetailsByClaimID(es.getEscrowList().get(0)
                .getEgClaim().getClaimId().toString());

        GetTransactionDetailsForCSR2 pGetTransactionDetailsForCSR2 = new GetTransactionDetailsForCSR2();

        Action action = new Action();
        action.setActionId(es.getEscrowList().get(0).getEgClaim()
                .getActionsList().get(0).getActionId());
        pGetTransactionDetailsForCSR2.openClaimByCSR2(action,
                SellerProtectionStatus.findEnumByDesc(egparams
                        .getSellerProtection()));

        pGetTransactionDetailsForCSR2.close();

    }

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
            TestType.Regression, Priorities.P1, TestAspect.Positive,
            "claimsCountInDashboard" }, dataProvider = "eGRaiseClaims", description = "Claims Count details in dashboard")
    public void claimsCountInDashboard(String resultMessage,
            TestObject testObject, UserRole role, EbayTransaction es,
            EscrowDates escrowdates) throws Exception {
        SignInPage pSignInPage = new SignInPage(true);
        pSignInPage.signIn(es.getEscrowList().get(0).getSeller()
                .getUserLoginName(), "password");
        EbayGuaranteeDashboard pEbayGuaranteeDashboard = new EbayGuaranteeDashboard(
                true);
        pEbayGuaranteeDashboard.verifyClaims(es);

    }
```

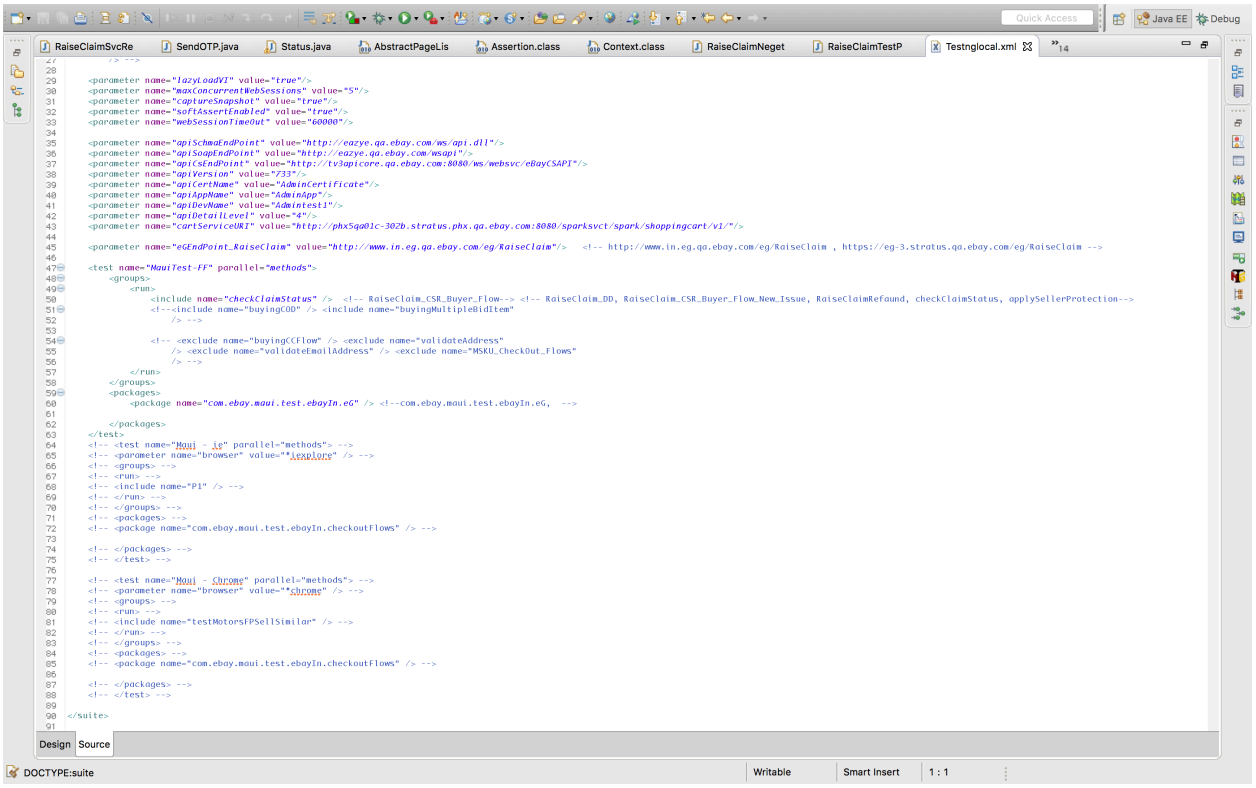


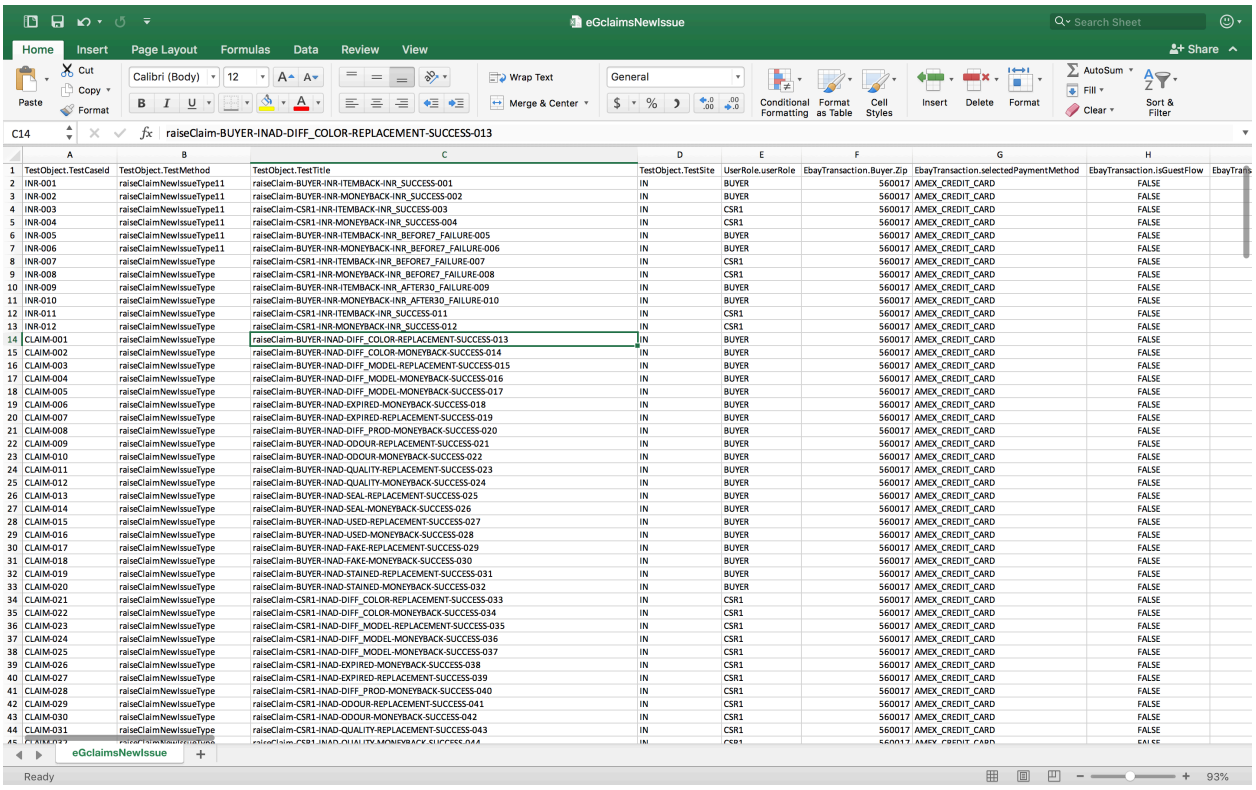
```
729
730     @DataProvider(name = "eGRaiseClaimsDD", parallel = true)
731     public static Iterator<Object[]> getUserInfoDD(Method m,
732             ITestContext testContex) throws Exception {
733         EasyFilter filter = EasyFilter.equalsIgnoreCase(TestObject.TEST_METHOD,
734                 m.getName());
735         filter = EasyFilter.and(filter, EasyFilter.equalsIgnoreCase(
736                 TestObject.TEST_SITE,
737                 ContextManager.getTestLevelContext(testContex).getSite()));
738
739         LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();
740
741         classMap.put("TestObject", TestObject.class);
742         classMap.put("UserRole", UserRole.class);
743         classMap.put("EbayTransaction", EbayTransaction.class);
744         classMap.put("EscrowDates", EscrowDates.class);
745
746         return SpreadSheetUtil.getEntitiesFromSpreadsheet(
747                 RaiseClaimTestPlan.class, classMap, "eGclaimsDD.csv", 0,
748                 null, filter);
749     }
750
751
752     //Demand Draft validation
753
754     @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
755             TestType.Regression, Priorities.P1, TestAspect.Positive,
756              "RaiseClaim_DD" }, dataProvider = "eGRaiseClaimsDD", description = "Raise Claim by Buyer for Demand Draft")
757     public void raiseClaimDD(String resultMessage,
758             TestObject testObject, UserRole role, EbayTransaction es,
759             EscrowDates escrowdates) throws Exception {
760         Logging.info("STEP-1: List items for raise Claim flow");
761         PowershipUtil.createTransaction_v1(es);
762         System.out
763                 .println("PPID= " + es.getEscrowList().get(0).getPaisaPayID());
764
765         Logging.info("STEP-2: Raise claim by updating the escrow status");
766         Thread.sleep(1000);
767         for (EbayEscrow escrow : es.getEscrowList()) {
768             EGClaim eGclaim = escrow.getEgClaim();
769
770             eGUtil.updateTrasanctionStatus(escrowdates, escrow, es);
771
772             if (eGclaim != null) {
773                 WebUIDriver.getWebUIDriver().createWebDriver();
774                 RaiseClaimSuccessPage pRaiseClaimSuccessPage = role
775                         .equals(UserRole.CSR1) ? eGUtil.raiseClaimDD(escrow,
776                         null, role) : eGUtil.raiseClaimDD(escrow,
777                         es.getBuyer(), role);
778
779                 RaiseClainStatusError status = eGUtil
780                         .geteGMessage(resultMessage, es);
781
782                 if (status.getStatus().equals(
783                         RaiseClainStatusError.CLAIM_SUCCESS)) {
784                     Integer claimId = 0;
785                     try {
786                         claimId = pRaiseClaimSuccessPage.getClaimId();
787                         eGclaim.setClaimId(claimId);
788                         if(role.equals(UserRole.BUYER) && eGclaim.getRefundPreference().equals(RefundPreferenceEnum.DD)){
789                             //String benAddress = pRaiseClaimSuccessPage.getBenAddress();
790                             eGclaim.setBeneficiaryAddress(pRaiseClaimSuccessPage.getBenAddress());
791                         }
792                     } catch (Exception ex) {
793                         ThreadHelper.waitForSeconds(5);
794                         claimId = Integer.valueOf(EGDBUtil.getClaimID(es
795                                 .getEscrowList().get(0).getPaisaPayID()));
```



```
769
770             eGUtil.updateTrasanctionStatus(escrowdates, escrow, es);
771
772             if (eGclaim != null) {
773                 WebUIDriver.getWebUIDriver().createWebDriver();
774                 RaiseClaimSuccessPage pRaiseClaimSuccessPage = role
775                         .equals(UserRole.CSR1) ? eGUtil.raiseClaimDD(escrow,
776                         null, role) : eGUtil.raiseClaimDD(escrow,
777                         es.getBuyer(), role);
778
779                 RaiseClainStatusError status = eGUtil
780                         .geteGMessage(resultMessage, es);
781
782                 if (status.getStatus().equals(
783                         RaiseClainStatusError.CLAIM_SUCCESS)) {
784                     Integer claimId = 0;
785                     try {
786                         claimId = pRaiseClaimSuccessPage.getClaimId();
787                         eGclaim.setClaimId(claimId);
788                         if(role.equals(UserRole.BUYER) && eGclaim.getRefundPreference().equals(RefundPreferenceEnum.DD)){
789                             //String benAddress = pRaiseClaimSuccessPage.getBenAddress();
790                             eGclaim.setBeneficiaryAddress(pRaiseClaimSuccessPage.getBenAddress());
791                         }
792                     } catch (Exception ex) {
793                         ThreadHelper.waitForSeconds(5);
794                         claimId = Integer.valueOf(EGDBUtil.getClaimID(es
795                                 .getEscrowList().get(0).getPaisaPayID()));
796                         eGclaim.setClaimId(claimId);
797                     }
798
799                     if (escrow.getSelectedShippingService().getDesc()
800                             .equalsIgnoreCase("Cash On Delivery"))
801                         eGclaim.setPaymentMethod("CASH ON DELIVERY");
802                     else
803                         eGclaim.setPaymentMethod("Credit Card");
804
805                     eGclaim.setCodConvenienceFee(EGDBUtil.getCODConvenienceFee(es.getPaisaPayID(), es.getBuyer().getUserLoginName()));
806                     eGUtil.validateClaiminDB(eGclaim, role,es.getBuyer());
807                     if(role.equals(UserRole.BUYER) && eGclaim.getRefundPreference().equals(RefundPreferenceEnum.DD)){
808                         eGUtil.validateBenAddressInDB(es);
809                     }
810                     try {
811                         String successMessage = pRaiseClaimSuccessPage
812                                 .getSuccessMessage();
813                         if (successMessage != null) {
814                             Assertion.softAssertEquals(
815                                     pRaiseClaimSuccessPage.getSuccessMessage(),
816                                     status.getErrorMsg(),
817                                     "success message are not matching and claim status is- "
818                                             + eGclaim.getClaimId());
819                         }
820                     } catch (Exception ex) {
821                         Assertion.assertTrue(claimId > 0, "claim ID not found");
822                     }
823                     Log.info("claim id is: " + claimId);
824
825                 } else {
826                     eGUtil.validateErrorStatus(status, pRaiseClaimSuccessPage);
827
828                 }
829             }
830         }
831
832     }
833
834 }
835
```


RaiseClaimSvcRe
SendOTP.java
Status.java
AbstractPageLis
Assertion.class
Context.class
RaiseClaimNeget
RaiseClaimTestP
Testnglocal.xml
Quick Access
Java EE
Debug
Design
Source
Writable
Smart Insert
1 : 1


RaiseClaimSvcRe
SendOTP.java
Status.java
AbstractPageLis
Assertion.class
Context.class
RaiseClaimNeget
RaiseClaimTestP
Testnglocal.xml
Quick Access
Java EE
Debug
Design
Source
DOCTYPE:suite
Writable
Smart Insert
1 : 1



eGclaimsNewIssue

Search Sheet

Home | Insert | Page Layout | Formulas | Data | Review | View | Share

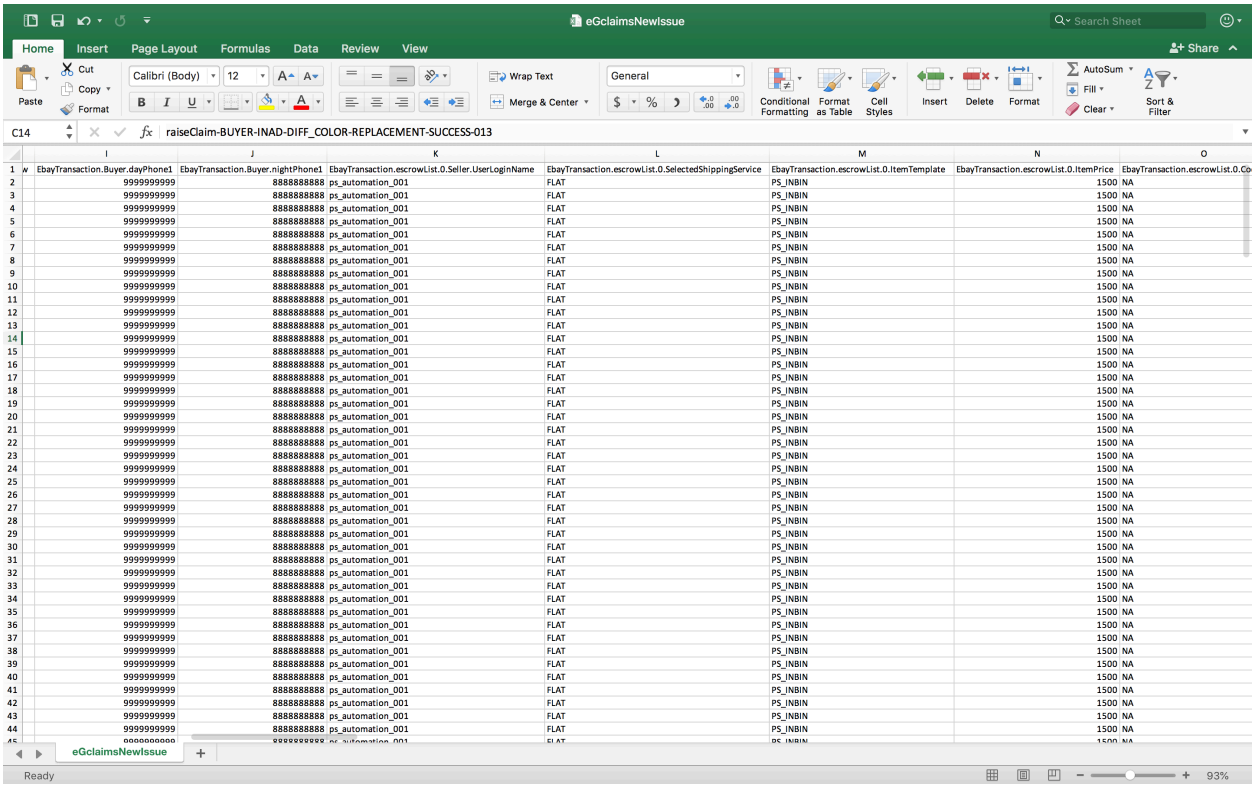
Paste | Cut | Copy | Format | Calibri (Body) | 12 | Wrap Text | Merge & Center | General | Conditional Formatting | Format as Table | Cell Styles | Insert | Delete | Format | AutoSum | Fill | Clear | Sort & Filter

C14 | raiseClaim-BUYER-INAD-DIFF_COLOR-REPLACEMENT-SUCCESS-013

	A	B	C	D	E	F	G	H
1	TestObject.TestCaseId	TestObject.TestMethod	TestObject.TestTitle	TestObject.TestSite	UserRole.userRole	EbayTransaction.Buyer.Zip	EbayTransaction.selectedPaymentMethod	EbayTransaction.isGuestFlow
2	INR-001	raiseClaimNewIssueType11	raiseClaim-BUYER-INR-ITEMBACK-INR_SUCCESS-001	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
3	INR-002	raiseClaimNewIssueType11	raiseClaim-BUYER-INR-MONEYBACK-INR_SUCCESS-002	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
4	INR-003	raiseClaimNewIssueType11	raiseClaim-CSR1-INR-ITEMBACK-INR_SUCCESS-003	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
5	INR-004	raiseClaimNewIssueType11	raiseClaim-CSR1-INR-MONEYBACK-INR_SUCCESS-004	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
6	INR-005	raiseClaimNewIssueType11	raiseClaim-BUYER-INR-ITEMBACK-INR_BEFORE7_FAILURE-005	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
7	INR-006	raiseClaimNewIssueType11	raiseClaim-BUYER-INR-MONEYBACK-INR_BEFORE7_FAILURE-006	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
8	INR-007	raiseClaimNewIssueType	raiseClaim-CSR1-INR-ITEMBACK-INR_BEFORE7_FAILURE-007	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
9	INR-008	raiseClaimNewIssueType	raiseClaim-CSR1-INR-MONEYBACK-INR_BEFORE7_FAILURE-008	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
10	INR-009	raiseClaimNewIssueType	raiseClaim-BUYER-INR-ITEMBACK-INR_AFTER30_FAILURE-009	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
11	INR-010	raiseClaimNewIssueType	raiseClaim-BUYER-INR-MONEYBACK-INR_AFTER30_FAILURE-010	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
12	INR-011	raiseClaimNewIssueType	raiseClaim-CSR1-INR-ITEMBACK-INR_SUCCESS-011	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
13	INR-012	raiseClaimNewIssueType	raiseClaim-CSR1-INR-MONEYBACK-INR_SUCCESS-012	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
14	CLAIM-001	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-DIFF_COLOR-REPLACEMENT-SUCCESS-013	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
15	CLAIM-002	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-DIFF_COLOR-MONEYBACK-SUCCESS-014	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
16	CLAIM-003	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-DIFF_MODEL-REPLACEMENT-SUCCESS-015	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
17	CLAIM-004	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-016	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
18	CLAIM-005	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-017	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
19	CLAIM-006	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-EXPIRED-MONEYBACK-SUCCESS-018	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
20	CLAIM-007	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-EXPIRED-REPLACEMENT-SUCCESS-019	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
21	CLAIM-008	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-DIFF_PROD-MONEYBACK-SUCCESS-020	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
22	CLAIM-009	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-ODOUR-REPLACEMENT-SUCCESS-021	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
23	CLAIM-010	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-ODOUR-MONEYBACK-SUCCESS-022	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
24	CLAIM-011	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-QUALITY-REPLACEMENT-SUCCESS-023	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
25	CLAIM-012	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-QUALITY-MONEYBACK-SUCCESS-024	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
26	CLAIM-013	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-SEAL-REPLACEMENT-SUCCESS-025	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
27	CLAIM-014	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-SEAL-MONEYBACK-SUCCESS-026	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
28	CLAIM-015	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-USED-REPLACEMENT-SUCCESS-027	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
29	CLAIM-016	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-USED-MONEYBACK-SUCCESS-028	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
30	CLAIM-017	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-FAKE-REPLACEMENT-SUCCESS-029	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
31	CLAIM-018	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-FAKE-MONEYBACK-SUCCESS-030	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
32	CLAIM-019	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-STAINED-REPLACEMENT-SUCCESS-031	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
33	CLAIM-020	raiseClaimNewIssueType	raiseClaim-BUYER-INAD-STAINED-MONEYBACK-SUCCESS-032	IN	BUYER	560017	AMEX_CREDIT_CARD	FALSE
34	CLAIM-021	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-DIFF_COLOR-REPLACEMENT-SUCCESS-033	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
35	CLAIM-022	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-DIFF_COLOR-MONEYBACK-SUCCESS-034	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
36	CLAIM-023	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-DIFF_MODEL-REPLACEMENT-SUCCESS-035	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
37	CLAIM-024	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-036	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
38	CLAIM-025	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-037	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
39	CLAIM-026	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-EXPIRED-MONEYBACK-SUCCESS-038	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
40	CLAIM-027	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-EXPIRED-REPLACEMENT-SUCCESS-039	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
41	CLAIM-028	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-DIFF_PROD-MONEYBACK-SUCCESS-040	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
42	CLAIM-029	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-ODOUR-REPLACEMENT-SUCCESS-041	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
43	CLAIM-030	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-ODOUR-MONEYBACK-SUCCESS-042	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE
44	CLAIM-031	raiseClaimNewIssueType	raiseClaim-CSR1-INAD-QUALITY-REPLACEMENT-SUCCESS-043	IN	CSR1	560017	AMEX_CREDIT_CARD	FALSE

eGclaimsNewIssue

Ready | 93%



eGclaimsNewIssue

Search Sheet

Home | Insert | Page Layout | Formulas | Data | Review | View | Share

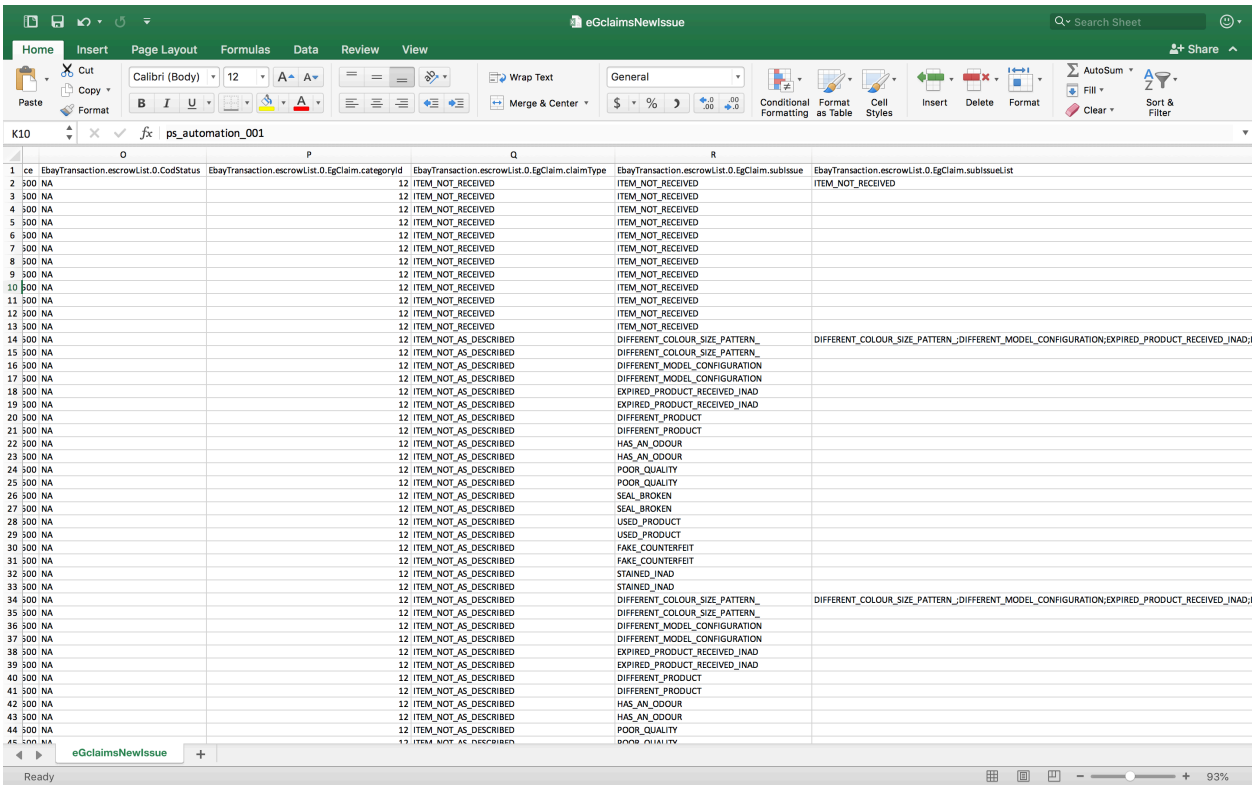
Paste | Cut | Copy | Format | Calibri (Body) | 12 | Wrap Text | Merge & Center | General | Conditional Formatting | Format as Table | Cell Styles | Insert | Delete | Format | AutoSum | Fill | Clear | Sort & Filter

C14 | raiseClaim-BUYER-INAD-DIFF_COLOR-REPLACEMENT-SUCCESS-013

	I	J	K	L	M	N	O
1	EbayTransaction.Buyer.dayPhone1	EbayTransaction.Buyer.nightPhone1	EbayTransaction.escrowList.0.Seller.UserLoginName	EbayTransaction.escrowList.0.SelectedShippingService	EbayTransaction.escrowList.0.ItemTemplate	EbayTransaction.escrowList.0.ItemPrice	EbayTransaction.escrowList.0.Co
2	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
3	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
4	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
5	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
6	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
7	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
8	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
9	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
10	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
11	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
12	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
13	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
14	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
15	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
16	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
17	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
18	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
19	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
20	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
21	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
22	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
23	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
24	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
25	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
26	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
27	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
28	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
29	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
30	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
31	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
32	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
33	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
34	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
35	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
36	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
37	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
38	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
39	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
40	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
41	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
42	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
43	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA
44	9999999999	8888888888	ps_automation_001	FLAT	PS_INBIN	1500	NA

eGclaimsNewIssue

Ready | 93%



eGclaimsNewIssue

Search Sheet

Home Insert Page Layout Formulas Data Review View Share

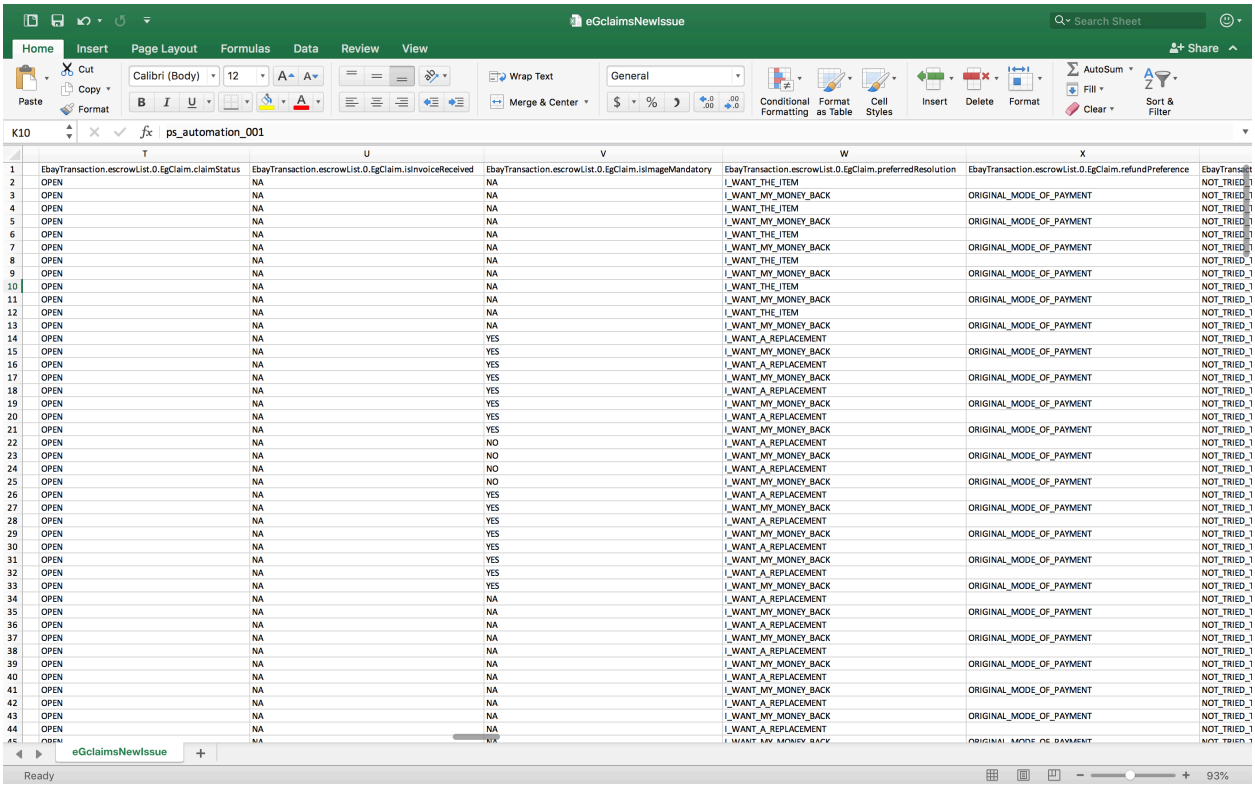
Paste Cut Copy Format Calibri (Body) 12 Wrap Text General Conditional Formatting Format as Table Cell Styles Insert Delete Format AutoSum Fill Clear Sort & Filter

K10 ps_automation_001

	O	P	Q	R
1	ce EbayTransaction.escrowList.0.CodStatus	EbayTransaction.escrowList.0.EgClaim.categoryId	EbayTransaction.escrowList.0.EgClaim.claimType	EbayTransaction.escrowList.0.EgClaim.subIssue
2	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
3	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
4	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
5	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
6	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
7	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
8	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
9	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
10	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
11	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
12	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
13	500 NA	12	ITEM_NOT_RECEIVED	ITEM_NOT_RECEIVED
14	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_COLOUR_SIZE_PATTERN_
15	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_COLOUR_SIZE_PATTERN_
16	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_MODEL_CONFIGURATION
17	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_MODEL_CONFIGURATION
18	500 NA	12	ITEM_NOT_AS_DESCRIBED	EXPIRED_PRODUCT_RECEIVED_INAD
19	500 NA	12	ITEM_NOT_AS_DESCRIBED	EXPIRED_PRODUCT_RECEIVED_INAD
20	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_PRODUCT
21	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_PRODUCT
22	500 NA	12	ITEM_NOT_AS_DESCRIBED	HAS_AN_ODOUR
23	500 NA	12	ITEM_NOT_AS_DESCRIBED	HAS_AN_ODOUR
24	500 NA	12	ITEM_NOT_AS_DESCRIBED	POOR_QUALITY
25	500 NA	12	ITEM_NOT_AS_DESCRIBED	POOR_QUALITY
26	500 NA	12	ITEM_NOT_AS_DESCRIBED	SEAL_BROKEN
27	500 NA	12	ITEM_NOT_AS_DESCRIBED	SEAL_BROKEN
28	500 NA	12	ITEM_NOT_AS_DESCRIBED	USED_PRODUCT
29	500 NA	12	ITEM_NOT_AS_DESCRIBED	USED_PRODUCT
30	500 NA	12	ITEM_NOT_AS_DESCRIBED	FAKE_COUNTERFEIT
31	500 NA	12	ITEM_NOT_AS_DESCRIBED	FAKE_COUNTERFEIT
32	500 NA	12	ITEM_NOT_AS_DESCRIBED	STAINED_INAD
33	500 NA	12	ITEM_NOT_AS_DESCRIBED	STAINED_INAD
34	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_COLOUR_SIZE_PATTERN_
35	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_COLOUR_SIZE_PATTERN_
36	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_MODEL_CONFIGURATION
37	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_MODEL_CONFIGURATION
38	500 NA	12	ITEM_NOT_AS_DESCRIBED	EXPIRED_PRODUCT_RECEIVED_INAD
39	500 NA	12	ITEM_NOT_AS_DESCRIBED	EXPIRED_PRODUCT_RECEIVED_INAD
40	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_PRODUCT
41	500 NA	12	ITEM_NOT_AS_DESCRIBED	DIFFERENT_PRODUCT
42	500 NA	12	ITEM_NOT_AS_DESCRIBED	HAS_AN_ODOUR
43	500 NA	12	ITEM_NOT_AS_DESCRIBED	HAS_AN_ODOUR
44	500 NA	12	ITEM_NOT_AS_DESCRIBED	POOR_QUALITY

eGclaimsNewIssue

Ready 93%



eGclaimsNewIssue

Search Sheet

Home Insert Page Layout Formulas Data Review View Share

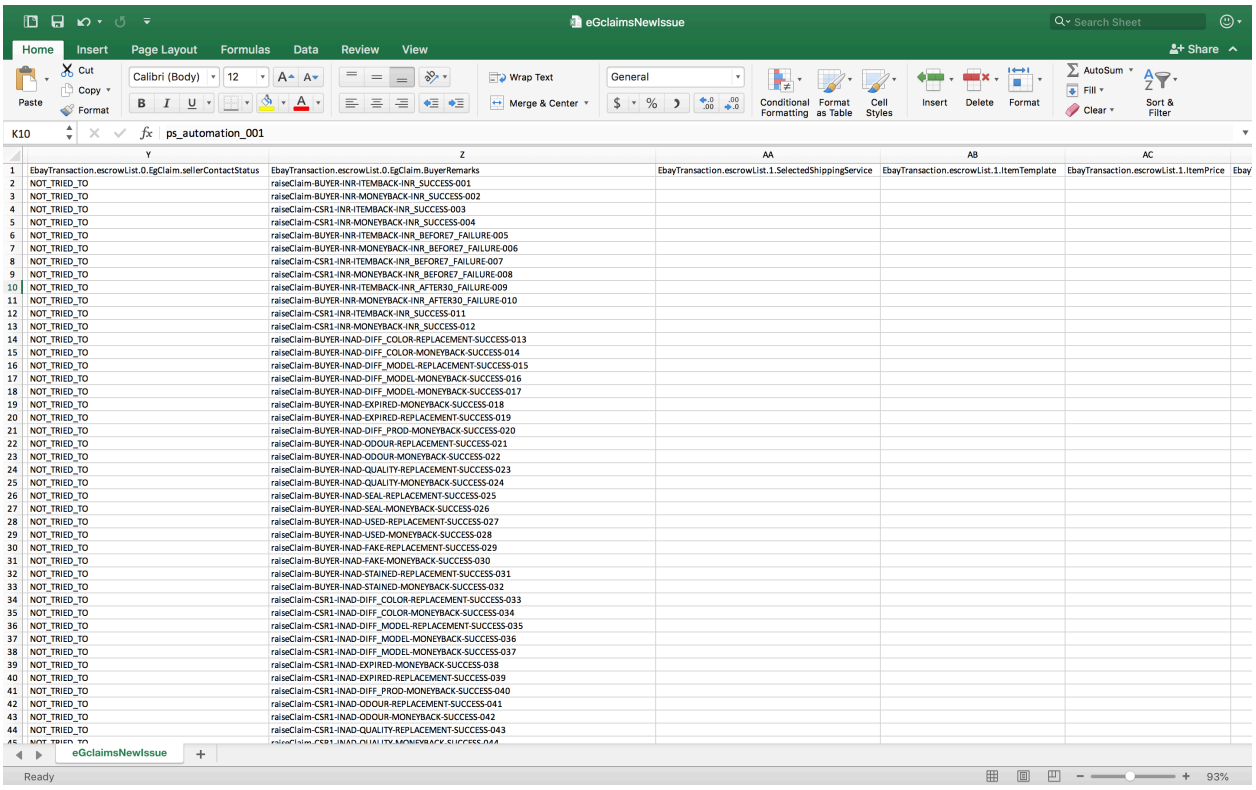
Paste Cut Copy Format Calibri (Body) 12 Wrap Text General Conditional Formatting Format as Table Cell Styles Insert Delete Format AutoSum Fill Clear Sort & Filter

K10 ps_automation_001

	T	U	V	W	X
1	EbayTransaction.escrowList.0.EgClaim.claimStatus	EbayTransaction.escrowList.0.EgClaim.isInvoiceReceived	EbayTransaction.escrowList.0.EgClaim.isImageMandatory	EbayTransaction.escrowList.0.EgClaim.preferredResolution	EbayTransaction.escrowList.0.EgClaim.refundPreference
2	OPEN	NA	NA	I_WANT_THE_ITEM	
3	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
4	OPEN	NA	NA	I_WANT_THE_ITEM	
5	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
6	OPEN	NA	NA	I_WANT_THE_ITEM	
7	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
8	OPEN	NA	NA	I_WANT_THE_ITEM	
9	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
10	OPEN	NA	NA	I_WANT_THE_ITEM	
11	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
12	OPEN	NA	NA	I_WANT_THE_ITEM	
13	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
14	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
15	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
16	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
17	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
18	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
19	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
20	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
21	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
22	OPEN	NA	NO	I_WANT_A_REPLACEMENT	
23	OPEN	NA	NO	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
24	OPEN	NA	NO	I_WANT_A_REPLACEMENT	
25	OPEN	NA	NO	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
26	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
27	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
28	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
29	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
30	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
31	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
32	OPEN	NA	YES	I_WANT_A_REPLACEMENT	
33	OPEN	NA	YES	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
34	OPEN	NA	NA	I_WANT_A_REPLACEMENT	
35	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
36	OPEN	NA	NA	I_WANT_A_REPLACEMENT	
37	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
38	OPEN	NA	NA	I_WANT_A_REPLACEMENT	
39	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
40	OPEN	NA	NA	I_WANT_A_REPLACEMENT	
41	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
42	OPEN	NA	NA	I_WANT_A_REPLACEMENT	
43	OPEN	NA	NA	I_WANT_MY_MONEY_BACK	ORIGINAL_MODE_OF_PAYMENT
44	OPEN	NA	NA	I_WANT_A_REPLACEMENT	

eGclaimsNewIssue

Ready 93%



eGclaimsNewIssue

Search Sheet

Home | Insert | Page Layout | Formulas | Data | Review | View | Share

Paste | Cut | Copy | Format | Calibri (Body) | 12 | Wrap Text | General | Merge & Center | Conditional Formatting | Format as Table | Cell Styles | Insert | Delete | Format | AutoSum | Fill | Clear | Sort & Filter

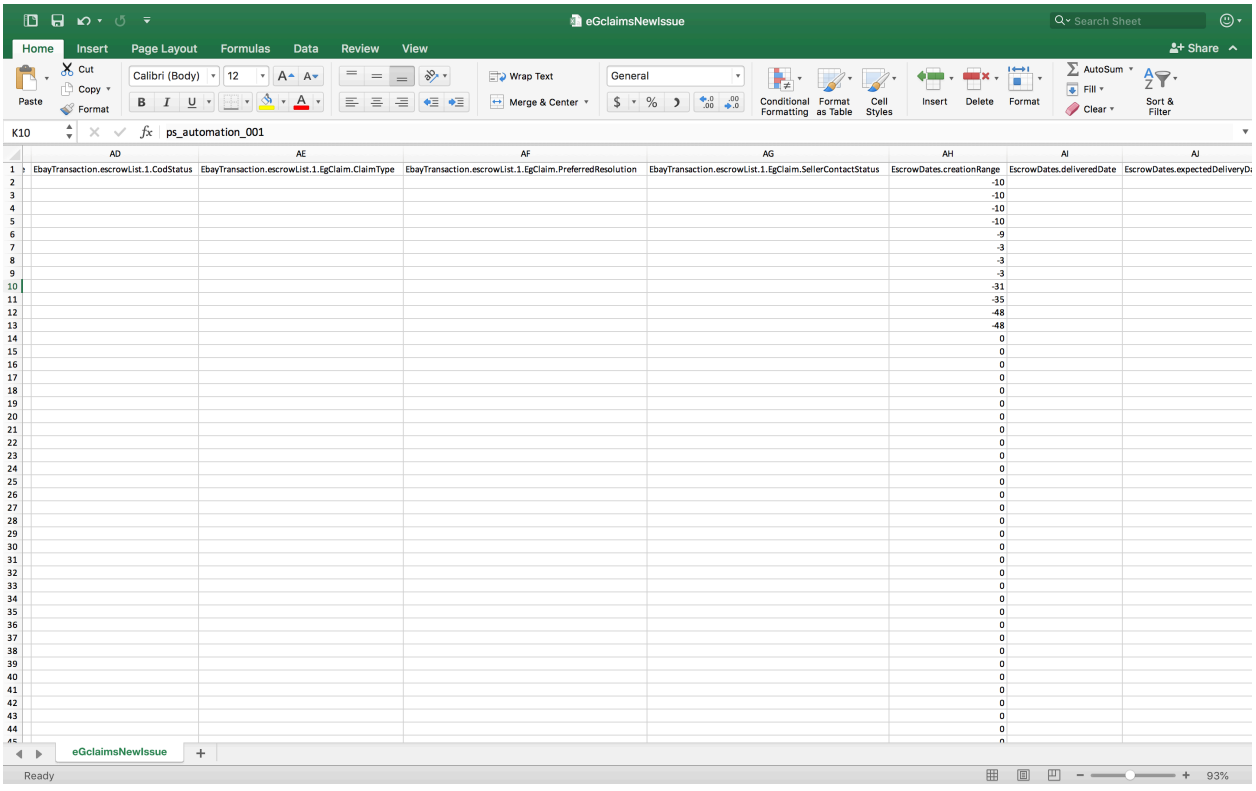
K10 | ps_automation_001

	Y	Z	AA	AB	AC
1	EbayTransaction.escrowList.0.EgClaim.sellerContactStatus	EbayTransaction.escrowList.0.EgClaim.BuyerRemarks	EbayTransaction.escrowList.1.SelectedShippingService	EbayTransaction.escrowList.1.ItemTemplate	EbayTransaction.escrowList.1.ItemPrice
2	NOT_TRIED_TO	raiseClaim-BUYER-INR-ITEMBACK-INR_SUCCESS-001			
3	NOT_TRIED_TO	raiseClaim-BUYER-INR-MONEYBACK-INR_SUCCESS-002			
4	NOT_TRIED_TO	raiseClaim-CSR1-INR-ITEMBACK-INR_SUCCESS-003			
5	NOT_TRIED_TO	raiseClaim-CSR1-INR-MONEYBACK-INR_SUCCESS-004			
6	NOT_TRIED_TO	raiseClaim-BUYER-INR-ITEMBACK-INR_BEFORE7_FAILURE-005			
7	NOT_TRIED_TO	raiseClaim-BUYER-INR-MONEYBACK-INR_BEFORE7_FAILURE-006			
8	NOT_TRIED_TO	raiseClaim-CSR1-INR-ITEMBACK-INR_BEFORE7_FAILURE-007			
9	NOT_TRIED_TO	raiseClaim-CSR1-INR-MONEYBACK-INR_BEFORE7_FAILURE-008			
10	NOT_TRIED_TO	raiseClaim-BUYER-INR-ITEMBACK-INR_AFTER30_FAILURE-009			
11	NOT_TRIED_TO	raiseClaim-BUYER-INR-MONEYBACK-INR_AFTER30_FAILURE-010			
12	NOT_TRIED_TO	raiseClaim-CSR1-INR-ITEMBACK-INR_SUCCESS-011			
13	NOT_TRIED_TO	raiseClaim-CSR1-INR-MONEYBACK-INR_SUCCESS-012			
14	NOT_TRIED_TO	raiseClaim-BUYER-INAD-DIFF_COLOR-REPLACEMENT-SUCCESS-013			
15	NOT_TRIED_TO	raiseClaim-BUYER-INAD-DIFF_COLOR-MONEYBACK-SUCCESS-014			
16	NOT_TRIED_TO	raiseClaim-BUYER-INAD-DIFF_MODEL-REPLACEMENT-SUCCESS-015			
17	NOT_TRIED_TO	raiseClaim-BUYER-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-016			
18	NOT_TRIED_TO	raiseClaim-BUYER-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-017			
19	NOT_TRIED_TO	raiseClaim-BUYER-INAD-EXPIRED-MONEYBACK-SUCCESS-018			
20	NOT_TRIED_TO	raiseClaim-BUYER-INAD-EXPIRED-REPLACEMENT-SUCCESS-019			
21	NOT_TRIED_TO	raiseClaim-BUYER-INAD-DIFF_PROD-MONEYBACK-SUCCESS-020			
22	NOT_TRIED_TO	raiseClaim-BUYER-INAD-ODOUR-REPLACEMENT-SUCCESS-021			
23	NOT_TRIED_TO	raiseClaim-BUYER-INAD-ODOUR-MONEYBACK-SUCCESS-022			
24	NOT_TRIED_TO	raiseClaim-BUYER-INAD-QUALITY-REPLACEMENT-SUCCESS-023			
25	NOT_TRIED_TO	raiseClaim-BUYER-INAD-QUALITY-MONEYBACK-SUCCESS-024			
26	NOT_TRIED_TO	raiseClaim-BUYER-INAD-SEAL-REPLACEMENT-SUCCESS-025			
27	NOT_TRIED_TO	raiseClaim-BUYER-INAD-SEAL-MONEYBACK-SUCCESS-026			
28	NOT_TRIED_TO	raiseClaim-BUYER-INAD-USED-REPLACEMENT-SUCCESS-027			
29	NOT_TRIED_TO	raiseClaim-BUYER-INAD-USED-MONEYBACK-SUCCESS-028			
30	NOT_TRIED_TO	raiseClaim-BUYER-INAD-FAKE-REPLACEMENT-SUCCESS-029			
31	NOT_TRIED_TO	raiseClaim-BUYER-INAD-FAKE-MONEYBACK-SUCCESS-030			
32	NOT_TRIED_TO	raiseClaim-BUYER-INAD-STAINED-REPLACEMENT-SUCCESS-031			
33	NOT_TRIED_TO	raiseClaim-BUYER-INAD-STAINED-MONEYBACK-SUCCESS-032			
34	NOT_TRIED_TO	raiseClaim-CSR1-INAD-DIFF_COLOR-REPLACEMENT-SUCCESS-033			
35	NOT_TRIED_TO	raiseClaim-CSR1-INAD-DIFF_COLOR-MONEYBACK-SUCCESS-034			
36	NOT_TRIED_TO	raiseClaim-CSR1-INAD-DIFF_MODEL-REPLACEMENT-SUCCESS-035			
37	NOT_TRIED_TO	raiseClaim-CSR1-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-036			
38	NOT_TRIED_TO	raiseClaim-CSR1-INAD-DIFF_MODEL-MONEYBACK-SUCCESS-037			
39	NOT_TRIED_TO	raiseClaim-CSR1-INAD-EXPIRED-MONEYBACK-SUCCESS-038			
40	NOT_TRIED_TO	raiseClaim-CSR1-INAD-EXPIRED-REPLACEMENT-SUCCESS-039			
41	NOT_TRIED_TO	raiseClaim-CSR1-INAD-DIFF_PROD-MONEYBACK-SUCCESS-040			
42	NOT_TRIED_TO	raiseClaim-CSR1-INAD-ODOUR-REPLACEMENT-SUCCESS-041			
43	NOT_TRIED_TO	raiseClaim-CSR1-INAD-ODOUR-MONEYBACK-SUCCESS-042			
44	NOT_TRIED_TO	raiseClaim-CSR1-INAD-QUALITY-REPLACEMENT-SUCCESS-043			

eGclaimsNewIssue

Ready

93%



eGclaimsNewIssue

Search Sheet

Home | Insert | Page Layout | Formulas | Data | Review | View | Share

Paste | Cut | Copy | Format | Calibri (Body) | 12 | Wrap Text | General | Merge & Center | Conditional Formatting | Format as Table | Cell Styles | Insert | Delete | Format | AutoSum | Fill | Clear | Sort & Filter

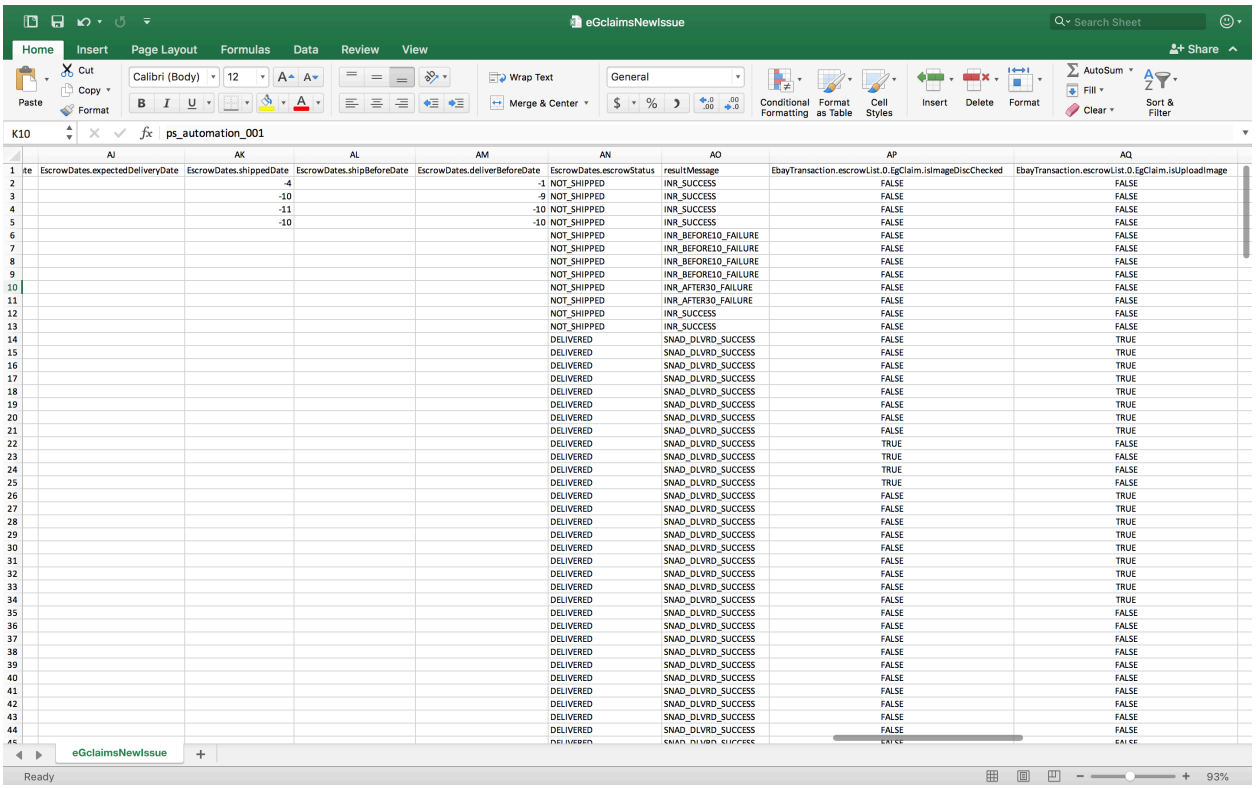
K10 | ps_automation_001

	AD	AE	AF	AG	AH	AI	AJ
1	EbayTransaction.escrowList.1.CodStatus	EbayTransaction.escrowList.1.EgClaim.ClaimType	EbayTransaction.escrowList.1.EgClaim.PreferredResolution	EbayTransaction.escrowList.1.EgClaim.SellerContactStatus	EscrowDates.creationRange	EscrowDates.deliveredDate	EscrowDates.expectedDeliveryDa
2					-10		
3					-10		
4					-10		
5					-10		
6					-9		
7					-3		
8					-3		
9					-3		
10					-31		
11					-35		
12					-48		
13					-48		
14					0		
15					0		
16					0		
17					0		
18					0		
19					0		
20					0		
21					0		
22					0		
23					0		
24					0		
25					0		
26					0		
27					0		
28					0		
29					0		
30					0		
31					0		
32					0		
33					0		
34					0		
35					0		
36					0		
37					0		
38					0		
39					0		
40					0		
41					0		
42					0		
43					0		
44					0		

eGclaimsNewIssue

Ready

93%



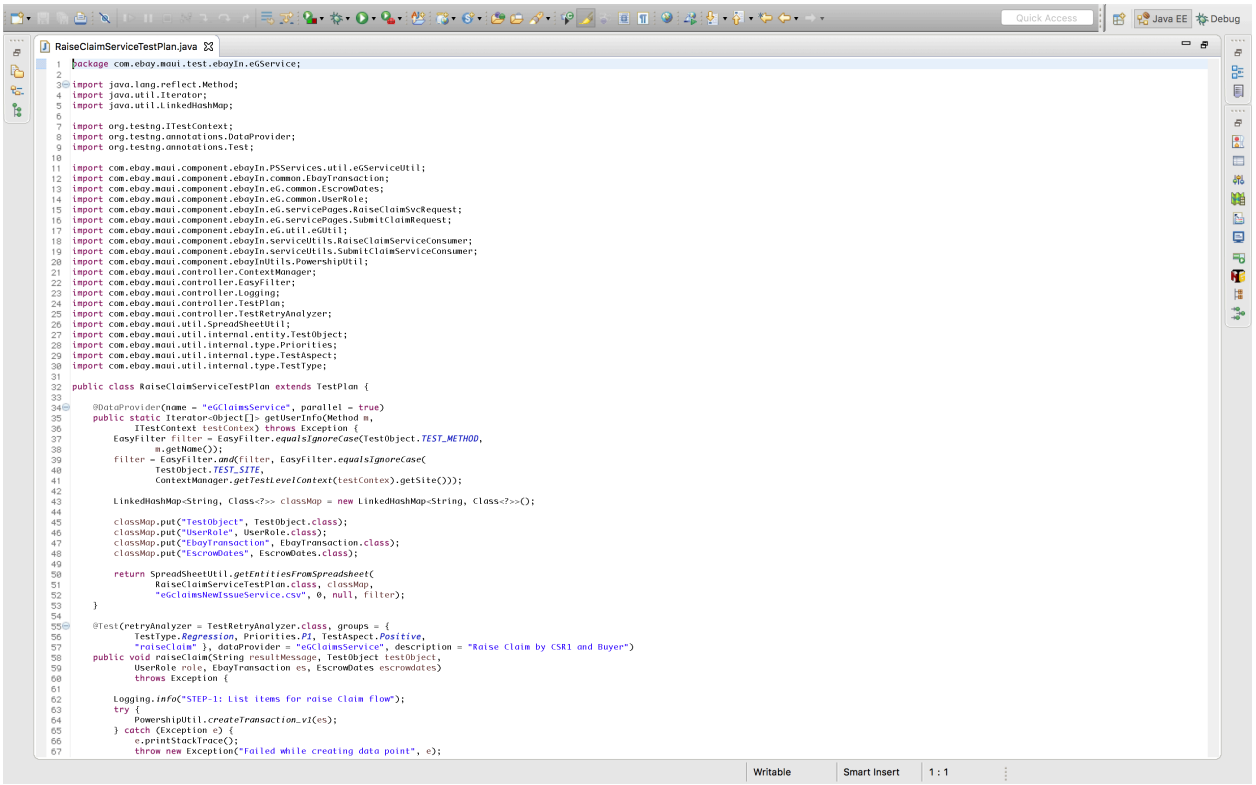
eGclaimsNewIssue

Home | Insert | Page Layout | Formulas | Data | Review | View

K10: ps_automation_001

	AI	AK	AL	AM	AN	AO	AP	AQ
1	EscrowDates.expectedDeliveryDate	EscrowDates.shippedDate	EscrowDates.shipBeforeDate	EscrowDates.deliverBeforeDate	EscrowDates.escrowStatus	resultMessage	EbayTransaction.escrowList.0.EgClaim.isImageDiscChecked	EbayTransaction.escrowList.0.EgClaim.isUploadImage
2		-4		-1	NOT_SHIPPED	INR_SUCCESS	FALSE	FALSE
3		-10		-9	NOT_SHIPPED	INR_SUCCESS	FALSE	FALSE
4		-11		-10	NOT_SHIPPED	INR_SUCCESS	FALSE	FALSE
5		-10		-10	NOT_SHIPPED	INR_SUCCESS	FALSE	FALSE
6					NOT_SHIPPED	INR_BEFORE10_FAILURE	FALSE	FALSE
7					NOT_SHIPPED	INR_BEFORE10_FAILURE	FALSE	FALSE
8					NOT_SHIPPED	INR_BEFORE10_FAILURE	FALSE	FALSE
9					NOT_SHIPPED	INR_BEFORE10_FAILURE	FALSE	FALSE
10					NOT_SHIPPED	INR_AFTER30_FAILURE	FALSE	FALSE
11					NOT_SHIPPED	INR_AFTER30_FAILURE	FALSE	FALSE
12					NOT_SHIPPED	INR_SUCCESS	FALSE	FALSE
13					NOT_SHIPPED	INR_SUCCESS	FALSE	FALSE
14					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
15					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
16					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
17					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
18					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
19					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
20					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
21					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
22					DELIVERED	SNAD_DLVRD_SUCCESS	TRUE	FALSE
23					DELIVERED	SNAD_DLVRD_SUCCESS	TRUE	FALSE
24					DELIVERED	SNAD_DLVRD_SUCCESS	TRUE	FALSE
25					DELIVERED	SNAD_DLVRD_SUCCESS	TRUE	FALSE
26					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
27					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
28					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
29					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
30					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
31					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
32					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
33					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
34					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	TRUE
35					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
36					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
37					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
38					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
39					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
40					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
41					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
42					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
43					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE
44					DELIVERED	SNAD_DLVRD_SUCCESS	FALSE	FALSE

Ready



RaiseClaimServiceTestPlan.java

```
package com.ebay.maui.test.ebayIn.eGService;

import java.lang.reflect.Method;
import java.util.Iterator;
import java.util.LinkedHashMap;

import org.testng.ITestContext;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

import com.ebay.maui.component.ebayIn.PSServices.util.eGServiceUtil;
import com.ebay.maui.component.ebayIn.common.EbayTransaction;
import com.ebay.maui.component.ebayIn.eG.common.EscrowDates;
import com.ebay.maui.component.ebayIn.eG.common.UserRole;
import com.ebay.maui.component.ebayIn.eG.servicePages.RaiseClaimSvcRequest;
import com.ebay.maui.component.ebayIn.eG.servicePages.SubmitClaimRequest;
import com.ebay.maui.component.ebayIn.eG.util.eGUtil;
import com.ebay.maui.component.ebayIn.serviceUtils.RaiseClaimServiceConsumer;
import com.ebay.maui.component.ebayIn.serviceUtils.SubmitClaimServiceConsumer;
import com.ebay.maui.component.ebayInUtils.PowershipUtil;
import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.EasyFilter;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.controller.TestPlan;
import com.ebay.maui.controller.TestRetryAnalyzer;
import com.ebay.maui.util.SpreadSheetUtil;
import com.ebay.maui.util.internal.entity.TestObject;
import com.ebay.maui.util.internal.type.Priorities;
import com.ebay.maui.util.internal.type.TestAspect;
import com.ebay.maui.util.internal.type.TestType;

public class RaiseClaimServiceTestPlan extends TestPlan {

    @DataProvider(name = "eGClaimsService", parallel = true)
    public static Iterator<Object[]> getUserInfo(Method m,
            ITestContext testContex) throws Exception {
        EasyFilter filter = EasyFilter.equalsIgnoreCase(TestObject.TEST_METHOD,
                m.getName());
        filter = EasyFilter.and(filter, EasyFilter.equalsIgnoreCase(
                TestObject.TEST_SITE,
                ContextManager.getTestLevelContext(testContex).getSite()));

        LinkedHashMap<String, Class<?>> classMap = new LinkedHashMap<String, Class<?>>();

        classMap.put("TestObject", TestObject.class);
        classMap.put("UserRole", UserRole.class);
        classMap.put("EbayTransaction", EbayTransaction.class);
        classMap.put("EscrowDates", EscrowDates.class);

        return SpreadSheetUtil.getEntitiesFromSpreadsheet(
                RaiseClaimServiceTestPlan.class, classMap,
                "eGClaimsNewIssueService.csv", 0, null, filter);
    }

    @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
            TestType.Regression, Priorities.P1, TestAspect.Positive,
            "raiseClaim" }, dataProvider = "eGClaimsService", description = "Raise Claim by CSR1 and Buyer")
    public void raiseClaim(String resultMessage, TestObject testObject,
            UserRole role, EbayTransaction es, EscrowDates escrowdates)
            throws Exception {

        Logging.info("STEP-1: List items for raise Claim flow");
        try {
            PowershipUtil.createTransaction_v1(es);
        } catch (Exception e) {
            e.printStackTrace();
            throw new Exception("Failed while creating data point", e);
```

Writable | Smart Insert | 1 : 1

RaiseClaimServiceTestPlan.java

```
54
55      @Test(retryAnalyzer = TestRetryAnalyzer.class, groups = {
56              TestType.Regression, Priorities.P1, TestAspect.Positive,
57              "raiseClaim" }, dataProvider = "eGClaimsService", description = "Raise Claim by CSR1 and Buyer")
58      public void raiseClaim(String resultMessage, TestObject testObject,
59              UserRole role, EbayTransaction es, EscrowDates escrowdates)
60              throws Exception {
61
62          Logging.info("STEP-1: List items for raise Claim flow");
63          try {
64              PowershipUtil.createTransaction_v1(es);
65          } catch (Exception e) {
66              e.printStackTrace();
67              throw new Exception("Failed while creating data point", e);
68          }
69
70          Logging.info("STEP-2: Mark the trasaction to deliver");
71          eGUtil.updateTrasanctionStatus(escrowdates, es.getEscrowList().get(0),
72                  es);
73
74          Logging.info("STEP-3: Built the request payload for raise claim");
75          RaiseClaimSvcRequest raiseClaimRequest = new RaiseClaimSvcRequest();
76          if (role.getUserRole().trim().equalsIgnoreCase("CSR1")) {
77              raiseClaimRequest.setLoggedInUserName(PowershipUtil.adminCred
78                      .getProperty("eg_csr1"));
79          } else
80              raiseClaimRequest.setLoggedInUserName(es.getBuyer()
81                      .getUserLoginName());
82
83          eGServiceUtil.RaiseClaimrequest(es, raiseClaimRequest);
84          RaiseClaimServiceConsumer reqConsumer = new RaiseClaimServiceConsumer();
85
86          Logging.info("STEP-4: Raise the claim");
87          reqConsumer.raiseClaim(raiseClaimRequest);
88          Logging.info("STEP-5: Validate the claim");
89          eGServiceUtil.RaiseClaimresponsevalidation(es);
90
91      }
92
93      @Test(retryAnalyzer = TestRetryAnalyzer.class, enabled = false, groups = {
94              TestType.Regression, Priorities.P1, TestAspect.Positive,
95              "submitClaim" }, dataProvider = "eGClaimsService", description = "Submit Claim by Buyer")
96      public void submitClaim(String updateTransactiondate, String deliveryDate,
97              TestObject testObject, UserRole role, EbayTransaction es)
98              throws Exception {
99
100         // List items for raise Claim flow
101         try {
102             PowershipUtil.createTransaction_v1(es);
103         } catch (Exception e) {
104             e.printStackTrace();
105             throw new Exception("Failed while creating data point", e);
106         }
107
108         SubmitClaimRequest submitClaimRequest = new SubmitClaimRequest();
109
110         eGServiceUtil.SubmitClaimrequest(es, submitClaimRequest);
111
112         SubmitClaimServiceConsumer submitConsumer = new SubmitClaimServiceConsumer();
113
114         submitConsumer.submitClaim(submitClaimRequest);
115
116         eGServiceUtil.submitClaimresponsevalidation(es);
117
118     }
119 }
120
```

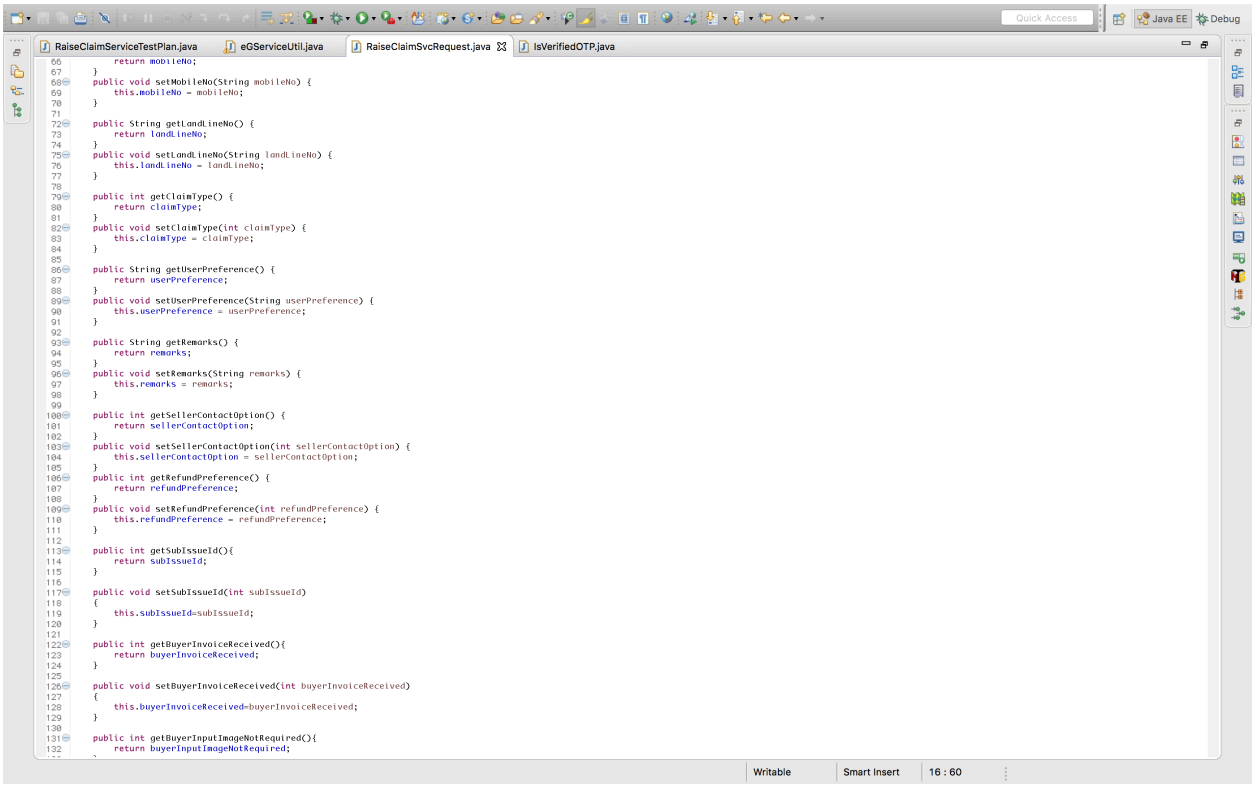
Writable | Smart Insert | 1 : 1

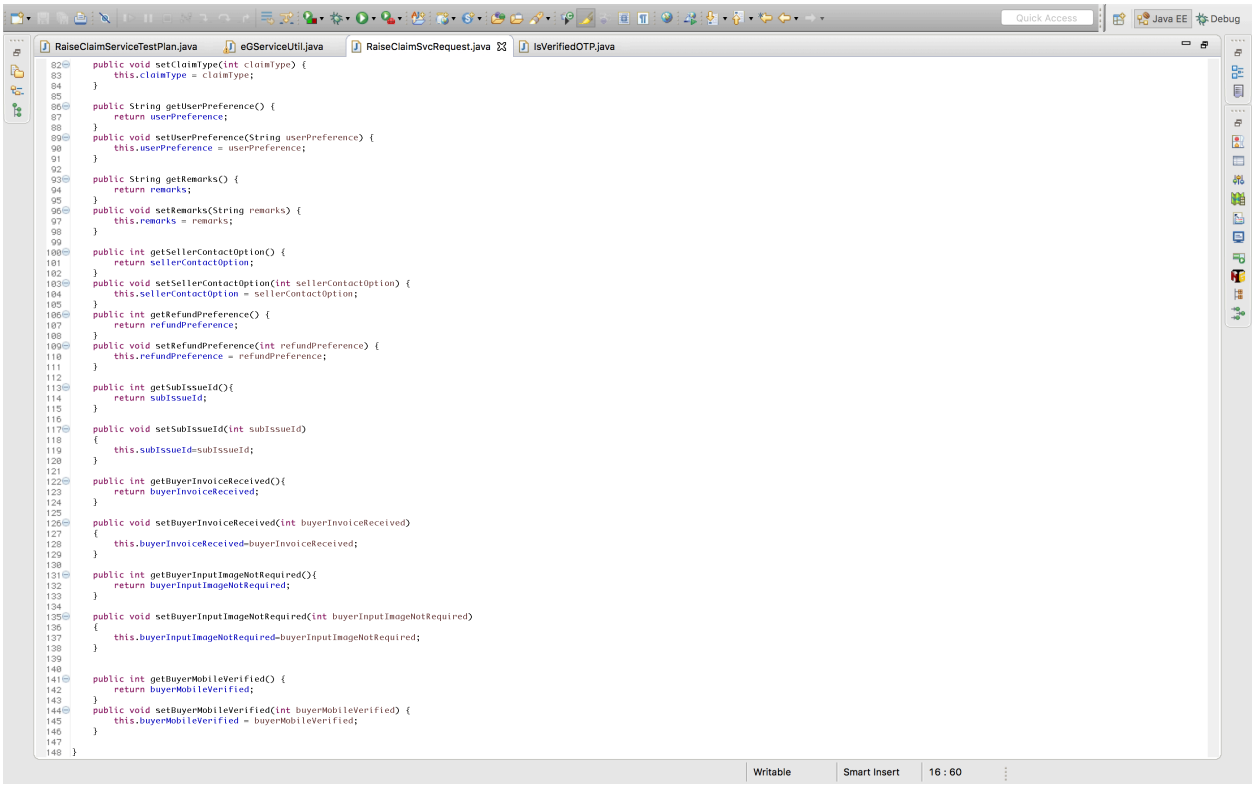
Service:

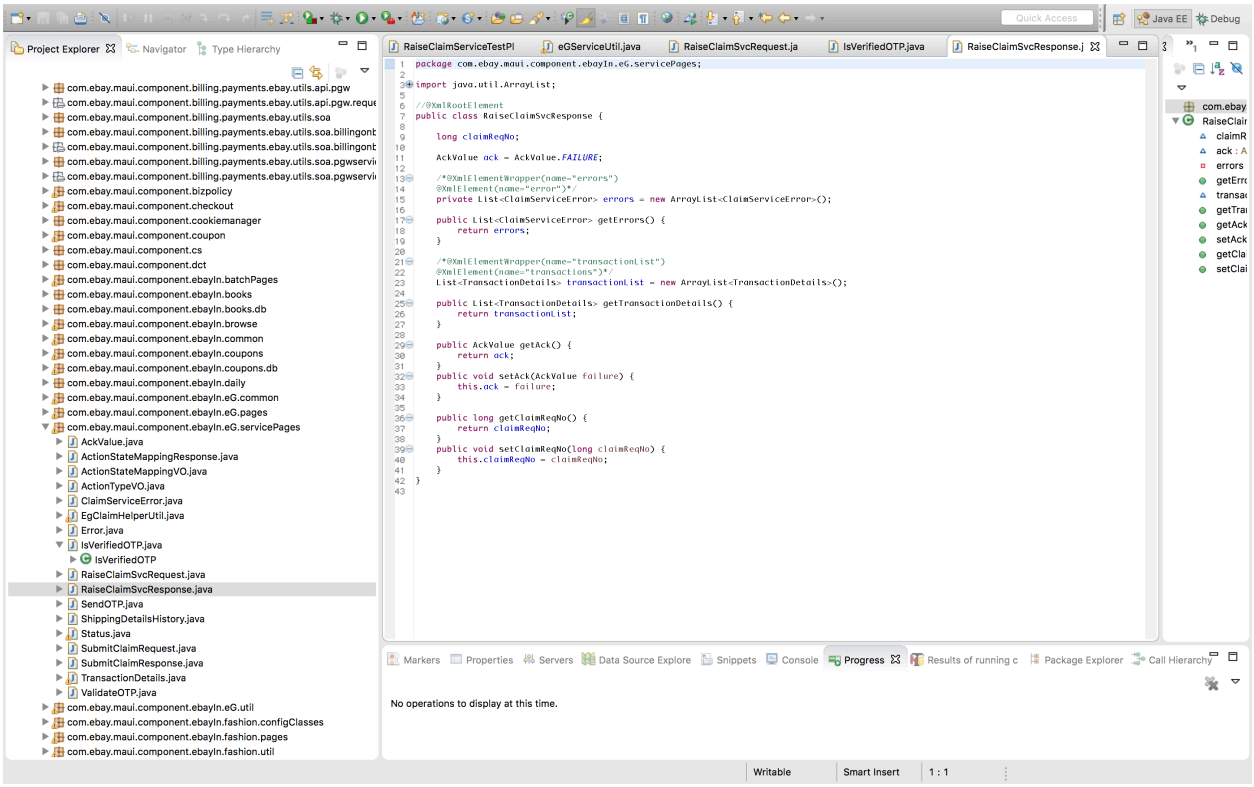
RaiseClaimServiceTestPlan.java | eGServiceUtil.java | RaiseClaimSvcRequest.java | IsVerifiedOTP.java

```
1   package com.ebay.maui.component.ebayIn.eG.servicePages;
2
3   import javax.xml.bind.annotation.XmlRootElement;
4
5   @XmlRootElement
6   public class RaiseClaimSvcRequest {
7
8       String loggedInUserName;  // Mandatory. Max length 75 character
9
10      long escrowId;            // Mandatory. Max length 20.
11      long txnId ;              // Mandatory. Max length 20.
12      long itemId;              // Mandatory. Max length 20.
13
14      // User selected values
15      String mobileNo;          // Mandatory. Max length 15.
16      String landLineNo;        // Mandatory. Max length 15.
17
18      int claimType;            // Mandatory. Values should be 7/8/9/10
19      String userPreference;    // Mandatory. Values should be 1/2/3
20      String remarks;           // Mandatory. Max length 4000.
21
22      //Added By Gourav
23      int subIssueId;
24      int buyerInvoiceReceived;
25      int buyerInputImageNotRequired;
26
27      /** Not Contacted, Not Tried - 1
28       Value  for Not Contacted, Tried to = Seller's phone not reachable (6)
29       Seller does not answer phone (7) /Seller does not reply to email (8)
30       Value Yes = contact by phone (3) / contact by email (4) /  contact by both (5) */
31
32      int sellerContactOption;     // Mandatory
33      int refundPreference;     // Mandatory. Values should be 1/2/3/4
34      int buyerMobileVerified;
35
36
37      public String getLoggedInUserName() {
38          return loggedInUserName;
39      }
40      public void setLoggedInUserName(String loggedInUserName) {
41          this.loggedInUserName = loggedInUserName;
42      }
43
44      public long getEscrowId() {
45          return escrowId;
46      }
47      public void setEscrowId(long escrowId) {
48          this.escrowId = escrowId;
49      }
50
51      public long getTxnId() {
52          return txnId;
53      }
54      public void setTxnId(long txnId) {
55          this.txnId = txnId;
56      }
57
58      public long getItemId() {
59          return itemId;
60      }
61      public void setItemId(long itemId) {
62          this.itemId = itemId;
63      }
64
65      public String getMobileNo() {
66          return mobileNo;
67      }
```

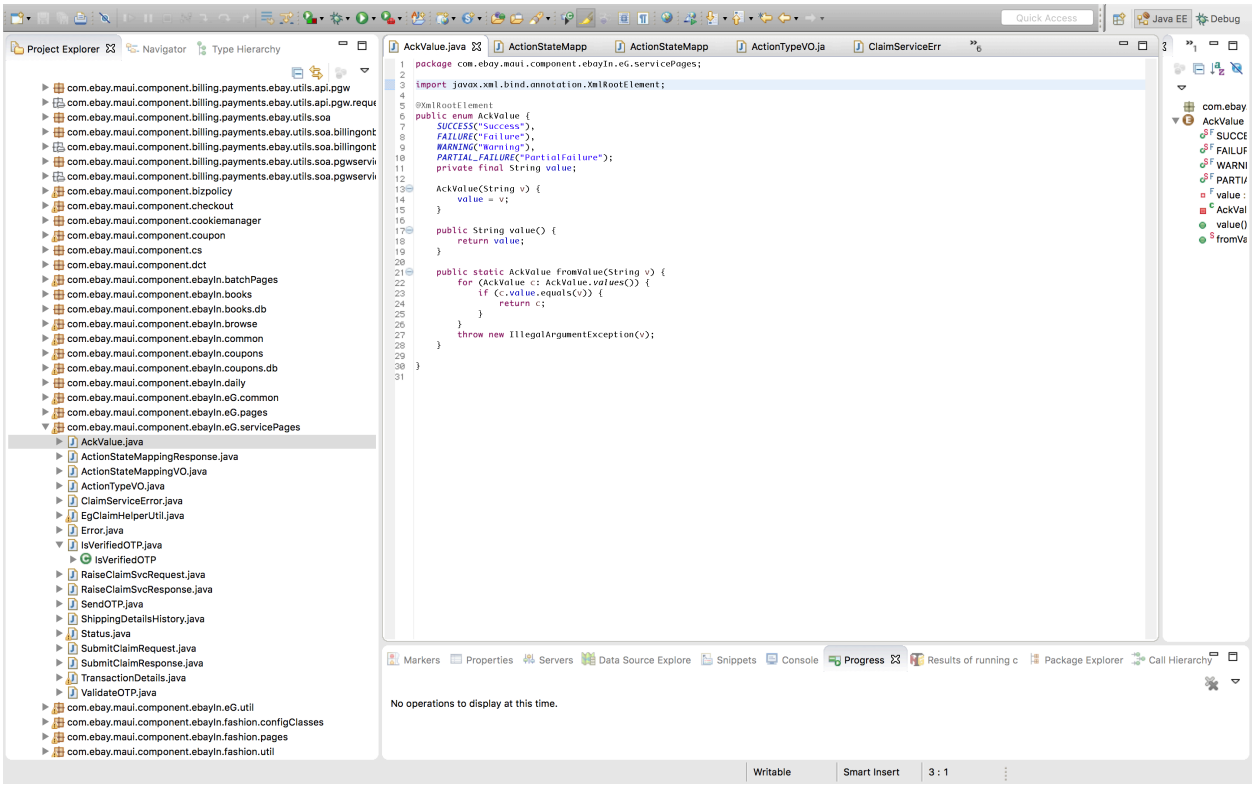
Writable | Smart Insert | 16 : 60


RaiseClaimServiceTestPlan.java
eGServiceUtil.java
RaiseClaimSvcRequest.java
IsVerifiedOTP.java
Writable
Smart Insert
16 : 60

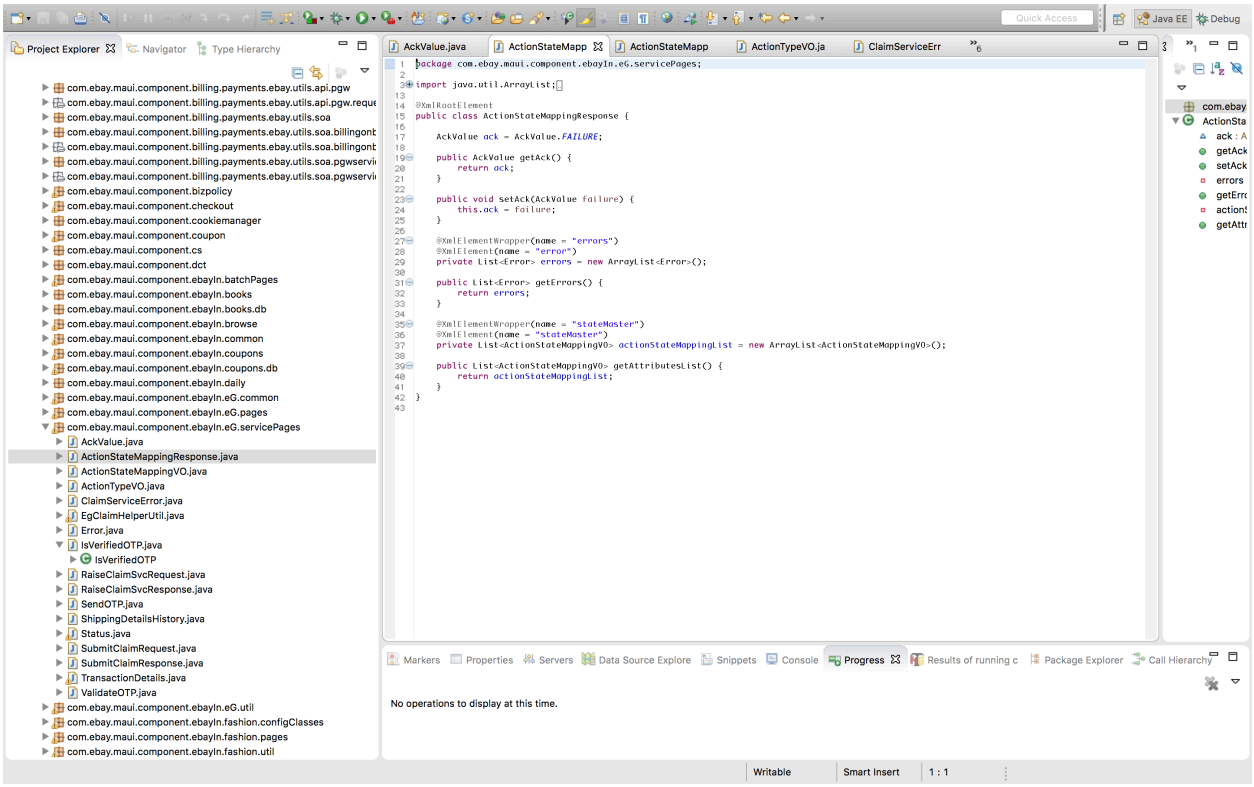

RaiseClaimServiceTestPlan.java
eGServiceUtil.java
RaiseClaimSvcRequest.java
IsVerifiedOTP.java
Writable
Smart Insert
16 : 60



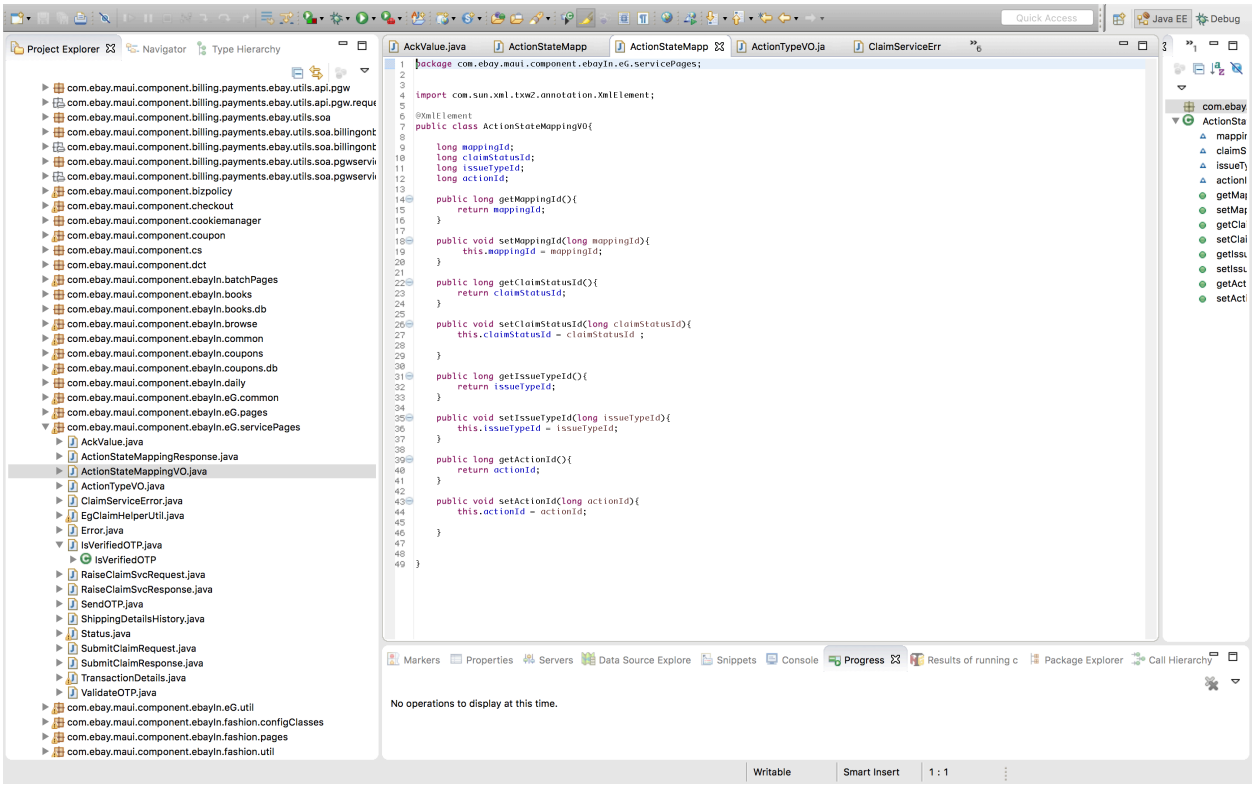
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

import java.util.ArrayList;

//@XmlRootElement
public class RaiseClaimSvcResponse {

    long claimReqNo;

    AckValue ack = AckValue.FAILURE;

    /*@XmlElementWrapper(name="errors")
    @XmlElement(name="error")*/
    private List<ClaimServiceError> errors = new ArrayList<ClaimServiceError>();

    public List<ClaimServiceError> getErrors() {
        return errors;
    }

    /*@XmlElementWrapper(name="transactionList")
    @XmlElement(name="transactions")*/
    List<TransactionDetails> transactionList = new ArrayList<TransactionDetails>();

    public List<TransactionDetails> getTransactionDetails() {
        return transactionList;
    }

    public AckValue getAck() {
        return ack;
    }
    public void setAck(AckValue failure) {
        this.ack = failure;
    }

    public long getClaimReqNo() {
        return claimReqNo;
    }
    public void setClaimReqNo(long claimReqNo) {
        this.claimReqNo = claimReqNo;
    }
}
```



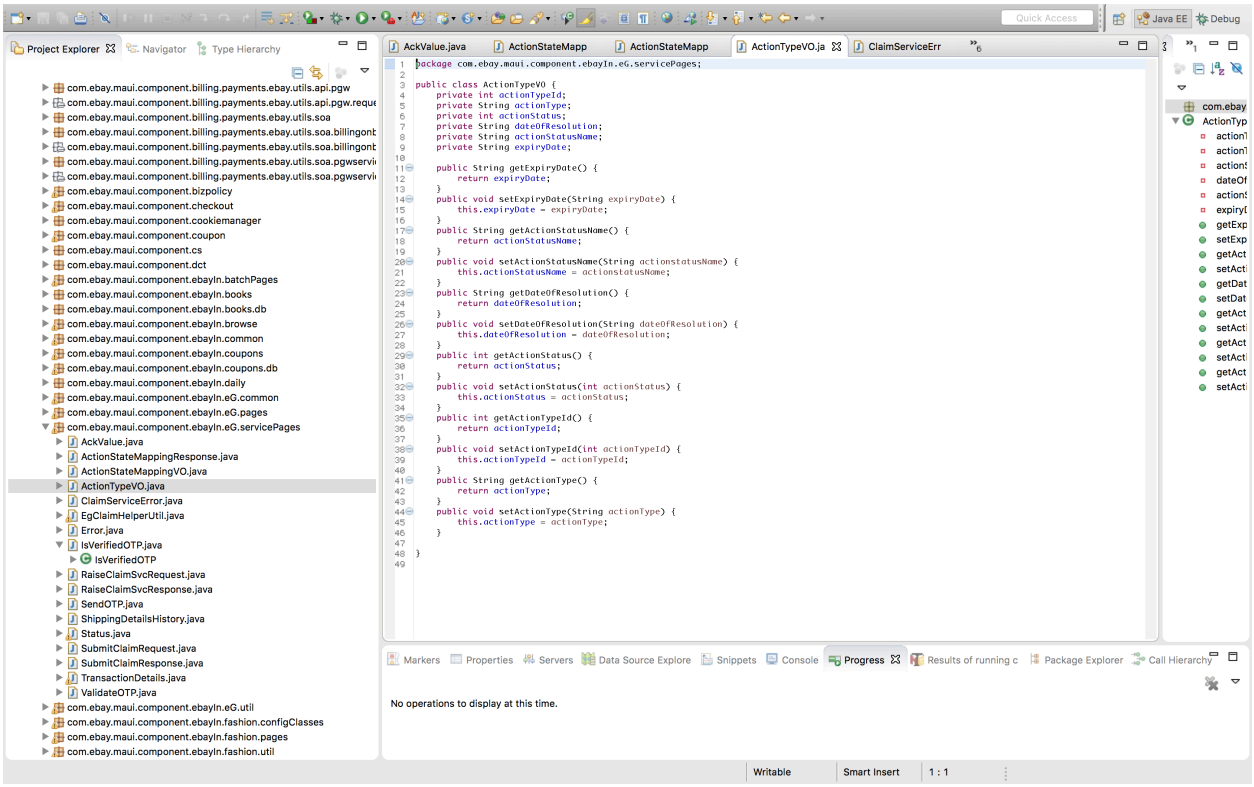
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public enum AckValue {
    SUCCESS("Success"),
    FAILURE("Failure"),
    WARNING("Warning"),
    PARTIAL_FAILURE("PartialFailure");
    private final String value;

    AckValue(String v) {
        value = v;
    }

    public String value() {
        return value;
    }

    public static AckValue fromValue(String v) {
        for (AckValue c: AckValue.values()) {
            if (c.value.equals(v)) {
                return c;
            }
        }
        throw new IllegalArgumentException(v);
    }

}
```



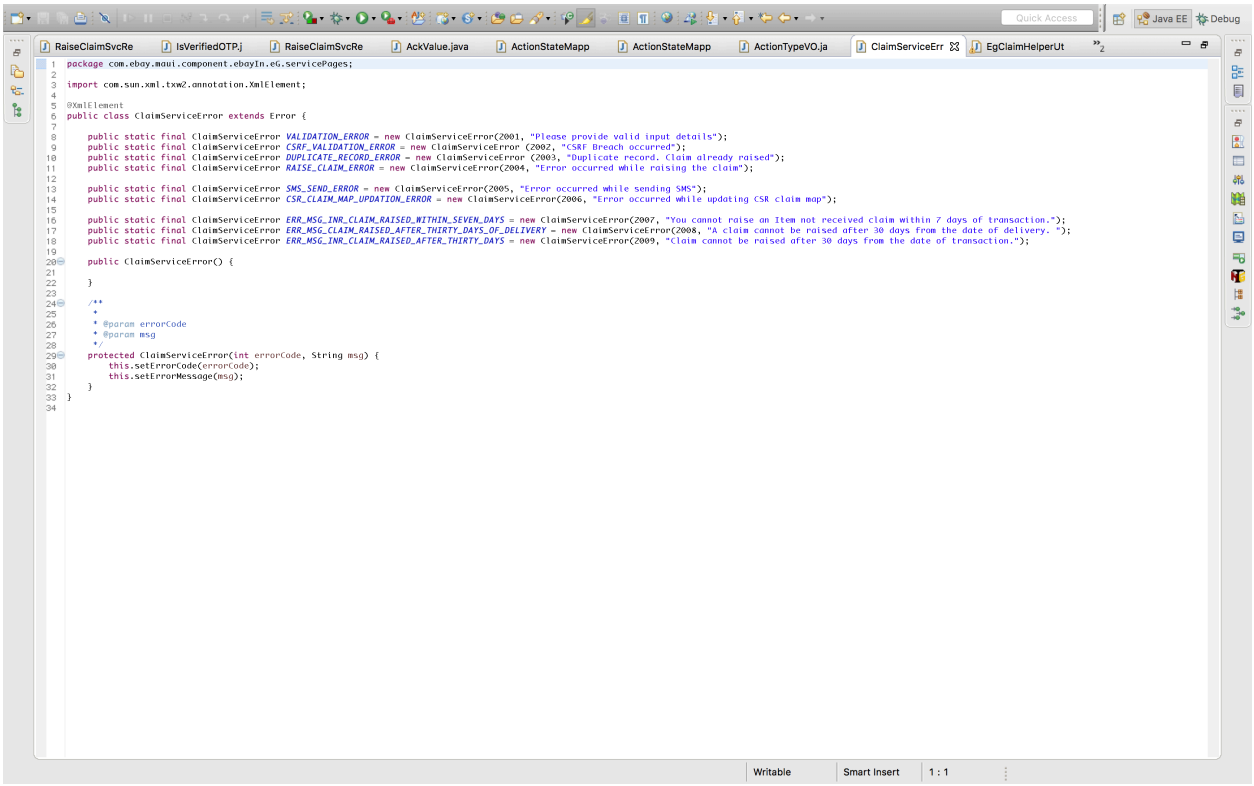

```java
package com.ebay.maui.component.ebayIn.eG.servicePages;

import java.util.ArrayList;

@XmlRootElement
public class ActionStateMappingResponse {

    AckValue ack = AckValue.FAILURE;

    public AckValue getAck() {
        return ack;
    }

    public void setAck(AckValue failure) {
        this.ack = failure;
    }

    @XmlElementWrapper(name = "errors")
    @XmlElement(name = "error")
    private List<Error> errors = new ArrayList<Error>();

    public List<Error> getErrors() {
        return errors;
    }

    @XmlElementWrapper(name = "stateMaster")
    @XmlElement(name = "stateMaster")
    private List<ActionStateMappingVO> actionStateMappingList = new ArrayList<ActionStateMappingVO>();

    public List<ActionStateMappingVO> getAttributesList() {
        return actionStateMappingList;
    }
}
```



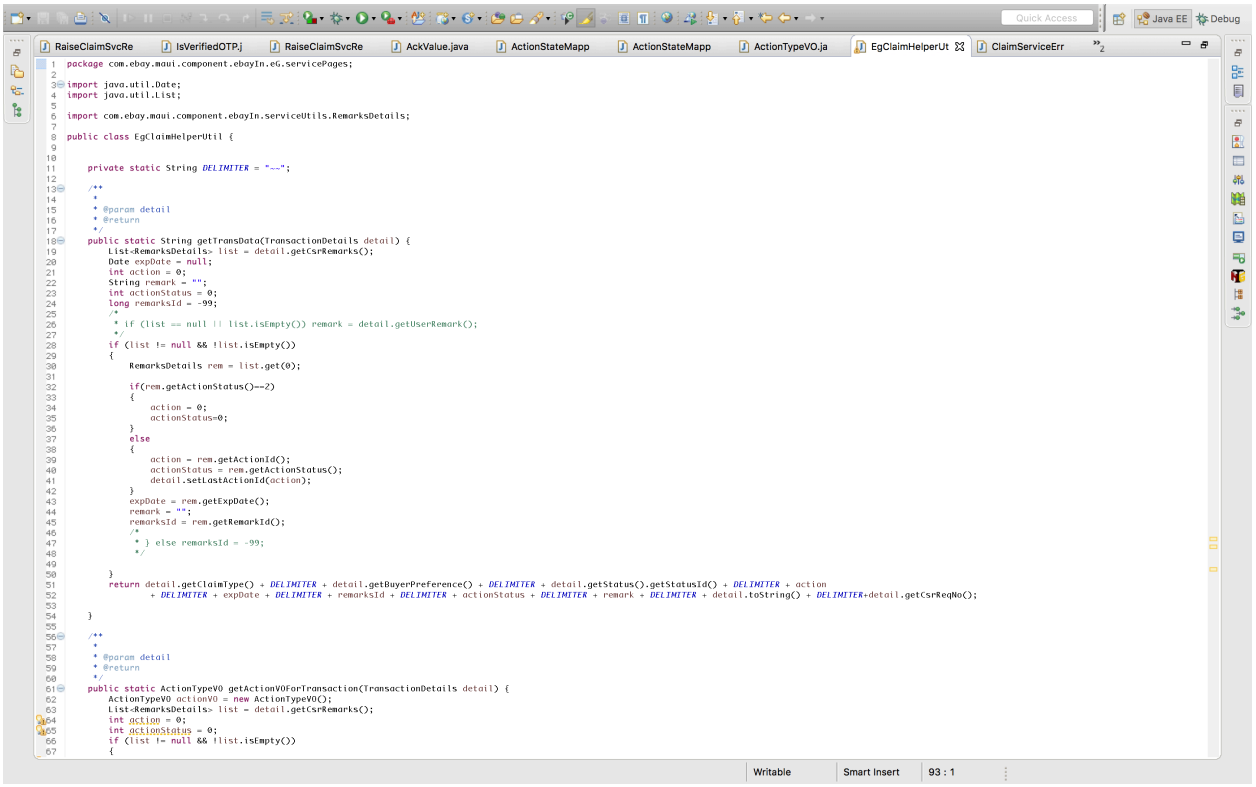

```java
package com.ebay.maui.component.ebayIn.eG.servicePages;


import com.sun.xml.txw2.annotation.XmlElement;

@XmlElement
public class ActionStateMappingVO{

    long mappingId;
    long claimStatusId;
    long issueTypeId;
    long actionId;

    public long getMappingId(){
        return mappingId;
    }

    public void setMappingId(long mappingId){
        this.mappingId = mappingId;
    }

    public long getClaimStatusId(){
        return claimStatusId;
    }

    public void setClaimStatusId(long claimStatusId){
        this.claimStatusId = claimStatusId ;

    }

    public long getIssueTypeId(){
        return issueTypeId;
    }

    public void setIssueTypeId(long issueTypeId){
        this.issueTypeId = issueTypeId;
    }

    public long getActionId(){
        return actionId;
    }

    public void setActionId(long actionId){
        this.actionId = actionId;

    }


}
```



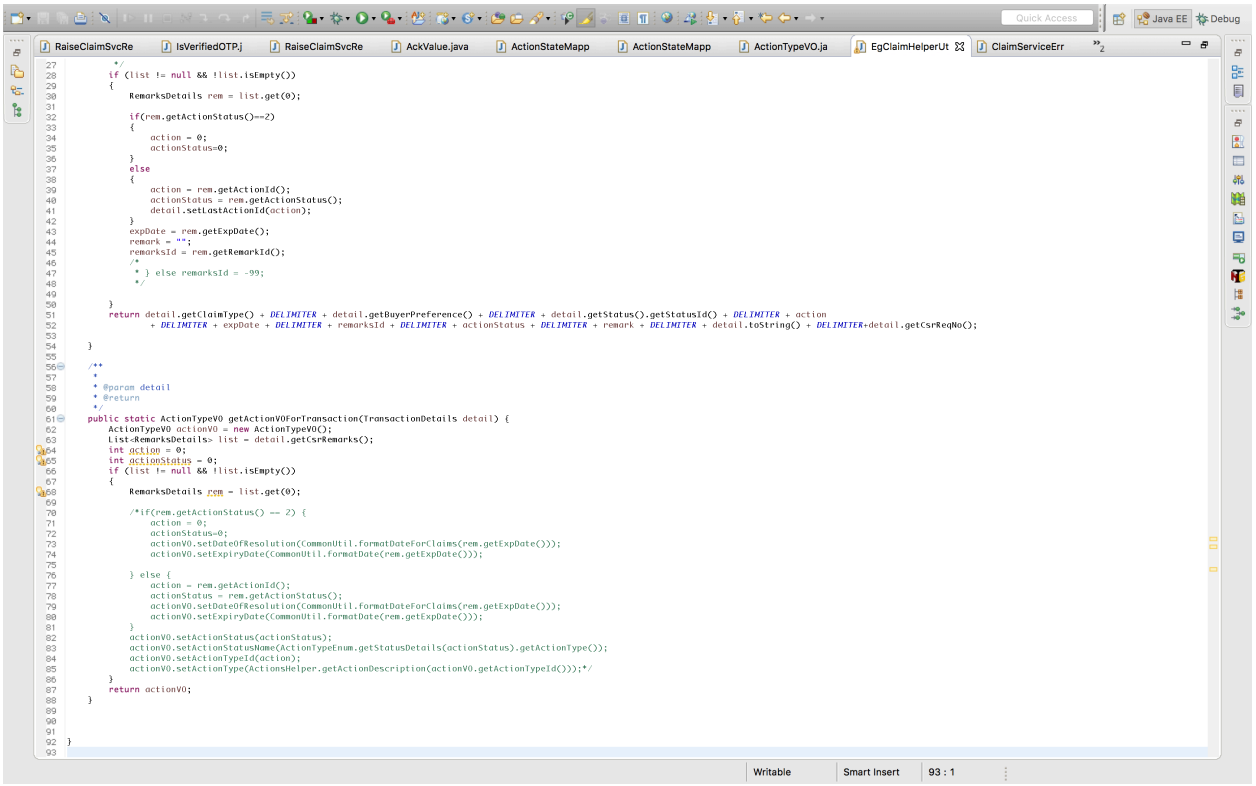
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

public class ActionTypeVO {
    private int actionTypeId;
    private String actionType;
    private int actionStatus;
    private String dateOfResolution;
    private String actionStatusName;
    private String expiryDate;

    public String getExpiryDate() {
        return expiryDate;
    }
    public void setExpiryDate(String expiryDate) {
        this.expiryDate = expiryDate;
    }
    public String getActionStatusName() {
        return actionStatusName;
    }
    public void setActionStatusName(String actionstatusName) {
        this.actionStatusName = actionstatusName;
    }
    public String getDateOfResolution() {
        return dateOfResolution;
    }
    public void setDateOfResolution(String dateOfResolution) {
        this.dateOfResolution = dateOfResolution;
    }
    public int getActionStatus() {
        return actionStatus;
    }
    public void setActionStatus(int actionStatus) {
        this.actionStatus = actionStatus;
    }
    public int getActionTypeId() {
        return actionTypeId;
    }
    public void setActionTypeId(int actionTypeId) {
        this.actionTypeId = actionTypeId;
    }
    public String getActionType() {
        return actionType;
    }
    public void setActionType(String actionType) {
        this.actionType = actionType;
    }
}
```



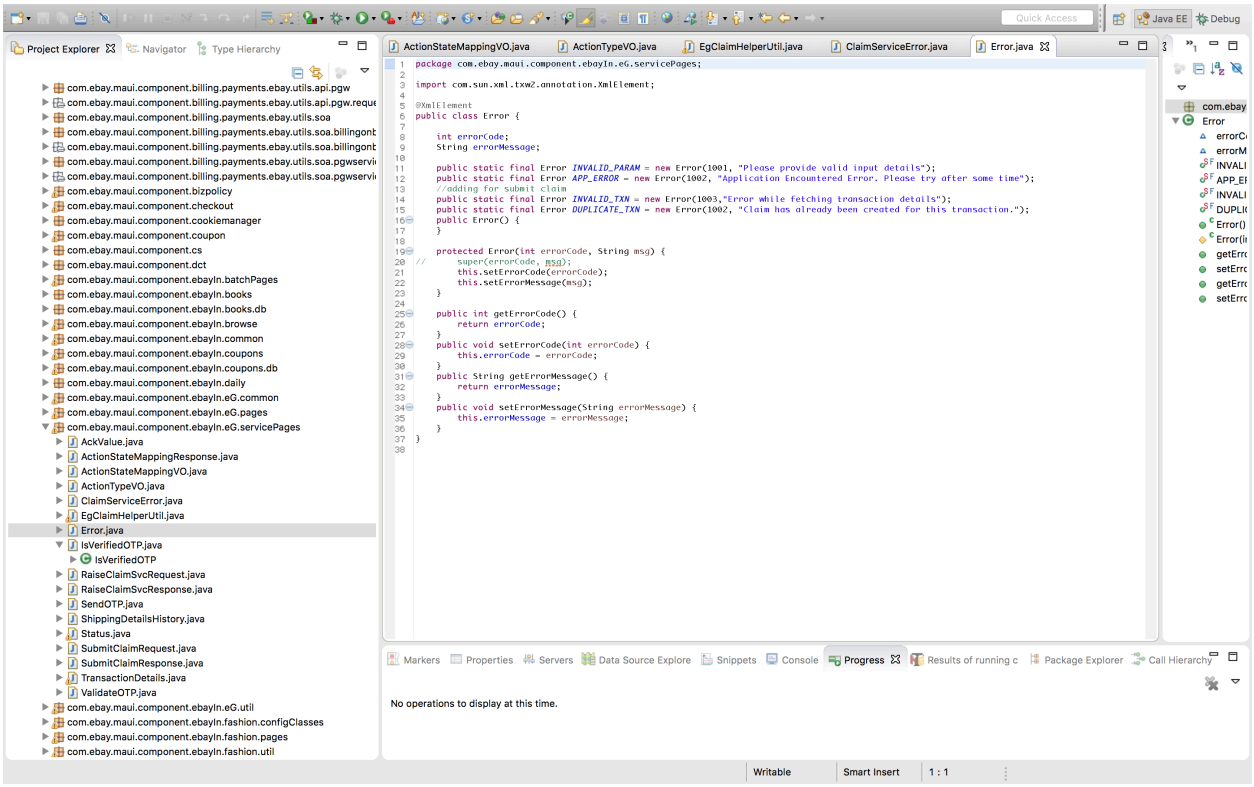
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

import com.sun.xml.txw2.annotation.XmlElement;

@XmlElement
public class ClaimServiceError extends Error {

    public static final ClaimServiceError VALIDATION_ERROR = new ClaimServiceError(2001, "Please provide valid input details");
    public static final ClaimServiceError CSRF_VALIDATION_ERROR = new ClaimServiceError (2002, "CSRF Breach occurred");
    public static final ClaimServiceError DUPLICATE_RECORD_ERROR = new ClaimServiceError (2003, "Duplicate record. Claim already raised");
    public static final ClaimServiceError RAISE_CLAIM_ERROR = new ClaimServiceError(2004, "Error occurred while raising the claim");

    public static final ClaimServiceError SMS_SEND_ERROR = new ClaimServiceError(2005, "Error occurred while sending SMS");
    public static final ClaimServiceError CSR_CLAIM_MAP_UPDATION_ERROR = new ClaimServiceError(2006, "Error occurred while updating CSR claim map");

    public static final ClaimServiceError ERR_MSG_INR_CLAIM_RAISED_WITHIN_SEVEN_DAYS = new ClaimServiceError(2007, "You cannot raise an Item not received claim within 7 days of transaction.");
    public static final ClaimServiceError ERR_MSG_CLAIM_RAISED_AFTER_THIRTY_DAYS_OF_DELIVERY = new ClaimServiceError(2008, "A claim cannot be raised after 30 days from the date of delivery. ");
    public static final ClaimServiceError ERR_MSG_INR_CLAIM_RAISED_AFTER_THIRTY_DAYS = new ClaimServiceError(2009, "Claim cannot be raised after 30 days from the date of transaction.");

    public ClaimServiceError() {

    }

    /**
     *
     * @param errorCode
     * @param msg
     */
    protected ClaimServiceError(int errorCode, String msg) {
        this.setErrorCode(errorCode);
        this.setErrorMessage(msg);
    }
}
```

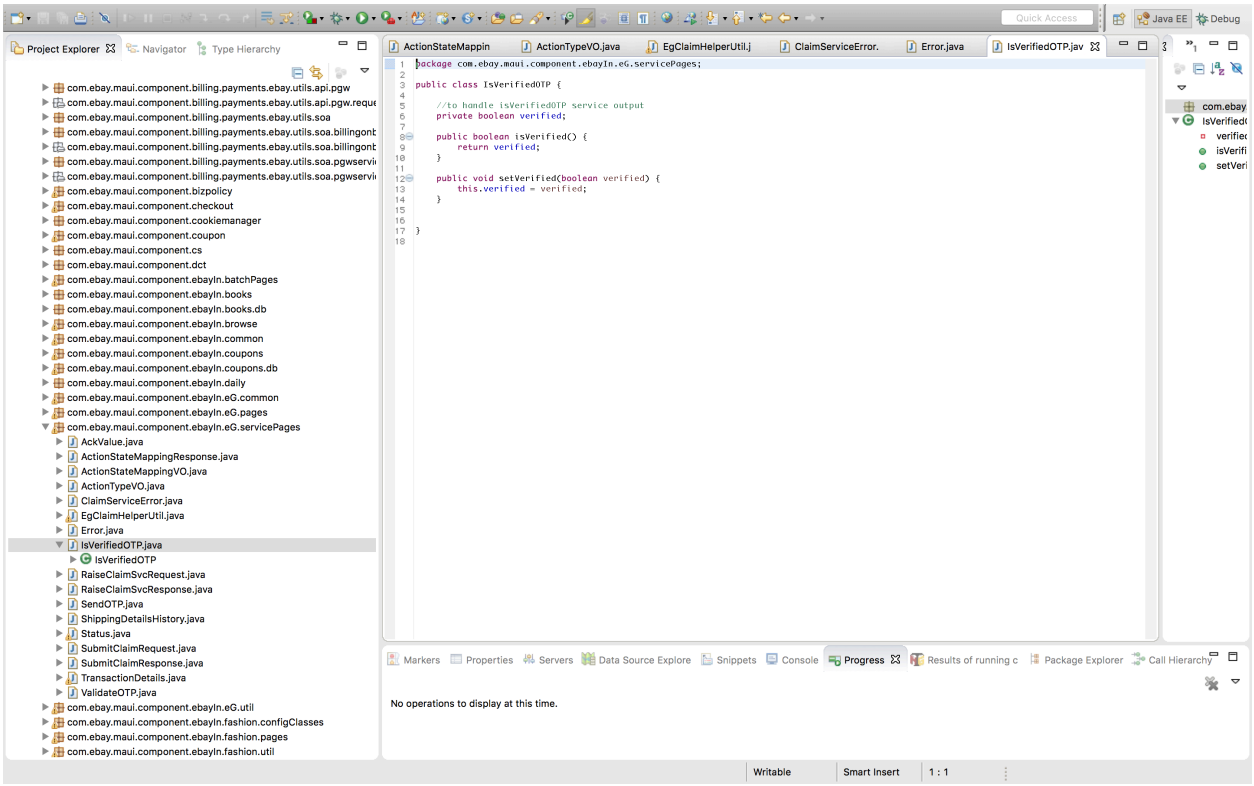


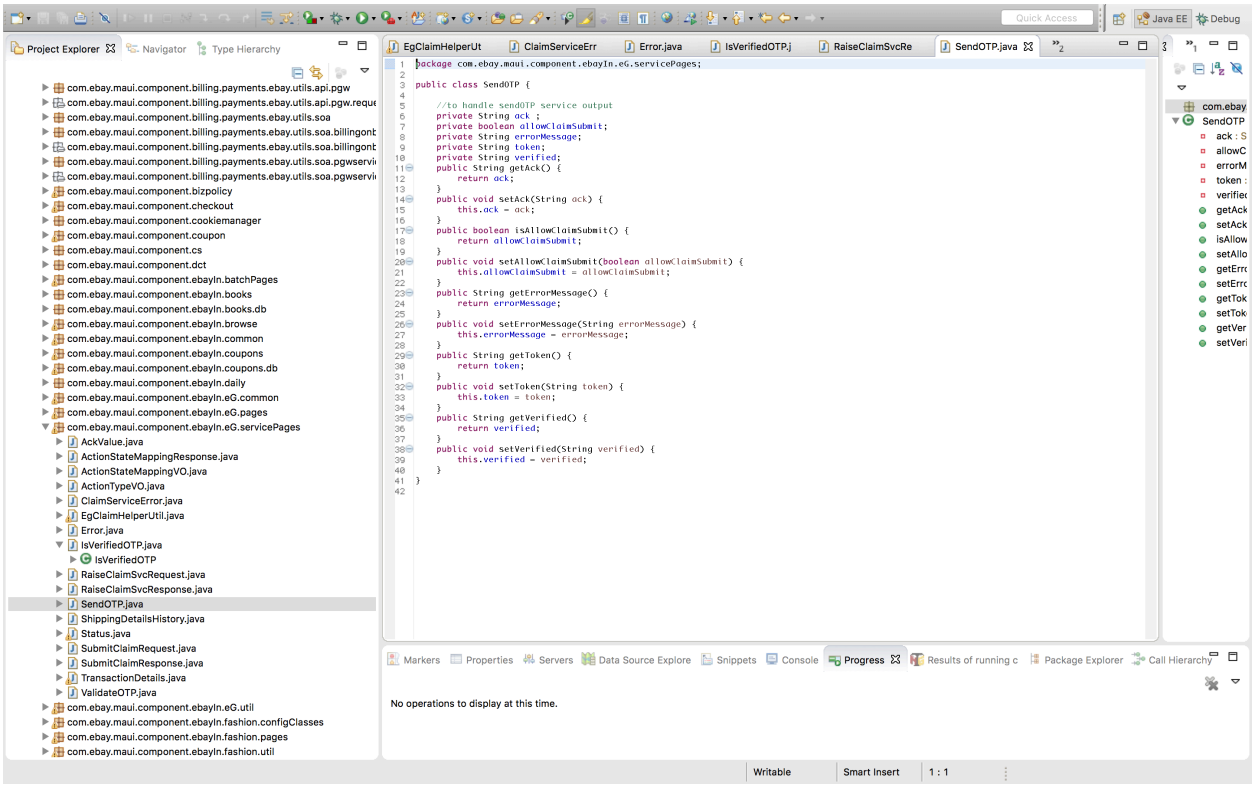
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

import java.util.Date;
import java.util.List;

import com.ebay.maui.component.ebayIn.serviceUtils.RemarksDetails;

public class EgClaimHelperUtil {


    private static String DELIMITER = "~~";

    /**
     *
     * @param detail
     * @return
     */
    public static String getTransData(TransactionDetails detail) {
        List<RemarksDetails> list = detail.getCsrRemarks();
        Date expDate = null;
        int action = 0;
        String remark = "";
        int actionStatus = 0;
        long remarksId = -99;
        /*
         * if (list == null || list.isEmpty()) remark = detail.getUserRemark();
         */
        if (list != null && !list.isEmpty())
        {
            RemarksDetails rem = list.get(0);

            if(rem.getActionStatus()==2)
            {
                action = 0;
                actionStatus=0;
            }
            else
            {
                action = rem.getActionId();
                actionStatus = rem.getActionStatus();
                detail.setLastActionId(action);
            }
            expDate = rem.getExpDate();
            remark = "";
            remarksId = rem.getRemarkId();
            /*
             * } else remarksId = -99;
             */

        }
        return detail.getClaimType() + DELIMITER + detail.getBuyerPreference() + DELIMITER + detail.getStatus().getStatusId() + DELIMITER + action
                + DELIMITER + expDate + DELIMITER + remarksId + DELIMITER + actionStatus + DELIMITER + remark + DELIMITER + detail.toString() + DELIMITER+detail.getCsrReqNo();

    }

    /**
     *
     * @param detail
     * @return
     */
    public static ActionTypeVO getActionVOForTransaction(TransactionDetails detail) {
        ActionTypeVO actionVO = new ActionTypeVO();
        List<RemarksDetails> list = detail.getCsrRemarks();
        int action = 0;
        int actionStatus = 0;
        if (list != null && !list.isEmpty())
        {
```



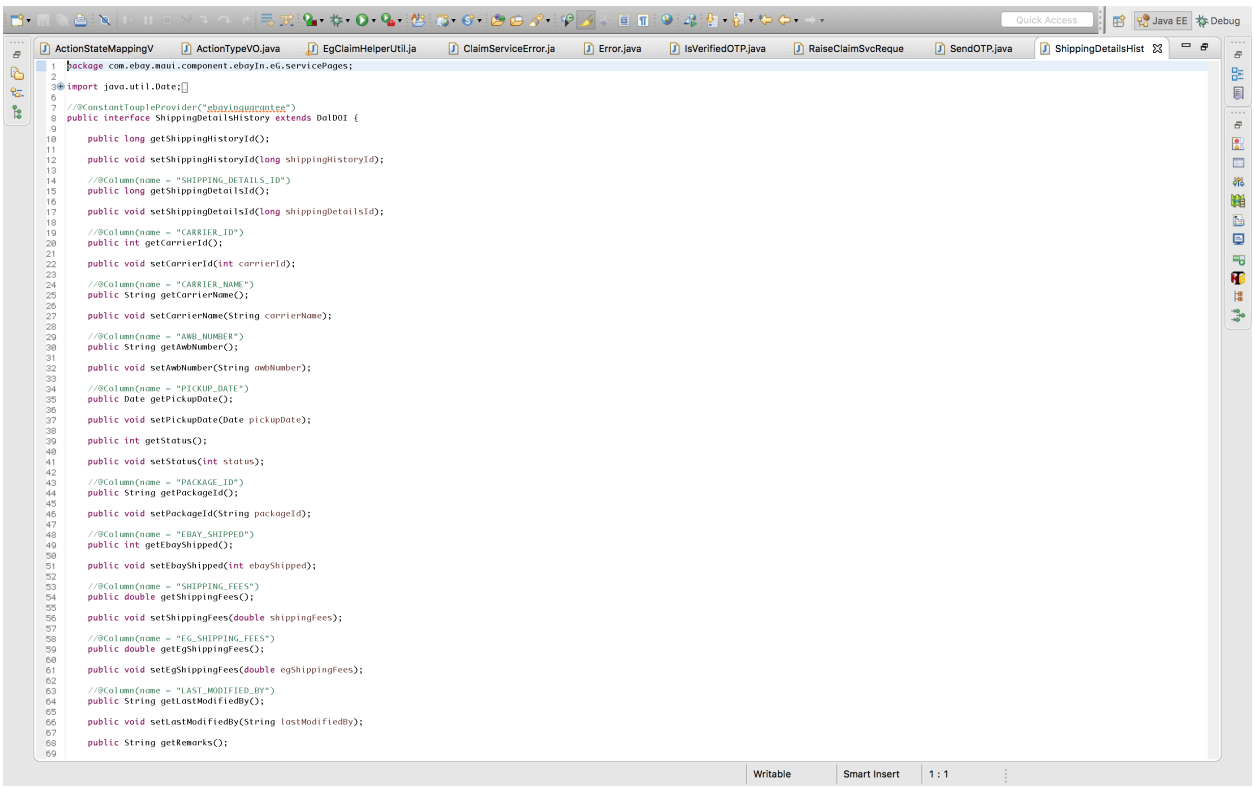
```
         */
        if (list != null && !list.isEmpty())
        {
            RemarksDetails rem = list.get(0);

            if(rem.getActionStatus()==2)
            {
                action = 0;
                actionStatus=0;
            }
            else
            {
                action = rem.getActionId();
                actionStatus = rem.getActionStatus();
                detail.setLastActionId(action);
            }
            expDate = rem.getExpDate();
            remark = "";
            remarksId = rem.getRemarkId();
            /*
             * } else remarksId = -99;
             */

        }
        return detail.getClaimType() + DELIMITER + detail.getBuyerPreference() + DELIMITER + detail.getStatus().getStatusId() + DELIMITER + action
                + DELIMITER + expDate + DELIMITER + remarksId + DELIMITER + actionStatus + DELIMITER + remark + DELIMITER + detail.toString() + DELIMITER+detail.getCsrReqNo();

    }

    /**
     *
     * @param detail
     * @return
     */
    public static ActionTypeVO getActionVOForTransaction(TransactionDetails detail) {
        ActionTypeVO actionVO = new ActionTypeVO();
        List<RemarksDetails> list = detail.getCsrRemarks();
        int action = 0;
        int actionStatus = 0;
        if (list != null && !list.isEmpty())
        {
            RemarksDetails rem = list.get(0);

            /*if(rem.getActionStatus() == 2) {
                action = 0;
                actionStatus=0;
                actionVO.setDateOfResolution(CommonUtil.formatDateForClaims(rem.getExpDate()));
                actionVO.setExpiryDate(CommonUtil.formatDate(rem.getExpDate()));

            } else {
                action = rem.getActionId();
                actionStatus = rem.getActionStatus();
                actionVO.setDateOfResolution(CommonUtil.formatDateForClaims(rem.getExpDate()));
                actionVO.setExpiryDate(CommonUtil.formatDate(rem.getExpDate()));
            }
            actionVO.setActionStatus(actionStatus);
            actionVO.setActionStatusName(ActionTypeEnum.getStatusDetails(actionStatus).getActionType());
            actionVO.setActionTypeId(action);
            actionVO.setActionType(ActionsHelper.getActionDescription(actionVO.getActionTypeId()));*/
        }
        return actionVO;
    }


}
```


Project Explorer
Navigator
Type Hierarchy
com.ebay.maui.component.billing.payments.ebay.utils.api.pgw
com.ebay.maui.component.billing.payments.ebay.utils.soa
com.ebay.maui.component.bizpolicy
com.ebay.maui.component.checkout
com.ebay.maui.component.cookiemanager
com.ebay.maui.component.coupon
com.ebay.maui.component.cs
com.ebay.maui.component.dct
com.ebay.maui.component.ebayIn.batchPages
com.ebay.maui.component.ebayIn.books
com.ebay.maui.component.ebayIn.books.db
com.ebay.maui.component.ebayIn.browse
com.ebay.maui.component.ebayIn.common
com.ebay.maui.component.ebayIn.coupons
com.ebay.maui.component.ebayIn.coupons.db
com.ebay.maui.component.ebayIn.daily
com.ebay.maui.component.ebayIn.eG.common
com.ebay.maui.component.ebayIn.eG.pages
com.ebay.maui.component.ebayIn.eG.servicePages
AckValue.java
ActionStateMappingResponse.java
ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
EgClaimHelperUtil.java
Error.java
IsVerifiedOTP.java
IsVerifiedOTP
RaiseClaimSvcRequest.java
RaiseClaimSvcResponse.java
SendOTP.java
ShippingDetailsHistory.java
Status.java
SubmitClaimRequest.java
SubmitClaimResponse.java
TransactionDetails.java
ValidateOTP.java
com.ebay.maui.component.ebayIn.eG.util
com.ebay.maui.component.ebayIn.fashion.configClasses
com.ebay.maui.component.ebayIn.fashion.pages
com.ebay.maui.component.ebayIn.fashion.util
ActionStateMappingVO.java
ActionTypeVO.java
EgClaimHelperUtil.java
ClaimServiceError.java
Error.java
1 package com.ebay.maui.component.ebayIn.eG.servicePages;
2
3 import com.sun.xml.txw2.annotation.XmlElement;
4
5 @XmlElement
6 public class Error {
7
8 int errorCode;
9 String errorMessage;
10
11 public static final Error INVALID_PARAM = new Error(1001, "Please provide valid input details");
12 public static final Error APP_ERROR = new Error(1002, "Application Encountered Error. Please try after some time");
13 //adding for submit claim
14 public static final Error INVALID_TXN = new Error(1003,"Error while fetching transaction details");
15 public static final Error DUPLICATE_TXN = new Error(1002, "Claim has already been created for this transaction.");
16 public Error() {
17 }
18
19 protected Error(int errorCode, String msg) {
20 // super(errorCode, msg);
21 this.setErrorCode(errorCode);
22 this.setErrorMessage(msg);
23 }
24
25 public int getErrorCode() {
26 return errorCode;
27 }
28 public void setErrorCode(int errorCode) {
29 this.errorCode = errorCode;
30 }
31 public String getErrorMessage() {
32 return errorMessage;
33 }
34 public void setErrorMessage(String errorMessage) {
35 this.errorMessage = errorMessage;
36 }
37 }
38
Markers
Properties
Servers
Data Source Explore
Snippets
Console
Progress
Results of running c
Package Explorer
Call Hierarchy
No operations to display at this time.
Writable
Smart Insert
1 : 1

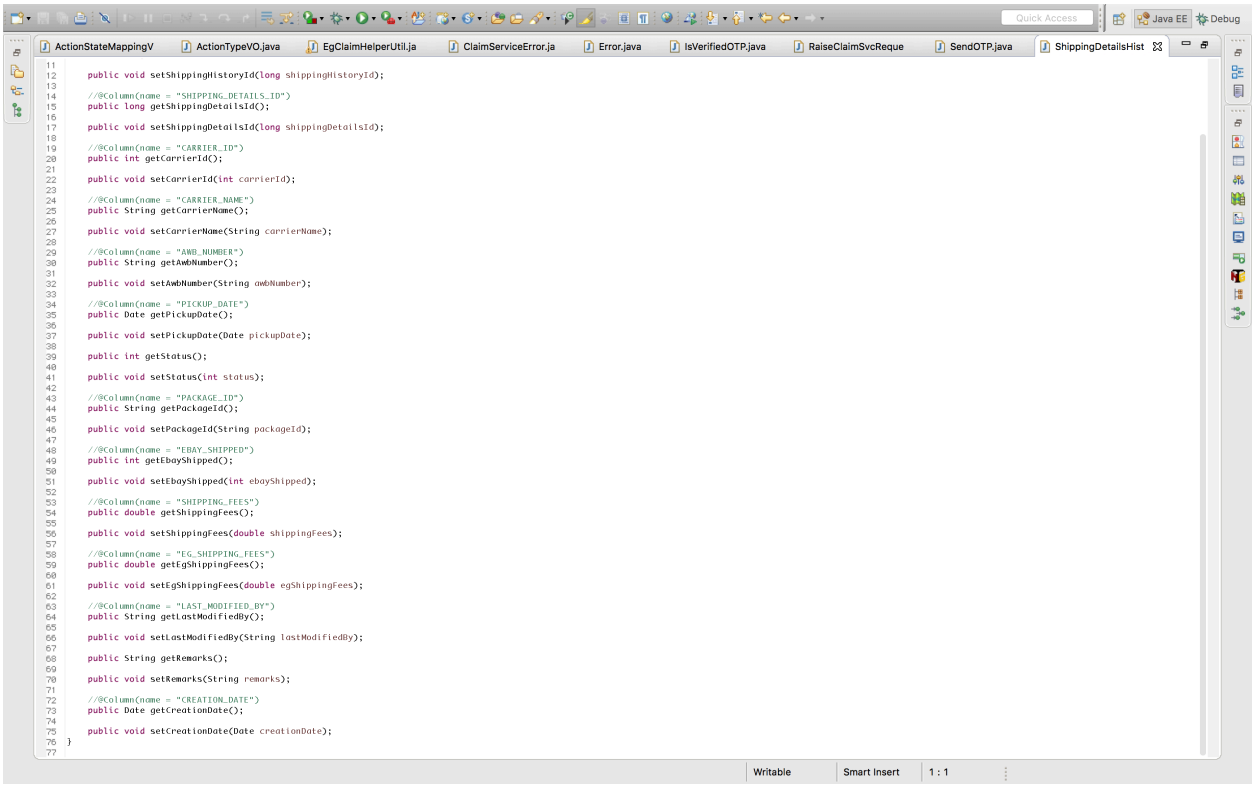

Project Explorer
Navigator
Type Hierarchy
com.ebay.maui.component.billing.payments.ebay.utils.api.pgw
com.ebay.maui.component.billing.payments.ebay.utils.soa
com.ebay.maui.component.bizpolicy
com.ebay.maui.component.checkout
com.ebay.maui.component.cookiemanager
com.ebay.maui.component.coupon
com.ebay.maui.component.cs
com.ebay.maui.component.dct
com.ebay.maui.component.ebayIn.batchPages
com.ebay.maui.component.ebayIn.books
com.ebay.maui.component.ebayIn.books.db
com.ebay.maui.component.ebayIn.browse
com.ebay.maui.component.ebayIn.common
com.ebay.maui.component.ebayIn.coupons
com.ebay.maui.component.ebayIn.coupons.db
com.ebay.maui.component.ebayIn.daily
com.ebay.maui.component.ebayIn.eG.common
com.ebay.maui.component.ebayIn.eG.pages
com.ebay.maui.component.ebayIn.eG.servicePages
AckValue.java
ActionStateMappingResponse.java
ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
EgClaimHelperUtil.java
Error.java
IsVerifiedOTP.java
IsVerifiedOTP
RaiseClaimSvcRequest.java
RaiseClaimSvcResponse.java
SendOTP.java
ShippingDetailsHistory.java
Status.java
SubmitClaimRequest.java
SubmitClaimResponse.java
TransactionDetails.java
ValidateOTP.java
com.ebay.maui.component.ebayIn.eG.util
com.ebay.maui.component.ebayIn.fashion.configClasses
com.ebay.maui.component.ebayIn.fashion.pages
com.ebay.maui.component.ebayIn.fashion.util
ActionTypeVO.java
EgClaimHelperUtil.j
ClaimServiceError.
Error.java
IsVerifiedOTP.jav
1 package com.ebay.maui.component.ebayIn.eG.servicePages;
2
3 public class IsVerifiedOTP {
4
5 //to handle isVerifiedOTP service output
6 private boolean verified;
7
8 public boolean isVerified() {
9 return verified;
10 }
11
12 public void setVerified(boolean verified) {
13 this.verified = verified;
14 }
15
16
17 }
18
Markers
Properties
Servers
Data Source Explore
Snippets
Console
Progress
Results of running c
Package Explorer
Call Hierarchy
No operations to display at this time.
Writable
Smart Insert
1 : 1



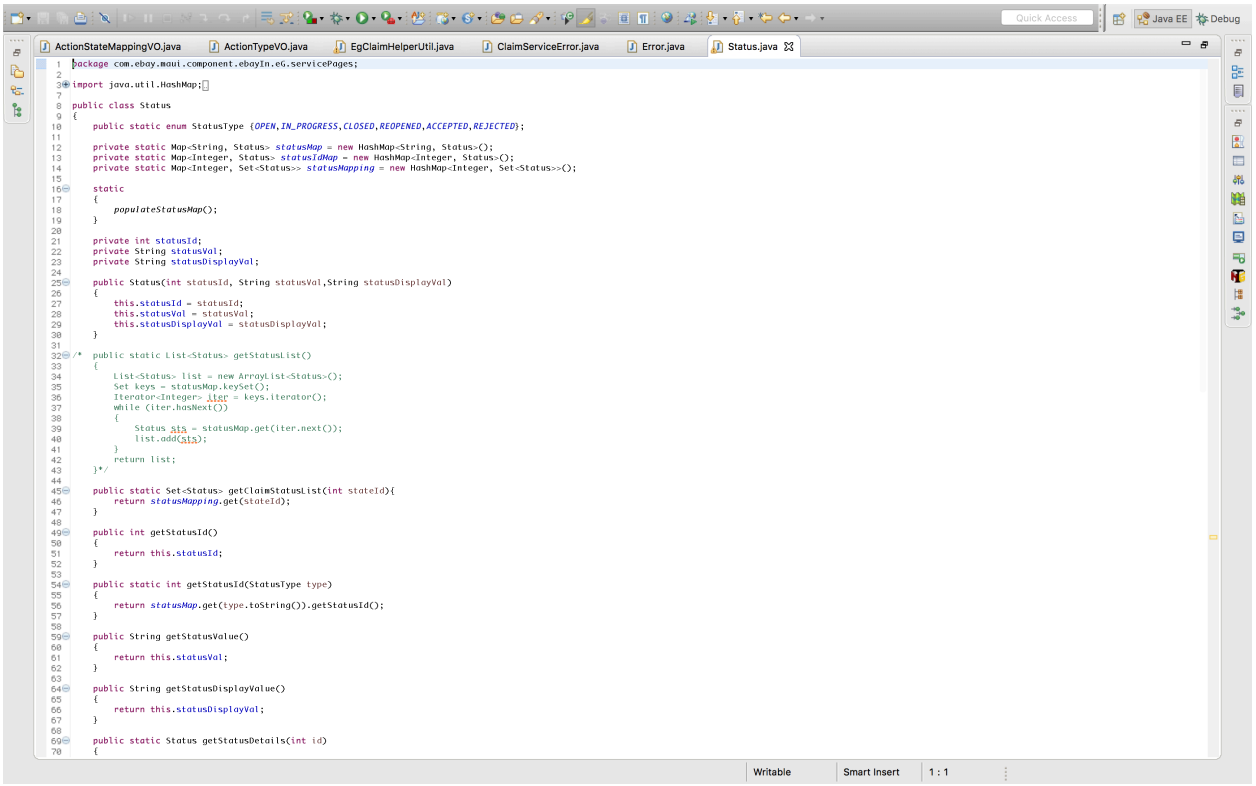
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

public class SendOTP {

    //to handle sendOTP service output
    private String ack ;
    private boolean allowClaimSubmit;
    private String errorMessage;
    private String token;
    private String verified;
    public String getAck() {
        return ack;
    }
    public void setAck(String ack) {
        this.ack = ack;
    }
    public boolean isAllowClaimSubmit() {
        return allowClaimSubmit;
    }
    public void setAllowClaimSubmit(boolean allowClaimSubmit) {
        this.allowClaimSubmit = allowClaimSubmit;
    }
    public String getErrorMessage() {
        return errorMessage;
    }
    public void setErrorMessage(String errorMessage) {
        this.errorMessage = errorMessage;
    }
    public String getToken() {
        return token;
    }
    public void setToken(String token) {
        this.token = token;
    }
    public String getVerified() {
        return verified;
    }
    public void setVerified(String verified) {
        this.verified = verified;
    }
}
```



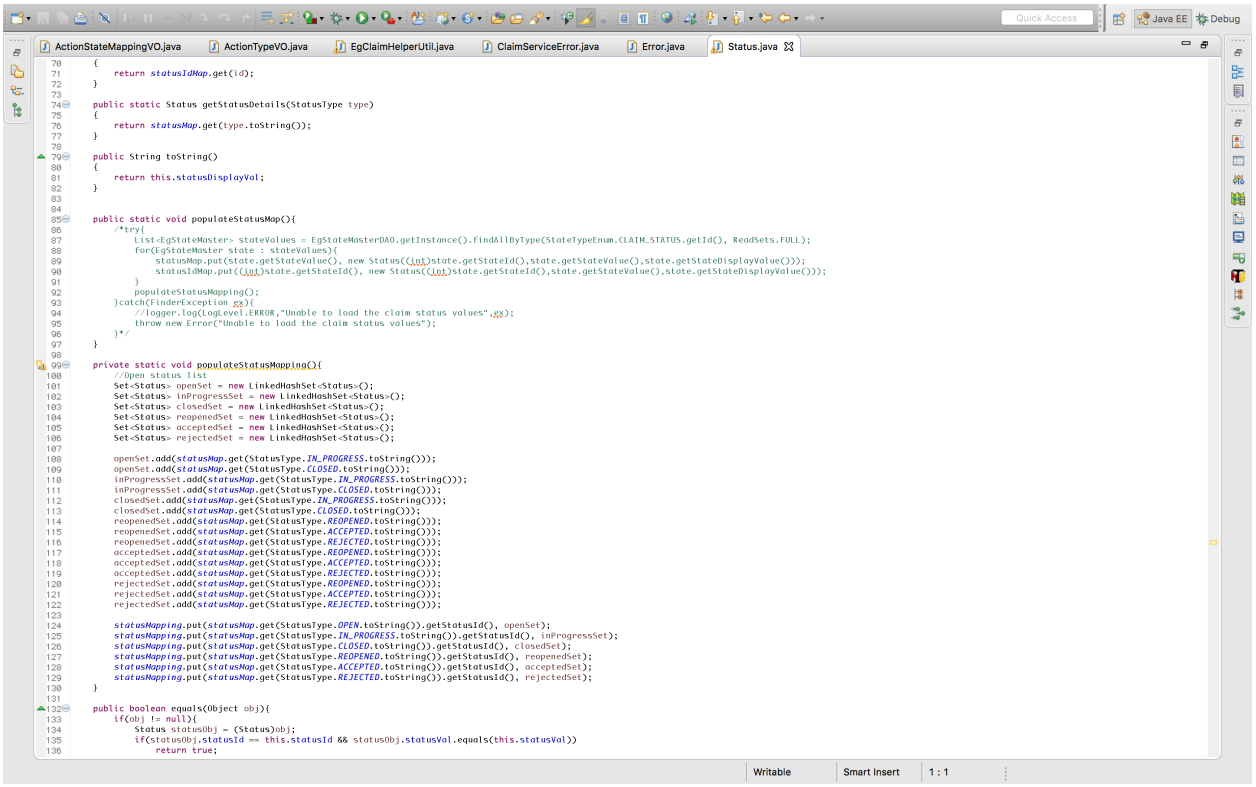
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

import java.util.Date;

//@ConstantTupleProvider("ebayinguarantee")
public interface ShippingDetailsHistory extends DalDOI {

    public long getShippingHistoryId();

    public void setShippingHistoryId(long shippingHistoryId);

    //@Column(name = "SHIPPING_DETAILS_ID")
    public long getShippingDetailsId();

    public void setShippingDetailsId(long shippingDetailsId);

    //@Column(name = "CARRIER_ID")
    public int getCarrierId();

    public void setCarrierId(int carrierId);

    //@Column(name = "CARRIER_NAME")
    public String getCarrierName();

    public void setCarrierName(String carrierName);

    //@Column(name = "AWB_NUMBER")
    public String getAwbNumber();

    public void setAwbNumber(String awbNumber);

    //@Column(name = "PICKUP_DATE")
    public Date getPickupDate();

    public void setPickupDate(Date pickupDate);

    public int getStatus();

    public void setStatus(int status);

    //@Column(name = "PACKAGE_ID")
    public String getPackageId();

    public void setPackageId(String packageId);

    //@Column(name = "EBAY_SHIPPED")
    public int getEbayShipped();

    public void setEbayShipped(int ebayShipped);

    //@Column(name = "SHIPPING_FEES")
    public double getShippingFees();

    public void setShippingFees(double shippingFees);

    //@Column(name = "EG_SHIPPING_FEES")
    public double getEgShippingFees();

    public void setEgShippingFees(double egShippingFees);

    //@Column(name = "LAST_MODIFIED_BY")
    public String getLastModifiedBy();

    public void setLastModifiedBy(String lastModifiedBy);

    public String getRemarks();
```



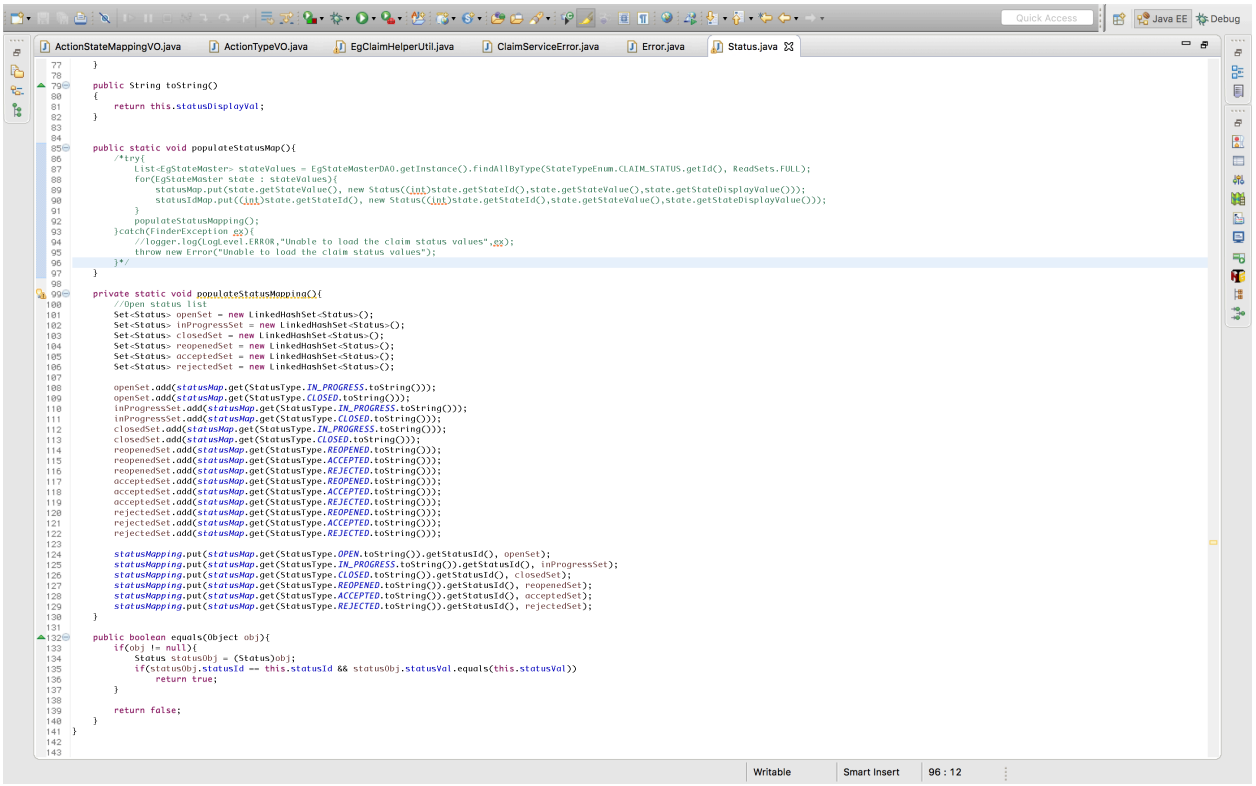
```
    public void setShippingHistoryId(long shippingHistoryId);

    //@Column(name = "SHIPPING_DETAILS_ID")
    public long getShippingDetailsId();

    public void setShippingDetailsId(long shippingDetailsId);

    //@Column(name = "CARRIER_ID")
    public int getCarrierId();

    public void setCarrierId(int carrierId);

    //@Column(name = "CARRIER_NAME")
    public String getCarrierName();

    public void setCarrierName(String carrierName);

    //@Column(name = "AWB_NUMBER")
    public String getAwbNumber();

    public void setAwbNumber(String awbNumber);

    //@Column(name = "PICKUP_DATE")
    public Date getPickupDate();

    public void setPickupDate(Date pickupDate);

    public int getStatus();

    public void setStatus(int status);

    //@Column(name = "PACKAGE_ID")
    public String getPackageId();

    public void setPackageId(String packageId);

    //@Column(name = "EBAY_SHIPPED")
    public int getEbayShipped();

    public void setEbayShipped(int ebayShipped);

    //@Column(name = "SHIPPING_FEES")
    public double getShippingFees();

    public void setShippingFees(double shippingFees);

    //@Column(name = "EG_SHIPPING_FEES")
    public double getEgShippingFees();

    public void setEgShippingFees(double egShippingFees);

    //@Column(name = "LAST_MODIFIED_BY")
    public String getLastModifiedBy();

    public void setLastModifiedBy(String lastModifiedBy);

    public String getRemarks();

    public void setRemarks(String remarks);

    //@Column(name = "CREATION_DATE")
    public Date getCreationDate();

    public void setCreationDate(Date creationDate);
}
```



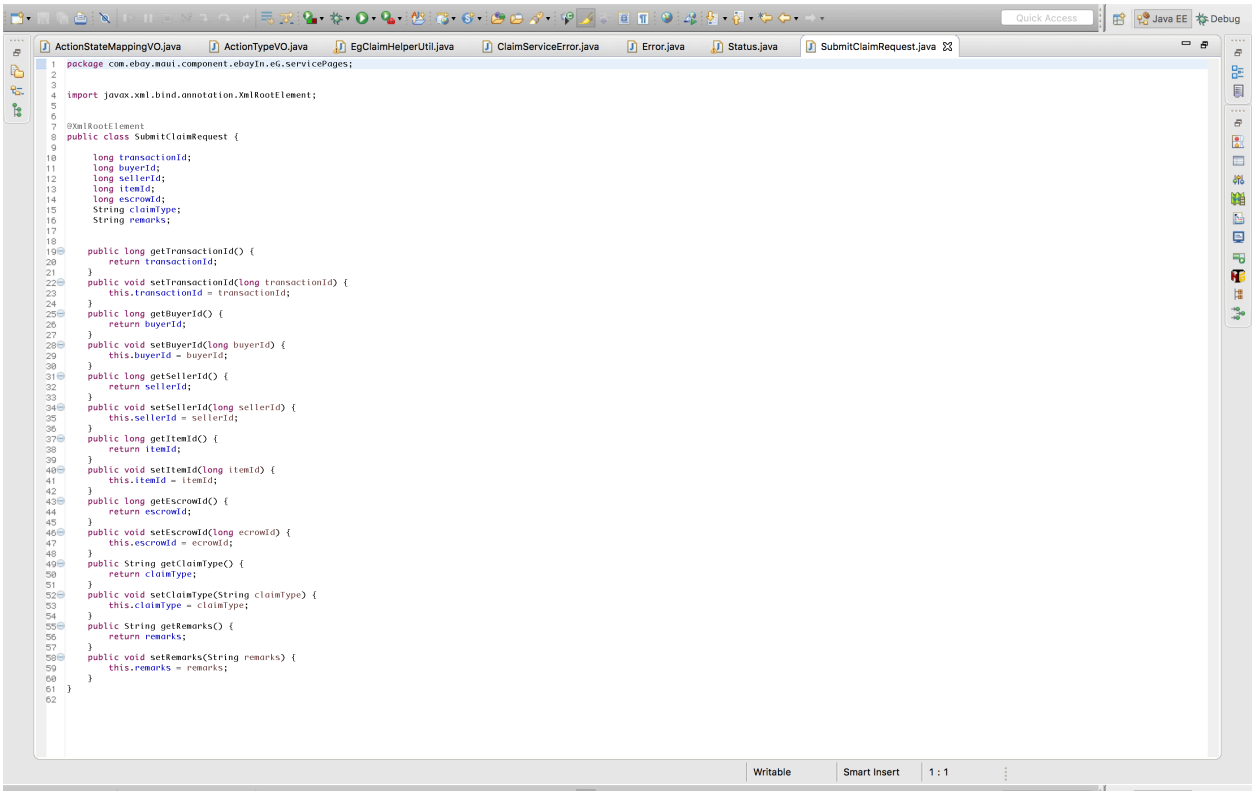
```
package com.ebay.maui.component.ebayIn.eG.servicePages;

import java.util.HashMap;

public class Status
{
    public static enum StatusType {OPEN,IN_PROGRESS,CLOSED,REOPENED,ACCEPTED,REJECTED};

    private static Map<String, Status> statusMap = new HashMap<String, Status>();
    private static Map<Integer, Status> statusIdMap = new HashMap<Integer, Status>();
    private static Map<Integer, Set<Status>> statusMapping = new HashMap<Integer, Set<Status>>();

    static
    {
        populateStatusMap();
    }

    private int statusId;
    private String statusVal;
    private String statusDisplayVal;

    public Status(int statusId, String statusVal,String statusDisplayVal)
    {
        this.statusId = statusId;
        this.statusVal = statusVal;
        this.statusDisplayVal = statusDisplayVal;
    }

/*  public static List<Status> getStatusList()
    {
        List<Status> list = new ArrayList<Status>();
        Set keys = statusMap.keySet();
        Iterator<Integer> iter = keys.iterator();
        while (iter.hasNext())
        {
            Status sts = statusMap.get(iter.next());
            list.add(sts);
        }
        return list;
    }*/

    public static Set<Status> getClaimStatusList(int stateId){
        return statusMapping.get(stateId);
    }

    public int getStatusId()
    {
        return this.statusId;
    }

    public static int getStatusId(StatusType type)
    {
        return statusMap.get(type.toString()).getStatusId();
    }

    public String getStatusValue()
    {
        return this.statusVal;
    }

    public String getStatusDisplayValue()
    {
        return this.statusDisplayVal;
    }

    public static Status getStatusDetails(int id)
    {
```



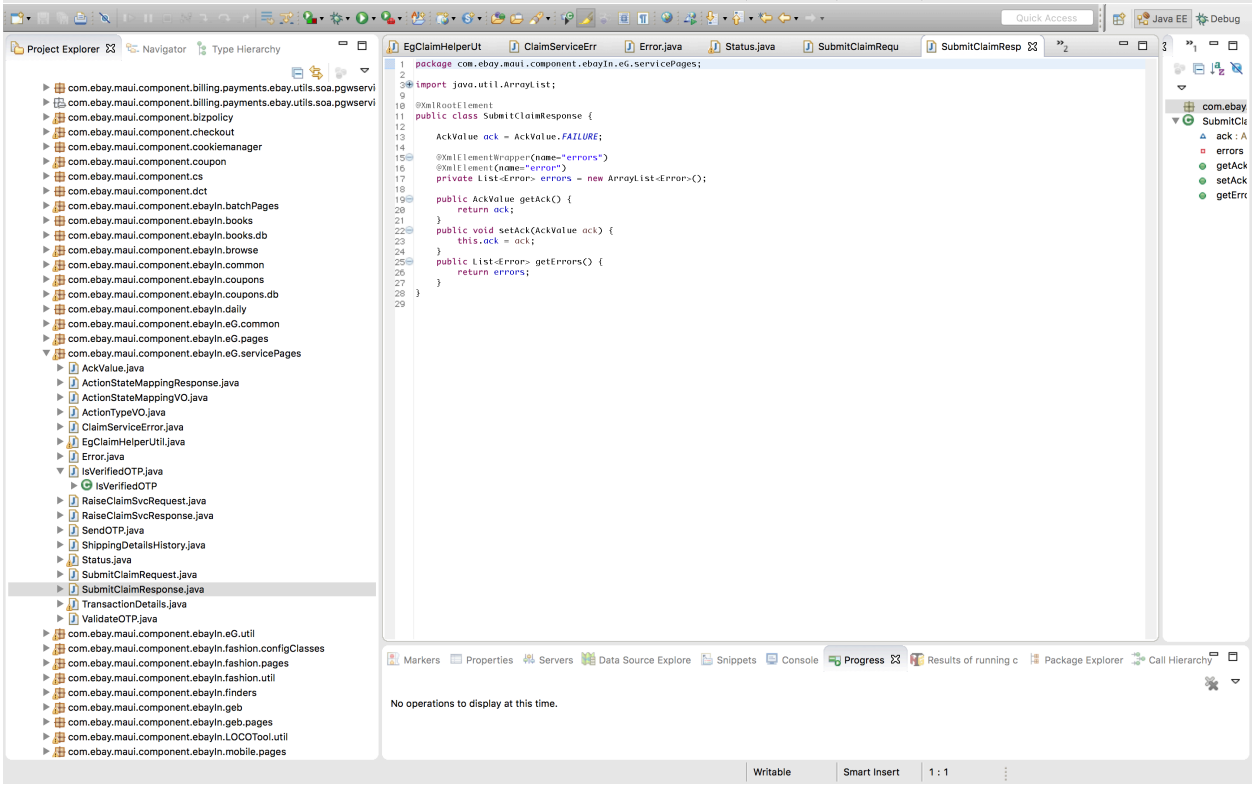
```
    {
        return statusIdMap.get(id);
    }

    public static Status getStatusDetails(StatusType type)
    {
        return statusMap.get(type.toString());
    }

    public String toString()
    {
        return this.statusDisplayVal;
    }


    public static void populateStatusMap(){
        /*try{
            List<EgStateMaster> stateValues = EgStateMasterDAO.getInstance().findAllByType(StateTypeEnum.CLAIM_STATUS.getId(), ReadSets.FULL);
            for(EgStateMaster state : stateValues){
                statusMap.put(state.getStateValue(), new Status((int)state.getStateId(),state.getStateValue(),state.getStateDisplayValue()));
                statusIdMap.put((int)state.getStateId(), new Status((int)state.getStateId(),state.getStateValue(),state.getStateDisplayValue()));
            }
            populateStatusMapping();
        }catch(FinderException ex){
            //logger.log(LogLevel.ERROR,"Unable to load the claim status values",ex);
            throw new Error("Unable to load the claim status values");
        }*/
    }

    private static void populateStatusMapping(){
        //Open status list
        Set<Status> openSet = new LinkedHashSet<Status>();
        Set<Status> inProgressSet = new LinkedHashSet<Status>();
        Set<Status> closedSet = new LinkedHashSet<Status>();
        Set<Status> reopenedSet = new LinkedHashSet<Status>();
        Set<Status> acceptedSet = new LinkedHashSet<Status>();
        Set<Status> rejectedSet = new LinkedHashSet<Status>();

        openSet.add(statusMap.get(StatusType.IN_PROGRESS.toString()));
        openSet.add(statusMap.get(StatusType.CLOSED.toString()));
        inProgressSet.add(statusMap.get(StatusType.IN_PROGRESS.toString()));
        inProgressSet.add(statusMap.get(StatusType.CLOSED.toString()));
        closedSet.add(statusMap.get(StatusType.IN_PROGRESS.toString()));
        closedSet.add(statusMap.get(StatusType.CLOSED.toString()));
        reopenedSet.add(statusMap.get(StatusType.REOPENED.toString()));
        reopenedSet.add(statusMap.get(StatusType.ACCEPTED.toString()));
        reopenedSet.add(statusMap.get(StatusType.REJECTED.toString()));
        acceptedSet.add(statusMap.get(StatusType.REOPENED.toString()));
        acceptedSet.add(statusMap.get(StatusType.ACCEPTED.toString()));
        acceptedSet.add(statusMap.get(StatusType.REJECTED.toString()));
        rejectedSet.add(statusMap.get(StatusType.REOPENED.toString()));
        rejectedSet.add(statusMap.get(StatusType.ACCEPTED.toString()));
        rejectedSet.add(statusMap.get(StatusType.REJECTED.toString()));

        statusMapping.put(statusMap.get(StatusType.OPEN.toString()).getStatusId(), openSet);
        statusMapping.put(statusMap.get(StatusType.IN_PROGRESS.toString()).getStatusId(), inProgressSet);
        statusMapping.put(statusMap.get(StatusType.CLOSED.toString()).getStatusId(), closedSet);
        statusMapping.put(statusMap.get(StatusType.REOPENED.toString()).getStatusId(), reopenedSet);
        statusMapping.put(statusMap.get(StatusType.ACCEPTED.toString()).getStatusId(), acceptedSet);
        statusMapping.put(statusMap.get(StatusType.REJECTED.toString()).getStatusId(), rejectedSet);
    }

    public boolean equals(Object obj){
        if(obj != null){
            Status statusObj = (Status)obj;
            if(statusObj.statusId == this.statusId && statusObj.statusVal.equals(this.statusVal))
                return true;
```



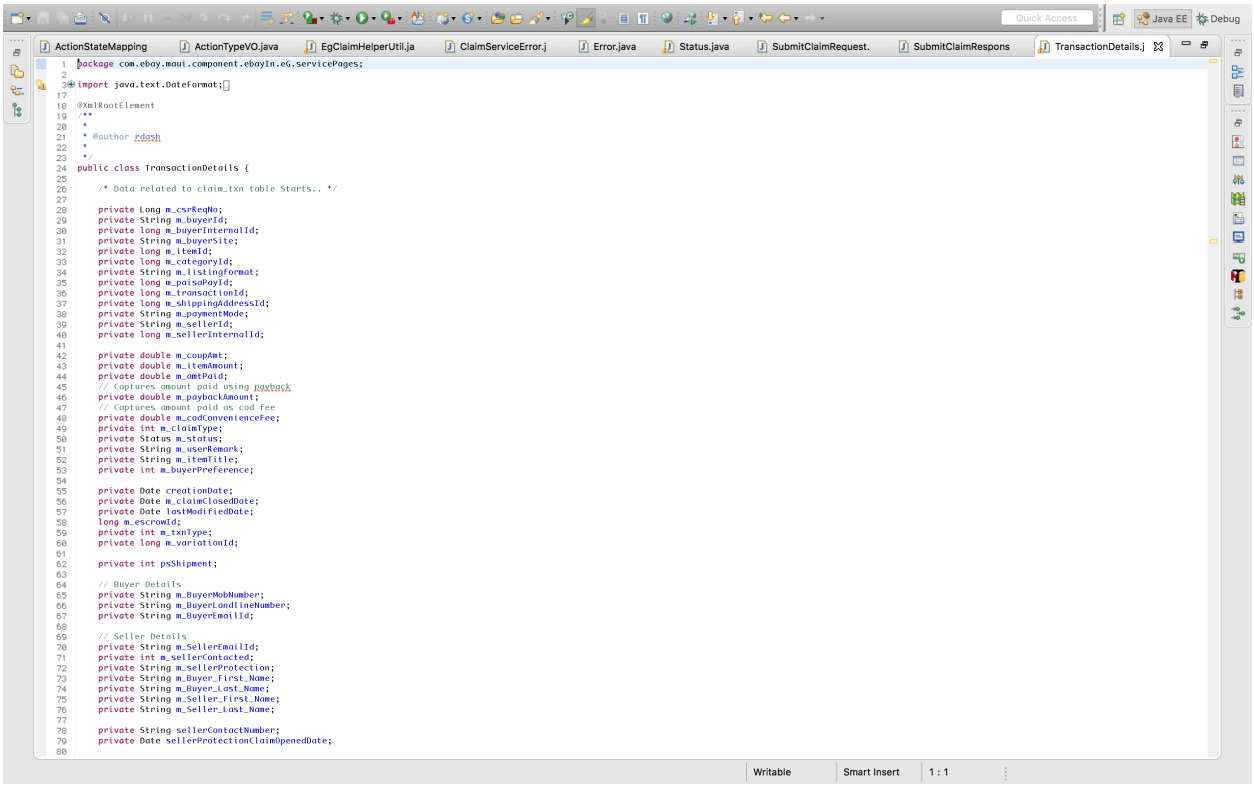
```
    }

    public String toString()
    {
        return this.statusDisplayVal;
    }


    public static void populateStatusMap(){
        /*try{
            List<EgStateMaster> stateValues = EgStateMasterDAO.getInstance().findAllByType(StateTypeEnum.CLAIM_STATUS.getId(), ReadSets.FULL);
            for(EgStateMaster state : stateValues){
                statusMap.put(state.getStateValue(), new Status((int)state.getStateId(),state.getStateValue(),state.getStateDisplayValue()));
                statusIdMap.put((int)state.getStateId(), new Status((int)state.getStateId(),state.getStateValue(),state.getStateDisplayValue()));
            }
            populateStatusMapping();
        }catch(FinderException ex){
            //logger.log(LogLevel.ERROR,"Unable to load the claim status values",ex);
            throw new Error("Unable to load the claim status values");
        }*/
    }

    private static void populateStatusMapping(){
        //Open status list
        Set<Status> openSet = new LinkedHashSet<Status>();
        Set<Status> inProgressSet = new LinkedHashSet<Status>();
        Set<Status> closedSet = new LinkedHashSet<Status>();
        Set<Status> reopenedSet = new LinkedHashSet<Status>();
        Set<Status> acceptedSet = new LinkedHashSet<Status>();
        Set<Status> rejectedSet = new LinkedHashSet<Status>();

        openSet.add(statusMap.get(StatusType.IN_PROGRESS.toString()));
        openSet.add(statusMap.get(StatusType.CLOSED.toString()));
        inProgressSet.add(statusMap.get(StatusType.IN_PROGRESS.toString()));
        inProgressSet.add(statusMap.get(StatusType.CLOSED.toString()));
        closedSet.add(statusMap.get(StatusType.IN_PROGRESS.toString()));
        closedSet.add(statusMap.get(StatusType.CLOSED.toString()));
        reopenedSet.add(statusMap.get(StatusType.REOPENED.toString()));
        reopenedSet.add(statusMap.get(StatusType.ACCEPTED.toString()));
        reopenedSet.add(statusMap.get(StatusType.REJECTED.toString()));
        acceptedSet.add(statusMap.get(StatusType.REOPENED.toString()));
        acceptedSet.add(statusMap.get(StatusType.ACCEPTED.toString()));
        acceptedSet.add(statusMap.get(StatusType.REJECTED.toString()));
        rejectedSet.add(statusMap.get(StatusType.REOPENED.toString()));
        rejectedSet.add(statusMap.get(StatusType.ACCEPTED.toString()));
        rejectedSet.add(statusMap.get(StatusType.REJECTED.toString()));

        statusMapping.put(statusMap.get(StatusType.OPEN.toString()).getStatusId(), openSet);
        statusMapping.put(statusMap.get(StatusType.IN_PROGRESS.toString()).getStatusId(), inProgressSet);
        statusMapping.put(statusMap.get(StatusType.CLOSED.toString()).getStatusId(), closedSet);
        statusMapping.put(statusMap.get(StatusType.REOPENED.toString()).getStatusId(), reopenedSet);
        statusMapping.put(statusMap.get(StatusType.ACCEPTED.toString()).getStatusId(), acceptedSet);
        statusMapping.put(statusMap.get(StatusType.REJECTED.toString()).getStatusId(), rejectedSet);
    }

    public boolean equals(Object obj){
        if(obj != null){
            Status statusObj = (Status)obj;
            if(statusObj.statusId == this.statusId && statusObj.statusVal.equals(this.statusVal))
                return true;
        }

        return false;
    }
}
```

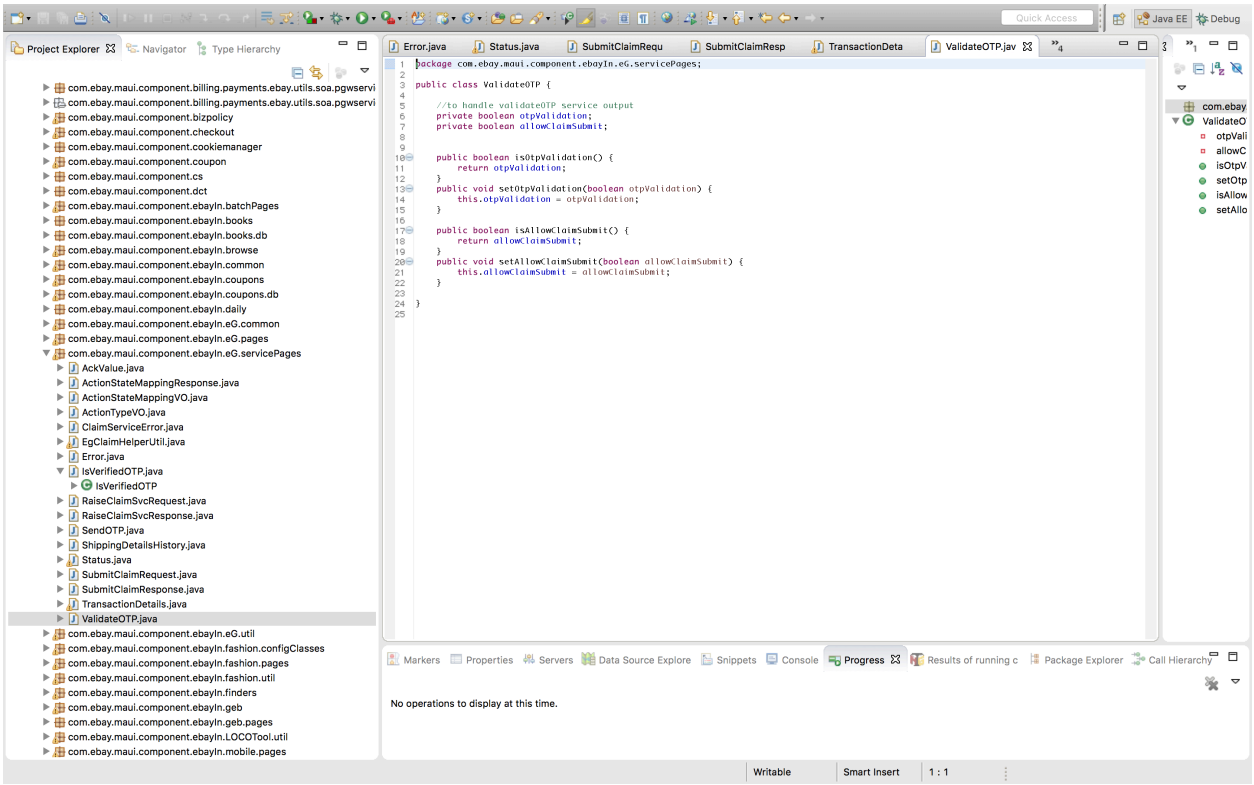


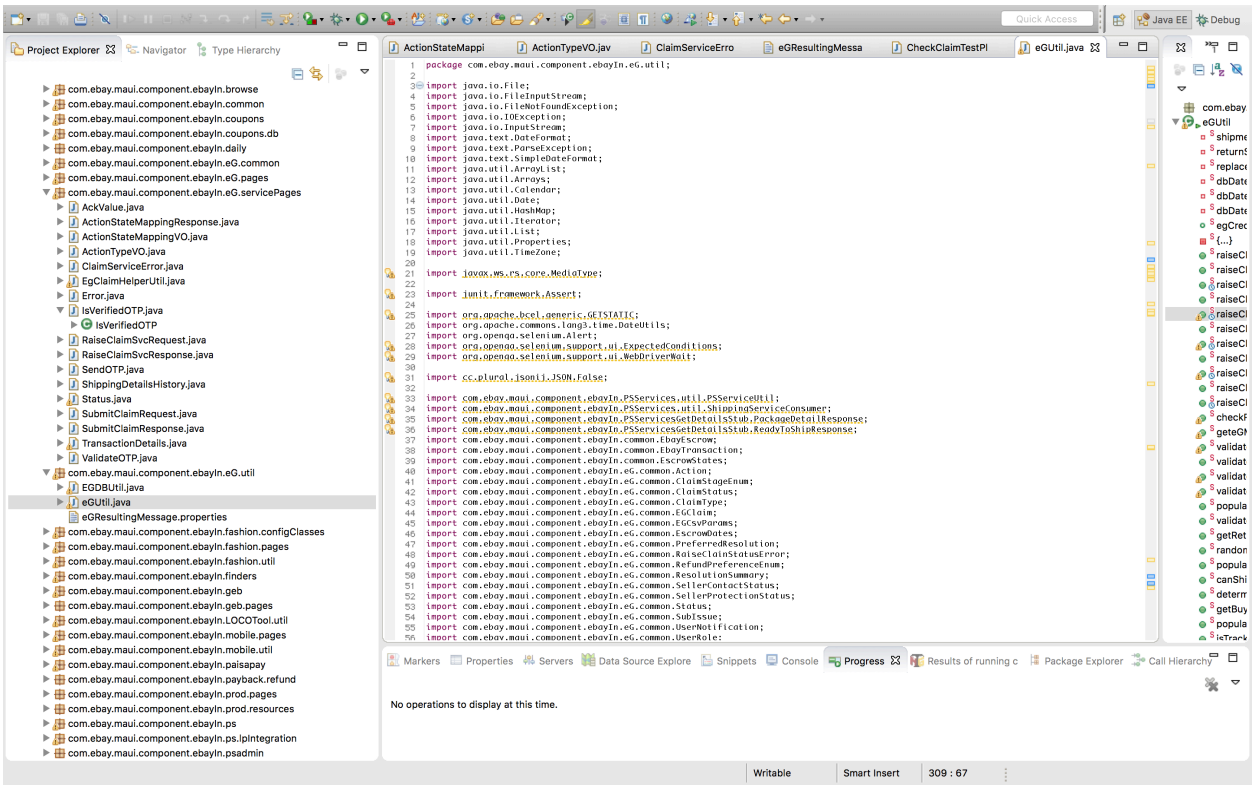

```java
package com.ebay.maui.component.ebayIn.eG.servicePages;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class SubmitClaimRequest {

    long transactionId;
    long buyerId;
    long sellerId;
    long itemId;
    long escrowId;
    String claimType;
    String remarks;


    public long getTransactionId() {
        return transactionId;
    }
    public void setTransactionId(long transactionId) {
        this.transactionId = transactionId;
    }
    public long getBuyerId() {
        return buyerId;
    }
    public void setBuyerId(long buyerId) {
        this.buyerId = buyerId;
    }
    public long getSellerId() {
        return sellerId;
    }
    public void setSellerId(long sellerId) {
        this.sellerId = sellerId;
    }
    public long getItemId() {
        return itemId;
    }
    public void setItemId(long itemId) {
        this.itemId = itemId;
    }
    public long getEscrowId() {
        return escrowId;
    }
    public void setEscrowId(long ecrowId) {
        this.escrowId = ecrowId;
    }
    public String getClaimType() {
        return claimType;
    }
    public void setClaimType(String claimType) {
        this.claimType = claimType;
    }
    public String getRemarks() {
        return remarks;
    }
    public void setRemarks(String remarks) {
        this.remarks = remarks;
    }
}
```



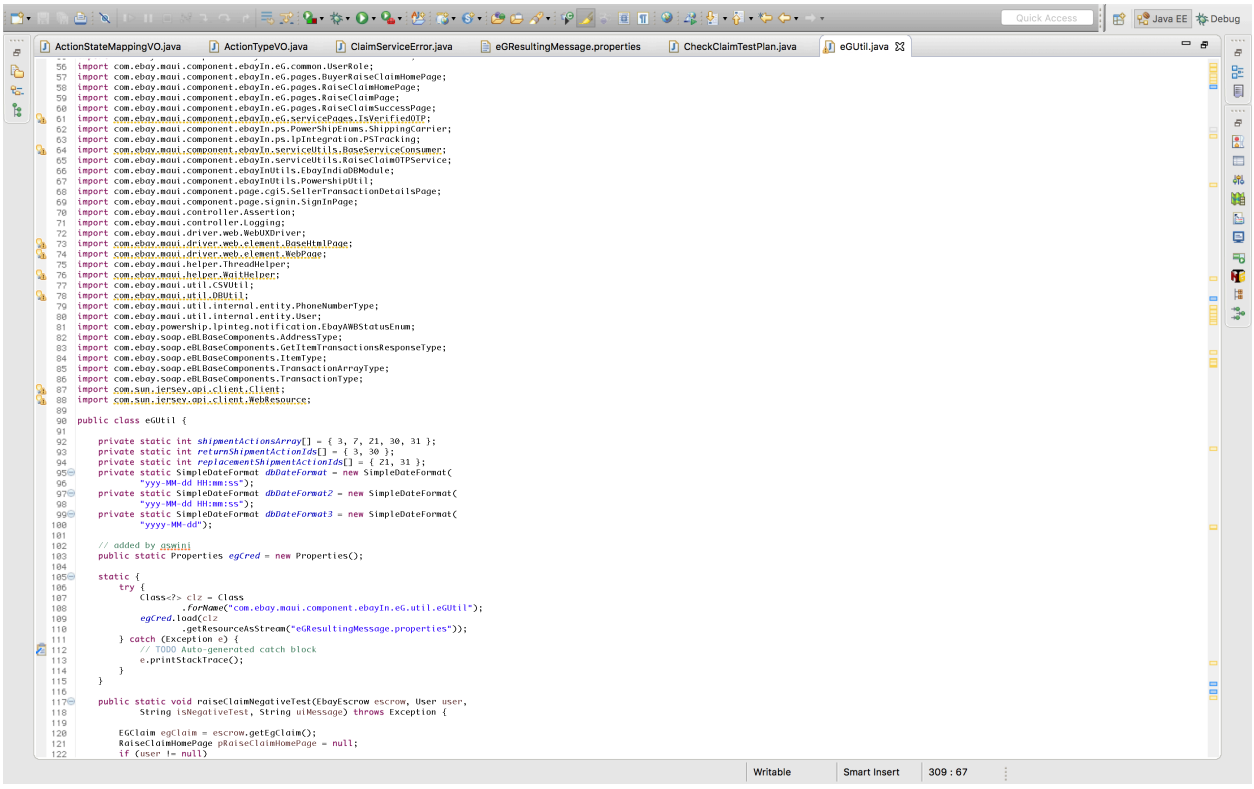

```java
package com.ebay.maui.component.ebayIn.eG.servicePages;

import java.util.ArrayList;

@XmlRootElement
public class SubmitClaimResponse {

    AckValue ack = AckValue.FAILURE;

    @XmlElementWrapper(name="errors")
    @XmlElement(name="error")
    private List<Error> errors = new ArrayList<Error>();

    public AckValue getAck() {
        return ack;
    }
    public void setAck(AckValue ack) {
        this.ack = ack;
    }
    public List<Error> getErrors() {
        return errors;
    }
}
```


ActionStateMapping
ActionTypeVO.java
EgClaimHelperUtil.ja
ClaimServiceError.j
Error.java
Status.java
SubmitClaimRequest.
SubmitClaimRespons
TransactionDetails.j
Writable
Smart Insert
1 : 1

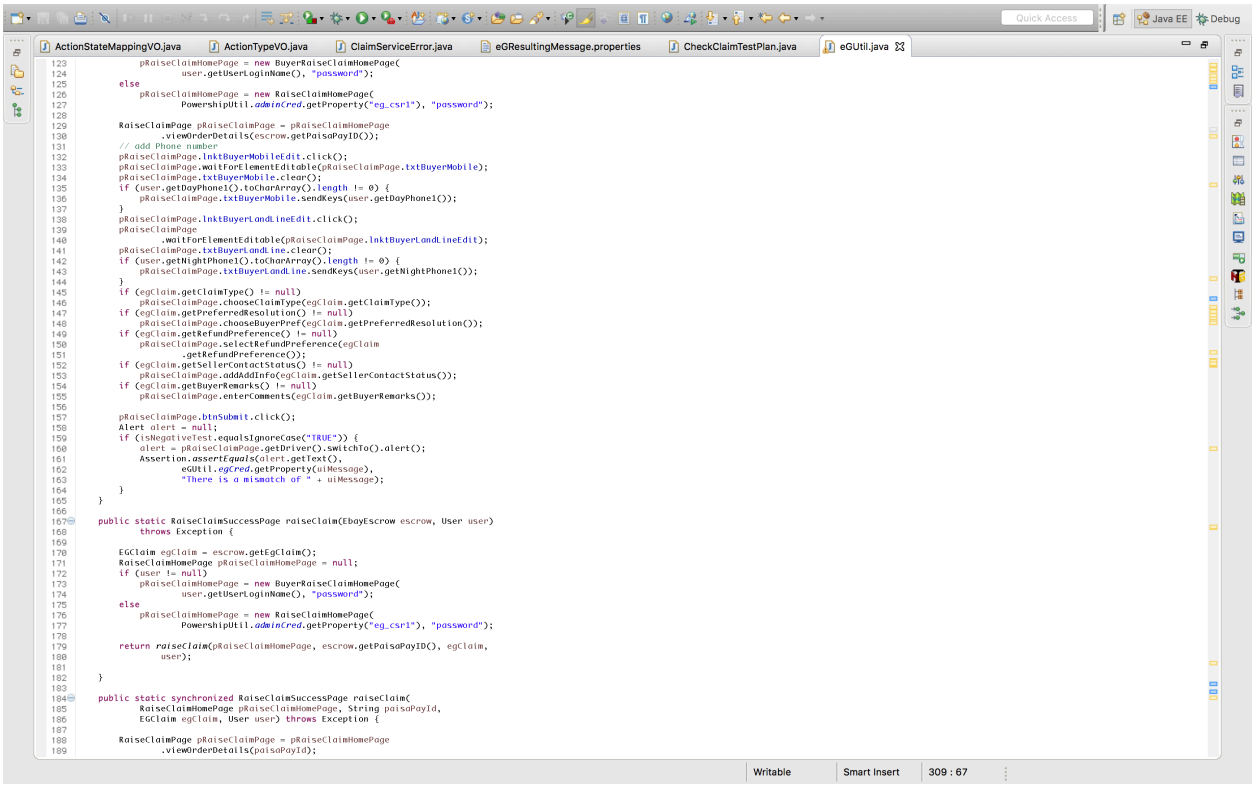

Project Explorer
Navigator
Type Hierarchy
Error.java
Status.java
SubmitClaimRequ
SubmitClaimResp
TransactionDeta
ValidateOTP.jav
No operations to display at this time.
Writable
Smart Insert
1 : 1



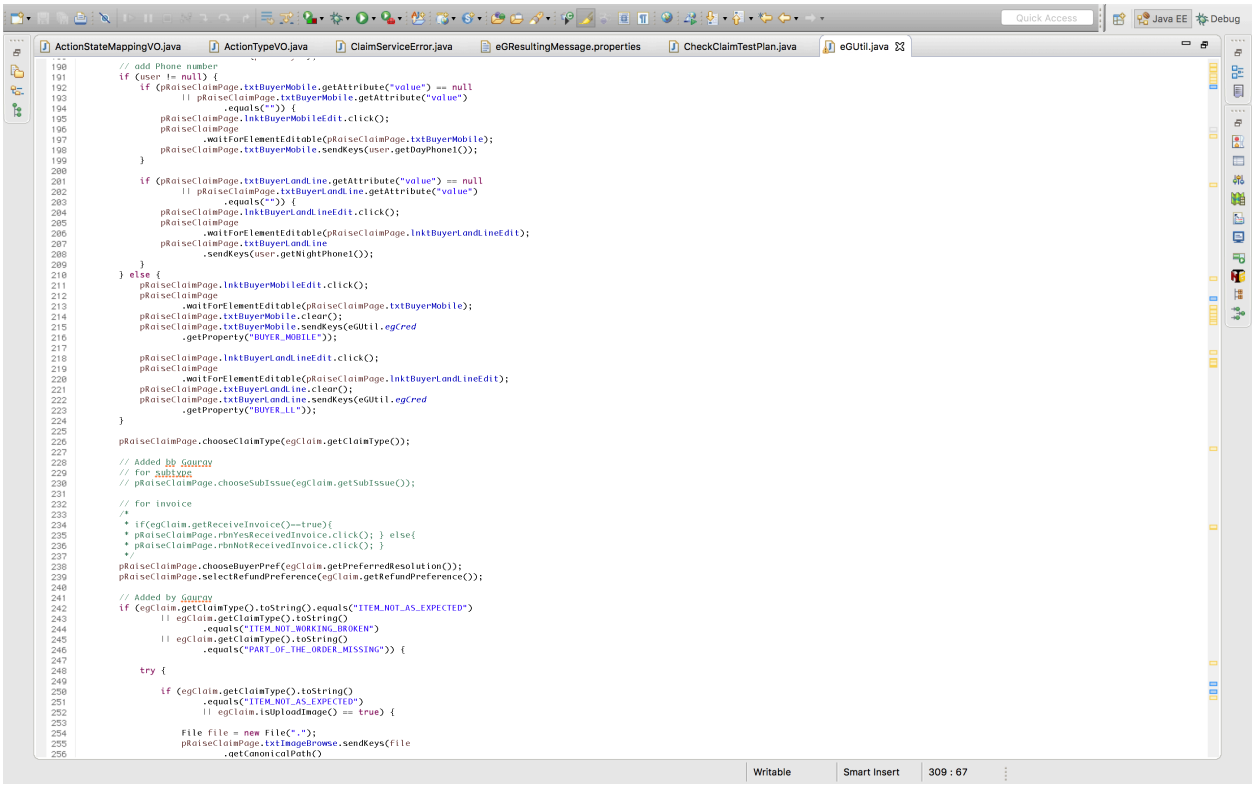
```
package com.ebay.maui.component.ebayIn.eG.util;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Properties;
import java.util.TimeZone;

import javax.ws.rs.core.MediaType;

import junit.framework.Assert;

import org.apache.bcel.generic.GETSTATIC;
import org.apache.commons.lang3.time.DateUtils;
import org.openqa.selenium.Alert;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import cc.plural.jsonij.JSON.False;

import com.ebay.maui.component.ebayIn.PSServices.util.PSServiceUtil;
import com.ebay.maui.component.ebayIn.PSServices.util.ShippingServiceConsumer;
import com.ebay.maui.component.ebayIn.PSServicesGetDetailsStub.PackageDetailResponse;
import com.ebay.maui.component.ebayIn.PSServicesGetDetailsStub.ReadyToShipResponse;
import com.ebay.maui.component.ebayIn.common.EbayEscrow;
import com.ebay.maui.component.ebayIn.common.EbayTransaction;
import com.ebay.maui.component.ebayIn.common.EscrowStates;
import com.ebay.maui.component.ebayIn.eG.common.Action;
import com.ebay.maui.component.ebayIn.eG.common.ClaimStageEnum;
import com.ebay.maui.component.ebayIn.eG.common.ClaimStatus;
import com.ebay.maui.component.ebayIn.eG.common.ClaimType;
import com.ebay.maui.component.ebayIn.eG.common.EGClaim;
import com.ebay.maui.component.ebayIn.eG.common.EGCsvParams;
import com.ebay.maui.component.ebayIn.eG.common.EscrowDates;
import com.ebay.maui.component.ebayIn.eG.common.PreferredResolution;
import com.ebay.maui.component.ebayIn.eG.common.RaiseClaimStatusError;
import com.ebay.maui.component.ebayIn.eG.common.RefundPreferenceEnum;
import com.ebay.maui.component.ebayIn.eG.common.ResolutionSummary;
import com.ebay.maui.component.ebayIn.eG.common.SellerContactStatus;
import com.ebay.maui.component.ebayIn.eG.common.SellerProtectionStatus;
import com.ebay.maui.component.ebayIn.eG.common.Status;
import com.ebay.maui.component.ebayIn.eG.common.SubIssue;
import com.ebay.maui.component.ebayIn.eG.common.UserNotification;
import com.ebay.maui.component.ebayIn.eG.common.UserRole;
```



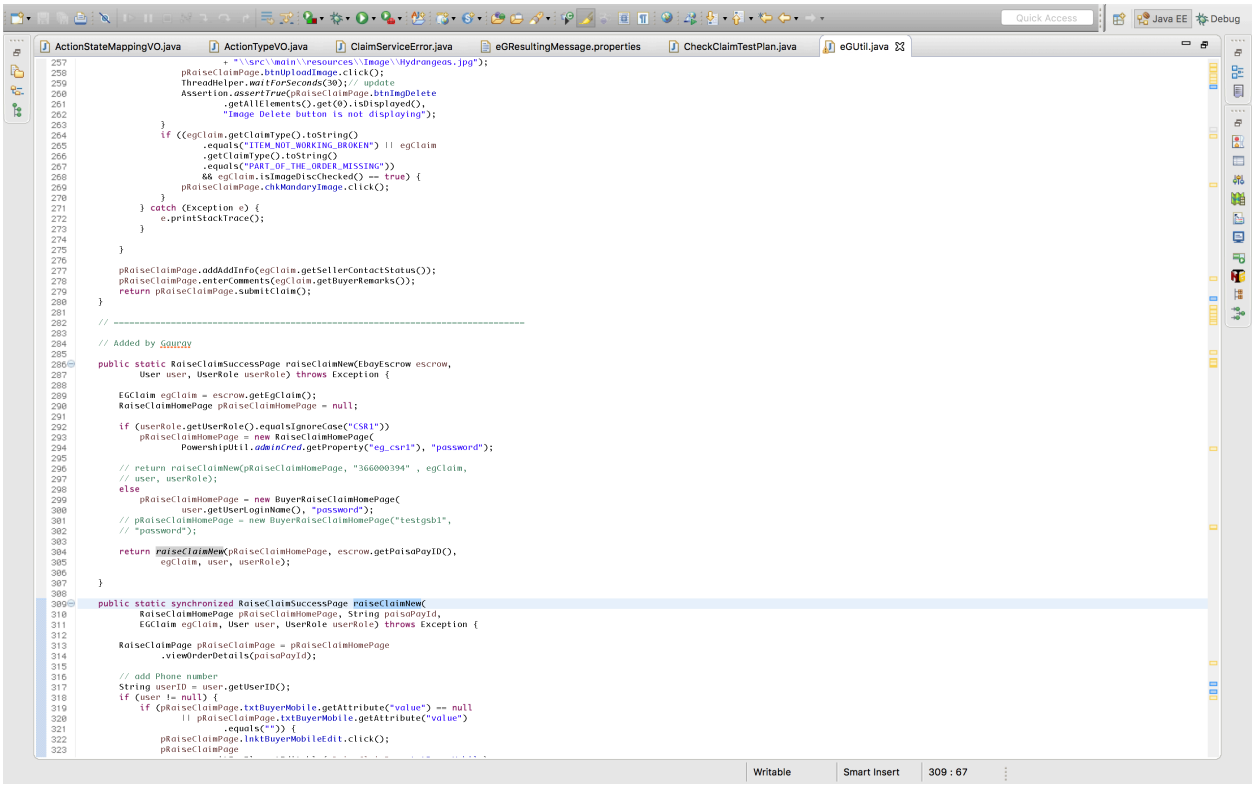
```
import com.ebay.maui.component.ebayIn.eG.common.UserRole;
import com.ebay.maui.component.ebayIn.eG.pages.BuyerRaiseClaimHomePage;
import com.ebay.maui.component.ebayIn.eG.pages.RaiseClaimHomePage;
import com.ebay.maui.component.ebayIn.eG.pages.RaiseClaimPage;
import com.ebay.maui.component.ebayIn.eG.pages.RaiseClaimSuccessPage;
import com.ebay.maui.component.ebayIn.eG.servicePages.IsVerifiedOTP;
import com.ebay.maui.component.ebayIn.ps.PowerShipEnums.ShippingCarrier;
import com.ebay.maui.component.ebayIn.ps.lpIntegration.PSTracking;
import com.ebay.maui.component.ebayIn.serviceUtils.BaseServiceConsumer;
import com.ebay.maui.component.ebayIn.serviceUtils.RaiseClaimOTPService;
import com.ebay.maui.component.ebayInUtils.EbayIndiaDBModule;
import com.ebay.maui.component.ebayInUtils.PowerShipUtil;
import com.ebay.maui.component.page.cgi5.SellerTransactionDetailsPage;
import com.ebay.maui.component.page.signin.SignInPage;
import com.ebay.maui.controller.Assertion;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.web.WebUXDriver;
import com.ebay.maui.driver.web.element.BaseHtmlPage;
import com.ebay.maui.driver.web.element.WebPage;
import com.ebay.maui.helper.ThreadHelper;
import com.ebay.maui.helper.WaitHelper;
import com.ebay.maui.util.CSVUtil;
import com.ebay.maui.util.DBUtil;
import com.ebay.maui.util.internal.entity.PhoneNumberType;
import com.ebay.maui.util.internal.entity.User;
import com.ebay.powership.lpinteg.notification.EbayAWBStatusEnum;
import com.ebay.soap.eBLBaseComponents.AddressType;
import com.ebay.soap.eBLBaseComponents.GetItemTransactionsResponseType;
import com.ebay.soap.eBLBaseComponents.ItemType;
import com.ebay.soap.eBLBaseComponents.TransactionArrayType;
import com.ebay.soap.eBLBaseComponents.TransactionType;
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;

public class eGUtil {

    private static int shipmentActionsArray[] = { 3, 7, 21, 30, 31 };
    private static int returnShipmentActionIds[] = { 3, 30 };
    private static int replacementShipmentActionIds[] = { 21, 31 };
    private static SimpleDateFormat dbDateFormat = new SimpleDateFormat(
            "yyy-MM-dd HH:mm:ss");
    private static SimpleDateFormat dbDateFormat2 = new SimpleDateFormat(
            "yyy-MM-dd HH:mm:ss");
    private static SimpleDateFormat dbDateFormat3 = new SimpleDateFormat(
            "yyyy-MM-dd");

    // added by aswini
    public static Properties egCred = new Properties();

    static {
        try {
            Class<?> clz = Class
                    .forName("com.ebay.maui.component.ebayIn.eG.util.eGUtil");
            egCred.load(clz
                    .getResourceAsStream("eGResultingMessage.properties"));
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public static void raiseClaimNegativeTest(EbayEscrow escrow, User user,
            String isNegativeTest, String uiMessage) throws Exception {

        EGClaim egClaim = escrow.getEgClaim();
        RaiseClaimHomePage pRaiseClaimHomePage = null;
        if (user != null)
```



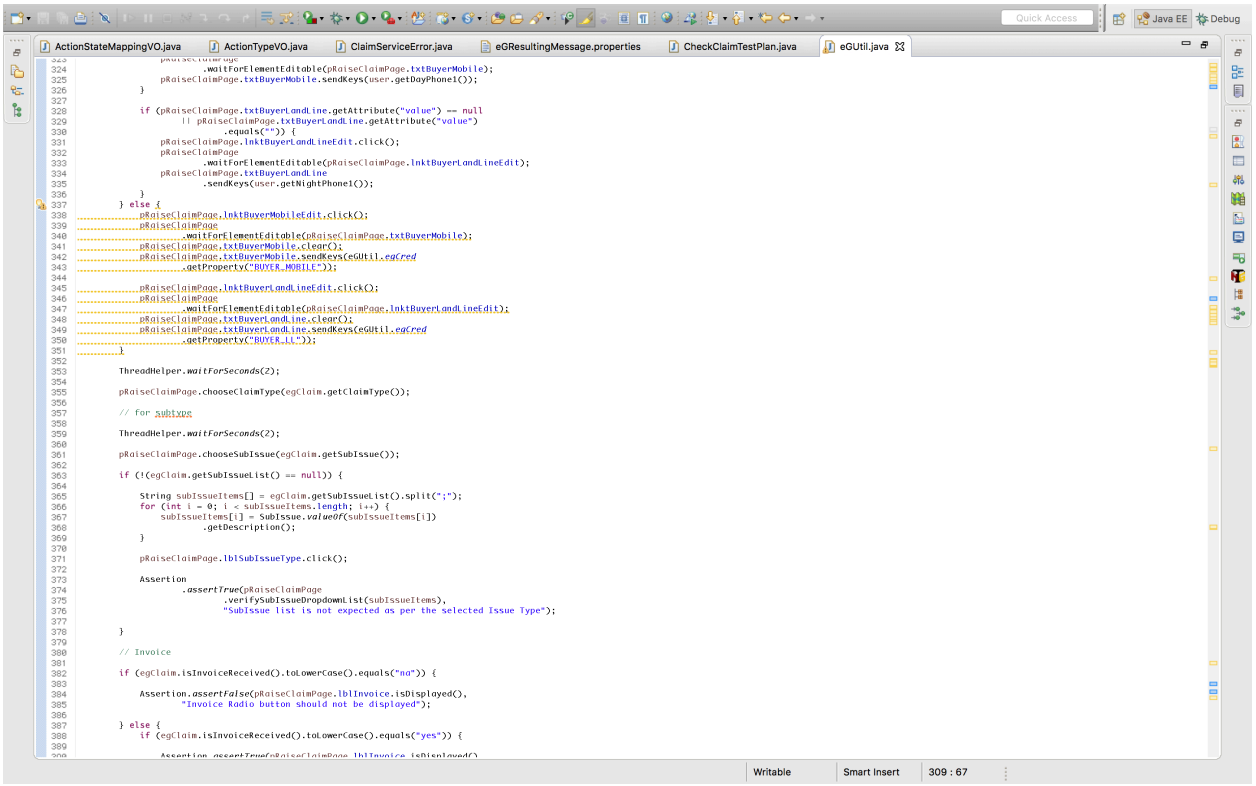
```
            pRaiseClaimHomePage = new BuyerRaiseClaimHomePage(
                    user.getUserLoginName(), "password");
        else
            pRaiseClaimHomePage = new RaiseClaimHomePage(
                    PowershipUtil.adminCred.getProperty("eg_csr1"), "password");

        RaiseClaimPage pRaiseClaimPage = pRaiseClaimHomePage
                .viewOrderDetails(escrow.getPaisaPayID());
        // add Phone number
        pRaiseClaimPage.lnktBuyerMobileEdit.click();
        pRaiseClaimPage.waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
        pRaiseClaimPage.txtBuyerMobile.clear();
        if (user.getDayPhone1().toCharArray().length != 0) {
            pRaiseClaimPage.txtBuyerMobile.sendKeys(user.getDayPhone1());
        }
        pRaiseClaimPage.lnktBuyerLandLineEdit.click();
        pRaiseClaimPage
                .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
        pRaiseClaimPage.txtBuyerLandLine.clear();
        if (user.getNightPhone1().toCharArray().length != 0) {
            pRaiseClaimPage.txtBuyerLandLine.sendKeys(user.getNightPhone1());
        }
        if (egClaim.getClaimType() != null)
            pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());
        if (egClaim.getPreferredResolution() != null)
            pRaiseClaimPage.chooseBuyerPref(egClaim.getPreferredResolution());
        if (egClaim.getRefundPreference() != null)
            pRaiseClaimPage.selectRefundPreference(egClaim
                    .getRefundPreference());
        if (egClaim.getSellerContactStatus() != null)
            pRaiseClaimPage.addAddInfo(egClaim.getSellerContactStatus());
        if (egClaim.getBuyerRemarks() != null)
            pRaiseClaimPage.enterComments(egClaim.getBuyerRemarks());

        pRaiseClaimPage.btnSubmit.click();
        Alert alert = null;
        if (isNegativeTest.equalsIgnoreCase("TRUE")) {
            alert = pRaiseClaimPage.getDriver().switchTo().alert();
            Assertion.assertEquals(alert.getText(),
                    eGUtil.egCred.getProperty(uiMessage),
                    "There is a mismatch of " + uiMessage);
        }
    }

    public static RaiseClaimSuccessPage raiseClaim(EbayEscrow escrow, User user)
            throws Exception {

        EGClaim egClaim = escrow.getEgClaim();
        RaiseClaimHomePage pRaiseClaimHomePage = null;
        if (user != null)
            pRaiseClaimHomePage = new BuyerRaiseClaimHomePage(
                    user.getUserLoginName(), "password");
        else
            pRaiseClaimHomePage = new RaiseClaimHomePage(
                    PowershipUtil.adminCred.getProperty("eg_csr1"), "password");

        return raiseClaim(pRaiseClaimHomePage, escrow.getPaisaPayID(), egClaim,
                user);

    }

    public static synchronized RaiseClaimSuccessPage raiseClaim(
            RaiseClaimHomePage pRaiseClaimHomePage, String paisaPayId,
            EGClaim egClaim, User user) throws Exception {

        RaiseClaimPage pRaiseClaimPage = pRaiseClaimHomePage
                .viewOrderDetails(paisaPayId);
```

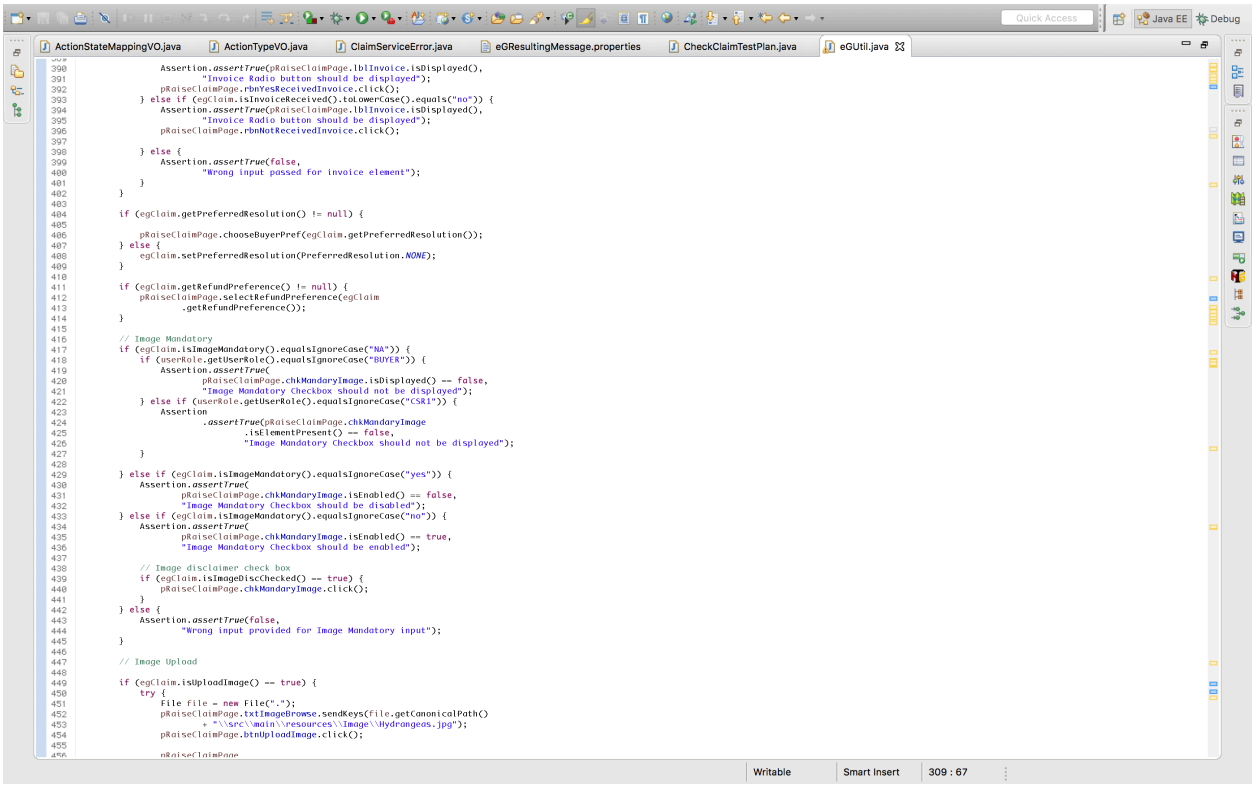
```
        // add Phone number
        if (user != null) {
            if (pRaiseClaimPage.txtBuyerMobile.getAttribute("value") == null
                    || pRaiseClaimPage.txtBuyerMobile.getAttribute("value")
                            .equals("")) {
                pRaiseClaimPage.lnktBuyerMobileEdit.click();
                pRaiseClaimPage
                        .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
                pRaiseClaimPage.txtBuyerMobile.sendKeys(user.getDayPhone1());
            }

            if (pRaiseClaimPage.txtBuyerLandLine.getAttribute("value") == null
                    || pRaiseClaimPage.txtBuyerLandLine.getAttribute("value")
                            .equals("")) {
                pRaiseClaimPage.lnktBuyerLandLineEdit.click();
                pRaiseClaimPage
                        .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
                pRaiseClaimPage.txtBuyerLandLine
                        .sendKeys(user.getNightPhone1());
            }
        } else {
            pRaiseClaimPage.lnktBuyerMobileEdit.click();
            pRaiseClaimPage
                    .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
            pRaiseClaimPage.txtBuyerMobile.clear();
            pRaiseClaimPage.txtBuyerMobile.sendKeys(eGUtil.egCred
                    .getProperty("BUYER_MOBILE"));

            pRaiseClaimPage.lnktBuyerLandLineEdit.click();
            pRaiseClaimPage
                    .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
            pRaiseClaimPage.txtBuyerLandLine.clear();
            pRaiseClaimPage.txtBuyerLandLine.sendKeys(eGUtil.egCred
                    .getProperty("BUYER_LL"));
        }

        pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());

        // Added bb Gaurav
        // for subtype
        // pRaiseClaimPage.chooseSubIssue(egClaim.getSubIssue());

        // for invoice
        /*
         * if(egClaim.getReceiveInvoice()==true){
         * pRaiseClaimPage.rbnYesReceivedInvoice.click(); } else{
         * pRaiseClaimPage.rbnNotReceivedInvoice.click(); }
         */
        pRaiseClaimPage.chooseBuyerPref(egClaim.getPreferredResolution());
        pRaiseClaimPage.selectRefundPreference(egClaim.getRefundPreference());

        // Added by Gaurav
        if (egClaim.getClaimType().toString().equals("ITEM_NOT_AS_EXPECTED")
                || egClaim.getClaimType().toString()
                        .equals("ITEM_NOT_WORKING_BROKEN")
                || egClaim.getClaimType().toString()
                        .equals("PART_OF_THE_ORDER_MISSING")) {

            try {

                if (egClaim.getClaimType().toString()
                        .equals("ITEM_NOT_AS_EXPECTED")
                        || egClaim.isUploadImage() == true) {

                    File file = new File(".");
                    pRaiseClaimPage.txtImageBrowse.sendKeys(file
                            .getCanonicalPath()
```

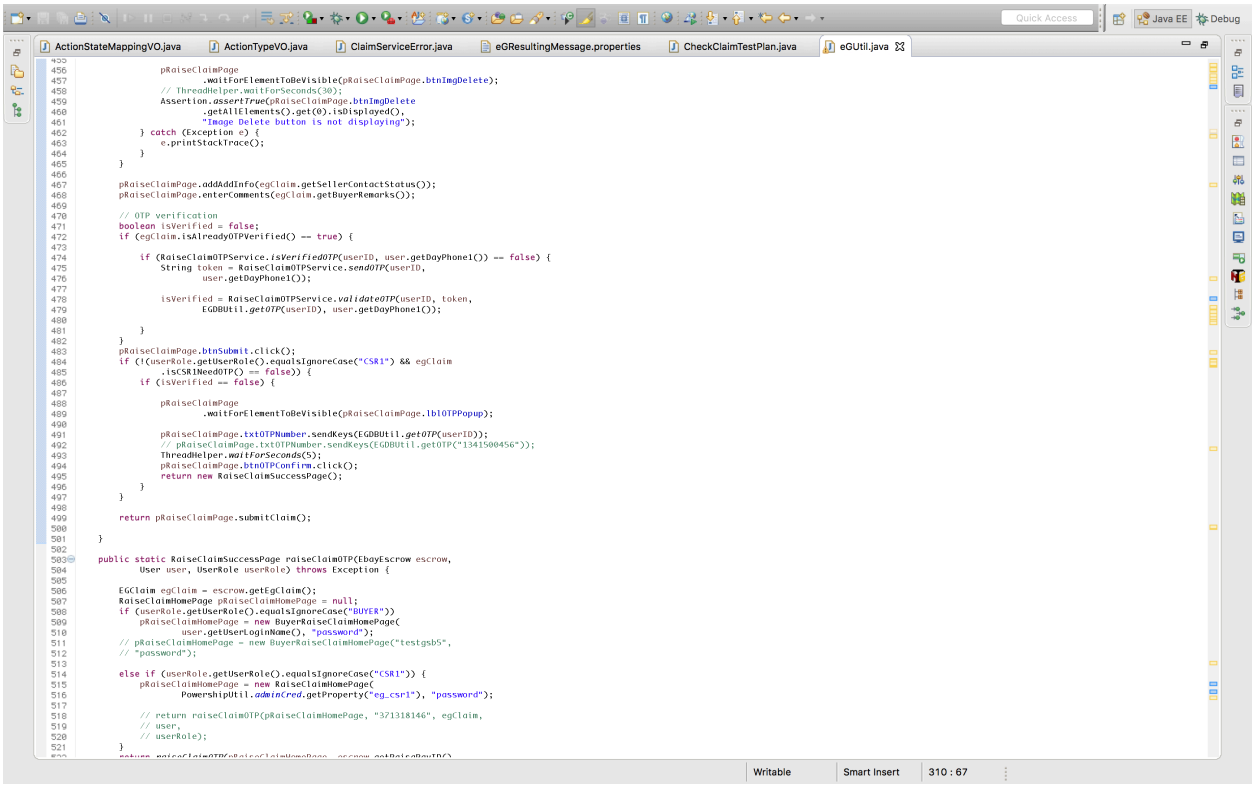


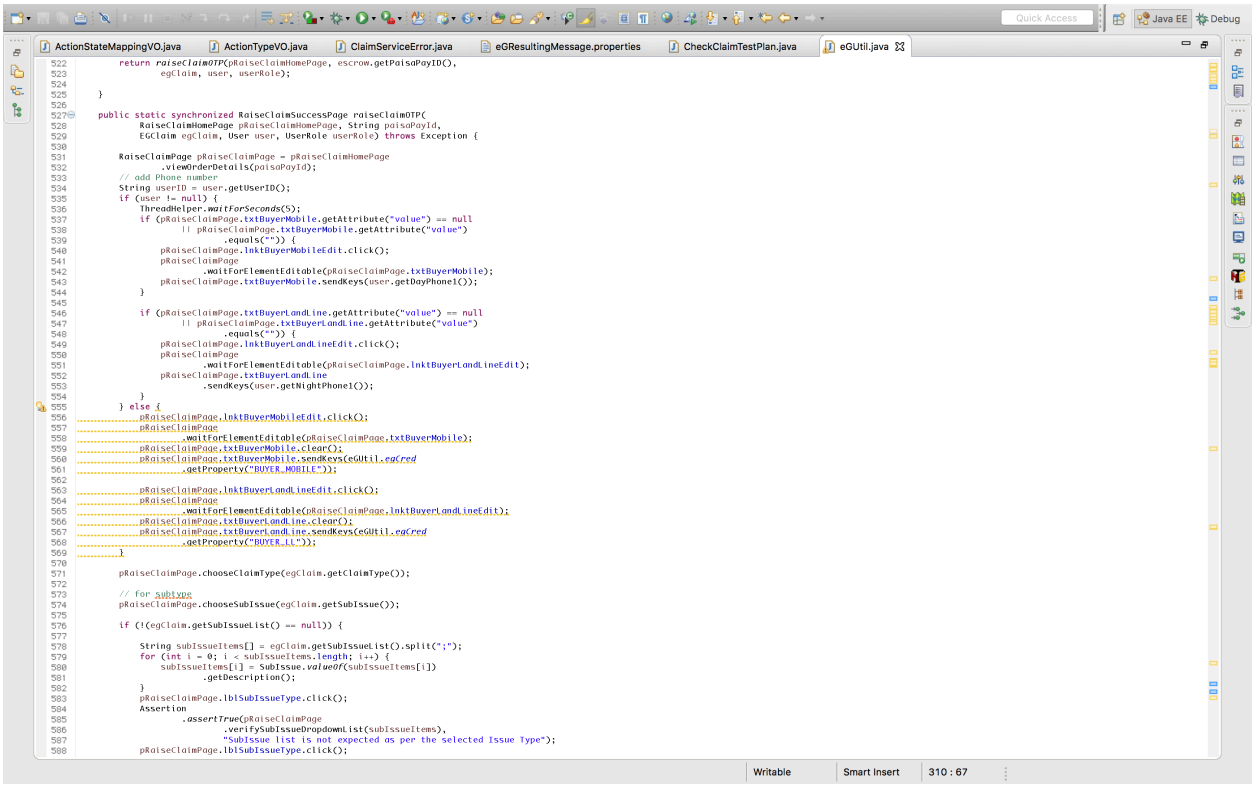
```
                        + "\\src\\main\\resources\\Image\\Hydrangeas.jpg");
                pRaiseClaimPage.btnUploadImage.click();
                ThreadHelper.waitForSeconds(30);// update
                Assertion.assertTrue(pRaiseClaimPage.btnImgDelete
                        .getAllElements().get(0).isDisplayed(),
                        "Image Delete button is not displaying");
            }
            if ((egClaim.getClaimType().toString()
                    .equals("ITEM_NOT_WORKING_BROKEN") || egClaim
                    .getClaimType().toString()
                    .equals("PART_OF_THE_ORDER_MISSING"))
                    && egClaim.isImageDiscChecked() == true) {
                pRaiseClaimPage.chkMandaryImage.click();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    pRaiseClaimPage.addAddInfo(egClaim.getSellerContactStatus());
    pRaiseClaimPage.enterComments(egClaim.getBuyerRemarks());
    return pRaiseClaimPage.submitClaim();
}

// ================================================================================

// Added by Gaurav

public static RaiseClaimSuccessPage raiseClaimNew(EbayEscrow escrow,
        User user, UserRole userRole) throws Exception {

    EGClaim egClaim = escrow.getEgClaim();
    RaiseClaimHomePage pRaiseClaimHomePage = null;

    if (userRole.getUserRole().equalsIgnoreCase("CSR1"))
        pRaiseClaimHomePage = new RaiseClaimHomePage(
                PowershipUtil.adminCred.getProperty("eg_csr1"), "password");

    // return raiseClaimNew(pRaiseClaimHomePage, "366000394" , egClaim,
    // user, userRole);
    else
        pRaiseClaimHomePage = new BuyerRaiseClaimHomePage(
                user.getUserLoginName(), "password");
    // pRaiseClaimHomePage = new BuyerRaiseClaimHomePage("testgsb1",
    // "password");

    return raiseClaimNew(pRaiseClaimHomePage, escrow.getPaisaPayID(),
            egClaim, user, userRole);

}

public static synchronized RaiseClaimSuccessPage raiseClaimNew(
        RaiseClaimHomePage pRaiseClaimHomePage, String paisaPayId,
        EGClaim egClaim, User user, UserRole userRole) throws Exception {

    RaiseClaimPage pRaiseClaimPage = pRaiseClaimHomePage
            .viewOrderDetails(paisaPayId);

    // add Phone number
    String userID = user.getUserID();
    if (user != null) {
        if (pRaiseClaimPage.txtBuyerMobile.getAttribute("value") == null
                || pRaiseClaimPage.txtBuyerMobile.getAttribute("value")
                        .equals("")) {
            pRaiseClaimPage.lnktBuyerMobileEdit.click();
            pRaiseClaimPage
```



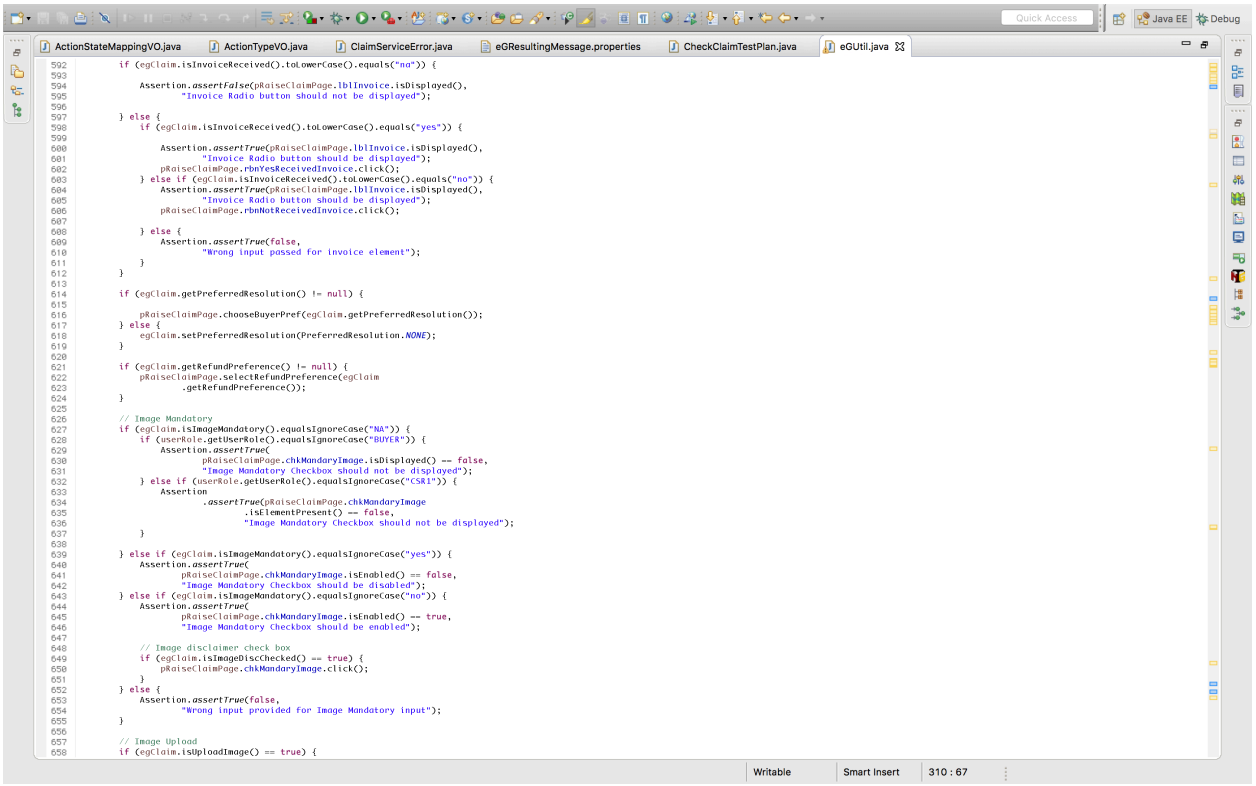
```
                    .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
            pRaiseClaimPage.txtBuyerMobile.sendKeys(user.getDayPhone1());
        }

        if (pRaiseClaimPage.txtBuyerLandLine.getAttribute("value") == null
                || pRaiseClaimPage.txtBuyerLandLine.getAttribute("value")
                        .equals("")) {
            pRaiseClaimPage.lnktBuyerLandLineEdit.click();
            pRaiseClaimPage
                    .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
            pRaiseClaimPage.txtBuyerLandLine
                    .sendKeys(user.getNightPhone1());
        }
    } else {
//..............pRaiseClaimPage.lnktBuyerMobileEdit.click();
//..............pRaiseClaimPage
//........................waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
//..............pRaiseClaimPage.txtBuyerMobile.clear();
//..............pRaiseClaimPage.txtBuyerMobile.sendKeys(ceGUtil.egCred
//........................getProperty("BUYER_MOBILE"));

//..............pRaiseClaimPage.lnktBuyerLandLineEdit.click();
//..............pRaiseClaimPage
//........................waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
//..............pRaiseClaimPage.txtBuyerLandLine.clear();
//..............pRaiseClaimPage.txtBuyerLandLine.sendKeys(ceGUtil.egCred
//........................getProperty("BUYER_LL"));
//..........}

        ThreadHelper.waitForSeconds(2);

        pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());

        // for subtype

        ThreadHelper.waitForSeconds(2);

        pRaiseClaimPage.chooseSubIssue(egClaim.getSubIssue());

        if (!(egClaim.getSubIssueList() == null)) {

            String subIssueItems[] = egClaim.getSubIssueList().split(";");
            for (int i = 0; i < subIssueItems.length; i++) {
                subIssueItems[i] = SubIssue.valueOf(subIssueItems[i])
                        .getDescription();
            }

            pRaiseClaimPage.lblSubIssueType.click();

            Assertion
                    .assertTrue(pRaiseClaimPage
                            .verifySubIssueDropdownList(subIssueItems),
                            "SubIssue list is not expected as per the selected Issue Type");

        }

        // Invoice

        if (egClaim.isInvoiceReceived().toLowerCase().equals("na")) {

            Assertion.assertFalse(pRaiseClaimPage.lblInvoice.isDisplayed(),
                    "Invoice Radio button should not be displayed");

        } else {
            if (egClaim.isInvoiceReceived().toLowerCase().equals("yes")) {
```


ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
eGResultingMessage.properties
CheckClaimTestPlan.java
eGUtil.java
Writable
Smart Insert
309 : 67

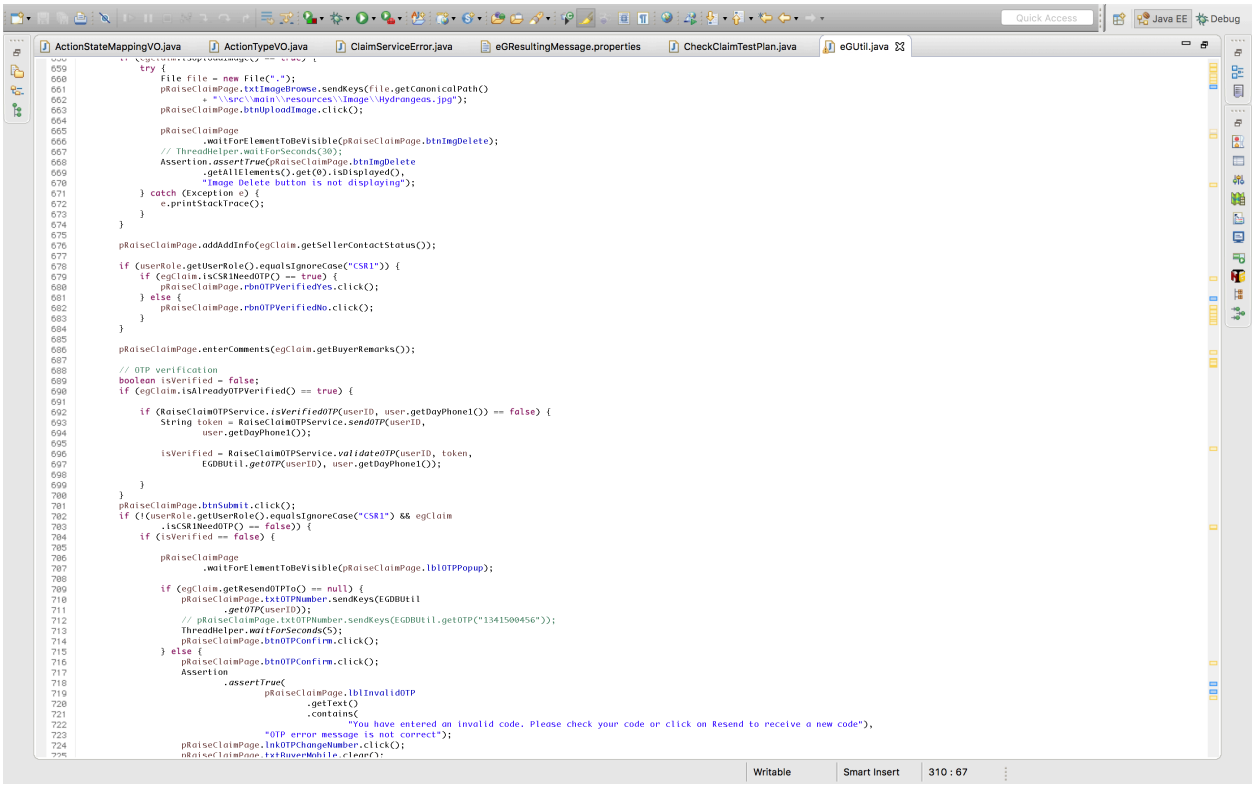

ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
eGResultingMessage.properties
CheckClaimTestPlan.java
eGUtil.java
Writable
Smart Insert
310 : 67



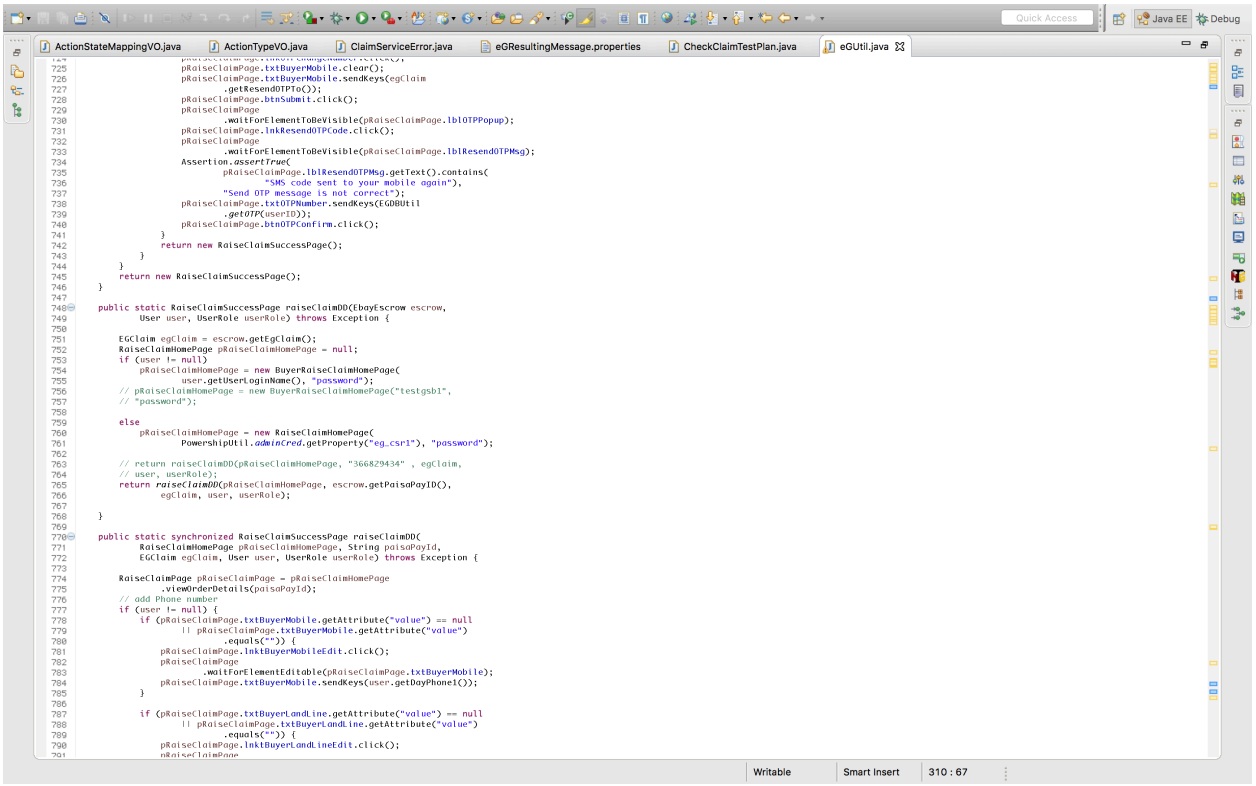
```
ActionStateMappingVO.java | ActionTypeVO.java | ClaimServiceError.java | eGResultingMessage.properties | CheckClaimTestPlan.java | eGUtil.java

522         return raiseClaimOTP(pRaiseClaimHomePage, escrow.getPaisaPayID(),
523                 egClaim, user, userRole);
524
525     }
526
527     public static synchronized RaiseClaimSuccessPage raiseClaimOTP(
528             RaiseClaimHomePage pRaiseClaimHomePage, String paisaPayId,
529             EGClaim egClaim, User user, UserRole userRole) throws Exception {
530
531         RaiseClaimPage pRaiseClaimPage = pRaiseClaimHomePage
532                 .viewOrderDetails(paisaPayId);
533         // add Phone number
534         String userID = user.getUserID();
535         if (user != null) {
536             ThreadHelper.waitForSeconds(5);
537             if (pRaiseClaimPage.txtBuyerMobile.getAttribute("value") == null
538                     || pRaiseClaimPage.txtBuyerMobile.getAttribute("value")
539                             .equals("")) {
540                 pRaiseClaimPage.lnktBuyerMobileEdit.click();
541                 pRaiseClaimPage
542                         .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
543                 pRaiseClaimPage.txtBuyerMobile.sendKeys(user.getDayPhone1());
544             }
545
546             if (pRaiseClaimPage.txtBuyerLandLine.getAttribute("value") == null
547                     || pRaiseClaimPage.txtBuyerLandLine.getAttribute("value")
548                             .equals("")) {
549                 pRaiseClaimPage.lnktBuyerLandLineEdit.click();
550                 pRaiseClaimPage
551                         .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
552                 pRaiseClaimPage.txtBuyerLandLine
553                         .sendKeys(user.getNightPhone1());
554             }
555         } else {
556             pRaiseClaimPage.lnktBuyerMobileEdit.click();
557             pRaiseClaimPage
558                     .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
559             pRaiseClaimPage.txtBuyerMobile.clear();
560             pRaiseClaimPage.txtBuyerMobile.sendKeys(eGUtil.egCred
561                     .getProperty("BUYER_MOBILE"));
562
563             pRaiseClaimPage.lnktBuyerLandLineEdit.click();
564             pRaiseClaimPage
565                     .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
566             pRaiseClaimPage.txtBuyerLandLine.clear();
567             pRaiseClaimPage.txtBuyerLandLine.sendKeys(eGUtil.egCred
568                     .getProperty("BUYER_LL"));
569         }
570
571         pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());
572
573         // for subtype
574         pRaiseClaimPage.chooseSubIssue(egClaim.getSubIssue());
575
576         if (!(egClaim.getSubIssueList() == null)) {
577
578             String subIssueItems[] = egClaim.getSubIssueList().split(";");
579             for (int i = 0; i < subIssueItems.length; i++) {
580                 subIssueItems[i] = SubIssue.valueOf(subIssueItems[i])
581                         .getDescription();
582             }
583             pRaiseClaimPage.lblSubIssueType.click();
584             Assertion
585                     .assertTrue(pRaiseClaimPage
586                             .verifySubIssueDropdownList(subIssueItems),
587                             "SubIssue list is not expected as per the selected Issue Type");
588             pRaiseClaimPage.lblSubIssueType.click();

Writable | Smart Insert | 310 : 67
```



```
ActionStateMappingVO.java | ActionTypeVO.java | ClaimServiceError.java | eGResultingMessage.properties | CheckClaimTestPlan.java | eGUtil.java

592         if (egClaim.isInvoiceReceived().toLowerCase().equals("na")) {
593
594             Assertion.assertFalse(pRaiseClaimPage.lblInvoice.isDisplayed(),
595                     "Invoice Radio button should not be displayed");
596
597         } else {
598             if (egClaim.isInvoiceReceived().toLowerCase().equals("yes")) {
599
600                 Assertion.assertTrue(pRaiseClaimPage.lblInvoice.isDisplayed(),
601                         "Invoice Radio button should be displayed");
602                 pRaiseClaimPage.rbnYesReceivedInvoice.click();
603             } else if (egClaim.isInvoiceReceived().toLowerCase().equals("no")) {
604                 Assertion.assertTrue(pRaiseClaimPage.lblInvoice.isDisplayed(),
605                         "Invoice Radio button should be displayed");
606                 pRaiseClaimPage.rbnNotReceivedInvoice.click();
607
608             } else {
609                 Assertion.assertTrue(false,
610                         "Wrong input passed for invoice element");
611             }
612         }
613
614         if (egClaim.getPreferredResolution() != null) {
615
616             pRaiseClaimPage.chooseBuyerPref(egClaim.getPreferredResolution());
617         } else {
618             egClaim.setPreferredResolution(PreferredResolution.NONE);
619         }
620
621         if (egClaim.getRefundPreference() != null) {
622             pRaiseClaimPage.selectRefundPreference(egClaim
623                     .getRefundPreference());
624         }
625
626         // Image Mandatory
627         if (egClaim.isImageMandatory().equalsIgnoreCase("NA")) {
628             if (userRole.getUserRole().equalsIgnoreCase("BUYER")) {
629                 Assertion.assertTrue(
630                         pRaiseClaimPage.chkMandaryImage.isDisplayed() == false,
631                         "Image Mandatory Checkbox should not be displayed");
632             } else if (userRole.getUserRole().equalsIgnoreCase("CSR1")) {
633                 Assertion
634                         .assertTrue(pRaiseClaimPage.chkMandaryImage
635                                 .isElementPresent() == false,
636                                 "Image Mandatory Checkbox should not be displayed");
637             }
638
639         } else if (egClaim.isImageMandatory().equalsIgnoreCase("yes")) {
640             Assertion.assertTrue(
641                     pRaiseClaimPage.chkMandaryImage.isEnabled() == false,
642                     "Image Mandatory Checkbox should be disabled");
643         } else if (egClaim.isImageMandatory().equalsIgnoreCase("no")) {
644             Assertion.assertTrue(
645                     pRaiseClaimPage.chkMandaryImage.isEnabled() == true,
646                     "Image Mandatory Checkbox should be enabled");
647
648             // Image disclaimer check box
649             if (egClaim.isImageDiscChecked() == true) {
650                 pRaiseClaimPage.chkMandaryImage.click();
651             }
652         } else {
653             Assertion.assertTrue(false,
654                     "Wrong input provided for Image Mandatory input");
655         }
656
657         // Image Upload
658         if (egClaim.isUploadImage() == true) {

Writable | Smart Insert | 310 : 67
```



```
659             try {
660                 File file = new File(".");
661                 pRaiseClaimPage.txtImageBrowse.sendKeys(file.getCanonicalPath()
662                         + "\\src\\main\\resources\\Image\\Hydrangeas.jpg");
663                 pRaiseClaimPage.btnUploadImage.click();
664
665                 pRaiseClaimPage
666                         .waitForElementToBeVisible(pRaiseClaimPage.btnImgDelete);
667                 // ThreadHelper.waitForSeconds(30);
668                 Assertion.assertTrue(pRaiseClaimPage.btnImgDelete
669                         .getAllElements().get(0).isDisplayed(),
670                         "Image Delete button is not displaying");
671             } catch (Exception e) {
672                 e.printStackTrace();
673             }
674         }
675
676         pRaiseClaimPage.addAddInfo(egClaim.getSellerContactStatus());
677
678         if (userRole.getUserRole().equalsIgnoreCase("CSR1")) {
679             if (egClaim.isCSR1NeedOTP() == true) {
680                 pRaiseClaimPage.rbnOTPVerifiedYes.click();
681             } else {
682                 pRaiseClaimPage.rbnOTPVerifiedNo.click();
683             }
684         }
685
686         pRaiseClaimPage.enterComments(egClaim.getBuyerRemarks());
687
688         // OTP verification
689         boolean isVerified = false;
690         if (egClaim.isAlreadyOTPVerified() == true) {
691
692             if (RaiseClaimOTPService.isVerifiedOTP(userID, user.getDayPhone1()) == false) {
693                 String token = RaiseClaimOTPService.sendOTP(userID,
694                         user.getDayPhone1());
695
696                 isVerified = RaiseClaimOTPService.validateOTP(userID, token,
697                         EGDBUtil.getOTP(userID), user.getDayPhone1());
698
699             }
700         }
701         pRaiseClaimPage.btnSubmit.click();
702         if (!(userRole.getUserRole().equalsIgnoreCase("CSR1") && egClaim
703                 .isCSR1NeedOTP() == false)) {
704             if (isVerified == false) {
705
706                 pRaiseClaimPage
707                         .waitForElementToBeVisible(pRaiseClaimPage.lblOTPPopup);
708
709                 if (egClaim.getResendOTPTo() == null) {
710                     pRaiseClaimPage.txtOTPNumber.sendKeys(EGDBUtil
711                             .getOTP(userID));
712                     // pRaiseClaimPage.txtOTPNumber.sendKeys(EGDBUtil.getOTP("1341500456"));
713                     ThreadHelper.waitForSeconds(5);
714                     pRaiseClaimPage.btnOTPConfirm.click();
715                 } else {
716                     pRaiseClaimPage.btnOTPConfirm.click();
717                     Assertion
718                             .assertTrue(
719                                     pRaiseClaimPage.lblInvalidOTP
720                                             .getText()
721                                             .contains(
722                                                     "You have entered an invalid code. Please check your code or click on Resend to receive a new code"),
723                             "OTP error message is not correct");
724                     pRaiseClaimPage.lnkOTPChangeNumber.click();
725                     pRaiseClaimPage.txtBuyerMobile.clear();
```

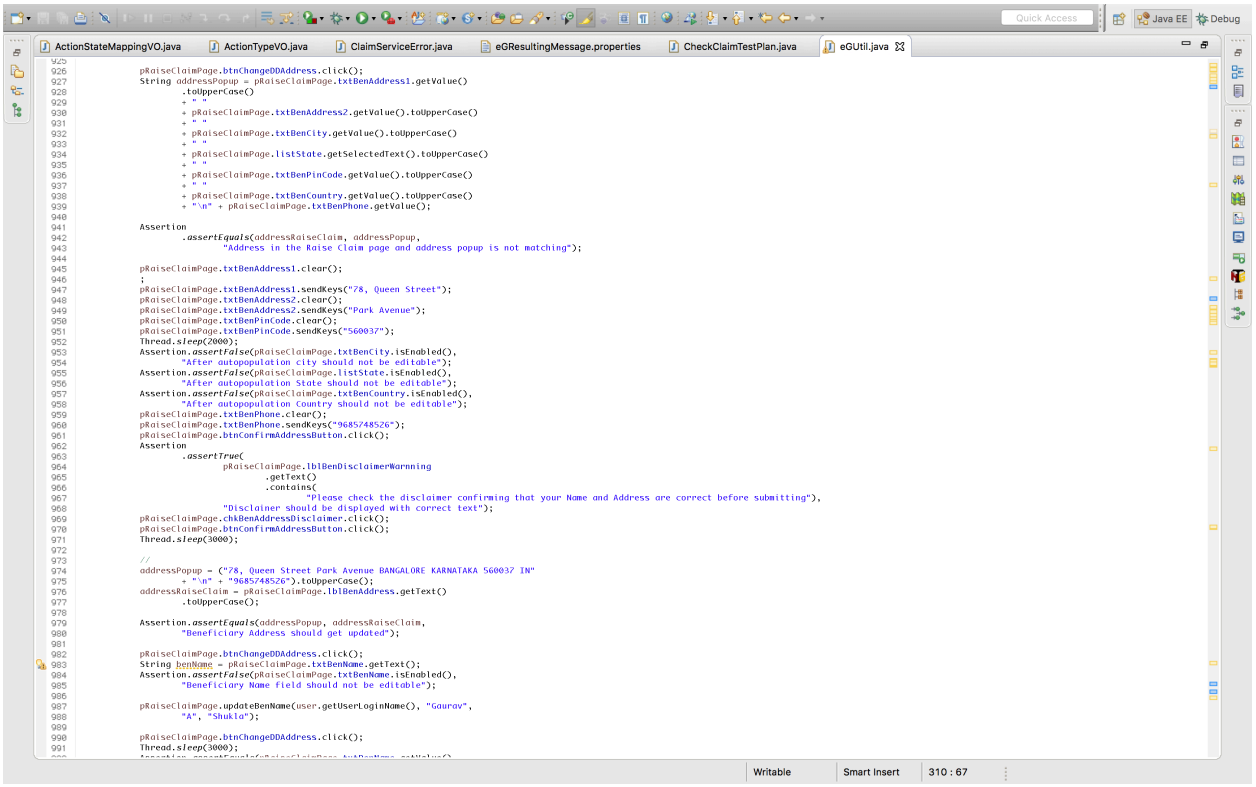
```
725                     pRaiseClaimPage.txtBuyerMobile.clear();
726                     pRaiseClaimPage.txtBuyerMobile.sendKeys(egClaim
727                             .getResendOTPTo());
728                     pRaiseClaimPage.btnSubmit.click();
729                     pRaiseClaimPage
730                             .waitForElementToBeVisible(pRaiseClaimPage.lblOTPPopup);
731                     pRaiseClaimPage.lnkResendOTPCode.click();
732                     pRaiseClaimPage
733                             .waitForElementToBeVisible(pRaiseClaimPage.lblResendOTPMsg);
734                     Assertion.assertTrue(
735                             pRaiseClaimPage.lblResendOTPMsg.getText().contains(
736                                     "SMS code sent to your mobile again"),
737                             "Send OTP message is not correct");
738                     pRaiseClaimPage.txtOTPNumber.sendKeys(EGDBUtil
739                             .getOTP(userID));
740                     pRaiseClaimPage.btnOTPConfirm.click();
741                 }
742                 return new RaiseClaimSuccessPage();
743             }
744         }
745         return new RaiseClaimSuccessPage();
746     }
747
748     public static RaiseClaimSuccessPage raiseClaimDD(EbayEscrow escrow,
749             User user, UserRole userRole) throws Exception {
750
751         EGClaim egClaim = escrow.getEgClaim();
752         RaiseClaimHomePage pRaiseClaimHomePage = null;
753         if (user != null)
754             pRaiseClaimHomePage = new BuyerRaiseClaimHomePage(
755                     user.getUserLoginName(), "password");
756         // pRaiseClaimHomePage = new BuyerRaiseClaimHomePage("testgsb1",
757         // "password");
758
759         else
760             pRaiseClaimHomePage = new RaiseClaimHomePage(
761                     PowershipUtil.adminCred.getProperty("eg_csr1"), "password");
762
763         // return raiseClaimDD(pRaiseClaimHomePage, "366829434" , egClaim,
764         // user, userRole);
765         return raiseClaimDD(pRaiseClaimHomePage, escrow.getPaisaPayID(),
766                 egClaim, user, userRole);
767
768     }
769
770     public static synchronized RaiseClaimSuccessPage raiseClaimDD(
771             RaiseClaimHomePage pRaiseClaimHomePage, String paisaPayId,
772             EGClaim egClaim, User user, UserRole userRole) throws Exception {
773
774         RaiseClaimPage pRaiseClaimPage = pRaiseClaimHomePage
775                 .viewOrderDetails(paisaPayId);
776         // add Phone number
777         if (user != null) {
778             if (pRaiseClaimPage.txtBuyerMobile.getAttribute("value") == null
779                     || pRaiseClaimPage.txtBuyerMobile.getAttribute("value")
780                             .equals("")) {
781                 pRaiseClaimPage.lnktBuyerMobileEdit.click();
782                 pRaiseClaimPage
783                         .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
784                 pRaiseClaimPage.txtBuyerMobile.sendKeys(user.getDayPhone1());
785             }
786
787             if (pRaiseClaimPage.txtBuyerLandLine.getAttribute("value") == null
788                     || pRaiseClaimPage.txtBuyerLandLine.getAttribute("value")
789                             .equals("")) {
790                 pRaiseClaimPage.lnktBuyerLandLineEdit.click();
```

```
                pRaiseClaimPage
                        .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
                pRaiseClaimPage.txtBuyerLandLine
                        .sendKeys(user.getNightPhone1());
            }
        } else {
            pRaiseClaimPage.lnktBuyerMobileEdit.click();
            pRaiseClaimPage
                    .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
            pRaiseClaimPage.txtBuyerMobile.clear();
            pRaiseClaimPage.txtBuyerMobile.sendKeys(eGUtil.egCred
                    .getProperty("BUYER_MOBILE"));

            pRaiseClaimPage.lnktBuyerLandLineEdit.click();
            pRaiseClaimPage
                    .waitForElementEditable(pRaiseClaimPage.lnktBuyerLandLineEdit);
            pRaiseClaimPage.txtBuyerLandLine.clear();
            pRaiseClaimPage.txtBuyerLandLine.sendKeys(eGUtil.egCred
                    .getProperty("BUYER_LL"));
        }

        pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());

        // for subtype

        pRaiseClaimPage.chooseSubIssue(egClaim.getSubIssue());

        // Invoice

        if (egClaim.isInvoiceReceived().toLowerCase().equals("na")) {

            Assertion.assertFalse(pRaiseClaimPage.lblInvoice.isDisplayed(),
                    "Invoice Radio button should not be displayed");

        } else {
            if (egClaim.isInvoiceReceived().toLowerCase().equals("yes")) {

                Assertion.assertTrue(pRaiseClaimPage.lblInvoice.isDisplayed(),
                        "Invoice Radio button should be displayed");
                pRaiseClaimPage.rbnYesReceivedInvoice.click();
            } else if (egClaim.isInvoiceReceived().toLowerCase().equals("no")) {
                Assertion.assertTrue(pRaiseClaimPage.lblInvoice.isDisplayed(),
                        "Invoice Radio button should be displayed");
                pRaiseClaimPage.rbnNotReceivedInvoice.click();

            } else {
                Assertion.assertTrue(false,
                        "Wrong input passed for invoice element");
            }
        }

        if (egClaim.getPreferredResolution() != null) {

            pRaiseClaimPage.chooseBuyerPref(egClaim.getPreferredResolution());
        } else {
            egClaim.setPreferredResolution(PreferredResolution.NONE);
        }

        if (egClaim.getRefundPreference() != null) {
            pRaiseClaimPage.selectRefundPreference(egClaim
                    .getRefundPreference());
        }

        // Image Mandatory
        if (egClaim.isImageMandatory().equalsIgnoreCase("NA")) {
            if (userRole.getUserRole().equalsIgnoreCase("BUYER")) {
                Assertion.assertTrue(
```

Writable | Smart Insert | 310 : 67

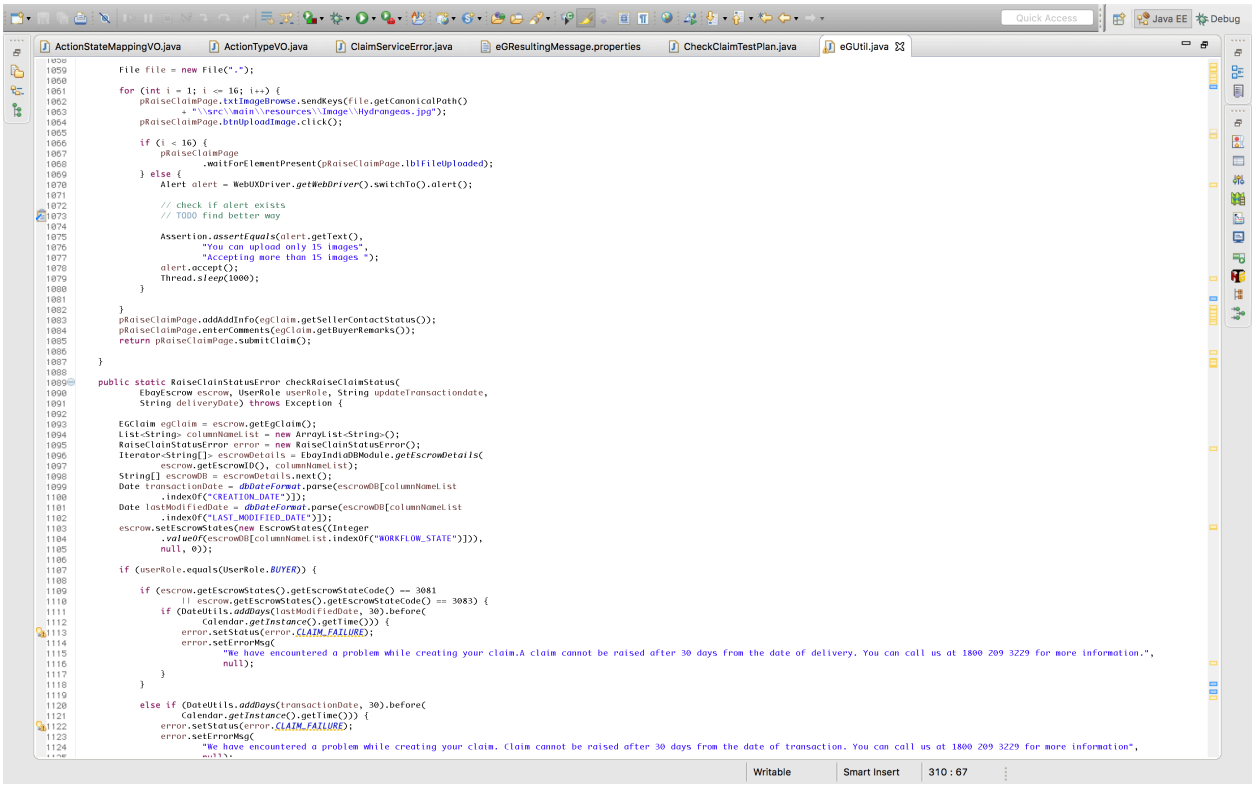
```
                        pRaiseClaimPage.chkMandaryImage.isDisplayed() == false,
                        "Image Mandatory Checkbox should not be displayed");
            } else if (userRole.getUserRole().equalsIgnoreCase("CSR1")) {
                Assertion
                        .assertTrue(pRaiseClaimPage.chkMandaryImage
                                .isElementPresent() == false,
                                "Image Mandatory Checkbox should not be displayed");
            }

        } else if (egClaim.isImageMandatory().equalsIgnoreCase("yes")) {
            Assertion.assertTrue(
                    pRaiseClaimPage.chkMandaryImage.isEnabled() == false,
                    "Image Mandatory Checkbox should be disabled");
        } else if (egClaim.isImageMandatory().equalsIgnoreCase("no")) {
            Assertion.assertTrue(
                    pRaiseClaimPage.chkMandaryImage.isEnabled() == true,
                    "Image Mandatory Checkbox should be enabled");

            // Image disclaimer check box
            if (egClaim.isImageDiscChecked() == true) {
                pRaiseClaimPage.chkMandaryImage.click();
            }
        } else {
            Assertion.assertTrue(false,
                    "Wrong input provided for Image Mandatory input");
        }

        // Image Upload

        if (egClaim.isUploadImage() == true) {
            try {
                File file = new File(".");
                pRaiseClaimPage.txtImageBrowse.sendKeys(file.getCanonicalPath()
                        + "\\src\\main\\resources\\Image\\Hydrangeas.jpg");
                pRaiseClaimPage.btnUploadImage.click();

                pRaiseClaimPage
                        .waitForElementToBeVisible(pRaiseClaimPage.btnImgDelete);
                // ThreadHelper.waitForSeconds(30);
                Assertion.assertTrue(pRaiseClaimPage.btnImgDelete
                        .getAllElements().get(0).isDisplayed(),
                        "Image Delete button is not displaying");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        // Beneficiary address
        if (userRole.getUserRole().equalsIgnoreCase("CSR1")) {
            Assertion.assertFalse(pRaiseClaimPage.lblBenAddressSectionLabel
                    .isElementPresent(),
                    "For CSR1 beneficiary section should not be available");

        } else if (userRole.getUserRole().equalsIgnoreCase("BUYER")
                && !egClaim.getRefundPreference().equals(
                        RefundPreferenceEnum.DD)) {
            Assertion.assertFalse(
                    pRaiseClaimPage.lblBenAddressSectionLabel.isDisplayed(),
                    "For CSR1 beneficiary section should not be available");
        } else if (userRole.getUserRole().equalsIgnoreCase("BUYER")
                && egClaim.getRefundPreference().toString().equals("DD")) {
            Assertion.assertTrue(pRaiseClaimPage.lblBenAddressSectionLabel
                    .isElementPresent(), "For Buyer and Refund Preference as"
                    + " Demand Draft beneficiary section should be available");

            String addressRaiseClaim = pRaiseClaimPage.lblBenAddress.getText()
                    .toUpperCase();
```

Writable | Smart Insert | 310 : 67

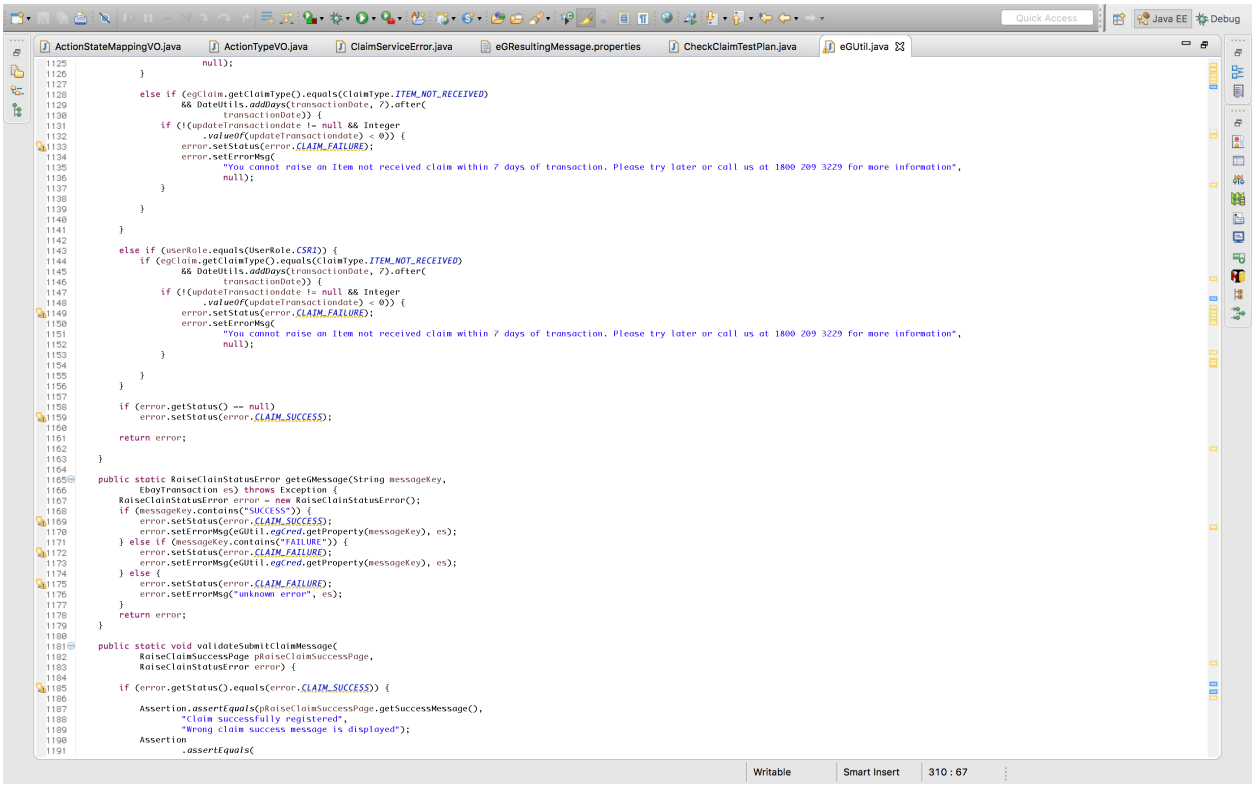


```
            pRaiseClaimPage.btnChangeDDAddress.click();
            String addressPopup = pRaiseClaimPage.txtBenAddress1.getValue()
                    .toUpperCase()
                    + " "
                    + pRaiseClaimPage.txtBenAddress2.getValue().toUpperCase()
                    + " "
                    + pRaiseClaimPage.txtBenCity.getValue().toUpperCase()
                    + " "
                    + pRaiseClaimPage.listState.getSelectedText().toUpperCase()
                    + " "
                    + pRaiseClaimPage.txtBenPinCode.getValue().toUpperCase()
                    + " "
                    + pRaiseClaimPage.txtBenCountry.getValue().toUpperCase()
                    + "\n" + pRaiseClaimPage.txtBenPhone.getValue();

            Assertion
                    .assertEquals(addressRaiseClaim, addressPopup,
                            "Address in the Raise Claim page and address popup is not matching");

            pRaiseClaimPage.txtBenAddress1.clear();
            ;
            pRaiseClaimPage.txtBenAddress1.sendKeys("78, Queen Street");
            pRaiseClaimPage.txtBenAddress2.clear();
            pRaiseClaimPage.txtBenAddress2.sendKeys("Park Avenue");
            pRaiseClaimPage.txtBenPinCode.clear();
            pRaiseClaimPage.txtBenPinCode.sendKeys("560037");
            Thread.sleep(2000);
            Assertion.assertFalse(pRaiseClaimPage.txtBenCity.isEnabled(),
                    "After autopopulation city should not be editable");
            Assertion.assertFalse(pRaiseClaimPage.listState.isEnabled(),
                    "After autopopulation State should not be editable");
            Assertion.assertFalse(pRaiseClaimPage.txtBenCountry.isEnabled(),
                    "After autopopulation Country should not be editable");
            pRaiseClaimPage.txtBenPhone.clear();
            pRaiseClaimPage.txtBenPhone.sendKeys("9685748526");
            pRaiseClaimPage.btnConfirmAddressButton.click();
            Assertion
                    .assertTrue(
                            pRaiseClaimPage.lblBenDisclaimerWarnning
                                    .getText()
                                    .contains(
                                            "Please check the disclaimer confirming that your Name and Address are correct before submitting"),
                            "Disclaimer should be displayed with correct text");
            pRaiseClaimPage.chkBenAddressDisclaimer.click();
            pRaiseClaimPage.btnConfirmAddressButton.click();
            Thread.sleep(3000);

            //
            addressPopup = ("78, Queen Street Park Avenue BANGALORE KARNATAKA 560037 IN"
                    + "\n" + "9685748526").toUpperCase();
            addressRaiseClaim = pRaiseClaimPage.lblBenAddress.getText()
                    .toUpperCase();

            Assertion.assertEquals(addressPopup, addressRaiseClaim,
                    "Beneficiary Address should get updated");

            pRaiseClaimPage.btnChangeDDAddress.click();
            String benName = pRaiseClaimPage.txtBenName.getText();
            Assertion.assertFalse(pRaiseClaimPage.txtBenName.isEnabled(),
                    "Beneficiary Name field should not be editable");

            pRaiseClaimPage.updateBenName(user.getUserLoginName(), "Gaurav",
                    "A", "Shukla");

            pRaiseClaimPage.btnChangeDDAddress.click();
            Thread.sleep(3000);
```

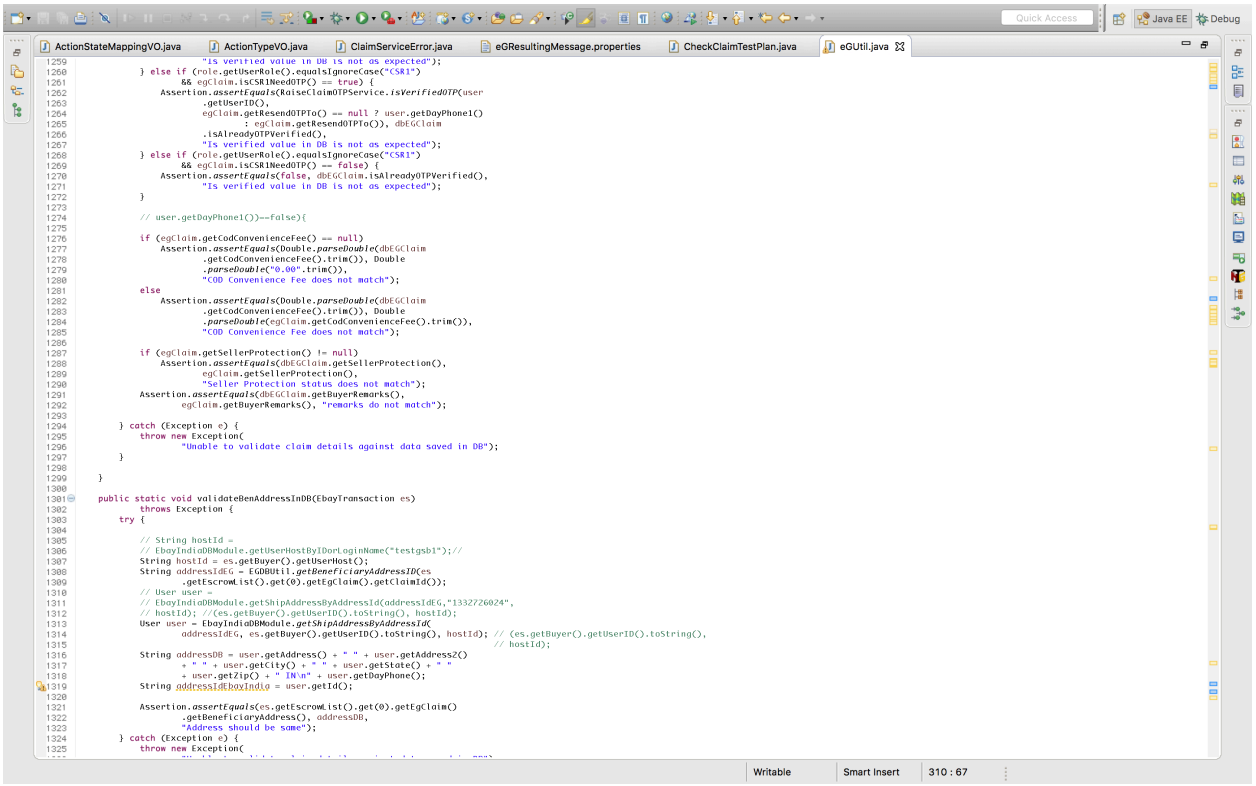
```
            Assertion.assertEquals(pRaiseClaimPage.txtBenName.getValue()
                    .toUpperCase(), "GAURAV A SHUKLA",
                    "Beneficiary Name should get updated");

            pRaiseClaimPage.btnCancel.click();

        }

        pRaiseClaimPage.addAddInfo(egClaim.getSellerContactStatus());
        pRaiseClaimPage.enterComments(egClaim.getBuyerRemarks());
        return pRaiseClaimPage.submitClaim();

    }

    // Added by Saronya
    public static RaiseClaimSuccessPage raiseClaimUploadingImage(
            EbayEscrow escrow, User user) throws Exception {

        EGClaim egClaim = escrow.getEgClaim();
        RaiseClaimHomePage pRaiseClaimHomePage = null;
        if (user != null)
            pRaiseClaimHomePage = new BuyerRaiseClaimHomePage(
                    user.getUserLoginName(), "password");
        else
            pRaiseClaimHomePage = new RaiseClaimHomePage(
                    PowershipUtil.adminCred.getProperty("eg_csr1"), "password");

        return raiseClaimUploadingImage(pRaiseClaimHomePage,
                escrow.getPaisaPayID(), egClaim, user);

    }

    // Added by Saronya

    public static synchronized RaiseClaimSuccessPage raiseClaimUploadingImage(
            RaiseClaimHomePage pRaiseClaimHomePage, String paisaPayId,
            EGClaim egClaim, User user) throws Exception {

        RaiseClaimPage pRaiseClaimPage = pRaiseClaimHomePage
                .viewOrderDetails(paisaPayId);

        // add Phone number
        if (pRaiseClaimPage.txtBuyerMobile.getAttribute("value") == null
                || pRaiseClaimPage.txtBuyerMobile.getAttribute("value").equals(
                        "")) {

            if (user == null
                    || (user.getDayPhone1() != null && !user.getDayPhone1()
                            .trim().equalsIgnoreCase(""))) {
                pRaiseClaimPage.lnktBuyerMobileEdit.click();
                pRaiseClaimPage
                        .waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
                if (user == null)
                    pRaiseClaimPage.txtBuyerMobile.sendKeys(eGUtil.egCred
                            .getProperty("BUYER_MOBILE"));
                else
                    pRaiseClaimPage.txtBuyerMobile
                            .sendKeys(user.getDayPhone1());

            }
        }

        pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());
        pRaiseClaimPage.chooseBuyerPref(egClaim.getPreferredResolution());
        pRaiseClaimPage.selectRefundPreference(egClaim.getRefundPreference());

```



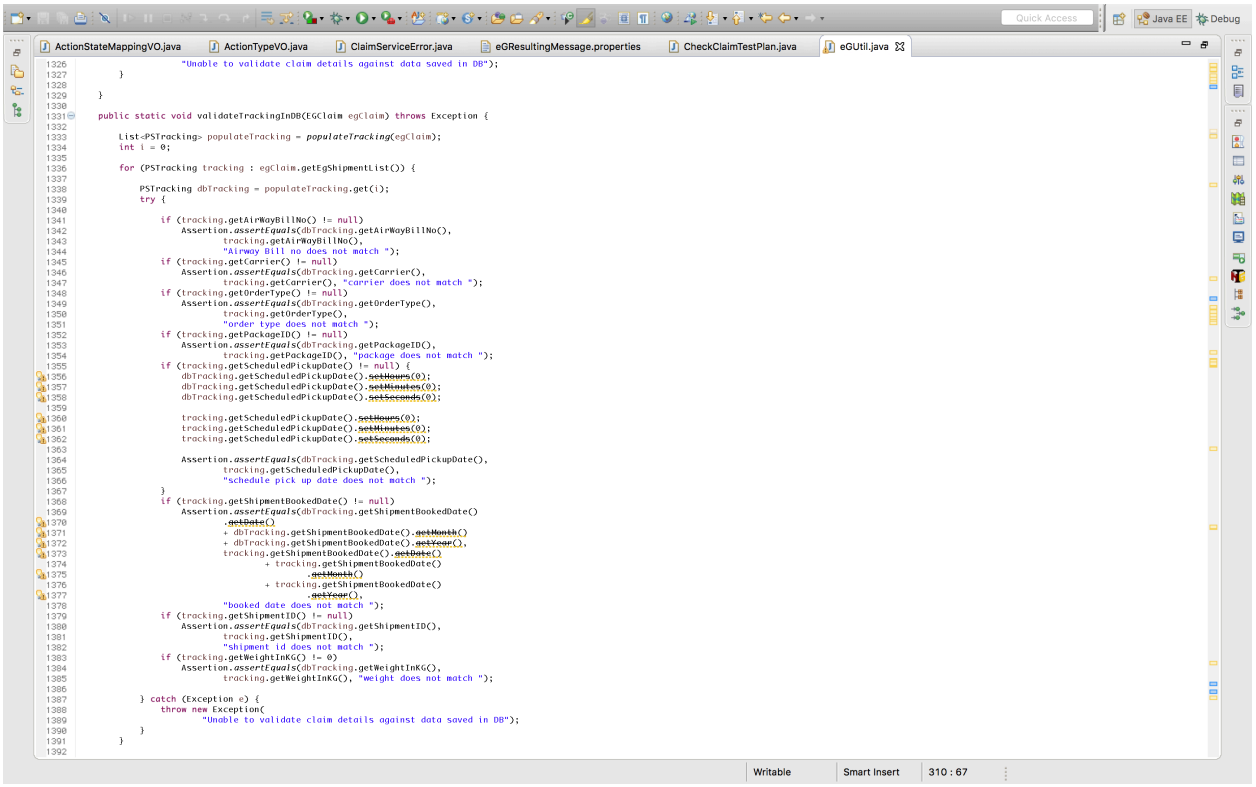
```
        File file = new File(".");

        for (int i = 1; i <= 16; i++) {
            pRaiseClaimPage.txtImageBrowse.sendKeys(file.getCanonicalPath()
                    + "\\src\\main\\resources\\Image\\Hydrangeas.jpg");
            pRaiseClaimPage.btnUploadImage.click();

            if (i < 16) {
                pRaiseClaimPage
                        .waitForElementPresent(pRaiseClaimPage.lblFileUploaded);
            } else {
                Alert alert = WebUXDriver.getWebDriver().switchTo().alert();

                // check if alert exists
                // TODO find better way

                Assertion.assertEquals(alert.getText(),
                        "You can upload only 15 images",
                        "Accepting more than 15 images ");
                alert.accept();
                Thread.sleep(1000);
            }

        }
        pRaiseClaimPage.addAddInfo(egClaim.getSellerContactStatus());
        pRaiseClaimPage.enterComments(egClaim.getBuyerRemarks());
        return pRaiseClaimPage.submitClaim();

    }

    public static RaiseClainStatusError checkRaiseClaimStatus(
            EbayEscrow escrow, UserRole userRole, String updateTransactiondate,
            String deliveryDate) throws Exception {

        EGClaim egClaim = escrow.getEgClaim();
        List<String> columnNameList = new ArrayList<String>();
        RaiseClainStatusError error = new RaiseClainStatusError();
        Iterator<String[]> escrowDetails = EbayIndiaDBModule.getEscrowDetails(
                escrow.getEscrowID(), columnNameList);
        String[] escrowDB = escrowDetails.next();
        Date transactionDate = dbDateFormat.parse(escrowDB[columnNameList
                .indexOf("CREATION_DATE")]);
        Date lastModifiedDate = dbDateFormat.parse(escrowDB[columnNameList
                .indexOf("LAST_MODIFIED_DATE")]);
        escrow.setEscrowStates(new EscrowStates((Integer
                .valueOf(escrowDB[columnNameList.indexOf("WORKFLOW_STATE")])),
                null, 0));

        if (userRole.equals(UserRole.BUYER)) {

            if (escrow.getEscrowStates().getEscrowStateCode() == 3081
                    || escrow.getEscrowStates().getEscrowStateCode() == 3083) {
                if (DateUtils.addDays(lastModifiedDate, 30).before(
                        Calendar.getInstance().getTime())) {
                    error.setStatus(error.CLAIM_FAILURE);
                    error.setErrorMsg(
                            "We have encountered a problem while creating your claim.A claim cannot be raised after 30 days from the date of delivery. You can call us at 1800 209 3229 for more information.",
                            null);
                }
            }

            else if (DateUtils.addDays(transactionDate, 30).before(
                    Calendar.getInstance().getTime())) {
                error.setStatus(error.CLAIM_FAILURE);
                error.setErrorMsg(
                        "We have encountered a problem while creating your claim. Claim cannot be raised after 30 days from the date of transaction. You can call us at 1800 209 3229 for more information",
                        null);
            }

            else if (egClaim.getClaimType().equals(ClaimType.ITEM_NOT_RECEIVED)
                    && DateUtils.addDays(transactionDate, 7).after(
                            transactionDate)) {
                if (!(updateTransactiondate != null && Integer
                        .valueOf(updateTransactiondate) < 0)) {
                    error.setStatus(error.CLAIM_FAILURE);
                    error.setErrorMsg(
                            "You cannot raise an Item not received claim within 7 days of transaction. Please try later or call us at 1800 209 3229 for more information",
                            null);
                }

            }

        }

        else if (userRole.equals(UserRole.CSR1)) {
            if (egClaim.getClaimType().equals(ClaimType.ITEM_NOT_RECEIVED)
                    && DateUtils.addDays(transactionDate, 7).after(
                            transactionDate)) {
                if (!(updateTransactiondate != null && Integer
                        .valueOf(updateTransactiondate) < 0)) {
                    error.setStatus(error.CLAIM_FAILURE);
                    error.setErrorMsg(
                            "You cannot raise an Item not received claim within 7 days of transaction. Please try later or call us at 1800 209 3229 for more information",
                            null);
                }

            }
        }

        if (error.getStatus() == null)
            error.setStatus(error.CLAIM_SUCCESS);

        return error;

    }

    public static RaiseClainStatusError geteGMessage(String messageKey,
            EbayTransaction es) throws Exception {
        RaiseClainStatusError error = new RaiseClainStatusError();
        if (messageKey.contains("SUCCESS")) {
            error.setStatus(error.CLAIM_SUCCESS);
            error.setErrorMsg(eGUtil.egCred.getProperty(messageKey), es);
        } else if (messageKey.contains("FAILURE")) {
            error.setStatus(error.CLAIM_FAILURE);
            error.setErrorMsg(eGUtil.egCred.getProperty(messageKey), es);
        } else {
            error.setStatus(error.CLAIM_FAILURE);
            error.setErrorMsg("unknown error", es);
        }
        return error;
    }

    public static void validateSubmitClaimMessage(
            RaiseClaimSuccessPage pRaiseClaimSuccessPage,
            RaiseClainStatusError error) {

        if (error.getStatus().equals(error.CLAIM_SUCCESS)) {

            Assertion.assertEquals(pRaiseClaimSuccessPage.getSuccessMessage(),
                    "Claim successfully registered",
                    "Wrong claim success message is displayed");
            Assertion
                    .assertEquals(
```



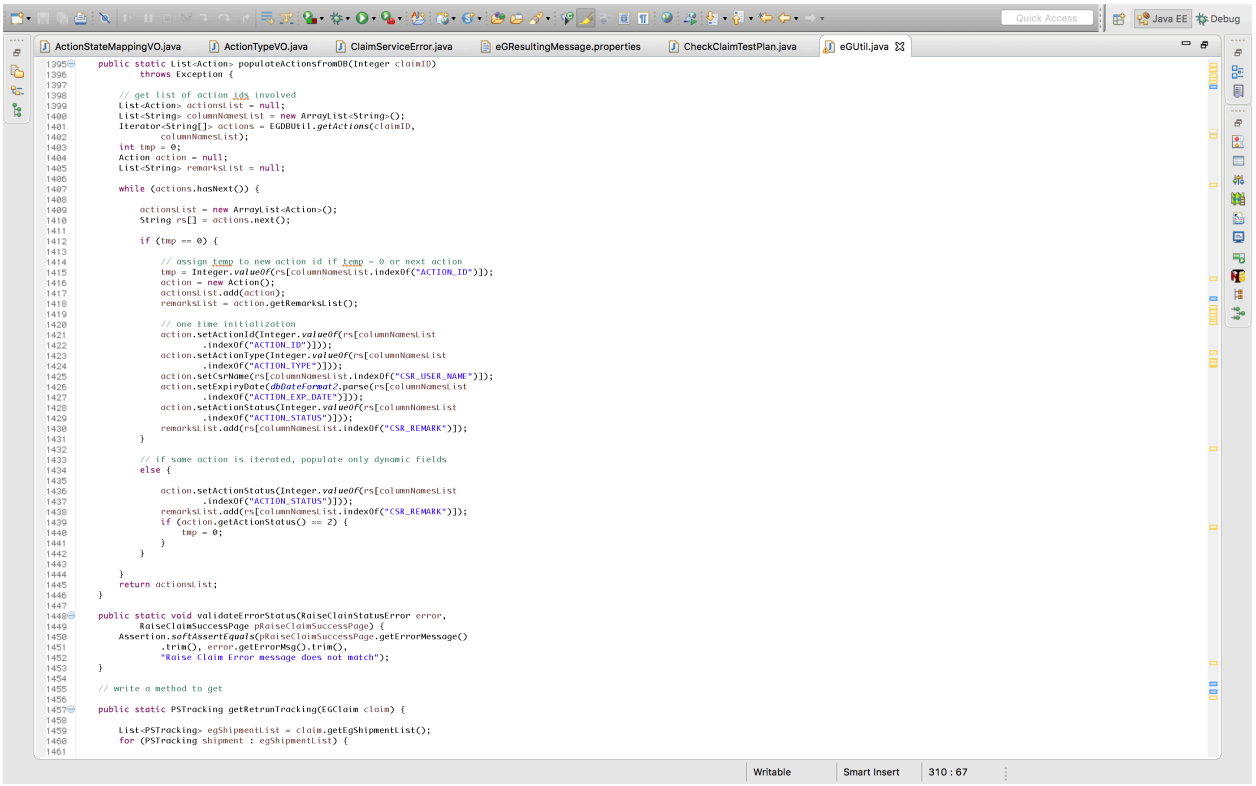
```
                        pRaiseClaimSuccessPage.getSuccessMessage1(),
                        "Please save the claimid for future reference and communication.",
                        "Wrong claim success message is displayed");
            Assertion
                    .assertEquals(
                        pRaiseClaimSuccessPage.getSuccessMessage1(),
                        "Please save the claimid for future reference and communication.",
                        "Wrong claim success message is displayed");
        } else {
            Assertion
                    .assertEquals(
                        pRaiseClaimSuccessPage.getSuccessMessage(),
                        "We have encountered a problem while creating your claim. Please try again later.",
                        "Wrong error message is displayed");
            Assertion.assertEquals(pRaiseClaimSuccessPage.getSuccessMessage1(),
                    error.getErrorMsg(), "Wrong error message is displayed");

            Assertion.assertEquals(pRaiseClaimSuccessPage.getSuccessMessage1(),
                    "Or call us at 1800 209 3229",
                    "Wrong error message is displayed");
        }
    }

    public static void validateClaiminDB(EGClaim egClaim, UserRole role,
            User user) throws Exception {
        try {
            EGClaim dbEGClaim = new EGClaim(egClaim.getClaimId());

            Assertion.assertEquals(dbEGClaim.getClaimStatus(),
                    egClaim.getClaimStatus(), "Claim status does not match");
            Assertion.assertEquals(dbEGClaim.getClaimType(),
                    egClaim.getClaimType(), "claim type does not match");
            // for subtype and invoice
            Assertion.assertEquals(dbEGClaim.getSubIssue(),
                    egClaim.getSubIssue(), "Sub Issue type does not match");

            Assertion.assertEquals(dbEGClaim.isInvoiceReceived().toUpperCase(),
                    egClaim.isInvoiceReceived().toUpperCase(),
                    "Invoice Received Question's value does not match");

            Assertion.assertEquals(dbEGClaim.getSellerContactStatus(),
                    egClaim.getSellerContactStatus(),
                    "Contact status does not match");
            Assertion.assertEquals(dbEGClaim.getPreferredResolution(),
                    egClaim.getPreferredResolution(),
                    "Preferred resolution does not match");
            Assertion.assertEquals(dbEGClaim.getPaymentMethod().trim()
                    .toUpperCase(), egClaim.getPaymentMethod().toUpperCase()
                    .trim(), "payment method does not match");
            if (role.getUserRole().equals("BUYER")) {
                Assertion.assertTrue(
                        dbEGClaim.isImageMandatory().contains(
                                egClaim.isImageMandatory().toUpperCase()),
                        "Image mandatory value is not matching in DB");
            } else if (role.getUserRole().equals("CSR1")) {
                Assertion.assertTrue(dbEGClaim.isImageMandatory()
                        .contains("NA"),
                        "Image mandatory value is not matching in DB");
            }

            // Number OTP Verified
            if (role.getUserRole().equalsIgnoreCase("BUYER")) {
                Assertion.assertEquals(RaiseClaimOTPService.isVerifiedOTP(user
                        .getUserID(),
                        egClaim.getResendOTPTo() == null ? user.getDayPhone1()
                                : egClaim.getResendOTPTo()), dbEGClaim
                        .isAlreadyOTPVerified(),
```



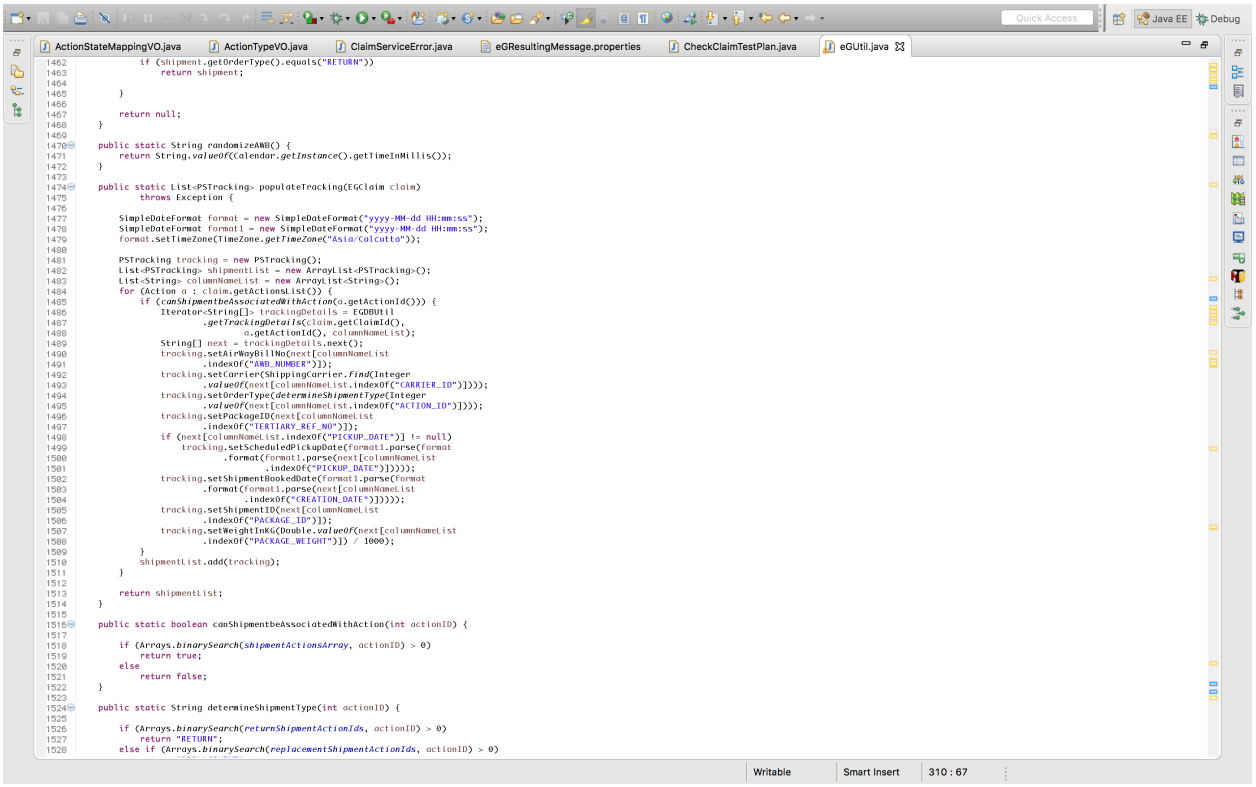
```
                        "Is verified value in DB is not as expected");
            } else if (role.getUserRole().equalsIgnoreCase("CSR1")
                    && egClaim.isCSR1NeedOTP() == true) {
                Assertion.assertEquals(RaiseClaimOTPService.isVerifiedOTP(user
                        .getUserID(),
                        egClaim.getResendOTPTo() == null ? user.getDayPhone1()
                                : egClaim.getResendOTPTo()), dbEGClaim
                        .isAlreadyOTPVerified(),
                        "Is verified value in DB is not as expected");
            } else if (role.getUserRole().equalsIgnoreCase("CSR1")
                    && egClaim.isCSR1NeedOTP() == false) {
                Assertion.assertEquals(false, dbEGClaim.isAlreadyOTPVerified(),
                        "Is verified value in DB is not as expected");
            }

            // user.getDayPhone1())==false){

            if (egClaim.getCodConvenienceFee() == null)
                Assertion.assertEquals(Double.parseDouble(dbEGClaim
                        .getCodConvenienceFee().trim()), Double
                        .parseDouble("0.00".trim()),
                        "COD Convenience Fee does not match");
            else
                Assertion.assertEquals(Double.parseDouble(dbEGClaim
                        .getCodConvenienceFee().trim()), Double
                        .parseDouble(egClaim.getCodConvenienceFee().trim()),
                        "COD Convenience Fee does not match");

            if (egClaim.getSellerProtection() != null)
                Assertion.assertEquals(dbEGClaim.getSellerProtection(),
                        egClaim.getSellerProtection(),
                        "Seller Protection status does not match");
            Assertion.assertEquals(dbEGClaim.getBuyerRemarks(),
                    egClaim.getBuyerRemarks(), "remarks do not match");

        } catch (Exception e) {
            throw new Exception(
                    "Unable to validate claim details against data saved in DB");
        }

    }

    public static void validateBenAddressInDB(EbayTransaction es)
            throws Exception {
        try {

            // String hostId =
            // EbayIndiaDBModule.getUserHostByIDorLoginName("testgsb1");//
            String hostId = es.getBuyer().getUserHost();
            String addressIdEG = EGDBUtil.getBeneficiaryAddressID(es
                    .getEscrowList().get(0).getEgClaim().getClaimId());
            // User user =
            // EbayIndiaDBModule.getShipAddressByAddressId(addressIdEG,"1332726024",
            // hostId); //(es.getBuyer().getUserID().toString(), hostId);
            User user = EbayIndiaDBModule.getShipAddressByAddressId(
                    addressIdEG, es.getBuyer().getUserID().toString(), hostId); // (es.getBuyer().getUserID().toString(),
                                                                                // hostId);
            String addressDB = user.getAddress() + " " + user.getAddress2()
                    + " " + user.getCity() + " " + user.getState() + " "
                    + user.getZip() + " IN\n" + user.getDayPhone();
            String addressIdEbayIndia = user.getId();

            Assertion.assertEquals(es.getEscrowList().get(0).getEgClaim()
                    .getBeneficiaryAddress(), addressDB,
                    "Address should be same");
        } catch (Exception e) {
            throw new Exception(
```



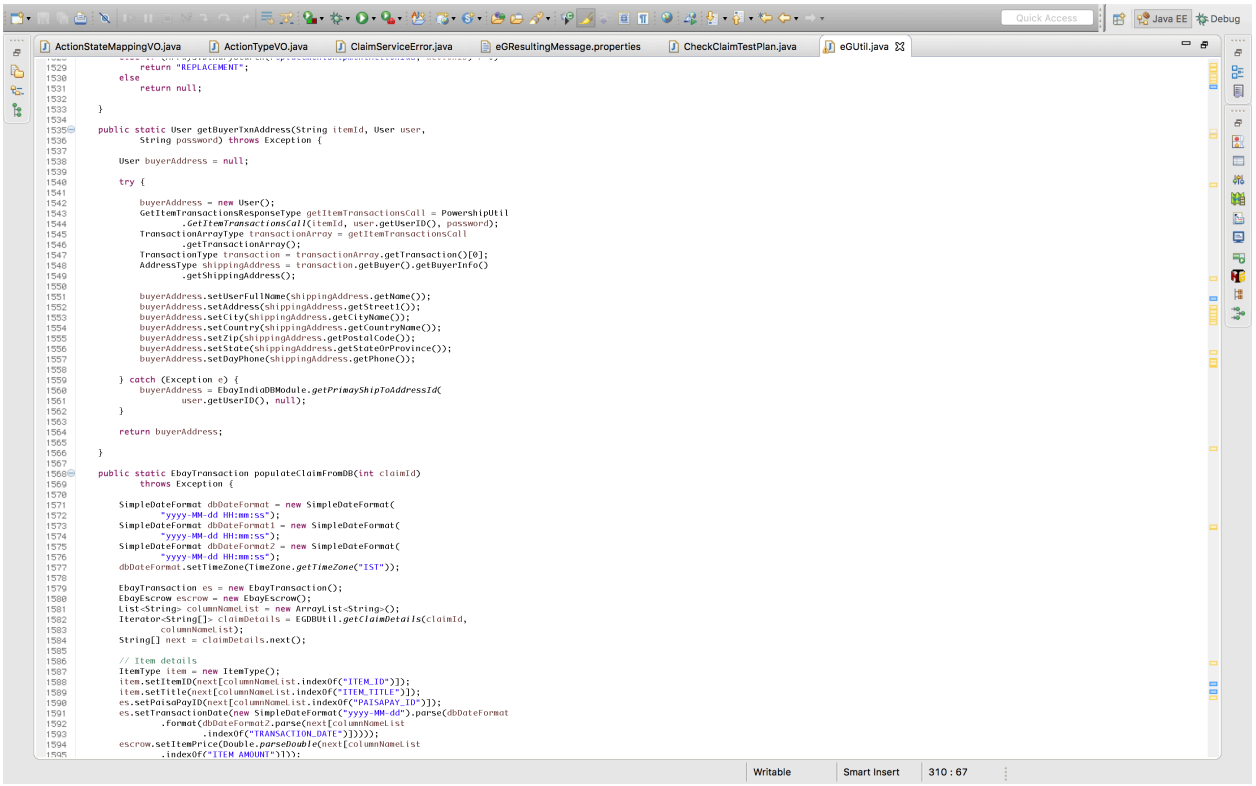
```
                    "Unable to validate claim details against data saved in DB");
        }
    }

    public static void validateTrackingInDB(EGClaim egClaim) throws Exception {

        List<PSTracking> populateTracking = populateTracking(egClaim);
        int i = 0;

        for (PSTracking tracking : egClaim.getEgShipmentList()) {

            PSTracking dbTracking = populateTracking.get(i);
            try {

                if (tracking.getAirWayBillNo() != null)
                    Assertion.assertEquals(dbTracking.getAirWayBillNo(),
                            tracking.getAirWayBillNo(),
                            "Airway Bill no does not match ");
                if (tracking.getCarrier() != null)
                    Assertion.assertEquals(dbTracking.getCarrier(),
                            tracking.getCarrier(), "carrier does not match ");
                if (tracking.getOrderType() != null)
                    Assertion.assertEquals(dbTracking.getOrderType(),
                            tracking.getOrderType(),
                            "order type does not match ");
                if (tracking.getPackageID() != null)
                    Assertion.assertEquals(dbTracking.getPackageID(),
                            tracking.getPackageID(), "package does not match ");
                if (tracking.getScheduledPickupDate() != null) {
                    dbTracking.getScheduledPickupDate().setHours(0);
                    dbTracking.getScheduledPickupDate().setMinutes(0);
                    dbTracking.getScheduledPickupDate().setSeconds(0);

                    tracking.getScheduledPickupDate().setHours(0);
                    tracking.getScheduledPickupDate().setMinutes(0);
                    tracking.getScheduledPickupDate().setSeconds(0);

                    Assertion.assertEquals(dbTracking.getScheduledPickupDate(),
                            tracking.getScheduledPickupDate(),
                            "schedule pick up date does not match ");
                }
                if (tracking.getShipmentBookedDate() != null)
                    Assertion.assertEquals(dbTracking.getShipmentBookedDate()
                            .getDate()
                            + dbTracking.getShipmentBookedDate().getMonth()
                            + dbTracking.getShipmentBookedDate().getYear(),
                            tracking.getShipmentBookedDate().getDate()
                                    + tracking.getShipmentBookedDate()
                                            .getMonth()
                                    + tracking.getShipmentBookedDate()
                                            .getYear(),
                            "booked date does not match ");
                if (tracking.getShipmentID() != null)
                    Assertion.assertEquals(dbTracking.getShipmentID(),
                            tracking.getShipmentID(),
                            "Shipment id does not match ");
                if (tracking.getWeightInKG() != 0)
                    Assertion.assertEquals(dbTracking.getWeightInKG(),
                            tracking.getWeightInKG(), "weight does not match ");

            } catch (Exception e) {
                throw new Exception(
                        "Unable to validate claim details against data saved in DB");
            }
        }
```



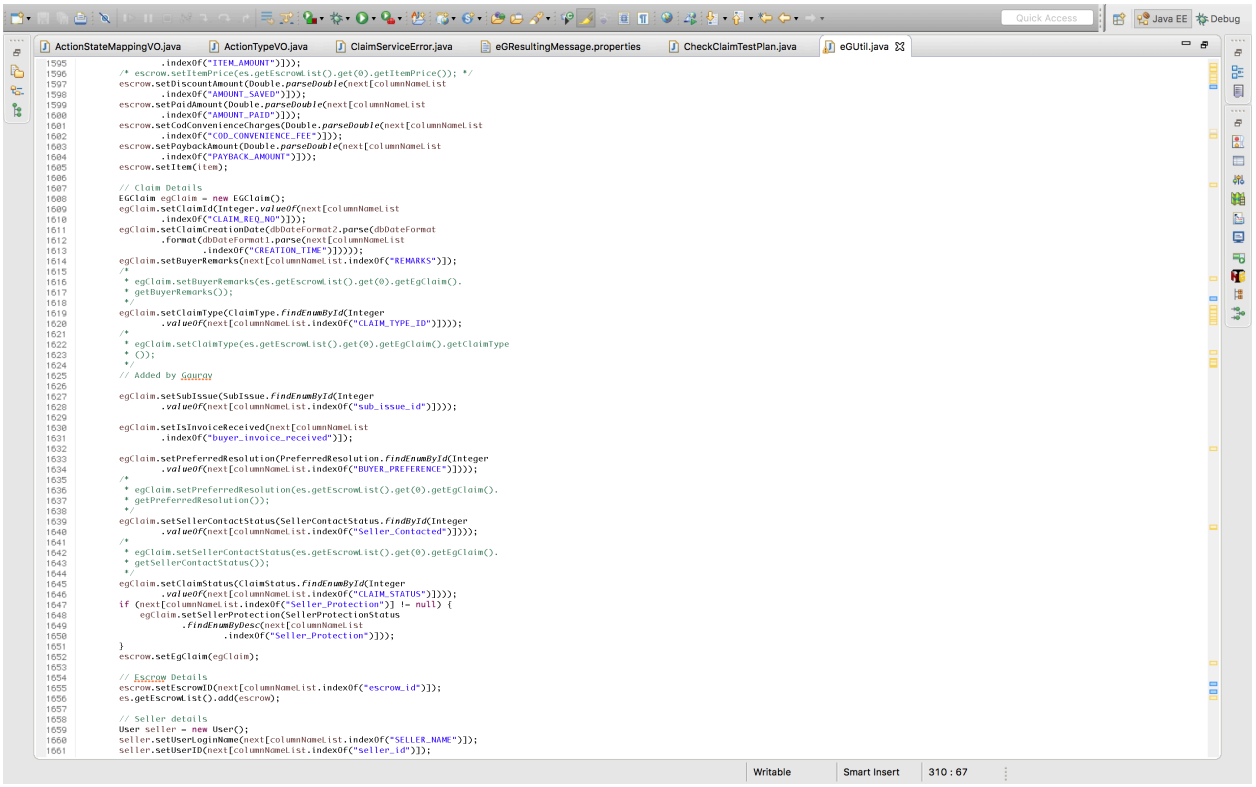
```
    public static List<Action> populateActionsfromDB(Integer claimID)
            throws Exception {

        // get list of action ids involved
        List<Action> actionsList = null;
        List<String> columnNamesList = new ArrayList<String>();
        Iterator<String[]> actions = EGDBUtil.getActions(claimID,
                columnNamesList);
        int tmp = 0;
        Action action = null;
        List<String> remarksList = null;

        while (actions.hasNext()) {

            actionsList = new ArrayList<Action>();
            String rs[] = actions.next();

            if (tmp == 0) {

                // assign temp to new action id if temp = 0 or next action
                tmp = Integer.valueOf(rs[columnNamesList.indexOf("ACTION_ID")]);
                action = new Action();
                actionsList.add(action);
                remarksList = action.getRemarksList();

                // one time initialization
                action.setActionId(Integer.valueOf(rs[columnNamesList
                        .indexOf("ACTION_ID")]));
                action.setActionType(Integer.valueOf(rs[columnNamesList
                        .indexOf("ACTION_TYPE")]));
                action.setCsrName(rs[columnNamesList.indexOf("CSR_USER_NAME")]);
                action.setExpiryDate(dbDateFormat2.parse(rs[columnNamesList
                        .indexOf("ACTION_EXP_DATE")]));
                action.setActionStatus(Integer.valueOf(rs[columnNamesList
                        .indexOf("ACTION_STATUS")]));
                remarksList.add(rs[columnNamesList.indexOf("CSR_REMARK")]);
            }

            // if same action is iterated, populate only dynamic fields
            else {

                action.setActionStatus(Integer.valueOf(rs[columnNamesList
                        .indexOf("ACTION_STATUS")]));
                remarksList.add(rs[columnNamesList.indexOf("CSR_REMARK")]);
                if (action.getActionStatus() == 2) {
                    tmp = 0;
                }
            }

        }
        return actionsList;
    }

    public static void validateErrorStatus(RaiseClainStatusError error,
            RaiseClaimSuccessPage pRaiseClaimSuccessPage) {
        Assertion.softAssertEquals(pRaiseClaimSuccessPage.getErrorMessage()
                .trim(), error.getErrorMsg().trim(),
                "Raise Claim Error message does not match");
    }

    // write a method to get

    public static PSTracking getRetrumTracking(EGClaim claim) {

        List<PSTracking> egShipmentList = claim.getEgShipmentList();
        for (PSTracking shipment : egShipmentList) {
```



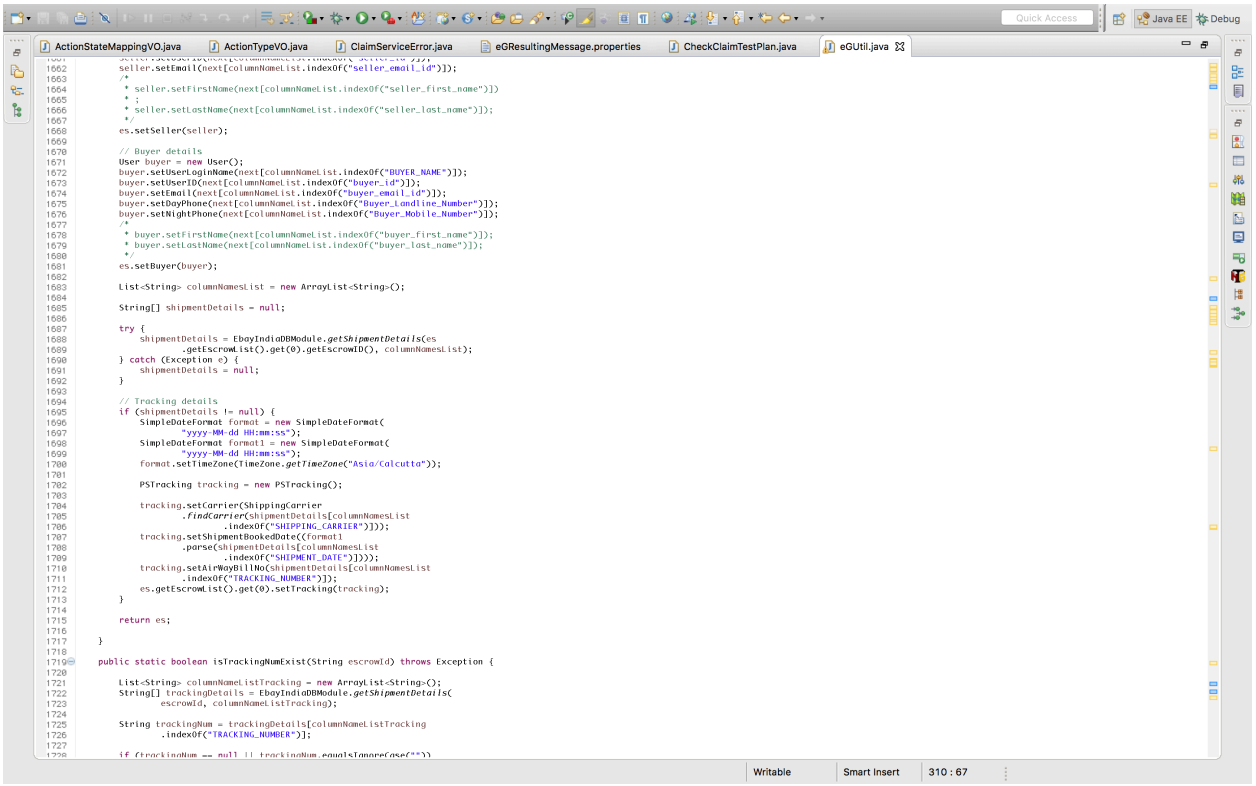
```
            if (shipment.getOrderType().equals("RETURN"))
                return shipment;

        }

        return null;
    }

    public static String randomizeAWB() {
        return String.valueOf(calendar.getInstance().getTimeInMillis());
    }

    public static List<PSTracking> populateTracking(EGClaim claim)
            throws Exception {

        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        format.setTimeZone(TimeZone.getTimeZone("Asia/Calcutta"));

        PSTracking tracking = new PSTracking();
        List<PSTracking> shipmentList = new ArrayList<PSTracking>();
        List<String> columnNameList = new ArrayList<String>();
        for (Action a : claim.getActionsList()) {
            if (canShipmentbeAssociatedWithAction(a.getActionId())) {
                Iterator<String[]> trackingDetails = EGDBUtil
                        .getTrackingDetails(claim.getClaimId(),
                                a.getActionId(), columnNameList);
                String[] next = trackingDetails.next();
                tracking.setAirWayBillNo(next[columnNameList
                        .indexOf("AWB_NUMBER")]);
                tracking.setCarrier(ShippingCarrier.find(Integer
                        .valueOf(next[columnNameList.indexOf("CARRIER_ID")])));
                tracking.setOrderType(determineShipmentType(Integer
                        .valueOf(next[columnNameList.indexOf("ACTION_ID")])));
                tracking.setPackageID(next[columnNameList
                        .indexOf("TERTIARY_REF_NO")]);
                if (next[columnNameList.indexOf("PICKUP_DATE")] != null)
                    tracking.setScheduledPickupDate(format1.parse(format
                            .format(format1.parse(next[columnNameList
                                    .indexOf("PICKUP_DATE")]))));
                tracking.setShipmentBookedDate(format1.parse(format
                        .format(format1.parse(next[columnNameList
                                .indexOf("CREATION_DATE")]))));
                tracking.setShipmentID(next[columnNameList
                        .indexOf("PACKAGE_ID")]);
                tracking.setWeightInKG(Double.valueOf(next[columnNameList
                        .indexOf("PACKAGE_WEIGHT")]) / 1000);
            }
            shipmentList.add(tracking);
        }

        return shipmentList;
    }

    public static boolean canShipmentbeAssociatedWithAction(int actionID) {

        if (Arrays.binarySearch(shipmentActionsArray, actionID) > 0)
            return true;
        else
            return false;
    }

    public static String determineShipmentType(int actionID) {

        if (Arrays.binarySearch(returnShipmentActionIds, actionID) > 0)
            return "RETURN";
        else if (Arrays.binarySearch(replacementShipmentActionIds, actionID) > 0)
```



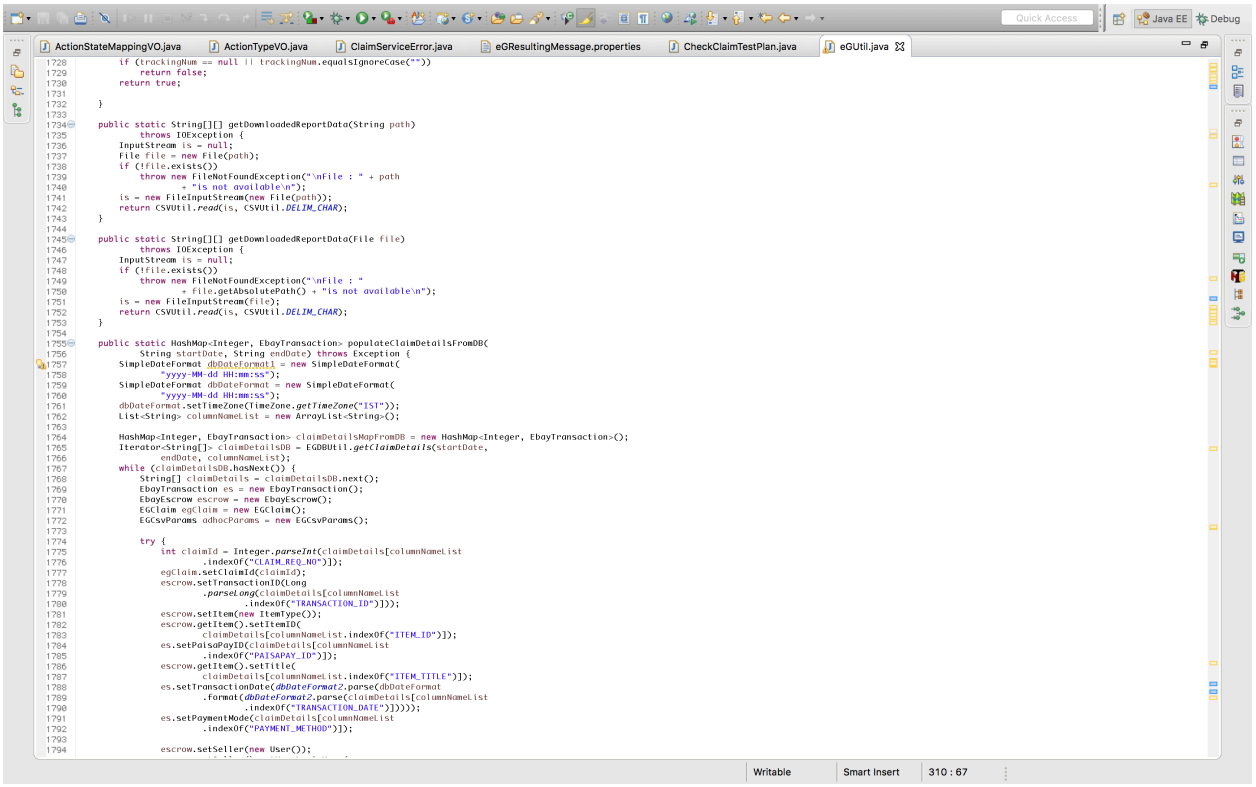
```
            return "REPLACEMENT";
        else
            return null;

    }

    public static User getBuyerTxnAddress(String itemId, User user,
            String password) throws Exception {

        User buyerAddress = null;

        try {

            buyerAddress = new User();
            GetItemTransactionsResponseType getItemTransactionsCall = PowershipUtil
                    .GetItemTransactionsCall(itemId, user.getUserID(), password);
            TransactionArrayType transactionArray = getItemTransactionsCall
                    .getTransactionArray();
            TransactionType transaction = transactionArray.getTransaction()[0];
            AddressType shippingAddress = transaction.getBuyer().getBuyerInfo()
                    .getShippingAddress();

            buyerAddress.setUserFullName(shippingAddress.getName());
            buyerAddress.setAddress(shippingAddress.getStreet1());
            buyerAddress.setCity(shippingAddress.getCityName());
            buyerAddress.setCountry(shippingAddress.getCountryName());
            buyerAddress.setZip(shippingAddress.getPostalCode());
            buyerAddress.setState(shippingAddress.getStateOrProvince());
            buyerAddress.setDayPhone(shippingAddress.getPhone());

        } catch (Exception e) {
            buyerAddress = EbayIndiaDBModule.getPrimaryShipToAddressId(
                    user.getUserID(), null);
        }

        return buyerAddress;

    }

    public static EbayTransaction populateClaimFromDB(int claimId)
            throws Exception {

        SimpleDateFormat dbDateFormat = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        SimpleDateFormat dbDateFormat2 = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        dbDateFormat.setTimeZone(TimeZone.getTimeZone("IST"));

        EbayTransaction es = new EbayTransaction();
        EbayEscrow escrow = new EbayEscrow();
        List<String> columnNameList = new ArrayList<String>();
        Iterator<String[]> claimDetails = EGDBUtil.getClaimDetails(claimId,
                columnNameList);
        String[] next = claimDetails.next();

        // Item details
        ItemType item = new ItemType();
        item.setItemID(next[columnNameList.indexOf("ITEM_ID")]);
        item.setTitle(next[columnNameList.indexOf("ITEM_TITLE")]);
        es.setPaisaPayID(next[columnNameList.indexOf("PAISAPAY_ID")]);
        es.setTransactionDate(new SimpleDateFormat("yyyy-MM-dd").parse(dbDateFormat
                .format(dbDateFormat2.parse(next[columnNameList
                        .indexOf("TRANSACTION_DATE")]))));
        escrow.setItemPrice(Double.parseDouble(next[columnNameList
                .indexOf("ITEM_AMOUNT")]));
```



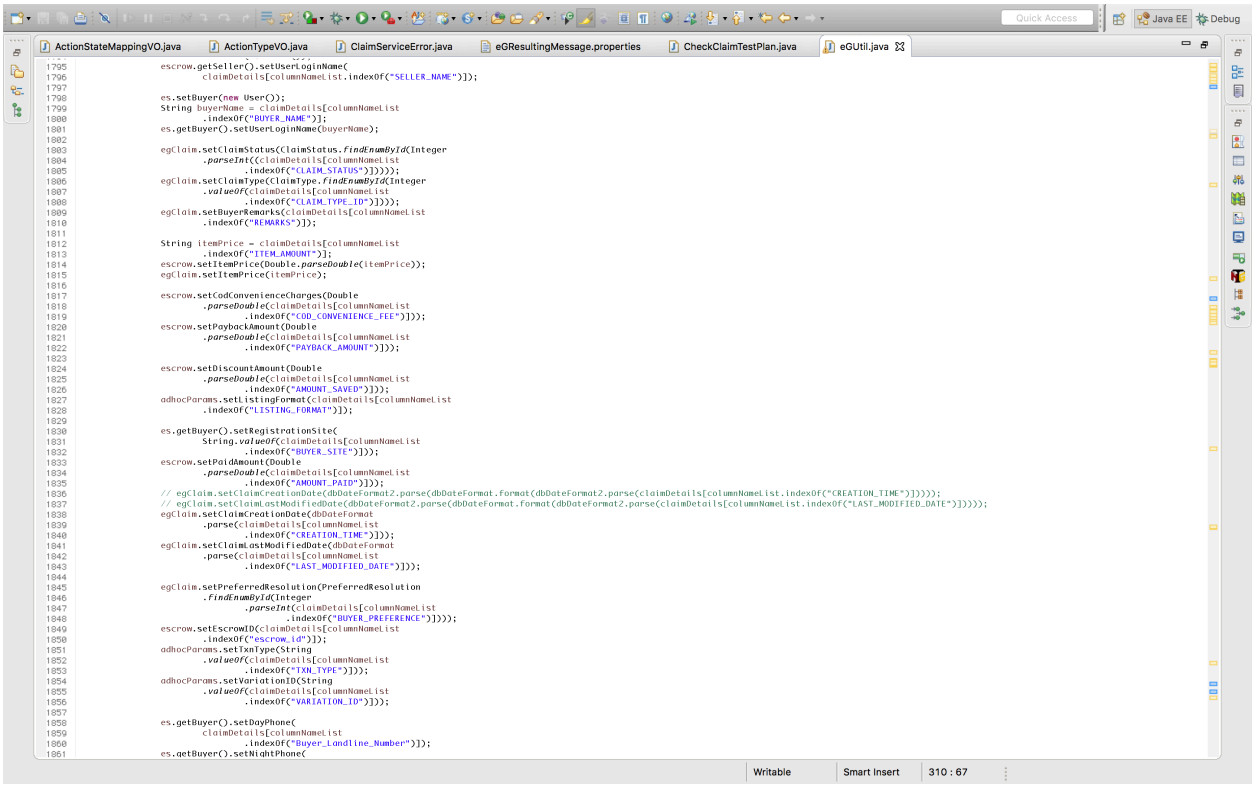
```
                .indexOf("ITEM_AMOUNT")]));
        /* escrow.setItemPrice(es.getEscrowList().get(0).getItemPrice()); */
        escrow.setDiscountAmount(Double.parseDouble(next[columnNameList
                .indexOf("AMOUNT_SAVED")]));
        escrow.setPaidAmount(Double.parseDouble(next[columnNameList
                .indexOf("AMOUNT_PAID")]));
        escrow.setCodConvenienceCharges(Double.parseDouble(next[columnNameList
                .indexOf("COD_CONVENIENCE_FEE")]));
        escrow.setPaybackAmount(Double.parseDouble(next[columnNameList
                .indexOf("PAYBACK_AMOUNT")]));
        escrow.setItem(item);

        // Claim Details
        EGClaim egClaim = new EGClaim();
        egClaim.setClaimId(Integer.valueOf(next[columnNameList
                .indexOf("CLAIM_REQ_NO")]));
        egClaim.setClaimCreationDate(dbDateFormat2.parse(dbDateFormat
                .format(dbDateFormat1.parse(next[columnNameList
                        .indexOf("CREATION_TIME")]))));
        egClaim.setBuyerRemarks(next[columnNameList.indexOf("REMARKS")]);
        /*
         * egClaim.setBuyerRemarks(es.getEscrowList().get(0).getEgClaim().
         * getBuyerRemarks());
         */
        egClaim.setClaimType(ClaimType.findEnumById(Integer
                .valueOf(next[columnNameList.indexOf("CLAIM_TYPE_ID")])));
        /*
         * egClaim.setClaimType(es.getEscrowList().get(0).getEgClaim().getClaimType
         * ());
         */
        // Added by Gourav

        egClaim.setSubIssue(SubIssue.findEnumById(Integer
                .valueOf(next[columnNameList.indexOf("sub_issue_id")])));

        egClaim.setIsInvoiceReceived(next[columnNameList
                .indexOf("buyer_invoice_received")]);

        egClaim.setPreferredResolution(PreferredResolution.findEnumById(Integer
                .valueOf(next[columnNameList.indexOf("BUYER_PREFERENCE")])));
        /*
         * egClaim.setPreferredResolution(es.getEscrowList().get(0).getEgClaim().
         * getPreferredResolution());
         */
        egClaim.setSellerContactStatus(SellerContactStatus.findById(Integer
                .valueOf(next[columnNameList.indexOf("Seller_Contacted")])));
        /*
         * egClaim.setSellerContactStatus(es.getEscrowList().get(0).getEgClaim().
         * getSellerContactStatus());
         */
        egClaim.setClaimStatus(ClaimStatus.findEnumById(Integer
                .valueOf(next[columnNameList.indexOf("CLAIM_STATUS")])));
        if (next[columnNameList.indexOf("Seller_Protection")] != null) {
            egClaim.setSellerProtection(SellerProtectionStatus
                    .findEnumByDesc(next[columnNameList
                            .indexOf("Seller_Protection")]));
        }
        escrow.setEgClaim(egClaim);

        // Escrow Details
        escrow.setEscrowID(next[columnNameList.indexOf("escrow_id")]);
        es.getEscrowList().add(escrow);

        // Seller details
        User seller = new User();
        seller.setUserLoginName(next[columnNameList.indexOf("SELLER_NAME")]);
        seller.setUserID(next[columnNameList.indexOf("seller_id")]);
```



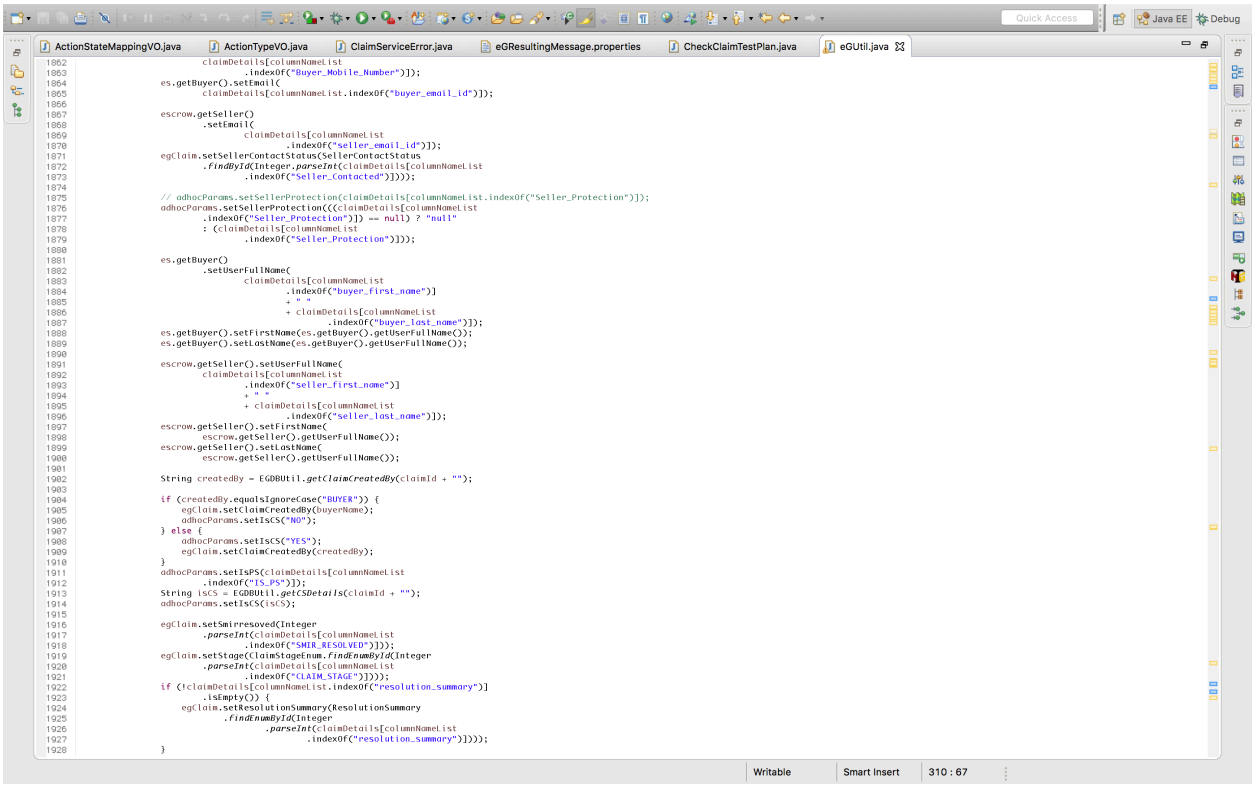
```
        seller.setEmail(next[columnNameList.indexOf("seller_email_id")]);
        /*
         * seller.setFirstName(next[columnNameList.indexOf("seller_first_name")])
         * ;
         * seller.setLastName(next[columnNameList.indexOf("seller_last_name")]);
         */
        es.setSeller(seller);

        // Buyer details
        User buyer = new User();
        buyer.setUserLoginName(next[columnNameList.indexOf("BUYER_NAME")]);
        buyer.setUserID(next[columnNameList.indexOf("buyer_id")]);
        buyer.setEmail(next[columnNameList.indexOf("buyer_email_id")]);
        buyer.setDayPhone(next[columnNameList.indexOf("Buyer_Landline_Number")]);
        buyer.setNightPhone(next[columnNameList.indexOf("Buyer_Mobile_Number")]);
        /*
         * buyer.setFirstName(next[columnNameList.indexOf("buyer_first_name")]);
         * buyer.setLastName(next[columnNameList.indexOf("buyer_last_name")]);
         */
        es.setBuyer(buyer);

        List<String> columnNamesList = new ArrayList<String>();

        String[] shipmentDetails = null;

        try {
            shipmentDetails = EbayIndiaDBModule.getShipmentDetails(es
                    .getEscrowList().get(0).getEscrowID(), columnNamesList);
        } catch (Exception e) {
            shipmentDetails = null;
        }

        // Tracking details
        if (shipmentDetails != null) {
            SimpleDateFormat format = new SimpleDateFormat(
                    "yyyy-MM-dd HH:mm:ss");
            SimpleDateFormat format1 = new SimpleDateFormat(
                    "yyyy-MM-dd HH:mm:ss");
            format.setTimeZone(TimeZone.getTimeZone("Asia/Calcutta"));

            PSTracking tracking = new PSTracking();

            tracking.setCarrier(ShippingCarrier
                    .findCarrier(shipmentDetails[columnNamesList
                            .indexOf("SHIPPING_CARRIER")]));
            tracking.setShipmentBookedDate((format1
                    .parse(shipmentDetails[columnNamesList
                            .indexOf("SHIPMENT_DATE")])));
            tracking.setAirWayBillNo(shipmentDetails[columnNamesList
                    .indexOf("TRACKING_NUMBER")]);
            es.getEscrowList().get(0).setTracking(tracking);
        }

        return es;

    }

    public static boolean isTrackingNumExist(String escrowId) throws Exception {

        List<String> columnNameListTracking = new ArrayList<String>();
        String[] trackingDetails = EbayIndiaDBModule.getShipmentDetails(
                escrowId, columnNameListTracking);

        String trackingNum = trackingDetails[columnNameListTracking
                .indexOf("TRACKING_NUMBER")];

        if (trackingNum == null || trackingNum.equalsIgnoreCase(""))
```



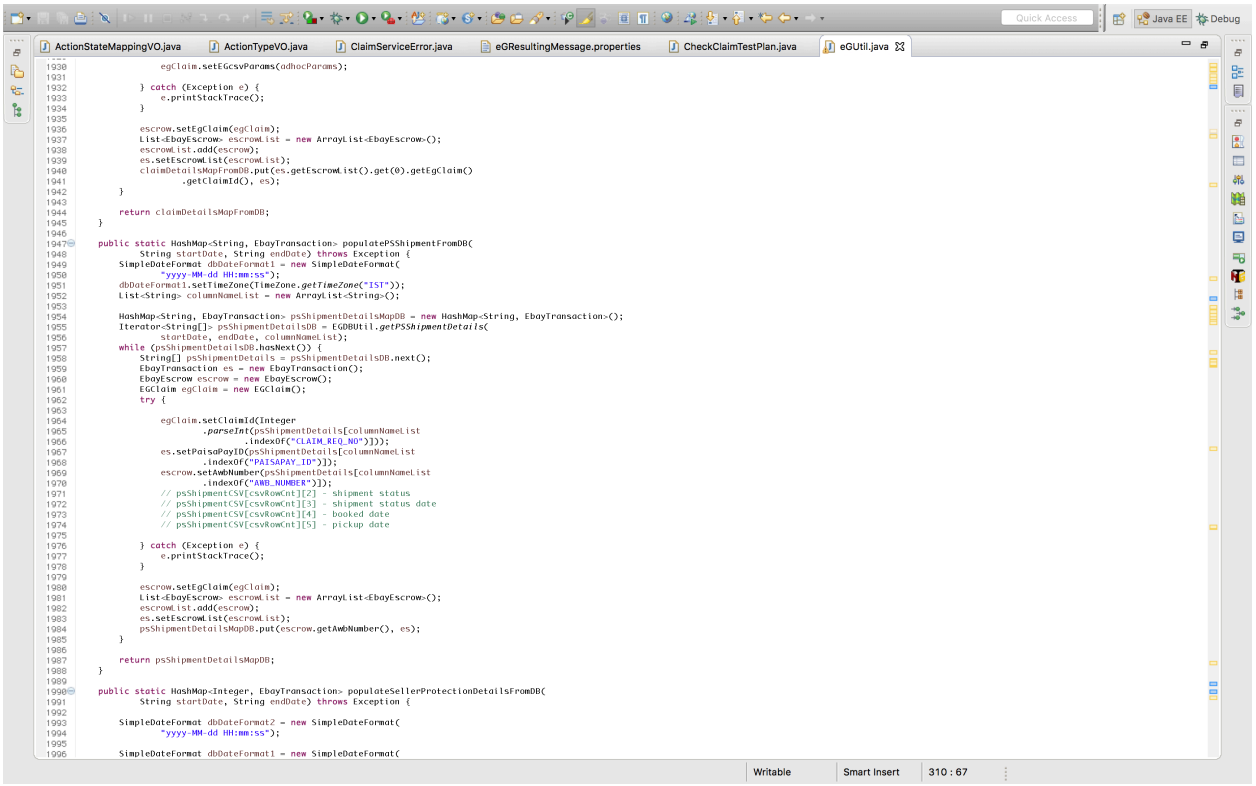
```
        if (trackingNum == null || trackingNum.equalsIgnoreCase(""))
            return false;
        return true;

    }

    public static String[][] getDownloadedReportData(String path)
            throws IOException {
        InputStream is = null;
        File file = new File(path);
        if (!file.exists())
            throw new FileNotFoundException("\nFile : " + path
                    + "is not available\n");
        is = new FileInputStream(new File(path));
        return CSVUtil.read(is, CSVUtil.DELIM_CHAR);
    }

    public static String[][] getDownloadedReportData(File file)
            throws IOException {
        InputStream is = null;
        if (!file.exists())
            throw new FileNotFoundException("\nFile : "
                    + file.getAbsolutePath() + "is not available\n");
        is = new FileInputStream(file);
        return CSVUtil.read(is, CSVUtil.DELIM_CHAR);
    }

    public static HashMap<Integer, EbayTransaction> populateClaimDetailsFromDB(
            String startDate, String endDate) throws Exception {
        SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        SimpleDateFormat dbDateFormat = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        dbDateFormat.setTimeZone(TimeZone.getTimeZone("IST"));
        List<String> columnNameList = new ArrayList<String>();

        HashMap<Integer, EbayTransaction> claimDetailsMapFromDB = new HashMap<Integer, EbayTransaction>();
        Iterator<String[]> claimDetailsDB = EGDBUtil.getClaimDetails(startDate,
                endDate, columnNameList);
        while (claimDetailsDB.hasNext()) {
            String[] claimDetails = claimDetailsDB.next();
            EbayTransaction es = new EbayTransaction();
            EbayEscrow escrow = new EbayEscrow();
            EGClaim egClaim = new EGClaim();
            EGCsvParams adhocParams = new EGCsvParams();

            try {
                int claimId = Integer.parseInt(claimDetails[columnNameList
                        .indexOf("CLAIM_REQ_NO")]);
                egClaim.setClaimId(claimId);
                escrow.setTransactionID(Long
                        .parseLong(claimDetails[columnNameList
                                .indexOf("TRANSACTION_ID")]));
                escrow.setItem(new ItemType());
                escrow.getItem().setItemID(
                        claimDetails[columnNameList.indexOf("ITEM_ID")]);
                es.setPaisaPayID(claimDetails[columnNameList
                        .indexOf("PAISAPAY_ID")]);
                escrow.getItem().setTitle(
                        claimDetails[columnNameList.indexOf("ITEM_TITLE")]);
                es.setTransactionDate(dbDateFormat2.parse(dbDateFormat
                        .format(dbDateFormat2.parse(claimDetails[columnNameList
                                .indexOf("TRANSACTION_DATE")]))));
                es.setPaymentMode(claimDetails[columnNameList
                        .indexOf("PAYMENT_METHOD")]);

                escrow.setSeller(new User());
```



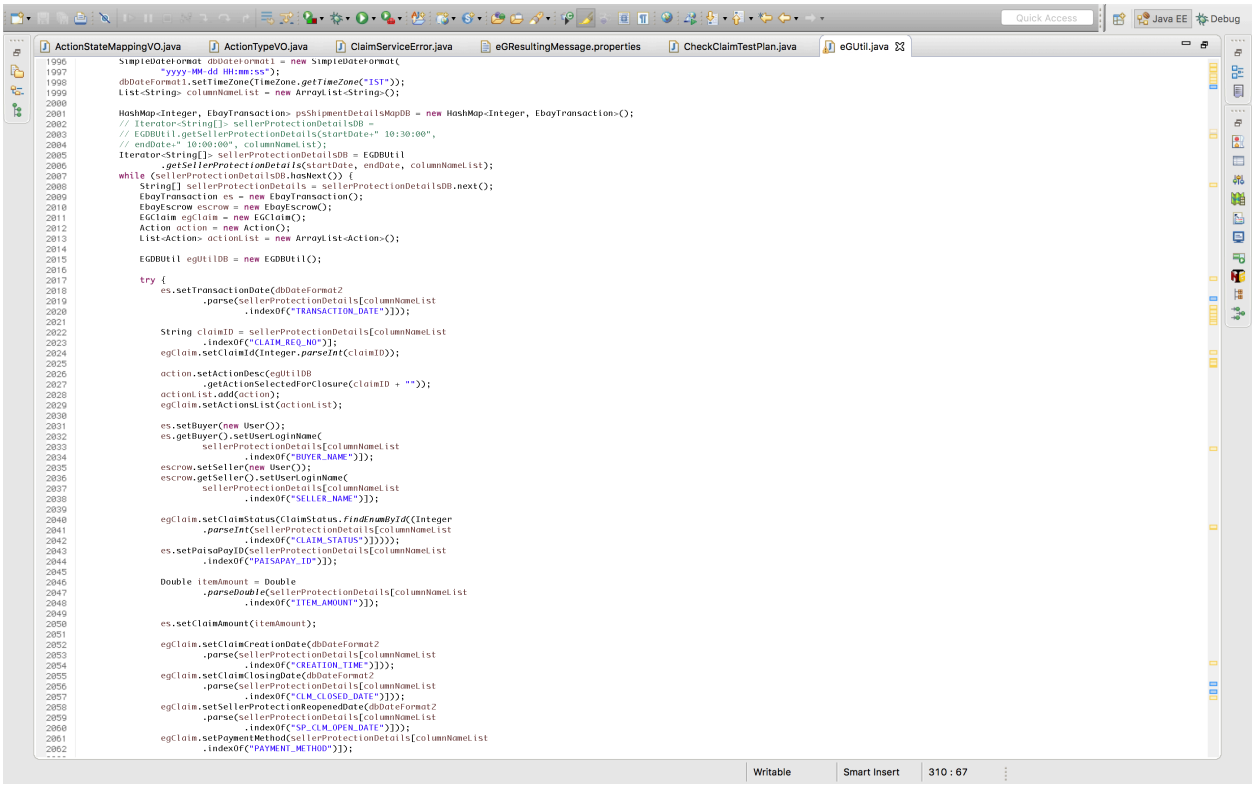
```
                escrow.getSeller().setUserLoginName(
                        claimDetails[columnNameList.indexOf("SELLER_NAME")]);

                es.setBuyer(new User());
                String buyerName = claimDetails[columnNameList
                        .indexOf("BUYER_NAME")];
                es.getBuyer().setUserLoginName(buyerName);

                egClaim.setClaimStatus(ClaimStatus.findEnumById(Integer
                        .parseInt((claimDetails[columnNameList
                                .indexOf("CLAIM_STATUS")]))));
                egClaim.setClaimType(ClaimType.FindEnumById(Integer
                        .valueOf(claimDetails[columnNameList
                                .indexOf("CLAIM_TYPE_ID")])));
                egClaim.setBuyerRemarks(claimDetails[columnNameList
                        .indexOf("REMARKS")]);

                String itemPrice = claimDetails[columnNameList
                        .indexOf("ITEM_AMOUNT")];
                escrow.setItemPrice(Double.parseDouble(itemPrice));
                egClaim.setItemPrice(itemPrice);

                escrow.setCodConvenienceCharges(Double
                        .parseDouble(claimDetails[columnNameList
                                .indexOf("COD_CONVENIENCE_FEE")]));
                escrow.setPaybackAmount(Double
                        .parseDouble(claimDetails[columnNameList
                                .indexOf("PAYBACK_AMOUNT")]));

                escrow.setDiscountAmount(Double
                        .parseDouble(claimDetails[columnNameList
                                .indexOf("AMOUNT_SAVED")]));
                adhocParams.setListingFormat(claimDetails[columnNameList
                        .indexOf("LISTING_FORMAT")]);

                es.getBuyer().setRegistrationSite(
                        String.valueOf(claimDetails[columnNameList
                                .indexOf("BUYER_SITE")]));
                escrow.setPaidAmount(Double
                        .parseDouble(claimDetails[columnNameList
                                .indexOf("AMOUNT_PAID")]));
                // egClaim.setClaimCreationDate(dbDateFormat2.parse(dbDateFormat.format(dbDateFormat2.parse(claimDetails[columnNameList.indexOf("CREATION_TIME")]))));
                // egClaim.setClaimLastModifiedDate(dbDateFormat2.parse(dbDateFormat.format(dbDateFormat2.parse(claimDetails[columnNameList.indexOf("LAST_MODIFIED_DATE")]))));
                egClaim.setClaimCreationDate(dbDateFormat
                        .parse(claimDetails[columnNameList
                                .indexOf("CREATION_TIME")]));
                egClaim.setClaimLastModifiedDate(dbDateFormat
                        .parse(claimDetails[columnNameList
                                .indexOf("LAST_MODIFIED_DATE")]));

                egClaim.setPreferredResolution(PreferredResolution
                        .findEnumById(Integer
                                .parseInt(claimDetails[columnNameList
                                        .indexOf("BUYER_PREFERENCE")])));
                escrow.setEscrowID(claimDetails[columnNameList
                        .indexOf("escrow_id")]);
                adhocParams.setTxnType(String
                        .valueOf(claimDetails[columnNameList
                                .indexOf("TXN_TYPE")]));
                adhocParams.setVariationID(String
                        .valueOf(claimDetails[columnNameList
                                .indexOf("VARIATION_ID")]));

                es.getBuyer().setDayPhone(
                        claimDetails[columnNameList
                                .indexOf("Buyer_Landline_Number")]);
                es.getBuyer().setNightPhone(
```



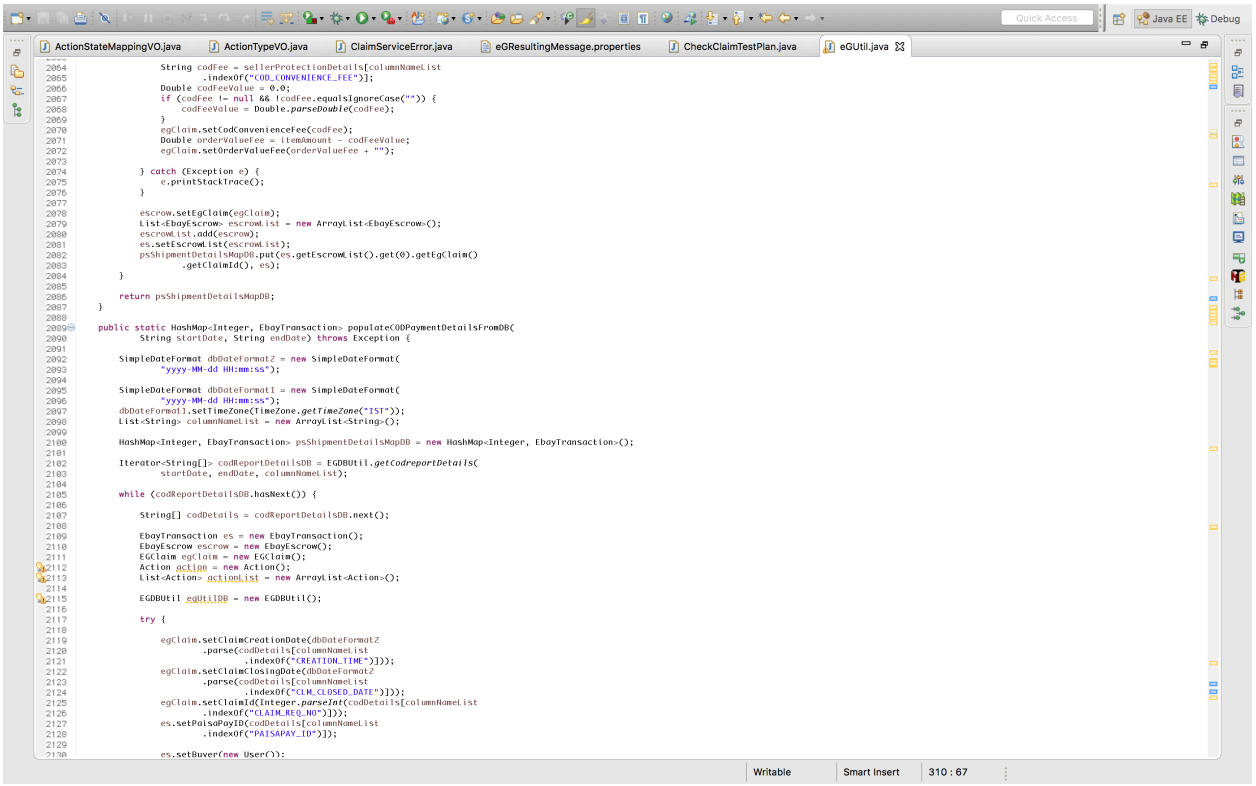
```
                        claimDetails[columnNameList
                                .indexOf("Buyer_Mobile_Number")]);
                es.getBuyer().setEmail(
                        claimDetails[columnNameList.indexOf("buyer_email_id")]);

                escrow.getSeller()
                        .setEmail(
                                claimDetails[columnNameList
                                        .indexOf("seller_email_id")]);
                egClaim.setSellerContactStatus(SellerContactStatus
                        .findById(Integer.parseInt(claimDetails[columnNameList
                                .indexOf("Seller_Contacted")])));

                // adhocParams.setSellerProtection(claimDetails[columnNameList.indexOf("Seller_Protection")]);
                adhocParams.setSellerProtection(((claimDetails[columnNameList
                        .indexOf("Seller_Protection")]) == null) ? "null"
                        : (claimDetails[columnNameList
                                .indexOf("Seller_Protection")]));

                es.getBuyer()
                        .setUserFullName(
                                claimDetails[columnNameList
                                        .indexOf("buyer_first_name")]
                                        + " "
                                        + claimDetails[columnNameList
                                                .indexOf("buyer_last_name")]);
                es.getBuyer().setFirstName(es.getBuyer().getUserFullName());
                es.getBuyer().setLastName(es.getBuyer().getUserFullName());

                escrow.getSeller().setUserFullName(
                        claimDetails[columnNameList
                                .indexOf("seller_first_name")]
                                + " "
                                + claimDetails[columnNameList
                                        .indexOf("seller_last_name")]);
                escrow.getSeller().setFirstName(
                        escrow.getSeller().getUserFullName());
                escrow.getSeller().setLastName(
                        escrow.getSeller().getUserFullName());

                String createdBy = EGDBUtil.getClaimCreatedBy(claimId + "");

                if (createdBy.equalsIgnoreCase("BUYER")) {
                    egClaim.setClaimCreatedBy(buyerName);
                    adhocParams.setIsCS("NO");
                } else {
                    adhocParams.setIsCS("YES");
                    egClaim.setClaimCreatedBy(createdBy);
                }
                adhocParams.setIsPS(claimDetails[columnNameList
                        .indexOf("IS_PS")]);
                String isCS = EGDBUtil.getCSDetails(claimId + "");
                adhocParams.setIsCS(isCS);

                egClaim.setSmirresoved(Integer
                        .parseInt(claimDetails[columnNameList
                                .indexOf("SMIR_RESOLVED")]));
                egClaim.setStage(ClaimStageEnum.findEnumById(Integer
                        .parseInt(claimDetails[columnNameList
                                .indexOf("CLAIM_STAGE")])));
                if (!claimDetails[columnNameList.indexOf("resolution_summary")]
                        .isEmpty()) {
                    egClaim.setResolutionSummary(ResolutionSummary
                            .findEnumById(Integer
                                    .parseInt(claimDetails[columnNameList
                                            .indexOf("resolution_summary")])));
                }
```



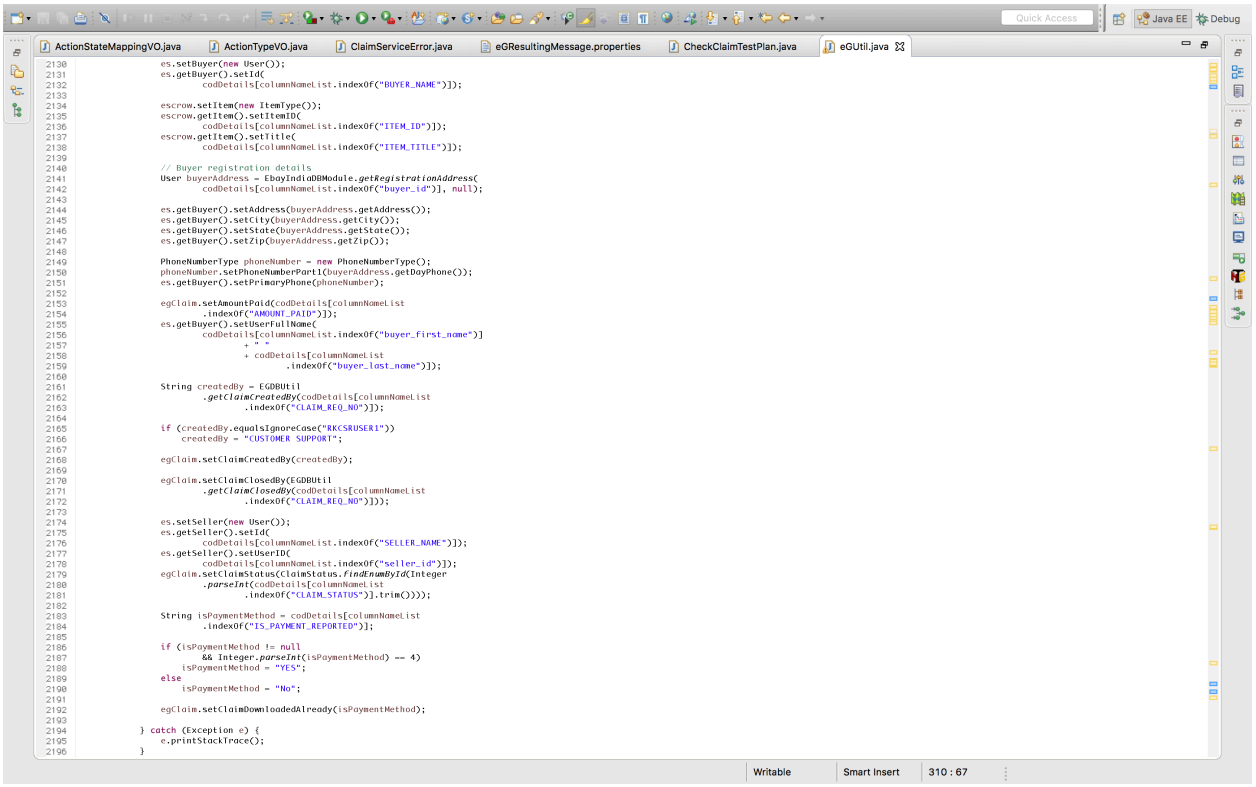
```
                egClaim.setEGcsvParams(adhocParams);

            } catch (Exception e) {
                e.printStackTrace();
            }

            escrow.setEgClaim(egClaim);
            List<EbayEscrow> escrowList = new ArrayList<EbayEscrow>();
            escrowList.add(escrow);
            es.setEscrowList(escrowList);
            claimDetailsMapFromDB.put(es.getEscrowList().get(0).getEgClaim()
                    .getClaimId(), es);
        }

        return claimDetailsMapFromDB;
    }

    public static HashMap<String, EbayTransaction> populatePSShipmentFromDB(
            String startDate, String endDate) throws Exception {
        SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        dbDateFormat1.setTimeZone(TimeZone.getTimeZone("IST"));
        List<String> columnNameList = new ArrayList<String>();

        HashMap<String, EbayTransaction> psShipmentDetailsMapDB = new HashMap<String, EbayTransaction>();
        Iterator<String[]> psShipmentDetailsDB = EGDBUtil.getPSShipmentDetails(
                startDate, endDate, columnNameList);
        while (psShipmentDetailsDB.hasNext()) {
            String[] psShipmentDetails = psShipmentDetailsDB.next();
            EbayTransaction es = new EbayTransaction();
            EbayEscrow escrow = new EbayEscrow();
            EGClaim egClaim = new EGClaim();
            try {

                egClaim.setClaimId(Integer
                        .parseInt(psShipmentDetails[columnNameList
                                .indexOf("CLAIM_REQ_NO")]));
                es.setPaisaPayID(psShipmentDetails[columnNameList
                        .indexOf("PAISAPAY_ID")]);
                escrow.setAwbNumber(psShipmentDetails[columnNameList
                        .indexOf("AWB_NUMBER")]);
                // psShipmentCSV[csvRowCnt][2] - shipment status
                // psShipmentCSV[csvRowCnt][3] - shipment status date
                // psShipmentCSV[csvRowCnt][4] - booked date
                // psShipmentCSV[csvRowCnt][5] - pickup date

            } catch (Exception e) {
                e.printStackTrace();
            }

            escrow.setEgClaim(egClaim);
            List<EbayEscrow> escrowList = new ArrayList<EbayEscrow>();
            escrowList.add(escrow);
            es.setEscrowList(escrowList);
            psShipmentDetailsMapDB.put(escrow.getAwbNumber(), es);
        }

        return psShipmentDetailsMapDB;
    }

    public static HashMap<Integer, EbayTransaction> populateSellerProtectionDetailsFromDB(
            String startDate, String endDate) throws Exception {

        SimpleDateFormat dbDateFormat2 = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");

        SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
```



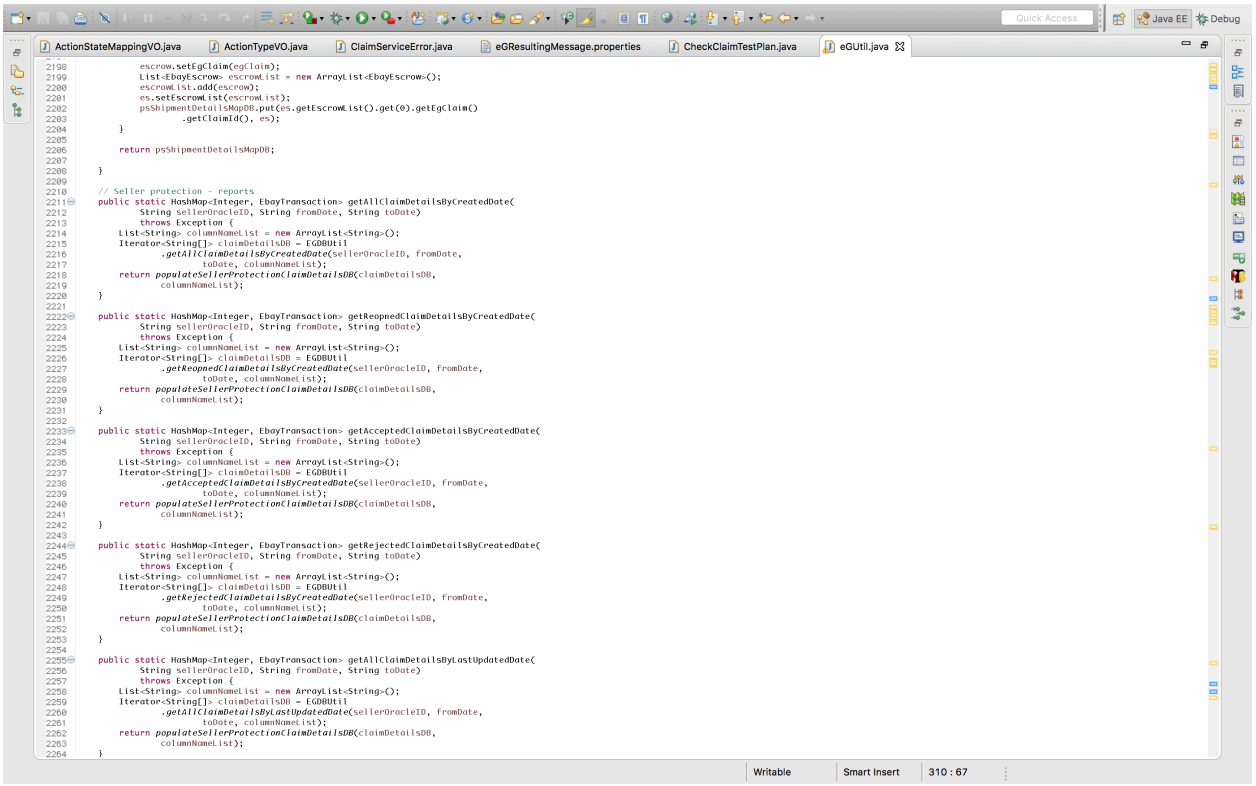
```
SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
        "yyyy-MM-dd HH:mm:ss");
dbDateFormat1.setTimeZone(TimeZone.getTimeZone("IST"));
List<String> columnNameList = new ArrayList<String>();

HashMap<Integer, EbayTransaction> psShipmentDetailsMapDB = new HashMap<Integer, EbayTransaction>();
// Iterator<String[]> sellerProtectionDetailsDB =
// EGDBUtil.getSellerProtectionDetails(startDate+" 10:30:00",
// endDate+" 10:00:00", columnNameList);
Iterator<String[]> sellerProtectionDetailsDB = EGDBUtil
        .getSellerProtectionDetails(startDate, endDate, columnNameList);
while (sellerProtectionDetailsDB.hasNext()) {
    String[] sellerProtectionDetails = sellerProtectionDetailsDB.next();
    EbayTransaction es = new EbayTransaction();
    EbayEscrow escrow = new EbayEscrow();
    EGClaim egClaim = new EGClaim();
    Action action = new Action();
    List<Action> actionList = new ArrayList<Action>();

    EGDBUtil egUtilDB = new EGDBUtil();

    try {
        es.setTransactionDate(dbDateFormat2
                .parse(sellerProtectionDetails[columnNameList
                        .indexOf("TRANSACTION_DATE")]));

        String claimID = sellerProtectionDetails[columnNameList
                .indexOf("CLAIM_REQ_NO")];
        egClaim.setClaimId(Integer.parseInt(claimID));

        action.setActionDesc(egUtilDB
                .getActionSelectedForClosure(claimID + ""));
        actionList.add(action);
        egClaim.setActionsList(actionList);

        es.setBuyer(new User());
        es.getBuyer().setUserLoginName(
                sellerProtectionDetails[columnNameList
                        .indexOf("BUYER_NAME")]);
        escrow.setSeller(new User());
        escrow.getSeller().setUserLoginName(
                sellerProtectionDetails[columnNameList
                        .indexOf("SELLER_NAME")]);

        egClaim.setClaimStatus(ClaimStatus.findEnumById(Integer
                .parseInt(sellerProtectionDetails[columnNameList
                        .indexOf("CLAIM_STATUS")])));
        es.setPaisaPayID(sellerProtectionDetails[columnNameList
                .indexOf("PAISAPAY_ID")]);

        Double itemAmount = Double
                .parseDouble(sellerProtectionDetails[columnNameList
                        .indexOf("ITEM_AMOUNT")]);

        es.setClaimAmount(itemAmount);

        egClaim.setClaimCreationDate(dbDateFormat2
                .parse(sellerProtectionDetails[columnNameList
                        .indexOf("CREATION_TIME")]));
        egClaim.setClaimClosingDate(dbDateFormat2
                .parse(sellerProtectionDetails[columnNameList
                        .indexOf("CLM_CLOSED_DATE")]));
        egClaim.setSellerProtectionReopenedDate(dbDateFormat2
                .parse(sellerProtectionDetails[columnNameList
                        .indexOf("SP_CLM_OPEN_DATE")]));
        egClaim.setPaymentMethod(sellerProtectionDetails[columnNameList
                .indexOf("PAYMENT_METHOD")]);
```

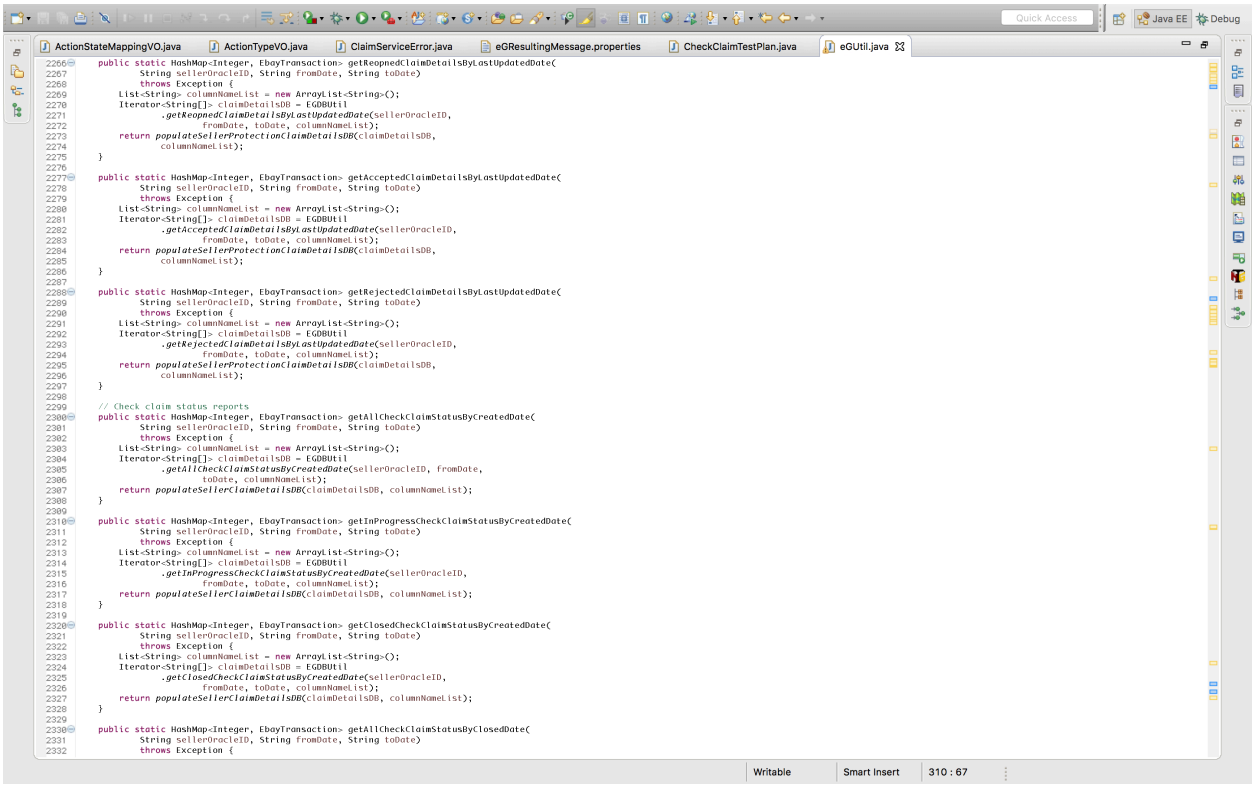
```
            String codFee = sellerProtectionDetails[columnNameList
                    .indexOf("COD_CONVENIENCE_FEE")];
            Double codFeeValue = 0.0;
            if (codFee != null && !codFee.equalsIgnoreCase("")) {
                codFeeValue = Double.parseDouble(codFee);
            }
            egClaim.setCodConvenienceFee(codFee);
            Double orderValueFee = itemAmount - codFeeValue;
            egClaim.setOrderValueFee(orderValuefee + "");

        } catch (Exception e) {
            e.printStackTrace();
        }

        escrow.setEgClaim(egClaim);
        List<EbayEscrow> escrowList = new ArrayList<EbayEscrow>();
        escrowList.add(escrow);
        es.setEscrowList(escrowList);
        psShipmentDetailsMapDB.put(es.getEscrowList().get(0).getEgClaim()
                .getClaimId(), es);
    }

    return psShipmentDetailsMapDB;
}

public static HashMap<Integer, EbayTransaction> populateCODPaymentDetailsFromDB(
        String startDate, String endDate) throws Exception {

    SimpleDateFormat dbDateFormat2 = new SimpleDateFormat(
            "yyyy-MM-dd HH:mm:ss");

    SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
            "yyyy-MM-dd HH:mm:ss");
    dbDateFormat1.setTimeZone(TimeZone.getTimeZone("IST"));
    List<String> columnNameList = new ArrayList<String>();

    HashMap<Integer, EbayTransaction> psShipmentDetailsMapDB = new HashMap<Integer, EbayTransaction>();

    Iterator<String[]> codReportDetailsDB = EGDBUtil.getCodreportDetails(
            startDate, endDate, columnNameList);

    while (codReportDetailsDB.hasNext()) {

        String[] codDetails = codReportDetailsDB.next();

        EbayTransaction es = new EbayTransaction();
        EbayEscrow escrow = new EbayEscrow();
        EGClaim egClaim = new EGClaim();
        Action action = new Action();
        List<Action> actionList = new ArrayList<Action>();

        EGDBUtil egUtilDB = new EGDBUtil();

        try {

            egClaim.setClaimCreationDate(dbDateFormat2
                    .parse(codDetails[columnNameList
                            .indexOf("CREATION_TIME")]));
            egClaim.setClaimClosingDate(dbDateFormat2
                    .parse(codDetails[columnNameList
                            .indexOf("CLM_CLOSED_DATE")]));
            egClaim.setClaimId(Integer.parseInt(codDetails[columnNameList
                    .indexOf("CLAIM_REQ_NO")]));
            es.setPaisaPayID(codDetails[columnNameList
                    .indexOf("PAISAPAY_ID")]);

            es.setBuyer(new User());
```



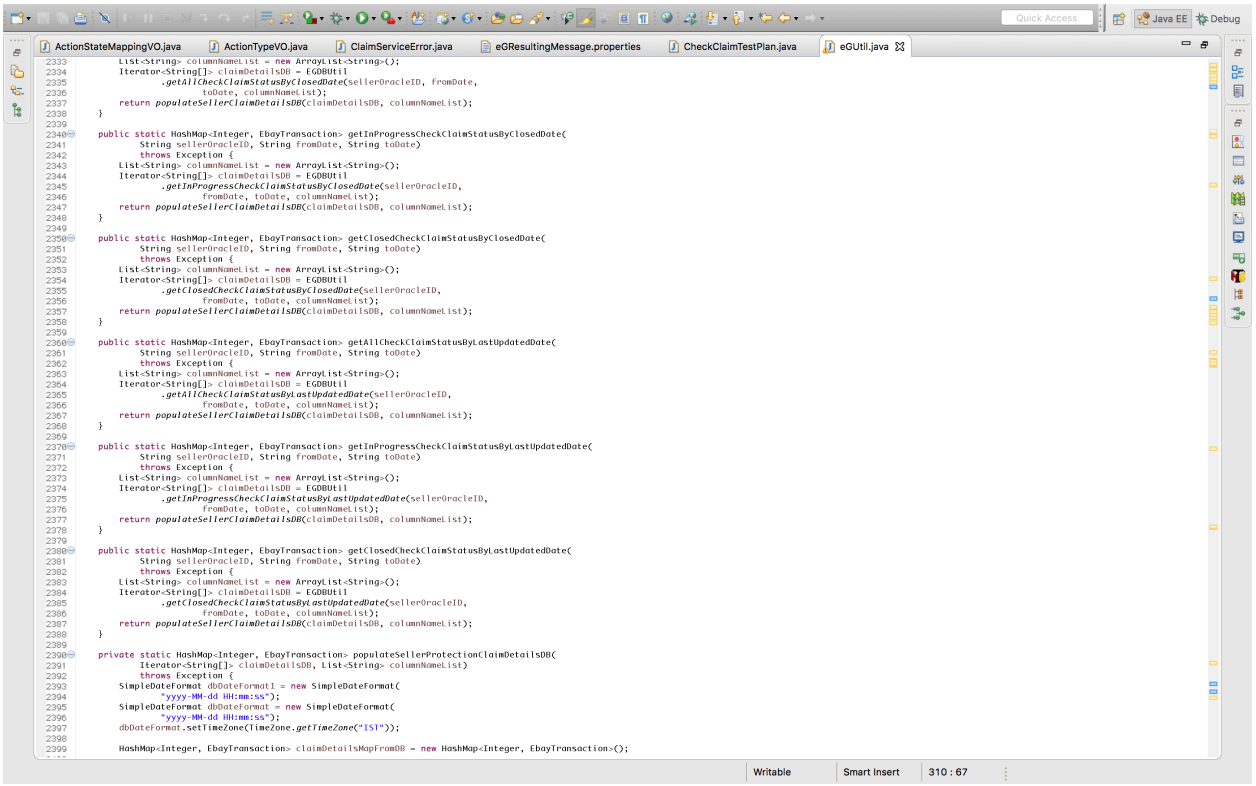
```
                es.setBuyer(new User());
                es.getBuyer().setId(
                        codDetails[columnNameList.indexOf("BUYER_NAME")]);

                escrow.setItem(new ItemType());
                escrow.getItem().setItemID(
                        codDetails[columnNameList.indexOf("ITEM_ID")]);
                escrow.getItem().setTitle(
                        codDetails[columnNameList.indexOf("ITEM_TITLE")]);

                // Buyer registration details
                User buyerAddress = EbayIndiaDBModule.getRegistrationAddress(
                        codDetails[columnNameList.indexOf("buyer_id")], null);

                es.getBuyer().setAddress(buyerAddress.getAddress());
                es.getBuyer().setCity(buyerAddress.getCity());
                es.getBuyer().setState(buyerAddress.getState());
                es.getBuyer().setZip(buyerAddress.getZip());

                PhoneNumberType phoneNumber = new PhoneNumberType();
                phoneNumber.setPhoneNumberPart1(buyerAddress.getDayPhone());
                es.getBuyer().setPrimaryPhone(phoneNumber);

                egClaim.setAmountPaid(codDetails[columnNameList
                        .indexOf("AMOUNT_PAID")]);
                es.getBuyer().setUserFullName(
                        codDetails[columnNameList.indexOf("buyer_first_name")]
                                + " "
                                + codDetails[columnNameList
                                        .indexOf("buyer_last_name")]);

                String createdBy = EGDBUtil
                        .getClaimCreatedBy(codDetails[columnNameList
                                .indexOf("CLAIM_REQ_NO")]);

                if (createdBy.equalsIgnoreCase("RKCSRUSER1"))
                    createdBy = "CUSTOMER SUPPORT";

                egClaim.setClaimCreatedBy(createdBy);

                egClaim.setClaimClosedBy(EGDBUtil
                        .getClaimClosedBy(codDetails[columnNameList
                                .indexOf("CLAIM_REQ_NO")]));

                es.setSeller(new User());
                es.getSeller().setId(
                        codDetails[columnNameList.indexOf("SELLER_NAME")]);
                es.getSeller().setUserID(
                        codDetails[columnNameList.indexOf("seller_id")]);
                egClaim.setClaimStatus(ClaimStatus.findEnumById(Integer
                        .parseInt(codDetails[columnNameList
                                .indexOf("CLAIM_STATUS")].trim())));

                String isPaymentMethod = codDetails[columnNameList
                        .indexOf("IS_PAYMENT_REPORTED")];

                if (isPaymentMethod != null
                        && Integer.parseInt(isPaymentMethod) == 4)
                    isPaymentMethod = "YES";
                else
                    isPaymentMethod = "No";

                egClaim.setClaimDownloadedAlready(isPaymentMethod);

            } catch (Exception e) {
                e.printStackTrace();
            }
```



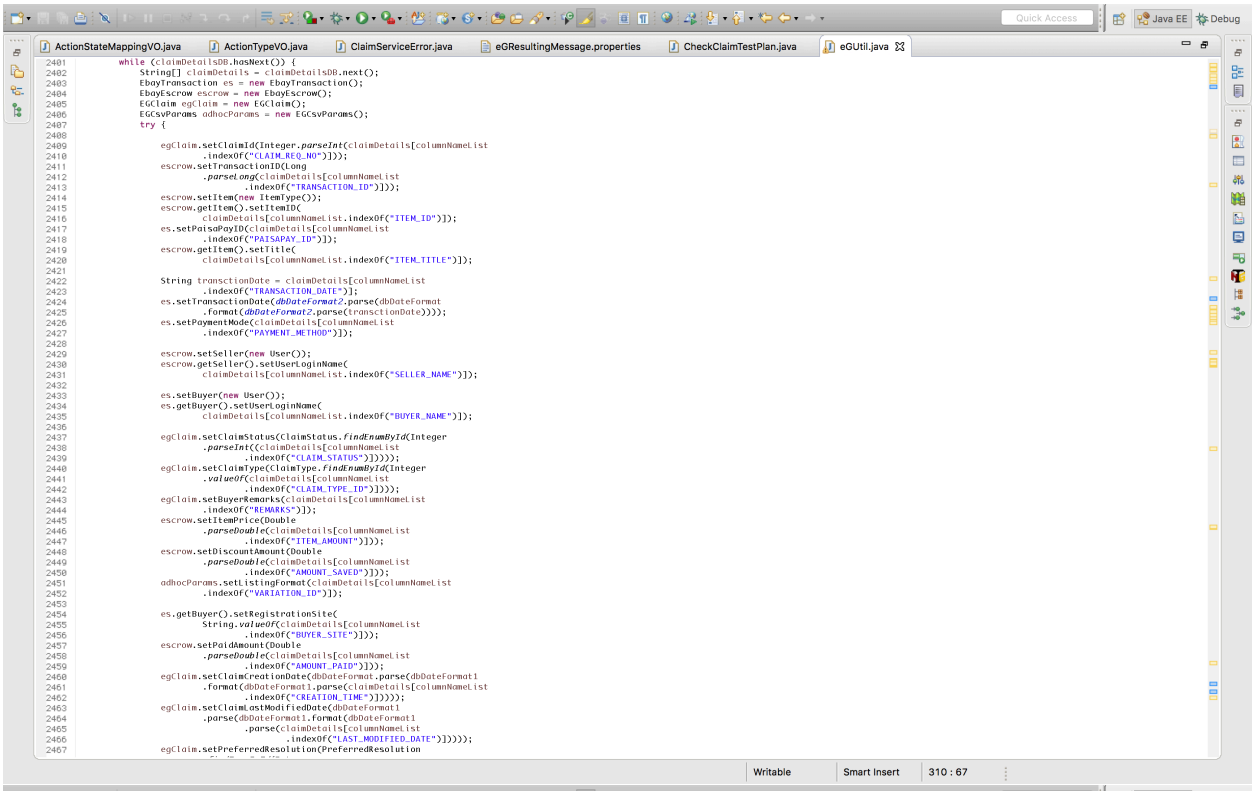
```
            escrow.setEgClaim(egClaim);
            List<EbayEscrow> escrowList = new ArrayList<EbayEscrow>();
            escrowList.add(escrow);
            es.setEscrowList(escrowList);
            psShipmentDetailsMapDB.put(es.getEscrowList().get(0).getEgClaim()
                    .getClaimId(), es);
        }

        return psShipmentDetailsMapDB;

    }

    // Seller protection - reports
    public static HashMap<Integer, EbayTransaction> getAllClaimDetailsByCreatedDate(
            String sellerOracleID, String fromDate, String toDate)
            throws Exception {
        List<String> columnNameList = new ArrayList<String>();
        Iterator<String[]> claimDetailsDB = EGDBUtil
                .getAllClaimDetailsByCreatedDate(sellerOracleID, fromDate,
                        toDate, columnNameList);
        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
                columnNameList);
    }

    public static HashMap<Integer, EbayTransaction> getReopnedClaimDetailsByCreatedDate(
            String sellerOracleID, String fromDate, String toDate)
            throws Exception {
        List<String> columnNameList = new ArrayList<String>();
        Iterator<String[]> claimDetailsDB = EGDBUtil
                .getReopnedClaimDetailsByCreatedDate(sellerOracleID, fromDate,
                        toDate, columnNameList);
        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
                columnNameList);
    }

    public static HashMap<Integer, EbayTransaction> getAcceptedClaimDetailsByCreatedDate(
            String sellerOracleID, String fromDate, String toDate)
            throws Exception {
        List<String> columnNameList = new ArrayList<String>();
        Iterator<String[]> claimDetailsDB = EGDBUtil
                .getAcceptedClaimDetailsByCreatedDate(sellerOracleID, fromDate,
                        toDate, columnNameList);
        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
                columnNameList);
    }

    public static HashMap<Integer, EbayTransaction> getRejectedClaimDetailsByCreatedDate(
            String sellerOracleID, String fromDate, String toDate)
            throws Exception {
        List<String> columnNameList = new ArrayList<String>();
        Iterator<String[]> claimDetailsDB = EGDBUtil
                .getRejectedClaimDetailsByCreatedDate(sellerOracleID, fromDate,
                        toDate, columnNameList);
        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
                columnNameList);
    }

    public static HashMap<Integer, EbayTransaction> getAllClaimDetailsByLastUpdatedDate(
            String sellerOracleID, String fromDate, String toDate)
            throws Exception {
        List<String> columnNameList = new ArrayList<String>();
        Iterator<String[]> claimDetailsDB = EGDBUtil
                .getAllClaimDetailsByLastUpdatedDate(sellerOracleID, fromDate,
                        toDate, columnNameList);
        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
                columnNameList);
    }
```



```
ActionStateMappingVO.java | ActionTypeVO.java | ClaimServiceError.java | eGResultingMessage.properties | CheckClaimTestPlan.java | eGUtil.java

2266    public static HashMap<Integer, EbayTransaction> getReopnedClaimDetailsByLastUpdatedDate(
2267            String sellerOracleID, String fromDate, String toDate)
2268            throws Exception {
2269        List<String> columnNameList = new ArrayList<String>();
2270        Iterator<String[]> claimDetailsDB = EGDBUtil
2271                .getReopnedClaimDetailsByLastUpdatedDate(sellerOracleID,
2272                        fromDate, toDate, columnNameList);
2273        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
2274                columnNameList);
2275    }
2276
2277    public static HashMap<Integer, EbayTransaction> getAcceptedClaimDetailsByLastUpdatedDate(
2278            String sellerOracleID, String fromDate, String toDate)
2279            throws Exception {
2280        List<String> columnNameList = new ArrayList<String>();
2281        Iterator<String[]> claimDetailsDB = EGDBUtil
2282                .getAcceptedClaimDetailsByLastUpdatedDate(sellerOracleID,
2283                        fromDate, toDate, columnNameList);
2284        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
2285                columnNameList);
2286    }
2287
2288    public static HashMap<Integer, EbayTransaction> getRejectedClaimDetailsByLastUpdatedDate(
2289            String sellerOracleID, String fromDate, String toDate)
2290            throws Exception {
2291        List<String> columnNameList = new ArrayList<String>();
2292        Iterator<String[]> claimDetailsDB = EGDBUtil
2293                .getRejectedClaimDetailsByLastUpdatedDate(sellerOracleID,
2294                        fromDate, toDate, columnNameList);
2295        return populateSellerProtectionClaimDetailsDB(claimDetailsDB,
2296                columnNameList);
2297    }
2298
2299    // Check claim status reports
2300    public static HashMap<Integer, EbayTransaction> getAllCheckClaimStatusByCreatedDate(
2301            String sellerOracleID, String fromDate, String toDate)
2302            throws Exception {
2303        List<String> columnNameList = new ArrayList<String>();
2304        Iterator<String[]> claimDetailsDB = EGDBUtil
2305                .getAllCheckClaimStatusByCreatedDate(sellerOracleID, fromDate,
2306                        toDate, columnNameList);
2307        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2308    }
2309
2310    public static HashMap<Integer, EbayTransaction> getInProgressCheckClaimStatusByCreatedDate(
2311            String sellerOracleID, String fromDate, String toDate)
2312            throws Exception {
2313        List<String> columnNameList = new ArrayList<String>();
2314        Iterator<String[]> claimDetailsDB = EGDBUtil
2315                .getInProgressCheckClaimStatusByCreatedDate(sellerOracleID,
2316                        fromDate, toDate, columnNameList);
2317        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2318    }
2319
2320    public static HashMap<Integer, EbayTransaction> getClosedCheckClaimStatusByCreatedDate(
2321            String sellerOracleID, String fromDate, String toDate)
2322            throws Exception {
2323        List<String> columnNameList = new ArrayList<String>();
2324        Iterator<String[]> claimDetailsDB = EGDBUtil
2325                .getClosedCheckClaimStatusByCreatedDate(sellerOracleID,
2326                        fromDate, toDate, columnNameList);
2327        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2328    }
2329
2330    public static HashMap<Integer, EbayTransaction> getAllCheckClaimStatusByClosedDate(
2331            String sellerOracleID, String fromDate, String toDate)
2332            throws Exception {

Writable | Smart Insert | 310 : 67
```

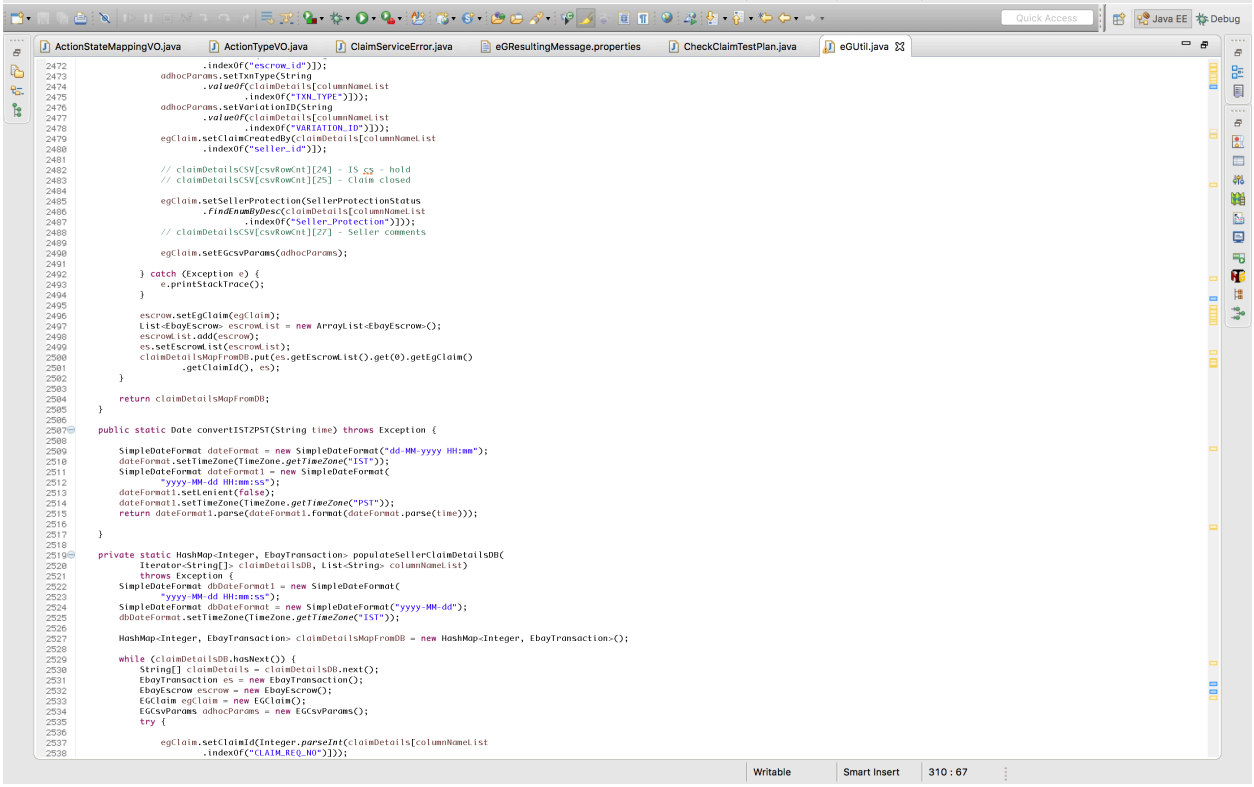


```
ActionStateMappingVO.java | ActionTypeVO.java | ClaimServiceError.java | eGResultingMessage.properties | CheckClaimTestPlan.java | eGUtil.java

2333        List<String> columnNameList = new ArrayList<String>();
2334        Iterator<String[]> claimDetailsDB = EGDBUtil
2335                .getAllCheckClaimStatusByClosedDate(sellerOracleID, fromDate,
2336                        toDate, columnNameList);
2337        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2338    }
2339
2340    public static HashMap<Integer, EbayTransaction> getInProgressCheckClaimStatusByClosedDate(
2341            String sellerOracleID, String fromDate, String toDate)
2342            throws Exception {
2343        List<String> columnNameList = new ArrayList<String>();
2344        Iterator<String[]> claimDetailsDB = EGDBUtil
2345                .getInProgressCheckClaimStatusByClosedDate(sellerOracleID,
2346                        fromDate, toDate, columnNameList);
2347        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2348    }
2349
2350    public static HashMap<Integer, EbayTransaction> getClosedCheckClaimStatusByClosedDate(
2351            String sellerOracleID, String fromDate, String toDate)
2352            throws Exception {
2353        List<String> columnNameList = new ArrayList<String>();
2354        Iterator<String[]> claimDetailsDB = EGDBUtil
2355                .getClosedCheckClaimStatusByClosedDate(sellerOracleID,
2356                        fromDate, toDate, columnNameList);
2357        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2358    }
2359
2360    public static HashMap<Integer, EbayTransaction> getAllCheckClaimStatusByLastUpdatedDate(
2361            String sellerOracleID, String fromDate, String toDate)
2362            throws Exception {
2363        List<String> columnNameList = new ArrayList<String>();
2364        Iterator<String[]> claimDetailsDB = EGDBUtil
2365                .getAllCheckClaimStatusByLastUpdatedDate(sellerOracleID,
2366                        fromDate, toDate, columnNameList);
2367        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2368    }
2369
2370    public static HashMap<Integer, EbayTransaction> getInProgressCheckClaimStatusByLastUpdatedDate(
2371            String sellerOracleID, String fromDate, String toDate)
2372            throws Exception {
2373        List<String> columnNameList = new ArrayList<String>();
2374        Iterator<String[]> claimDetailsDB = EGDBUtil
2375                .getInProgressCheckClaimStatusByLastUpdatedDate(sellerOracleID,
2376                        fromDate, toDate, columnNameList);
2377        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2378    }
2379
2380    public static HashMap<Integer, EbayTransaction> getClosedCheckClaimStatusByLastUpdatedDate(
2381            String sellerOracleID, String fromDate, String toDate)
2382            throws Exception {
2383        List<String> columnNameList = new ArrayList<String>();
2384        Iterator<String[]> claimDetailsDB = EGDBUtil
2385                .getClosedCheckClaimStatusByLastUpdatedDate(sellerOracleID,
2386                        fromDate, toDate, columnNameList);
2387        return populateSellerClaimDetailsDB(claimDetailsDB, columnNameList);
2388    }
2389
2390    private static HashMap<Integer, EbayTransaction> populateSellerProtectionClaimDetailsDB(
2391            Iterator<String[]> claimDetailsDB, List<String> columnNameList)
2392            throws Exception {
2393        SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
2394                "yyyy-MM-dd HH:mm:ss");
2395        SimpleDateFormat dbDateFormat = new SimpleDateFormat(
2396                "yyyy-MM-dd HH:mm:ss");
2397        dbDateFormat.setTimeZone(TimeZone.getTimeZone("IST"));
2398
2399        HashMap<Integer, EbayTransaction> claimDetailsMapFromDB = new HashMap<Integer, EbayTransaction>();

Writable | Smart Insert | 310 : 67
```



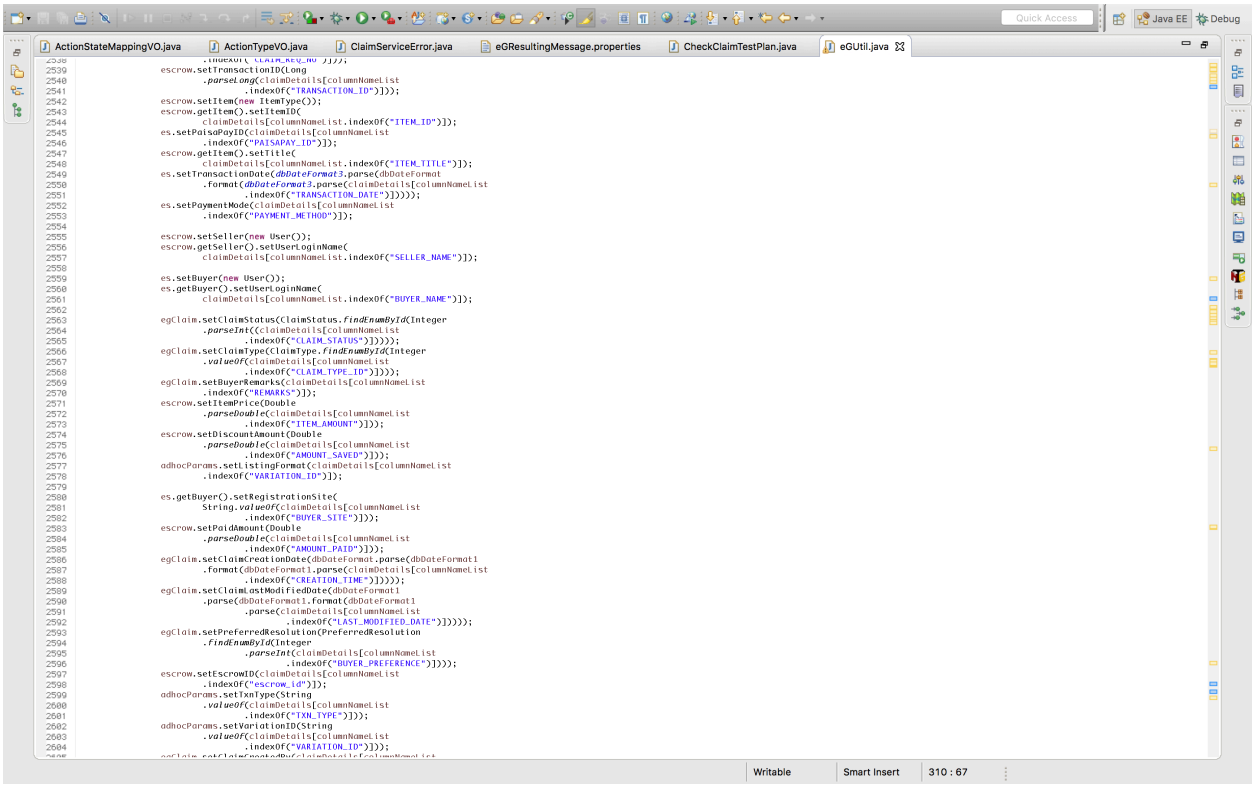
```
        while (claimDetailsDB.hasNext()) {
            String[] claimDetails = claimDetailsDB.next();
            EbayTransaction es = new EbayTransaction();
            EbayEscrow escrow = new EbayEscrow();
            EGClaim egClaim = new EGClaim();
            EGCsvParams adhocParams = new EGCsvParams();
            try {

                egClaim.setClaimId(Integer.parseInt(claimDetails[columnNameList
                        .indexOf("CLAIM_REQ_NO")]));
                escrow.setTransactionID(Long
                        .parseLong(claimDetails[columnNameList
                                .indexOf("TRANSACTION_ID")]));
                escrow.setItem(new ItemType());
                escrow.getItem().setItemID(
                        claimDetails[columnNameList.indexOf("ITEM_ID")]);
                es.setPaisaPayID(claimDetails[columnNameList
                        .indexOf("PAISAPAY_ID")]);
                escrow.getItem().setTitle(
                        claimDetails[columnNameList.indexOf("ITEM_TITLE")]);

                String transctionDate = claimDetails[columnNameList
                        .indexOf("TRANSACTION_DATE")];
                es.setTransactionDate(dbDateFormat2.parse(dbDateFormat
                        .format(dbDateFormat2.parse(transctionDate))));
                es.setPaymentMode(claimDetails[columnNameList
                        .indexOf("PAYMENT_METHOD")]);

                escrow.setSeller(new User());
                escrow.getSeller().setUserLoginName(
                        claimDetails[columnNameList.indexOf("SELLER_NAME")]);

                es.setBuyer(new User());
                es.getBuyer().setUserLoginName(
                        claimDetails[columnNameList.indexOf("BUYER_NAME")]);

                egClaim.setClaimStatus(ClaimStatus.findEnumById(Integer
                        .parseInt((claimDetails[columnNameList
                                .indexOf("CLAIM_STATUS")]))));
                egClaim.setClaimType(ClaimType.findEnumById(Integer
                        .valueOf(claimDetails[columnNameList
                                .indexOf("CLAIM_TYPE_ID")])));
                egClaim.setBuyerRemarks(claimDetails[columnNameList
                        .indexOf("REMARKS")]);
                escrow.setItemPrice(Double
                        .parseDouble(claimDetails[columnNameList
                                .indexOf("ITEM_AMOUNT")]));
                escrow.setDiscountAmount(Double
                        .parseDouble(claimDetails[columnNameList
                                .indexOf("AMOUNT_SAVED")]));
                adhocParams.setListingFormat(claimDetails[columnNameList
                        .indexOf("VARIATION_ID")]);

                es.getBuyer().setRegistrationSite(
                        String.valueOf(claimDetails[columnNameList
                                .indexOf("BUYER_SITE")]));
                escrow.setPaidAmount(Double
                        .parseDouble(claimDetails[columnNameList
                                .indexOf("AMOUNT_PAID")]));
                egClaim.setClaimCreationDate(dbDateFormat.parse(dbDateFormat1
                        .format(dbDateFormat1.parse(claimDetails[columnNameList
                                .indexOf("CREATION_TIME")]))));
                egClaim.setClaimLastModifiedDate(dbDateFormat1
                        .parse(dbDateFormat1.format(dbDateFormat1
                                .parse(claimDetails[columnNameList
                                        .indexOf("LAST_MODIFIED_DATE")])))));
                egClaim.setPreferredResolution(PreferredResolution
```

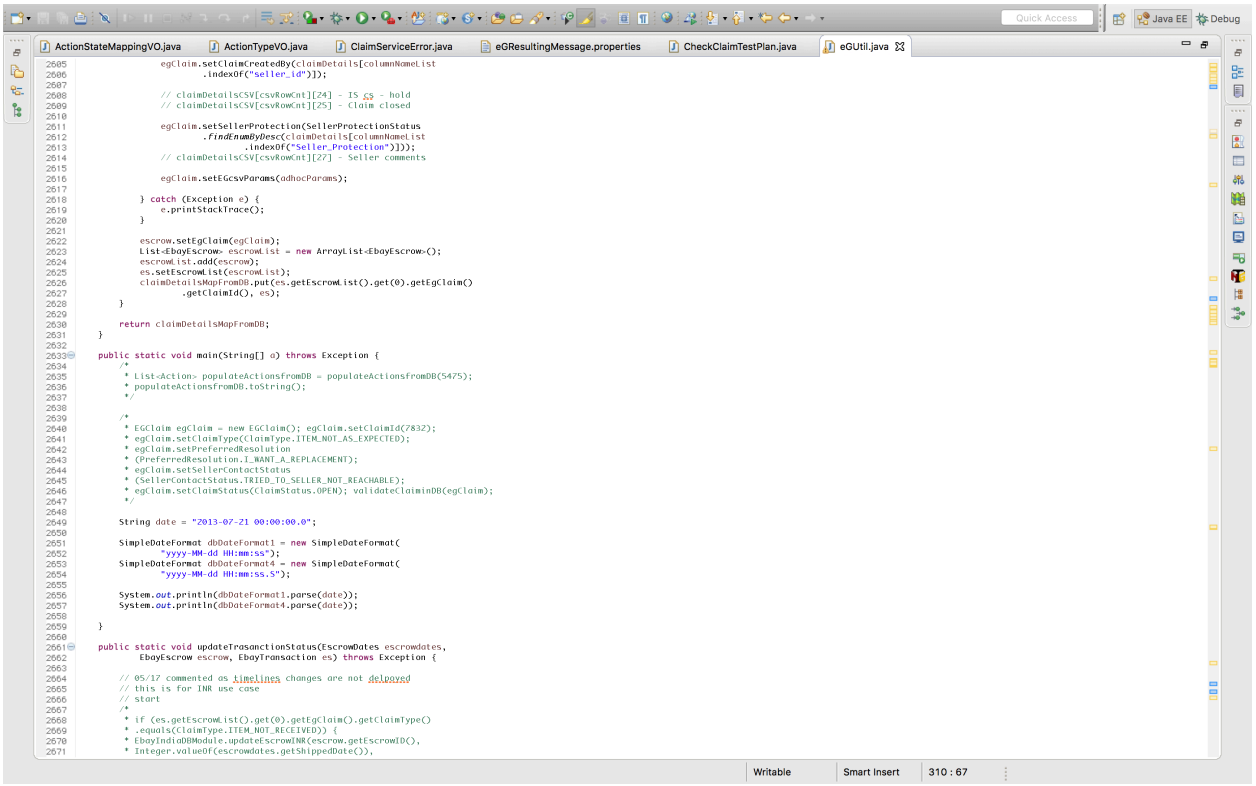
Writable | Smart Insert | 310 : 67

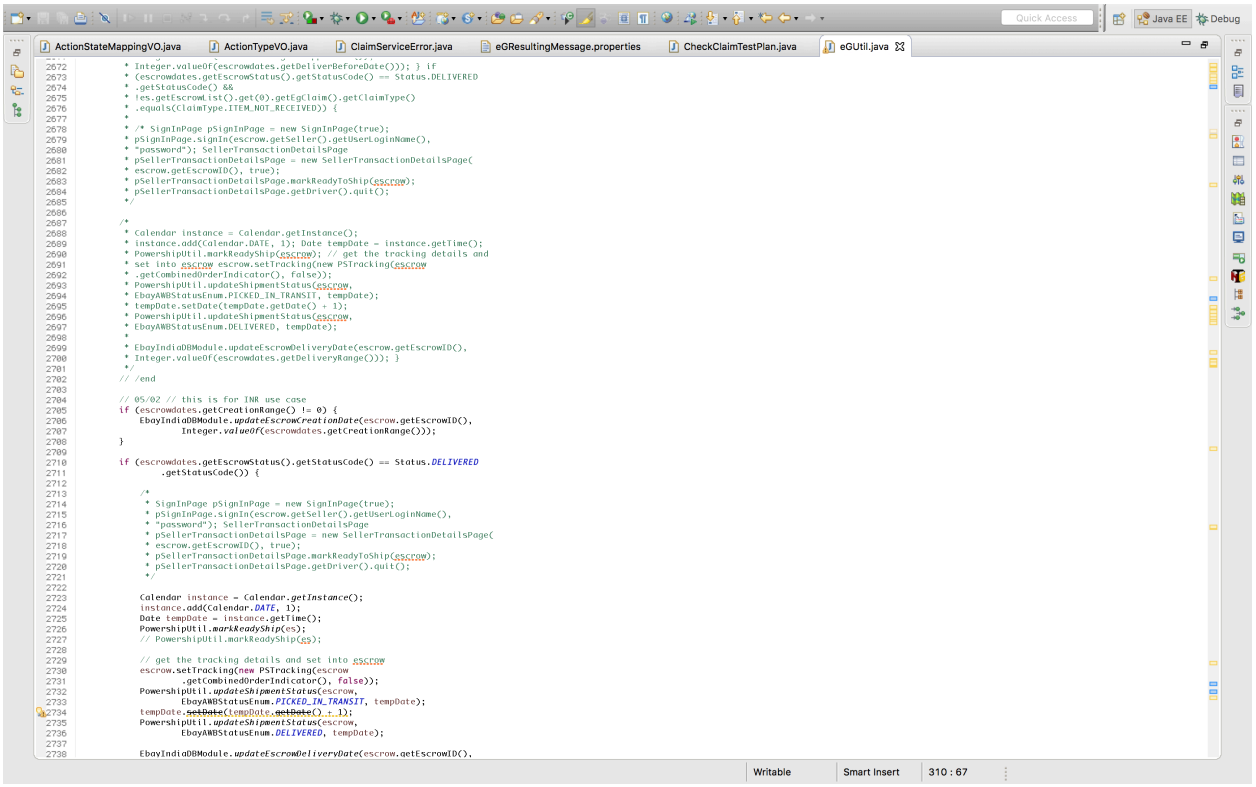


```
                        .indexOf("escrow_id")]);
                adhocParams.setTxnType(String
                        .valueOf(claimDetails[columnNameList
                                .indexOf("TXN_TYPE")]));
                adhocParams.setVariationID(String
                        .valueOf(claimDetails[columnNameList
                                .indexOf("VARIATION_ID")]));
                egClaim.setClaimCreatedBy(claimDetails[columnNameList
                        .indexOf("seller_id")]);

                // claimDetailsCSV[csvRowCnt][24] - IS cs - hold
                // claimDetailsCSV[csvRowCnt][25] - Claim closed

                egClaim.setSellerProtection(SellerProtectionStatus
                        .findEnumByDesc(claimDetails[columnNameList
                                .indexOf("Seller_Protection")]));
                // claimDetailsCSV[csvRowCnt][27] - Seller comments

                egClaim.setEGcsvParams(adhocParams);

            } catch (Exception e) {
                e.printStackTrace();
            }

            escrow.setEgClaim(egClaim);
            List<EbayEscrow> escrowList = new ArrayList<EbayEscrow>();
            escrowList.add(escrow);
            es.setEscrowList(escrowList);
            claimDetailsMapFromDB.put(es.getEscrowList().get(0).getEgClaim()
                    .getClaimId(), es);
        }

        return claimDetailsMapFromDB;
    }

    public static Date convertIST2PST(String time) throws Exception {

        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy HH:mm");
        dateFormat.setTimeZone(TimeZone.getTimeZone("IST"));
        SimpleDateFormat dateFormat1 = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        dateFormat1.setLenient(false);
        dateFormat1.setTimeZone(TimeZone.getTimeZone("PST"));
        return dateFormat1.parse(dateFormat1.format(dateFormat.parse(time)));

    }

    private static HashMap<Integer, EbayTransaction> populateSellerClaimDetailsDB(
            Iterator<String[]> claimDetailsDB, List<String> columnNameList)
            throws Exception {
        SimpleDateFormat dbDateFormat1 = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        SimpleDateFormat dbDateFormat = new SimpleDateFormat("yyyy-MM-dd");
        dbDateFormat.setTimeZone(TimeZone.getTimeZone("IST"));

        HashMap<Integer, EbayTransaction> claimDetailsMapFromDB = new HashMap<Integer, EbayTransaction>();

        while (claimDetailsDB.hasNext()) {
            String[] claimDetails = claimDetailsDB.next();
            EbayTransaction es = new EbayTransaction();
            EbayEscrow escrow = new EbayEscrow();
            EGClaim egClaim = new EGClaim();
            EGCsvParams adhocParams = new EGCsvParams();
            try {

                egClaim.setClaimId(Integer.parseInt(claimDetails[columnNameList
                        .indexOf("CLAIM_REQ_NO")]));
```

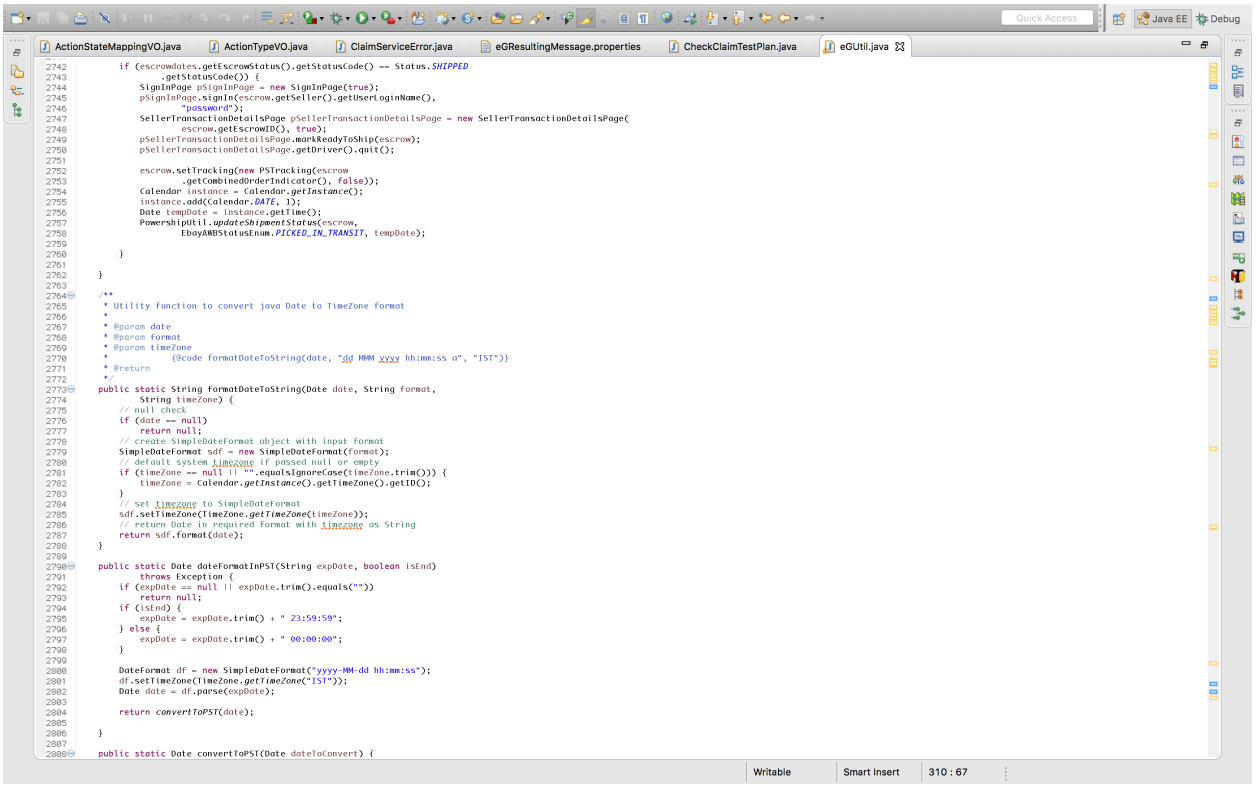
Writable | Smart Insert | 310 : 67


Quick Access
Java EE
Debug
ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
eGResultingMessage.properties
CheckClaimTestPlan.java
eGUtil.java
Writable
Smart Insert
310 : 67

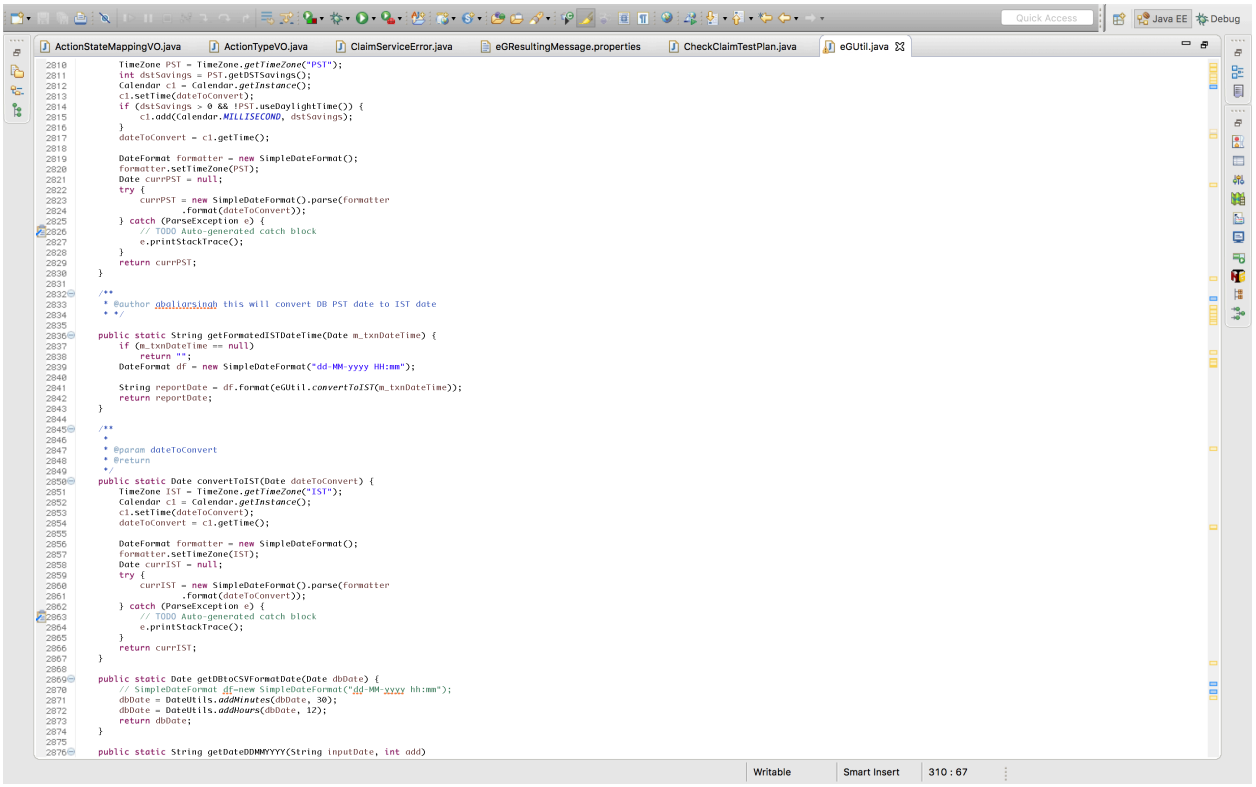

Quick Access
Java EE
Debug
ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
eGResultingMessage.properties
CheckClaimTestPlan.java
eGUtil.java
Writable
Smart Insert
310 : 67

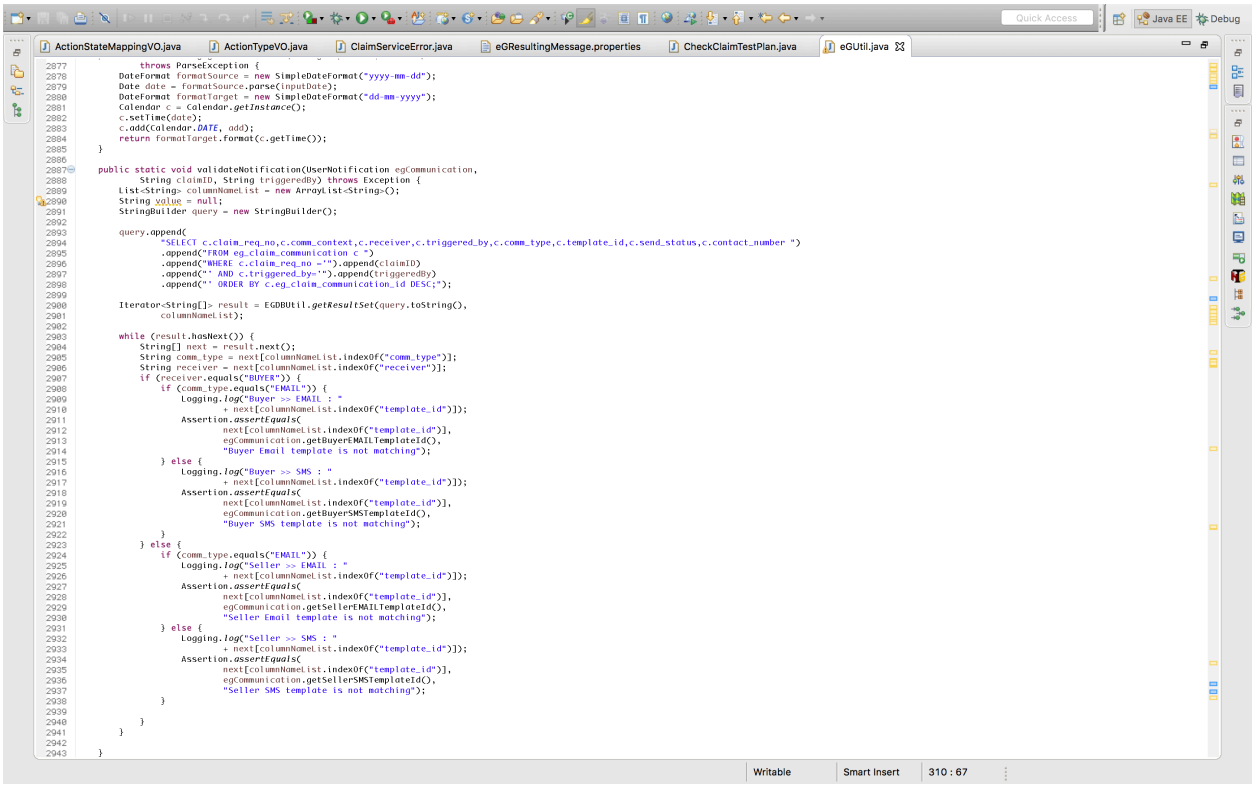


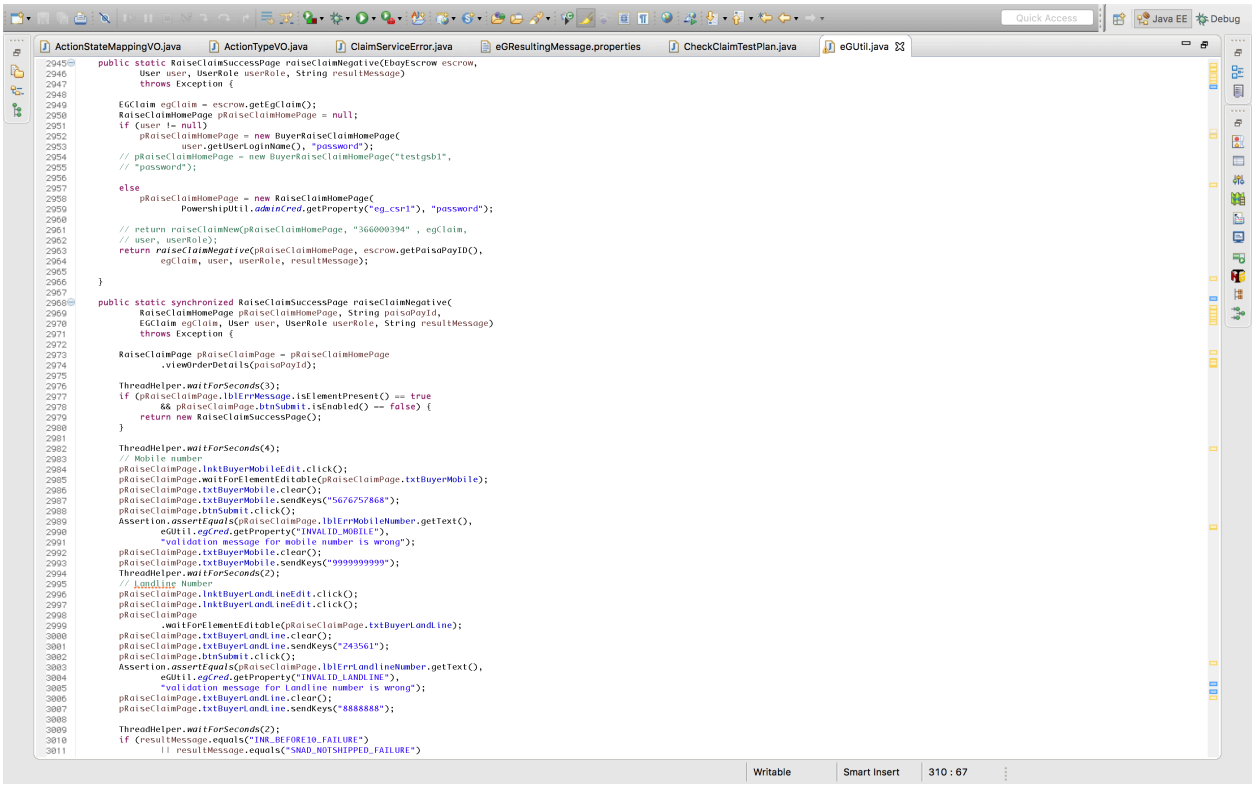
```
 * Integer.valueOf(escrowdates.getDeliverBeforeDate())); } if
 * (escrowdates.getEscrowStatus().getStatusCode() == Status.DELIVERED
 * .getStatusCode() &&
 * !es.getEscrowList().get(0).getEgClaim().getClaimType()
 * .equals(ClaimType.ITEM_NOT_RECEIVED)) {
 *
 * /* SignInPage pSignInPage = new SignInPage(true);
 * pSignInPage.signIn(escrow.getSeller().getUserLoginName(),
 * "password"); SellerTransactionDetailsPage
 * pSellerTransactionDetailsPage = new SellerTransactionDetailsPage(
 * escrow.getEscrowID(), true);
 * pSellerTransactionDetailsPage.markReadyToShip(escrow);
 * pSellerTransactionDetailsPage.getDriver().quit();
 */

/*
 * Calendar instance = Calendar.getInstance();
 * instance.add(Calendar.DATE, 1); Date tempDate = instance.getTime();
 * PowershipUtil.markReadyShip(escrow); // get the tracking details and
 * set into escrow escrow.setTracking(new PSTracking(escrow
 * .getCombinedOrderIndicator(), false));
 * PowershipUtil.updateShipmentStatus(escrow,
 * EbayAWBStatusEnum.PICKED_IN_TRANSIT, tempDate);
 * tempDate.setDate(tempDate.getDate() + 1);
 * PowershipUtil.updateShipmentStatus(escrow,
 * EbayAWBStatusEnum.DELIVERED, tempDate);
 *
 * EbayIndiaDBModule.updateEscrowDeliveryDate(escrow.getEscrowID(),
 * Integer.valueOf(escrowdates.getDeliveryRange())); }
 */
// /end

// 05/02 // this is for INR use case
if (escrowdates.getCreationRange() != 0) {
    EbayIndiaDBModule.updateEscrowCreationDate(escrow.getEscrowID(),
            Integer.valueOf(escrowdates.getCreationRange()));
}

if (escrowdates.getEscrowStatus().getStatusCode() == Status.DELIVERED
        .getStatusCode()) {

    /*
     * SignInPage pSignInPage = new SignInPage(true);
     * pSignInPage.signIn(escrow.getSeller().getUserLoginName(),
     * "password"); SellerTransactionDetailsPage
     * pSellerTransactionDetailsPage = new SellerTransactionDetailsPage(
     * escrow.getEscrowID(), true);
     * pSellerTransactionDetailsPage.markReadyToShip(escrow);
     * pSellerTransactionDetailsPage.getDriver().quit();
     */

    Calendar instance = Calendar.getInstance();
    instance.add(Calendar.DATE, 1);
    Date tempDate = instance.getTime();
    PowershipUtil.markReadyShip(es);
    // PowershipUtil.markReadyShip(es);

    // get the tracking details and set into escrow
    escrow.setTracking(new PSTracking(escrow
            .getCombinedOrderIndicator(), false));
    PowershipUtil.updateShipmentStatus(escrow,
            EbayAWBStatusEnum.PICKED_IN_TRANSIT, tempDate);
    tempDate.setDate(tempDate.getDate() + 1);
    PowershipUtil.updateShipmentStatus(escrow,
            EbayAWBStatusEnum.DELIVERED, tempDate);

    EbayIndiaDBModule.updateEscrowDeliveryDate(escrow.getEscrowID(),
```



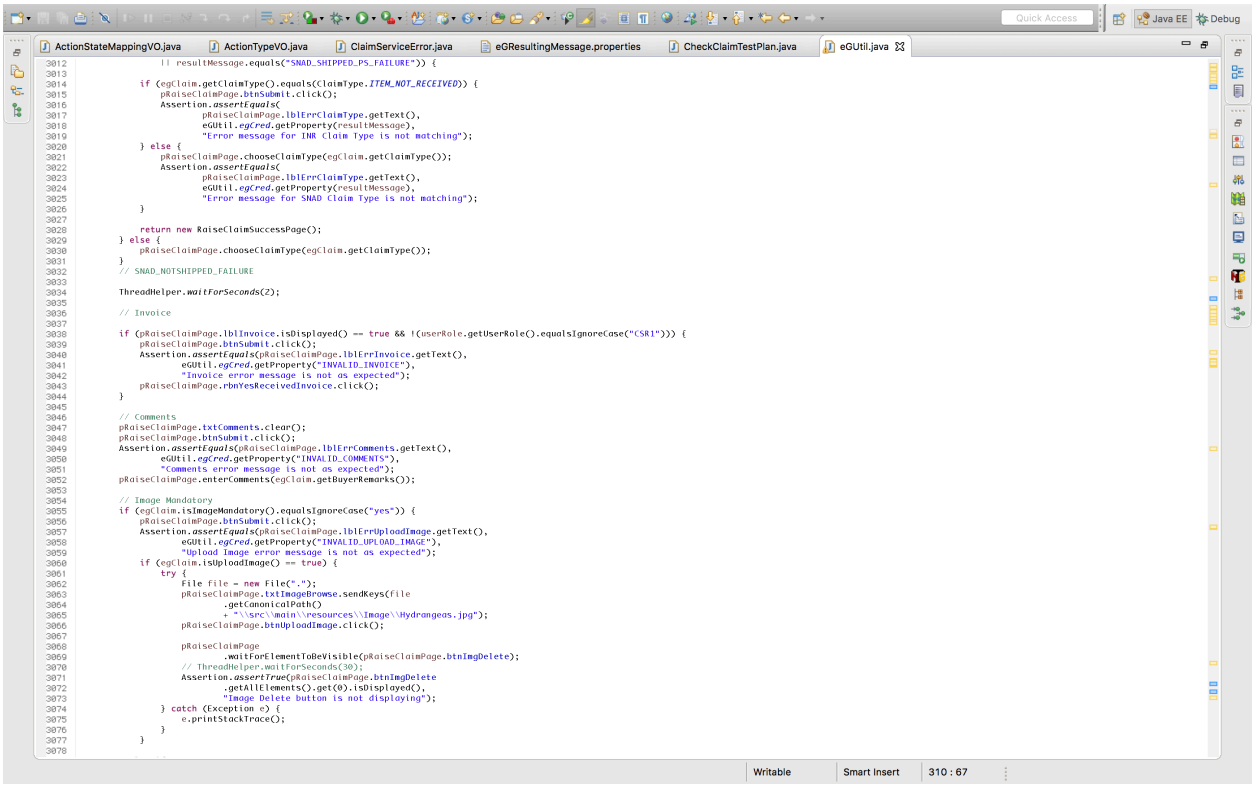
```
    if (escrowdates.getEscrowStatus().getStatusCode() == Status.SHIPPED
            .getStatusCode()) {
        SignInPage pSignInPage = new SignInPage(true);
        pSignInPage.signIn(escrow.getSeller().getUserLoginName(),
                "password");
        SellerTransactionDetailsPage pSellerTransactionDetailsPage = new SellerTransactionDetailsPage(
                escrow.getEscrowID(), true);
        pSellerTransactionDetailsPage.markReadyToShip(escrow);
        pSellerTransactionDetailsPage.getDriver().quit();

        escrow.setTracking(new PSTracking(escrow
                .getCombinedOrderIndicator(), false));
        Calendar instance = Calendar.getInstance();
        instance.add(Calendar.DATE, 1);
        Date tempDate = instance.getTime();
        PowershipUtil.updateShipmentStatus(escrow,
                EbayAWBStatusEnum.PICKED_IN_TRANSIT, tempDate);

    }

}

/**
 * Utility function to convert java Date to TimeZone format
 *
 * @param date
 * @param format
 * @param timeZone
 *            {@code formatDateToString(date, "dd MMM yyyy hh:mm:ss a", "IST")}
 * @return
 */
public static String formatDateToString(Date date, String format,
        String timeZone) {
    // null check
    if (date == null)
        return null;
    // create SimpleDateFormat object with input format
    SimpleDateFormat sdf = new SimpleDateFormat(format);
    // default system timezone if passed null or empty
    if (timeZone == null || "".equalsIgnoreCase(timeZone.trim())) {
        timeZone = Calendar.getInstance().getTimeZone().getID();
    }
    // set timezone to SimpleDateFormat
    sdf.setTimeZone(TimeZone.getTimeZone(timeZone));
    // return Date in required format with timezone as String
    return sdf.format(date);
}

public static Date dateFormatInPST(String expDate, boolean isEnd)
        throws Exception {
    if (expDate == null || expDate.trim().equals(""))
        return null;
    if (isEnd) {
        expDate = expDate.trim() + " 23:59:59";
    } else {
        expDate = expDate.trim() + " 00:00:00";
    }

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    df.setTimeZone(TimeZone.getTimeZone("IST"));
    Date date = df.parse(expDate);

    return convertToPST(date);

}

public static Date convertToPST(Date dateToConvert) {
```


ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
eGResultingMessage.properties
CheckClaimTestPlan.java
eGUtil.java
Writable
Smart Insert
310 : 67

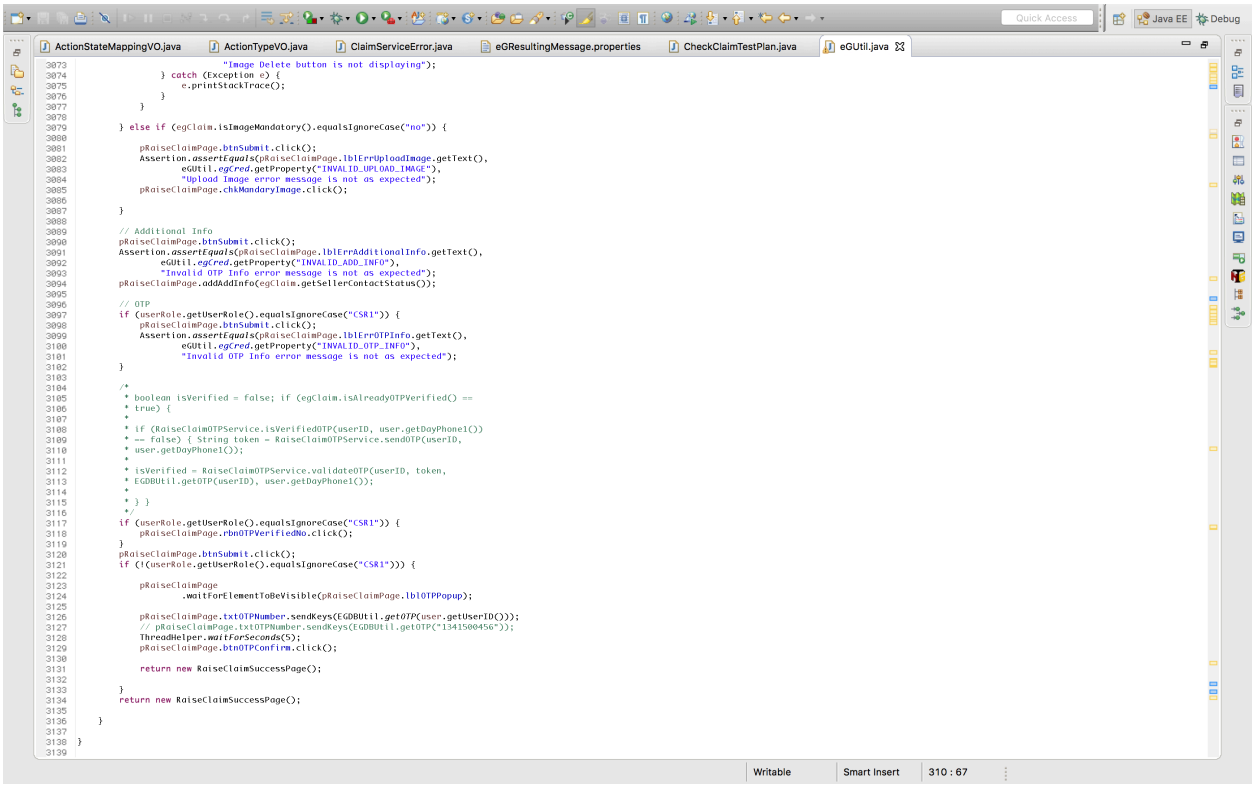

ActionStateMappingVO.java
ActionTypeVO.java
ClaimServiceError.java
eGResultingMessage.properties
CheckClaimTestPlan.java
eGUtil.java
Writable
Smart Insert
310 : 67



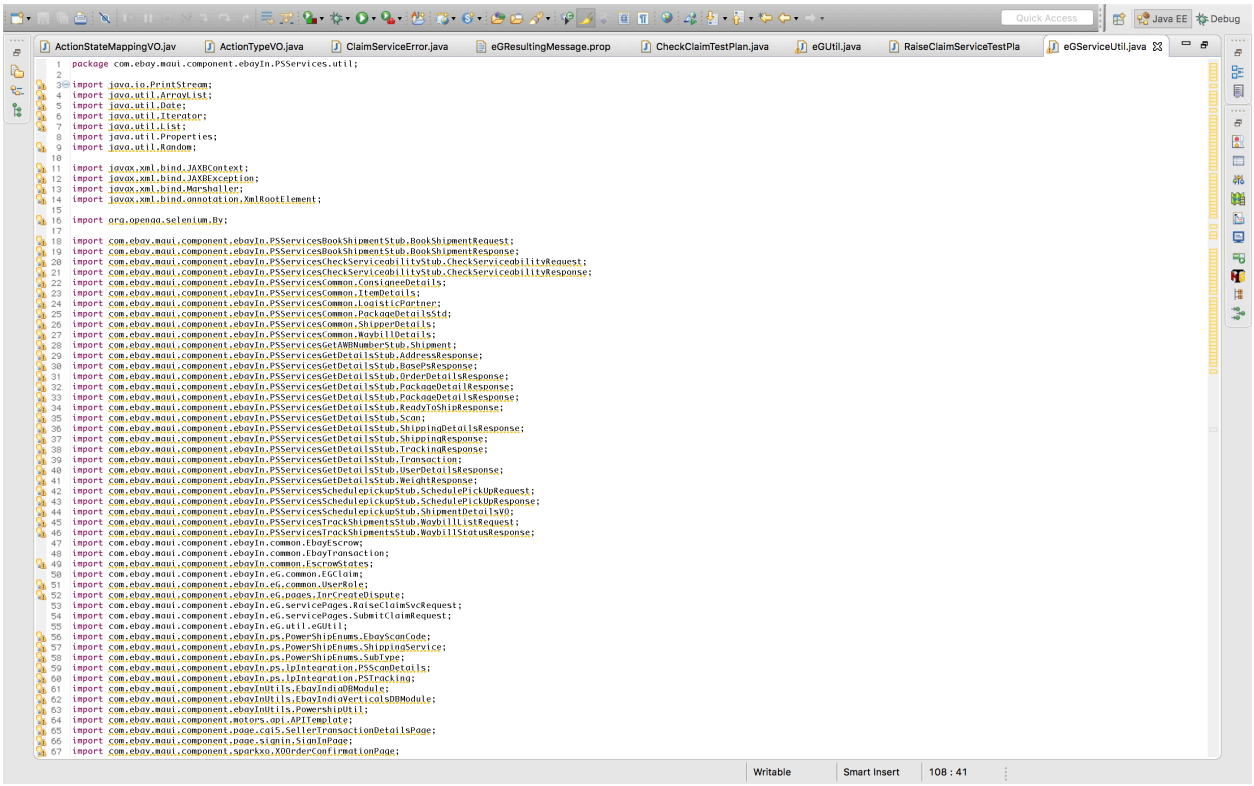
```
2945    public static RaiseClaimSuccessPage raiseClaimNegative(EbayEscrow escrow,
2946            User user, UserRole userRole, String resultMessage)
2947            throws Exception {
2948
2949        EGClaim egClaim = escrow.getEgClaim();
2950        RaiseClaimHomePage pRaiseClaimHomePage = null;
2951        if (user != null)
2952            pRaiseClaimHomePage = new BuyerRaiseClaimHomePage(
2953                    user.getUserLoginName(), "password");
2954        // pRaiseClaimHomePage = new BuyerRaiseClaimHomePage("testgsb1",
2955        // "password");
2956
2957        else
2958            pRaiseClaimHomePage = new RaiseClaimHomePage(
2959                    PowershipUtil.adminCred.getProperty("eg_csr1"), "password");
2960
2961        // return raiseClaimNew(pRaiseClaimHomePage, "366000394" , egClaim,
2962        // user, userRole);
2963        return raiseClaimNegative(pRaiseClaimHomePage, escrow.getPaisaPayID(),
2964                egClaim, user, userRole, resultMessage);
2965
2966    }
2967
2968    public static synchronized RaiseClaimSuccessPage raiseClaimNegative(
2969            RaiseClaimHomePage pRaiseClaimHomePage, String paisaPayId,
2970            EGClaim egClaim, User user, UserRole userRole, String resultMessage)
2971            throws Exception {
2972
2973        RaiseClaimPage pRaiseClaimPage = pRaiseClaimHomePage
2974                .viewOrderDetails(paisaPayId);
2975
2976        ThreadHelper.waitForSeconds(3);
2977        if (pRaiseClaimPage.lblErrMessage.isElementPresent() == true
2978                && pRaiseClaimPage.btnSubmit.isEnabled() == false) {
2979            return new RaiseClaimSuccessPage();
2980        }
2981
2982        ThreadHelper.waitForSeconds(4);
2983        // Mobile number
2984        pRaiseClaimPage.lnktBuyerMobileEdit.click();
2985        pRaiseClaimPage.waitForElementEditable(pRaiseClaimPage.txtBuyerMobile);
2986        pRaiseClaimPage.txtBuyerMobile.clear();
2987        pRaiseClaimPage.txtBuyerMobile.sendKeys("5676757868");
2988        pRaiseClaimPage.btnSubmit.click();
2989        Assertion.assertEquals(pRaiseClaimPage.lblErrMobileNumber.getText(),
2990                eGUtil.egCred.getProperty("INVALID_MOBILE"),
2991                "validation message for mobile number is wrong");
2992        pRaiseClaimPage.txtBuyerMobile.clear();
2993        pRaiseClaimPage.txtBuyerMobile.sendKeys("9999999999");
2994        ThreadHelper.waitForSeconds(2);
2995        // Landline Number
2996        pRaiseClaimPage.lnktBuyerLandLineEdit.click();
2997        pRaiseClaimPage.lnktBuyerLandLineEdit.click();
2998        pRaiseClaimPage
2999                .waitForElementEditable(pRaiseClaimPage.txtBuyerLandLine);
3000        pRaiseClaimPage.txtBuyerLandLine.clear();
3001        pRaiseClaimPage.txtBuyerLandLine.sendKeys("243561");
3002        pRaiseClaimPage.btnSubmit.click();
3003        Assertion.assertEquals(pRaiseClaimPage.lblErrLandlineNumber.getText(),
3004                eGUtil.egCred.getProperty("INVALID_LANDLINE"),
3005                "validation message for Landline number is wrong");
3006        pRaiseClaimPage.txtBuyerLandLine.clear();
3007        pRaiseClaimPage.txtBuyerLandLine.sendKeys("8888888");
3008
3009        ThreadHelper.waitForSeconds(2);
3010        if (resultMessage.equals("INR_BEFORE10_FAILURE")
3011                || resultMessage.equals("SNAD_NOTSHIPPED_FAILURE")
```

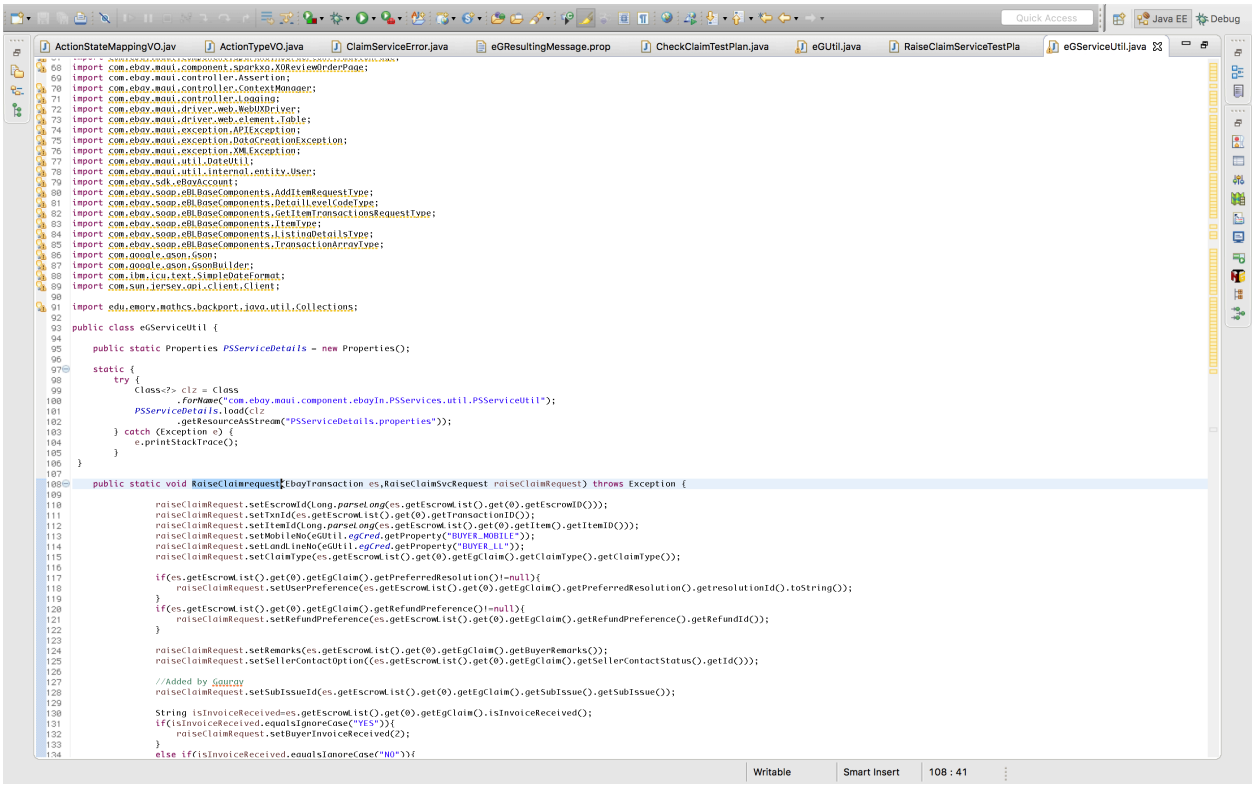


```
3012                || resultMessage.equals("SNAD_SHIPPED_PS_FAILURE")) {
3013
3014            if (egClaim.getClaimType().equals(ClaimType.ITEM_NOT_RECEIVED)) {
3015                pRaiseClaimPage.btnSubmit.click();
3016                Assertion.assertEquals(
3017                        pRaiseClaimPage.lblErrClaimType.getText(),
3018                        eGUtil.egCred.getProperty(resultMessage),
3019                        "Error message for INR Claim Type is not matching");
3020            } else {
3021                pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());
3022                Assertion.assertEquals(
3023                        pRaiseClaimPage.lblErrClaimType.getText(),
3024                        eGUtil.egCred.getProperty(resultMessage),
3025                        "Error message for SNAD Claim Type is not matching");
3026            }
3027
3028            return new RaiseClaimSuccessPage();
3029        } else {
3030            pRaiseClaimPage.chooseClaimType(egClaim.getClaimType());
3031        }
3032        // SNAD_NOTSHIPPED_FAILURE
3033
3034        ThreadHelper.waitForSeconds(2);
3035
3036        // Invoice
3037
3038        if (pRaiseClaimPage.lblInvoice.isDisplayed() == true && !(userRole.getUserRole().equalsIgnoreCase("CSR1"))) {
3039            pRaiseClaimPage.btnSubmit.click();
3040            Assertion.assertEquals(pRaiseClaimPage.lblErrInvoice.getText(),
3041                    eGUtil.egCred.getProperty("INVALID_INVOICE"),
3042                    "Invoice error message is not as expected");
3043            pRaiseClaimPage.rbnYesReceivedInvoice.click();
3044        }
3045
3046        // Comments
3047        pRaiseClaimPage.txtComments.clear();
3048        pRaiseClaimPage.btnSubmit.click();
3049        Assertion.assertEquals(pRaiseClaimPage.lblErrComments.getText(),
3050                eGUtil.egCred.getProperty("INVALID_COMMENTS"),
3051                "Comments error message is not as expected");
3052        pRaiseClaimPage.enterComments(egClaim.getBuyerRemarks());
3053
3054        // Image Mandatory
3055        if (egClaim.isImageMandatory().equalsIgnoreCase("yes")) {
3056            pRaiseClaimPage.btnSubmit.click();
3057            Assertion.assertEquals(pRaiseClaimPage.lblErrUploadImage.getText(),
3058                    eGUtil.egCred.getProperty("INVALID_UPLOAD_IMAGE"),
3059                    "Upload Image error message is not as expected");
3060            if (egClaim.isUploadImage() == true) {
3061                try {
3062                    File file = new File(".");
3063                    pRaiseClaimPage.txtImageBrowse.sendKeys(file
3064                            .getCanonicalPath()
3065                            + "\\src\\main\\resources\\Image\\Hydrangeas.jpg");
3066                    pRaiseClaimPage.btnUploadImage.click();
3067
3068                    pRaiseClaimPage
3069                            .waitForElementToBeVisible(pRaiseClaimPage.btnImgDelete);
3070                    // ThreadHelper.waitForSeconds(30);
3071                    Assertion.assertTrue(pRaiseClaimPage.btnImgDelete
3072                            .getAllElements().get(0).isDisplayed(),
3073                            "Image Delete button is not displaying");
3074                } catch (Exception e) {
3075                    e.printStackTrace();
3076                }
3077            }
3078
```



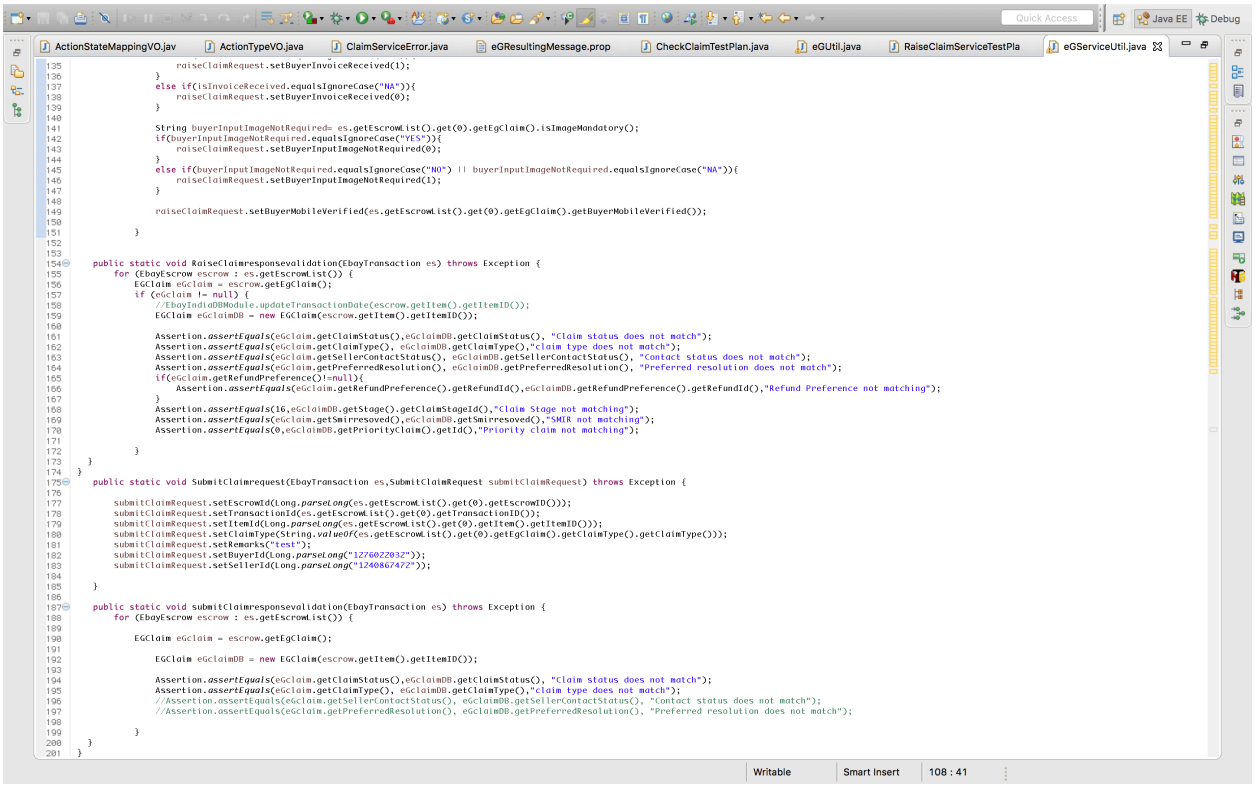
Service Util:





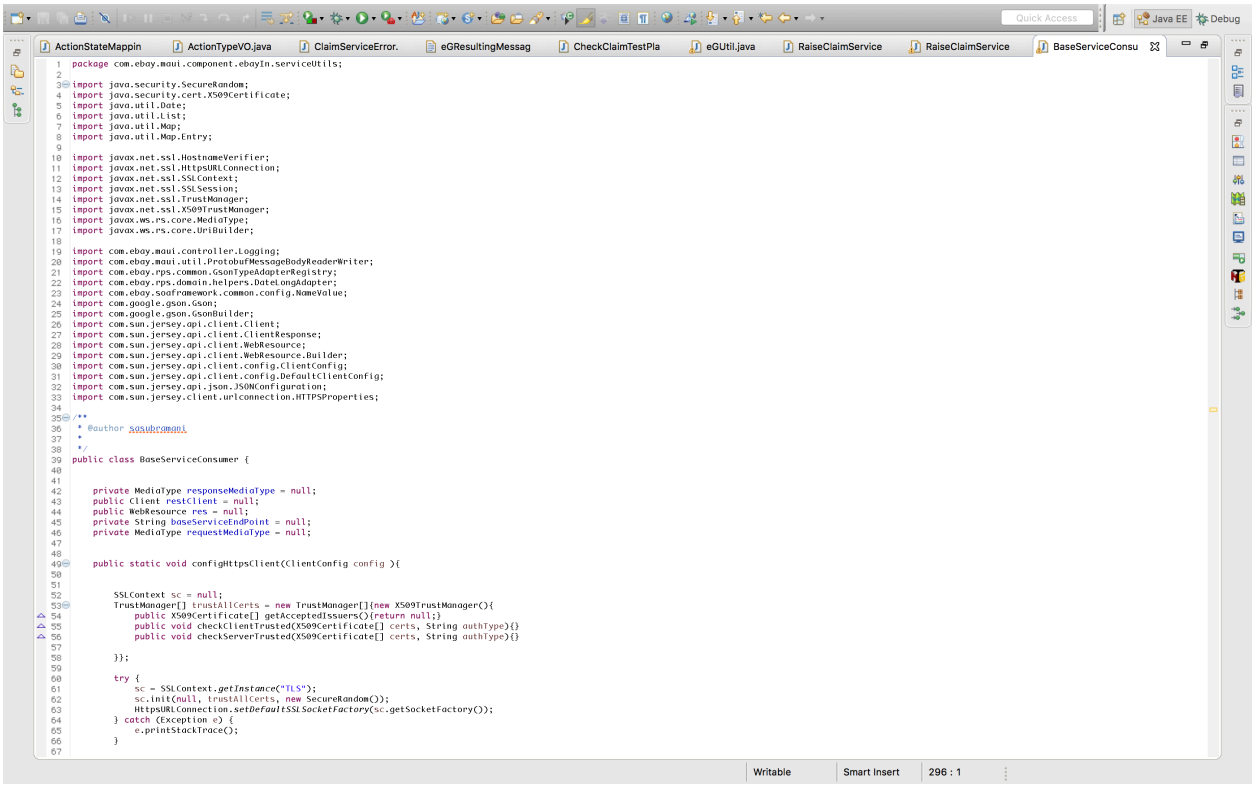
```
import com.ebay.maui.component.sparkxa.XOReviewOrderPage;
import com.ebay.maui.controller.Assertion;
import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.web.WebUXDriver;
import com.ebay.maui.driver.web.element.Table;
import com.ebay.maui.exception.APIException;
import com.ebay.maui.exception.DataCreationException;
import com.ebay.maui.exception.XMLException;
import com.ebay.maui.util.DateUtil;
import com.ebay.maui.util.internal.entity.User;
import com.ebay.sdk.eBayAccount;
import com.ebay.soap.eBLBaseComponents.AddItemRequestType;
import com.ebay.soap.eBLBaseComponents.DetailLevelCodeType;
import com.ebay.soap.eBLBaseComponents.GetItemTransactionsRequestType;
import com.ebay.soap.eBLBaseComponents.ItemType;
import com.ebay.soap.eBLBaseComponents.ListingDetailsType;
import com.ebay.soap.eBLBaseComponents.TransactionArrayType;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.ibm.icu.text.SimpleDateFormat;
import com.sun.jersey.api.client.Client;

import edu.emory.mathcs.backport.java.util.Collections;

public class eGServiceUtil {

    public static Properties PSServiceDetails = new Properties();

    static {
        try {
            Class<?> clz = Class
                    .forName("com.ebay.maui.component.ebayIn.PSServices.util.PSServiceUtil");
            PSServiceDetails.load(clz
                    .getResourceAsStream("PSServiceDetails.properties"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void RaiseClaimrequest(EbayTransaction es,RaiseClaimSvcRequest raiseClaimRequest) throws Exception {

                raiseClaimRequest.setEscrowId(Long.parseLong(es.getEscrowList().get(0).getEscrowID()));
                raiseClaimRequest.setTxnId(es.getEscrowList().get(0).getTransactionID());
                raiseClaimRequest.setItemId(Long.parseLong(es.getEscrowList().get(0).getItem().getItemID()));
                raiseClaimRequest.setMobileNo(eGUtil.egCred.getProperty("BUYER_MOBILE"));
                raiseClaimRequest.setLandLineNo(eGUtil.egCred.getProperty("BUYER_LL"));
                raiseClaimRequest.setClaimType(es.getEscrowList().get(0).getEgClaim().getClaimType().getClaimType());

                if(es.getEscrowList().get(0).getEgClaim().getPreferredResolution()!=null){
                    raiseClaimRequest.setUserPreference(es.getEscrowList().get(0).getEgClaim().getPreferredResolution().getresolutionId().toString());
                }
                if(es.getEscrowList().get(0).getEgClaim().getRefundPreference()!=null){
                    raiseClaimRequest.setRefundPreference(es.getEscrowList().get(0).getEgClaim().getRefundPreference().getRefundId());
                }

                raiseClaimRequest.setRemarks(es.getEscrowList().get(0).getEgClaim().getBuyerRemarks());
                raiseClaimRequest.setSellerContactOption((es.getEscrowList().get(0).getEgClaim().getSellerContactStatus().getId()));

                //Added by Gaurav
                raiseClaimRequest.setSubIssueId(es.getEscrowList().get(0).getEgClaim().getSubIssue().getSubIssue());

                String isInvoiceReceived=es.getEscrowList().get(0).getEgClaim().isInvoiceReceived();
                if(isInvoiceReceived.equalsIgnoreCase("YES")){
                    raiseClaimRequest.setBuyerInvoiceReceived(2);
                }
                else if(isInvoiceReceived.equalsIgnoreCase("NO")){
```

Writable | Smart Insert | 108 : 41

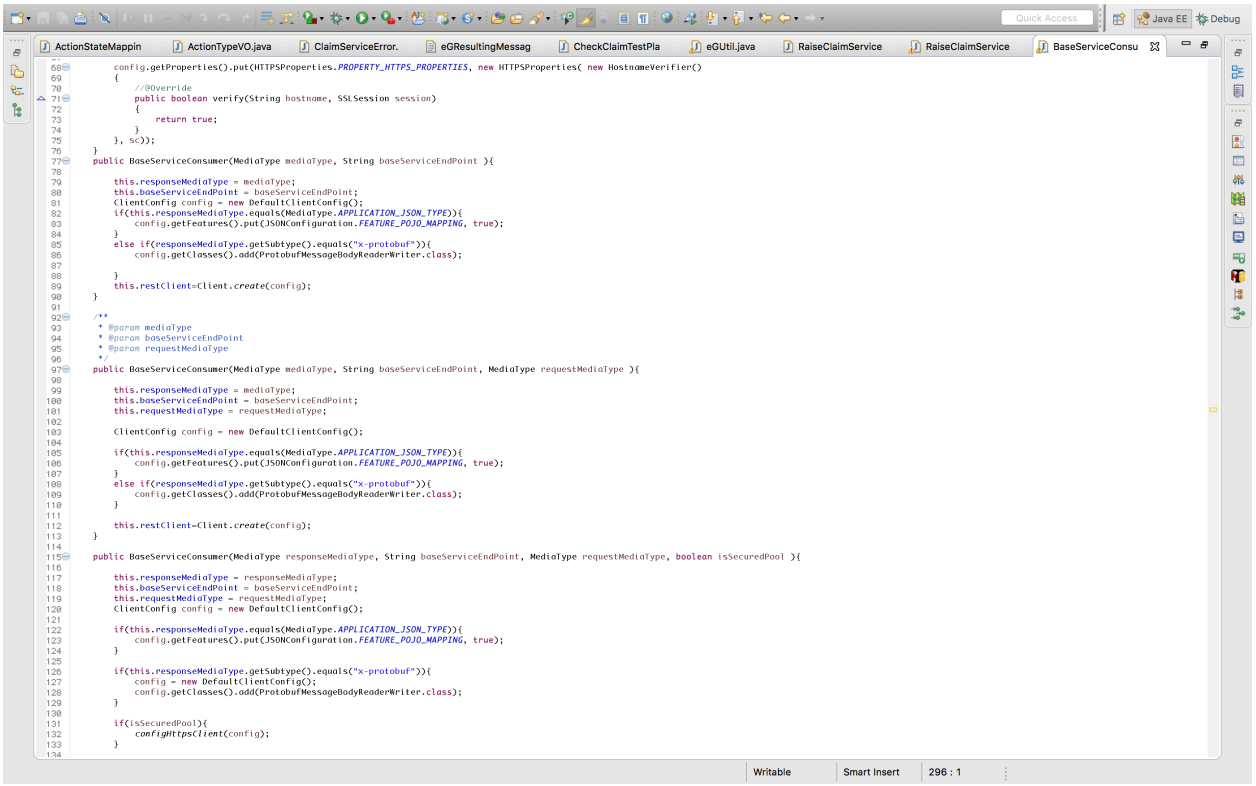


```
                    raiseClaimRequest.setBuyerInvoiceReceived(1);
                }
                else if(isInvoiceReceived.equalsIgnoreCase("NA")){
                    raiseClaimRequest.setBuyerInvoiceReceived(0);
                }

                String buyerInputImageNotRequired= es.getEscrowList().get(0).getEgClaim().isImageMandatory();
                if(buyerInputImageNotRequired.equalsIgnoreCase("YES")){
                    raiseClaimRequest.setBuyerInputImageNotRequired(0);
                }
                else if(buyerInputImageNotRequired.equalsIgnoreCase("NO") || buyerInputImageNotRequired.equalsIgnoreCase("NA")){
                    raiseClaimRequest.setBuyerInputImageNotRequired(1);
                }

                raiseClaimRequest.setBuyerMobileVerified(es.getEscrowList().get(0).getEgClaim().getBuyerMobileVerified());

            }


    public static void RaiseClaimresponsevalidation(EbayTransaction es) throws Exception {
        for (EbayEscrow escrow : es.getEscrowList()) {
            EGClaim eGclaim = escrow.getEgClaim();
            if (eGclaim != null) {
                //EbayIndiaDBModule.updateTransactionDate(escrow.getItem().getItemID());
                EGClaim eGclaimDB = new EGClaim(escrow.getItem().getItemID());

                Assertion.assertEquals(eGclaim.getClaimStatus(),eGclaimDB.getClaimStatus(), "Claim status does not match");
                Assertion.assertEquals(eGclaim.getClaimType(), eGclaimDB.getClaimType(),"claim type does not match");
                Assertion.assertEquals(eGclaim.getSellerContactStatus(), eGclaimDB.getSellerContactStatus(), "Contact status does not match");
                Assertion.assertEquals(eGclaim.getPreferredResolution(), eGclaimDB.getPreferredResolution(), "Preferred resolution does not match");
                if(eGclaim.getRefundPreference()!=null){
                    Assertion.assertEquals(eGclaim.getRefundPreference().getRefundId(),eGclaimDB.getRefundPreference().getRefundId(),"Refund Preference not matching");
                }
                Assertion.assertEquals(16,eGclaimDB.getStage().getClaimStageId(),"Claim Stage not matching");
                Assertion.assertEquals(eGclaim.getSmirresoved(),eGclaimDB.getSmirresoved(),"SMIR not matching");
                Assertion.assertEquals(0,eGclaimDB.getPriorityClaim().getId(),"Priority claim not matching");

            }
        }
    }
    public static void SubmitClaimrequest(EbayTransaction es,SubmitClaimRequest submitClaimRequest) throws Exception {

        submitClaimRequest.setEscrowId(Long.parseLong(es.getEscrowList().get(0).getEscrowID()));
        submitClaimRequest.setTransactionId(es.getEscrowList().get(0).getTransactionID());
        submitClaimRequest.setItemId(Long.parseLong(es.getEscrowList().get(0).getItem().getItemID()));
        submitClaimRequest.setClaimType(String.valueOf(es.getEscrowList().get(0).getEgClaim().getClaimType().getClaimType()));
        submitClaimRequest.setRemarks("test");
        submitClaimRequest.setBuyerId(Long.parseLong("1276022032"));
        submitClaimRequest.setSellerId(Long.parseLong("1240867472"));

    }

    public static void submitClaimresponsevalidation(EbayTransaction es) throws Exception {
        for (EbayEscrow escrow : es.getEscrowList()) {

            EGClaim eGclaim = escrow.getEgClaim();

                EGClaim eGclaimDB = new EGClaim(escrow.getItem().getItemID());

                Assertion.assertEquals(eGclaim.getClaimStatus(),eGclaimDB.getClaimStatus(), "Claim status does not match");
                Assertion.assertEquals(eGclaim.getClaimType(), eGclaimDB.getClaimType(),"claim type does not match");
                //Assertion.assertEquals(eGclaim.getSellerContactStatus(), eGclaimDB.getSellerContactStatus(), "Contact status does not match");
                //Assertion.assertEquals(eGclaim.getPreferredResolution(), eGclaimDB.getPreferredResolution(), "Preferred resolution does not match");

            }
    }
}
```

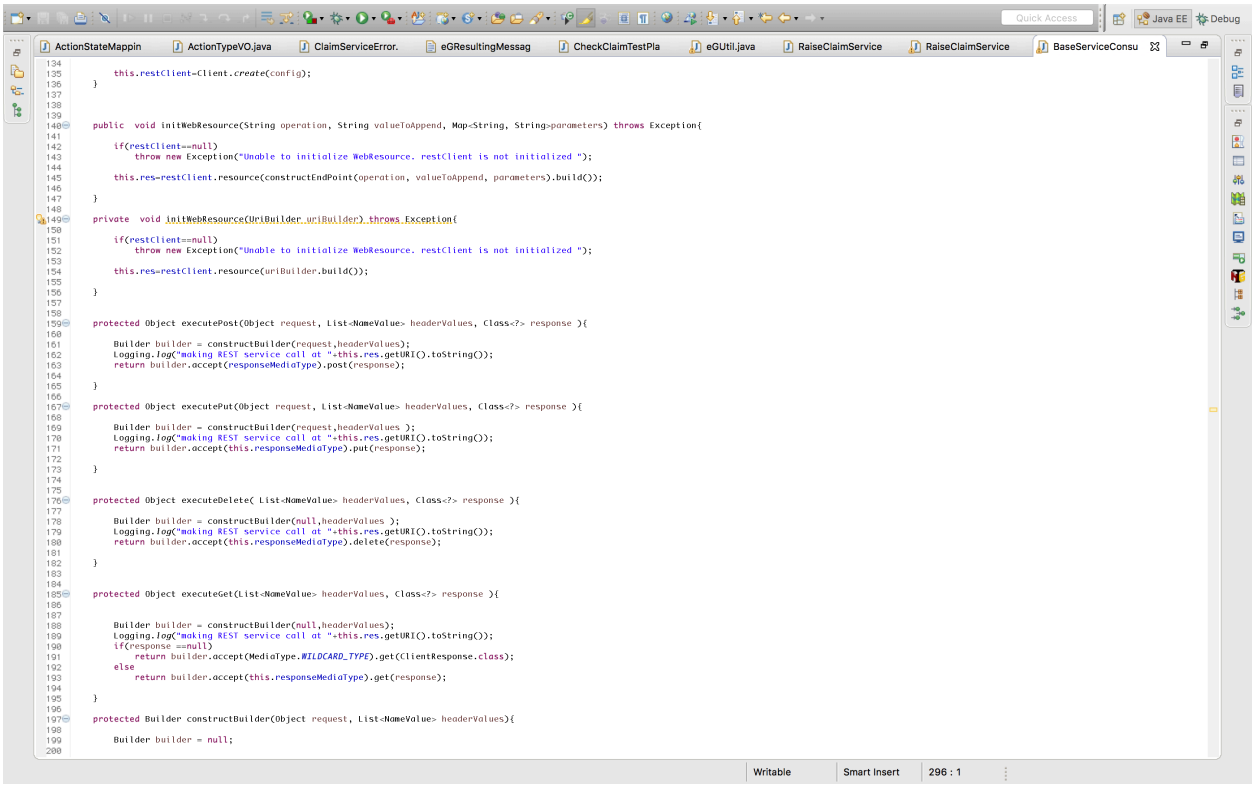
Writable | Smart Insert | 108 : 41



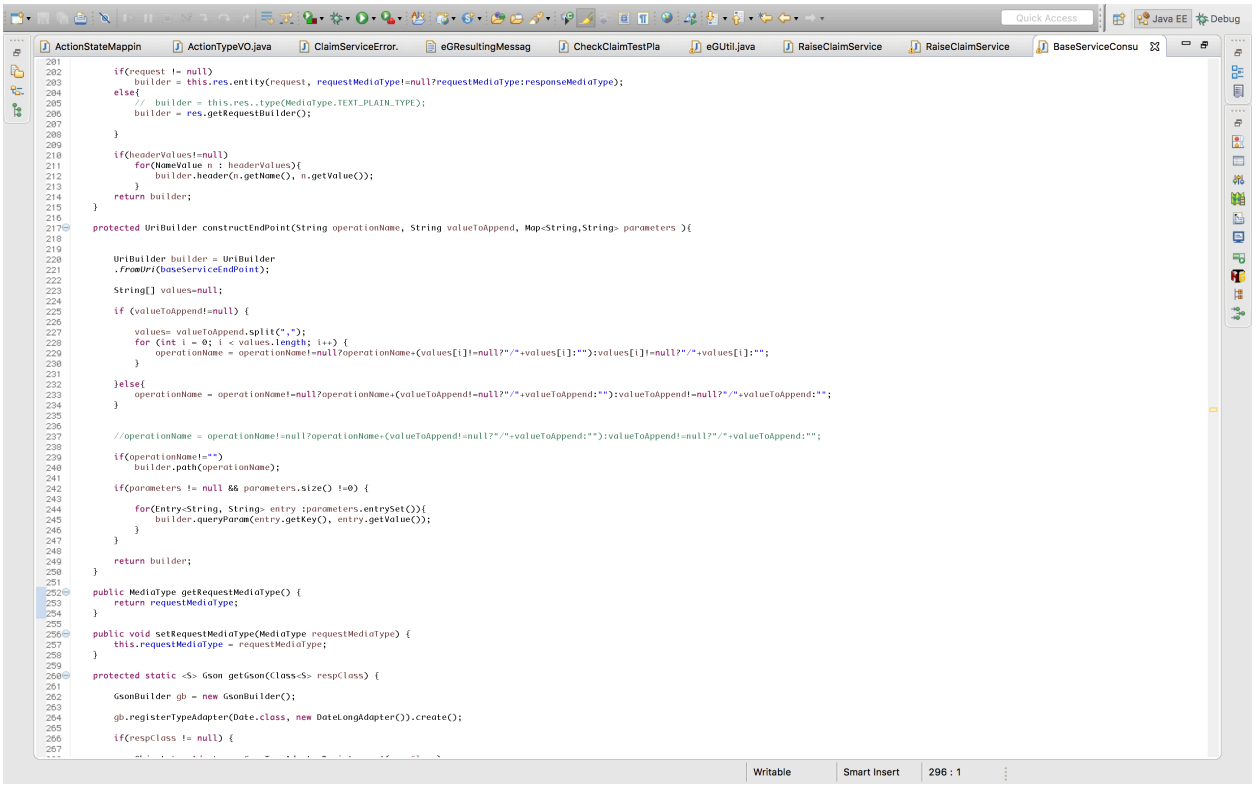
```
package com.ebay.maui.component.ebayIn.serviceUtils;

import java.security.SecureRandom;
import java.security.cert.X509Certificate;
import java.util.Date;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriBuilder;

import com.ebay.maui.controller.Logging;
import com.ebay.maui.util.ProtobufMessageBodyReaderWriter;
import com.ebay.rps.common.GsonTypeAdapterRegistry;
import com.ebay.rps.domain.helpers.DateLongAdapter;
import com.ebay.soaframework.common.config.NameValue;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.WebResource.Builder;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import com.sun.jersey.api.json.JSONConfiguration;
import com.sun.jersey.client.urlconnection.HTTPSProperties;

/**
 * @author sasubramani
 *
 */
public class BaseServiceConsumer {


    private MediaType responseMediaType = null;
    public Client restClient = null;
    public WebResource res = null;
    private String baseServiceEndPoint = null;
    private MediaType requestMediaType = null;


    public static void configHttpsClient(ClientConfig config ){


        SSLContext sc = null;
        TrustManager[] trustAllCerts = new TrustManager[]{new X509TrustManager(){
            public X509Certificate[] getAcceptedIssuers(){return null;}
            public void checkClientTrusted(X509Certificate[] certs, String authType){}
            public void checkServerTrusted(X509Certificate[] certs, String authType){}

        }};

        try {
            sc = SSLContext.getInstance("TLS");
            sc.init(null, trustAllCerts, new SecureRandom());
            HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
        } catch (Exception e) {
            e.printStackTrace();
        }
```



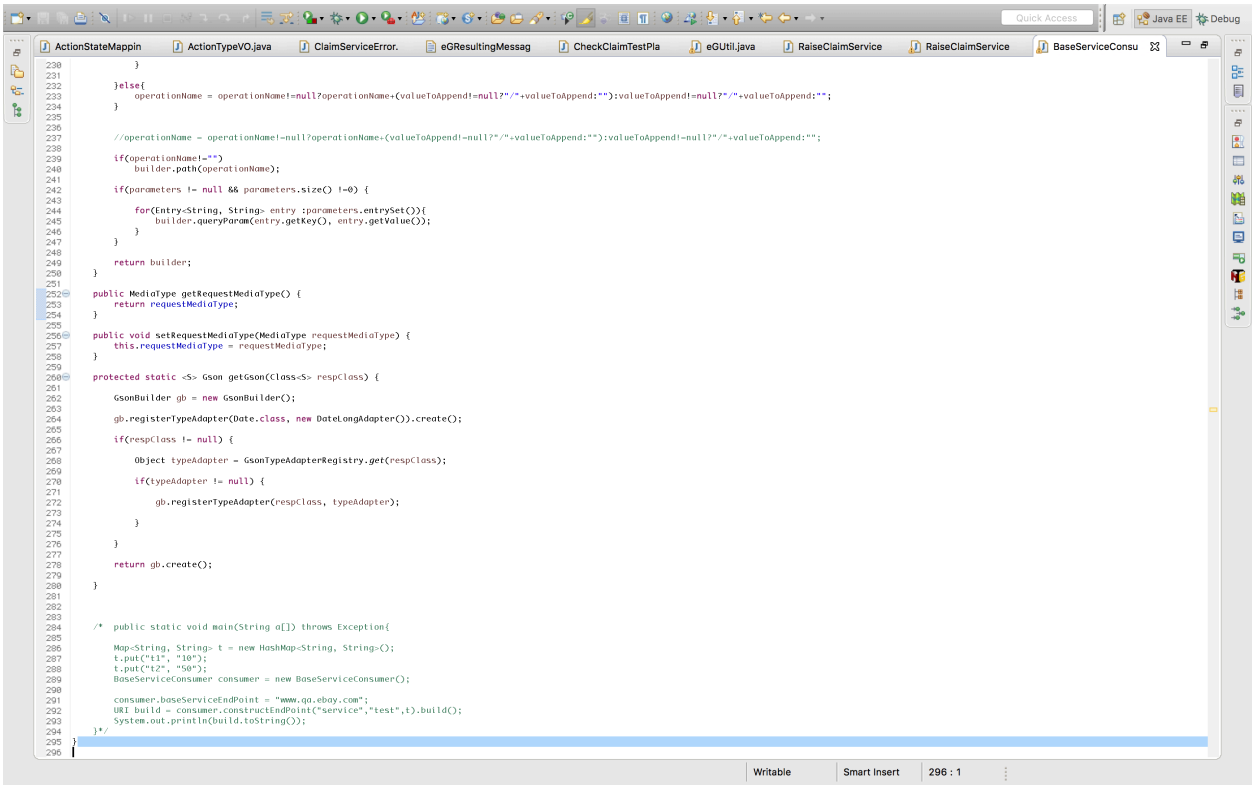
```
        config.getProperties().put(HTTPSProperties.PROPERTY_HTTPS_PROPERTIES, new HTTPSProperties( new HostnameVerifier()
        {
            //@Override
            public boolean verify(String hostname, SSLSession session)
            {
                return true;
            }
        }, sc));
    }
    public BaseServiceConsumer(MediaType mediaType, String baseServiceEndPoint ){

        this.responseMediaType = mediaType;
        this.baseServiceEndPoint = baseServiceEndPoint;
        ClientConfig config = new DefaultClientConfig();
        if(this.responseMediaType.equals(MediaType.APPLICATION_JSON_TYPE)){
            config.getFeatures().put(JSONConfiguration.FEATURE_POJO_MAPPING, true);
        }
        else if(responseMediaType.getSubtype().equals("x-protobuf")){
            config.getClasses().add(ProtobufMessageBodyReaderWriter.class);

        }
        this.restClient=Client.create(config);
    }

    /**
     * @param mediaType
     * @param baseServiceEndPoint
     * @param requestMediaType
     */
    public BaseServiceConsumer(MediaType mediaType, String baseServiceEndPoint, MediaType requestMediaType ){

        this.responseMediaType = mediaType;
        this.baseServiceEndPoint = baseServiceEndPoint;
        this.requestMediaType = requestMediaType;

        ClientConfig config = new DefaultClientConfig();

        if(this.responseMediaType.equals(MediaType.APPLICATION_JSON_TYPE)){
            config.getFeatures().put(JSONConfiguration.FEATURE_POJO_MAPPING, true);
        }
        else if(responseMediaType.getSubtype().equals("x-protobuf")){
            config.getClasses().add(ProtobufMessageBodyReaderWriter.class);
        }

        this.restClient=Client.create(config);
    }

    public BaseServiceConsumer(MediaType responseMediaType, String baseServiceEndPoint, MediaType requestMediaType, boolean isSecuredPool ){

        this.responseMediaType = responseMediaType;
        this.baseServiceEndPoint = baseServiceEndPoint;
        this.requestMediaType = requestMediaType;
        ClientConfig config = new DefaultClientConfig();

        if(this.responseMediaType.equals(MediaType.APPLICATION_JSON_TYPE)){
            config.getFeatures().put(JSONConfiguration.FEATURE_POJO_MAPPING, true);
        }

        if(this.responseMediaType.getSubtype().equals("x-protobuf")){
            config = new DefaultClientConfig();
            config.getClasses().add(ProtobufMessageBodyReaderWriter.class);
        }

        if(isSecuredPool){
            configHttpsClient(config);
        }
```

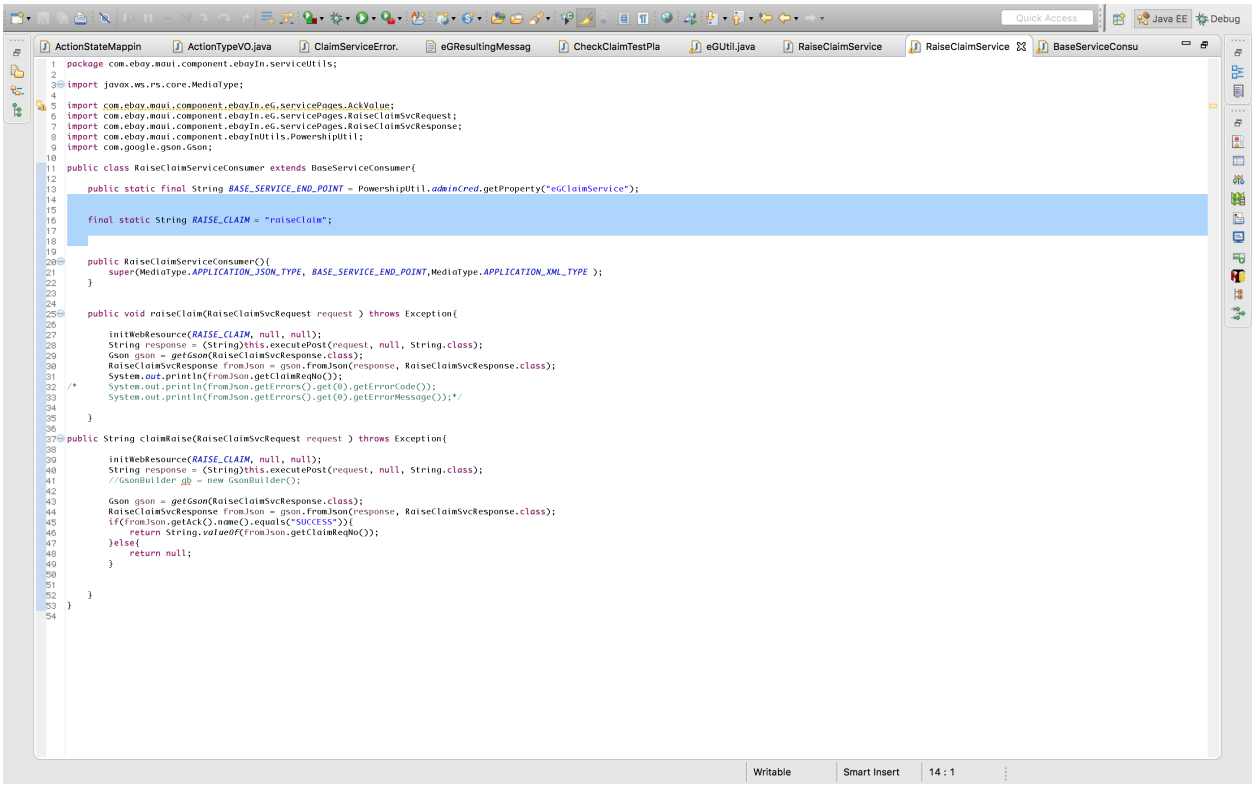


```
        this.restClient=Client.create(config);
    }


    public  void initWebResource(String operation, String valueToAppend, Map<String, String>parameters) throws Exception{

        if(restClient==null)
            throw new Exception("Unable to initialize WebResource. restClient is not initialized ");

        this.res=restClient.resource(constructEndPoint(operation, valueToAppend, parameters).build());

    }
    private  void initWebResource(UriBuilder uriBuilder) throws Exception{

        if(restClient==null)
            throw new Exception("Unable to initialize WebResource. restClient is not initialized ");

        this.res=restClient.resource(uriBuilder.build());

    }


    protected Object executePost(Object request, List<NameValue> headerValues, Class<?> response ){

        Builder builder = constructBuilder(request,headerValues);
        Logging.log("making REST service call at "+this.res.getURI().toString());
        return builder.accept(responseMediaType).post(response);

    }

    protected Object executePut(Object request, List<NameValue> headerValues, Class<?> response ){

        Builder builder = constructBuilder(request,headerValues );
        Logging.log("making REST service call at "+this.res.getURI().toString());
        return builder.accept(this.responseMediaType).put(response);

    }


    protected Object executeDelete( List<NameValue> headerValues, Class<?> response ){

        Builder builder = constructBuilder(null,headerValues );
        Logging.log("making REST service call at "+this.res.getURI().toString());
        return builder.accept(this.responseMediaType).delete(response);

    }


    protected Object executeGet(List<NameValue> headerValues, Class<?> response ){


        Builder builder = constructBuilder(null,headerValues);
        Logging.log("making REST service call at "+this.res.getURI().toString());
        if(response ==null)
            return builder.accept(MediaType.WILDCARD_TYPE).get(ClientResponse.class);
        else
            return builder.accept(this.responseMediaType).get(response);

    }

    protected Builder constructBuilder(Object request, List<NameValue> headerValues){

        Builder builder = null;
```



```
        if(request != null)
            builder = this.res.entity(request, requestMediaType!=null?requestMediaType:responseMediaType);
        else{
            //  builder = this.res..type(MediaType.TEXT_PLAIN_TYPE);
            builder = res.getRequestBuilder();

        }

        if(headerValues!=null)
            for(NameValue n : headerValues){
                builder.header(n.getName(), n.getValue());
            }
        return builder;
    }
    protected UriBuilder constructEndPoint(String operationName, String valueToAppend, Map<String,String> parameters ){


        UriBuilder builder = UriBuilder
        .fromUri(baseServiceEndPoint);

        String[] values=null;

        if (valueToAppend!=null) {

            values= valueToAppend.split(",");
            for (int i = 0; i < values.length; i++) {
                operationName = operationName!=null?operationName+(values[i]!=null?"/"+values[i]:""):values[i]!=null?"/"+values[i]:"";
            }

        }else{
            operationName = operationName!=null?operationName+(valueToAppend!=null?"/"+valueToAppend:""):valueToAppend!=null?"/"+valueToAppend:"";
        }


        //operationName = operationName!=null?operationName+(valueToAppend!=null?"/"+valueToAppend:""):valueToAppend!=null?"/"+valueToAppend:"";

        if(operationName!="")
            builder.path(operationName);

        if(parameters != null && parameters.size() !=0) {

            for(Entry<String, String> entry :parameters.entrySet()){
                builder.queryParam(entry.getKey(), entry.getValue());
            }
        }

        return builder;
    }
    public MediaType getRequestMediaType() {
        return requestMediaType;
    }

    public void setRequestMediaType(MediaType requestMediaType) {
        this.requestMediaType = requestMediaType;
    }

    protected static <S> Gson getGson(Class<S> respClass) {

        GsonBuilder gb = new GsonBuilder();

        gb.registerTypeAdapter(Date.class, new DateLongAdapter()).create();

        if(respClass != null) {
```


ActionStateMappin
ActionTypeVO.java
ClaimServiceError.
eGResultingMessag
CheckClaimTestPla
eGUtil.java
RaiseClaimService
RaiseClaimService
BaseServiceConsu
Quick Access
Java EE
Debug
}
}else{
operationName = operationName!=null?operationName+(valueToAppend!=null?"/"+valueToAppend:""):valueToAppend!=null?"/"+valueToAppend:"";
}
//operationName = operationName!=null?operationName+(valueToAppend!=null?"/"+valueToAppend:""):valueToAppend!=null?"/"+valueToAppend:"";
if(operationName!="")
builder.path(operationName);
if(parameters != null && parameters.size() !=0) {
for(Entry<String, String> entry :parameters.entrySet()){
builder.queryParam(entry.getKey(), entry.getValue());
}
}
return builder;
}
public MediaType getRequestMediaType() {
return requestMediaType;
}
public void setRequestMediaType(MediaType requestMediaType) {
this.requestMediaType = requestMediaType;
}
protected static <S> Gson getGson(Class<S> respClass) {
GsonBuilder gb = new GsonBuilder();
gb.registerTypeAdapter(Date.class, new DateLongAdapter()).create();
if(respClass != null) {
Object typeAdapter = GsonTypeAdapterRegistry.get(respClass);
if(typeAdapter != null) {
gb.registerTypeAdapter(respClass, typeAdapter);
}
}
return gb.create();
}
/* public static void main(String a[]) throws Exception{
Map<String, String> t = new HashMap<String, String>();
t.put("t1", "10");
t.put("t2", "50");
BaseServiceConsumer consumer = new BaseServiceConsumer();
consumer.baseServiceEndPoint = "www.qa.ebay.com";
URI build = consumer.constructEndPoint("service","test",t).build();
System.out.println(build.toString());
}*/
}
Writable
Smart Insert
296 : 1


ActionStateMappin
ActionTypeVO.java
ClaimServiceError.
eGResultingMessag
CheckClaimTestPla
eGUtil.java
RaiseClaimService
RaiseClaimService
BaseServiceConsu
Quick Access
Java EE
Debug
package com.ebay.maui.component.ebayIn.serviceUtils;
import javax.ws.rs.core.MediaType;
import com.ebay.maui.component.ebayIn.eG.servicePages.AckValue;
import com.ebay.maui.component.ebayIn.eG.servicePages.RaiseClaimSvcRequest;
import com.ebay.maui.component.ebayIn.eG.servicePages.RaiseClaimSvcResponse;
import com.ebay.maui.component.ebayInUtils.PowershipUtil;
import com.google.gson.Gson;
public class RaiseClaimServiceConsumer extends BaseServiceConsumer{
public static final String BASE_SERVICE_END_POINT = PowershipUtil.adminCred.getProperty("eGClaimService");
final static String RAISE_CLAIM = "raiseClaim";
public RaiseClaimServiceConsumer(){
super(MediaType.APPLICATION_JSON_TYPE, BASE_SERVICE_END_POINT,MediaType.APPLICATION_XML_TYPE);
}
public void raiseClaim(RaiseClaimSvcRequest request) throws Exception{
initWebResource(RAISE_CLAIM, null, null);
String response = (String)this.executePost(request, null, String.class);
Gson gson = getGson(RaiseClaimSvcResponse.class);
RaiseClaimSvcResponse fromJson = gson.fromJson(response, RaiseClaimSvcResponse.class);
System.out.println(fromJson.getClaimReqNo());
/*
System.out.println(fromJson.getErrors().get(0).getErrorCode());
System.out.println(fromJson.getErrors().get(0).getErrorMessage());*/
}
public String claimRaise(RaiseClaimSvcRequest request) throws Exception{
initWebResource(RAISE_CLAIM, null, null);
String response = (String)this.executePost(request, null, String.class);
//GsonBuilder gb = new GsonBuilder();
Gson gson = getGson(RaiseClaimSvcResponse.class);
RaiseClaimSvcResponse fromJson = gson.fromJson(response, RaiseClaimSvcResponse.class);
if(fromJson.getAck().name().equals("SUCCESS")){
return String.valueOf(fromJson.getClaimReqNo());
}else{
return null;
}
}
}
Writable
Smart Insert
14 : 1