UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

5COSC019C

Object Oriented Programming

Name: Herath Mudiyanselage Shashini Nilukshi

UOW number :19540652

IIT number:20221258

01.TESTING AND SYSTEM VALIDATION

01.01. Implement automated testing

WestminsterShoppingManager Class

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.InputStream;
import java.io.PrintStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

public class WestminsterShoppingManagerTest {

    private final ByteArrayOutputStream outContent = new ByteArrayOutputStream();
    private final ByteArrayOutputStream errContent = new ByteArrayOutputStream();
    private final InputStream originalIn = System.in;
    private final PrintStream originalOut = System.out;
    private final PrintStream originalErr = System.err;

    private WestminsterShoppingManager shoppingManager;

    @Before
    public void setUpStreams() {
```

```
        System.setOut(new PrintStream(outContent));
        System.setErr(new PrintStream(errContent));
        shoppingManager = new WestminsterShoppingManager();
    }

    @After
    public void restoreStreams() {
        System.setIn(originalIn);
        System.setOut(originalOut);
        System.setErr(originalErr);
    }

    @Test
    public void testAddElectronicProduct() {
        String input = "1\n1\n123\nTestElectronic\n5\n50.0\nTestManufacturer\nTestBrand\n12
months\n6\n";
        System.setIn(new ByteArrayInputStream(input.getBytes()));
        shoppingManager.runMenu();
        assertTrue(outContent.toString().contains("Electronic product is added."));
    }

    @Test
    public void testRemoveClothingProduct() {
        String input = "2\n2\n456\n";
        System.setIn(new ByteArrayInputStream(input.getBytes()));

        // Add a clothing product to be removed
        Clothing clothing = new Clothing("456", "TestClothing", 3, 25.0, "TestManufacturer", "M",
"Blue", "Clothing");
        shoppingManager.addClothingProduct(clothing);

        shoppingManager.runMenu();
```

```
        assertTrue(outContent.toString().contains("Number of existing products: 0"));
    }

    @Test
    public void testSaveAndLoadFromFile() {
        // Save a product to file
        String inputSave = "1\n1\n123\nTestElectronic\n5\n50.0\nTestManufacturer\nTestBrand\n12
months\n4\n";
        System.setIn(new ByteArrayInputStream(inputSave.getBytes()));
        shoppingManager.runMenu();

        // Load from file
        String inputLoad = "5\n";
        System.setIn(new ByteArrayInputStream(inputLoad.getBytes()));
        shoppingManager.runMenu();

        assertTrue(outContent.toString().contains("Loaded From Files."));
        assertNotNull(shoppingManager.loadAllFromFile("All Products.txt"));
    }

    @Test
    public void testInvalidInput() {
        String input = "invalid\n1\n123\nTestElectronic\n5\n50.0\nTestManufacturer\nTestBrand\n12
months\n6\n";
        System.setIn(new ByteArrayInputStream(input.getBytes()));
        shoppingManager.runMenu();
        assertTrue(outContent.toString().contains("Invalid input. Please enter a number."));
    }

    @Test
    public void testInvalidDataTypeInput() {
        String input = "1\n1\n123\nTestElectronic\ninvalid\n50.0\nTestManufacturer\nTestBrand\n12
months\n6\n";
```

```
    System.setIn(new ByteArrayInputStream(input.getBytes()));
    shoppingManager.runMenu();
    assertTrue(outContent.toString().contains("Invalid input. Please enter the correct data type."));
}

@Test
public void testRemoveNonexistentProduct() {
    String input = "2\n2\n999\n";
    System.setIn(new ByteArrayInputStream(input.getBytes()));
    shoppingManager.runMenu();
    assertTrue(outContent.toString().contains("The product with ID 999 is not found."));
}

@Test
public void testSaveToFileIOException() {
    // Force an IOException during save to file
    String filePath = "nonexistentDirectory/All Products.txt";
    shoppingManager.saveProductsToFile(List.of(), filePath);
    assertTrue(outContent.toString().contains("Error reading or writing the file."));
}

@Test
public void testRemoveProductFromFileIOException() {
    // Force an IOException during remove product from file
    String productID = "123";
    shoppingManager.removeProductFromFile(productID, "nonexistentDirectory/All Products.txt");
    assertTrue(outContent.toString().contains("Error reading or writing the file."));
}

@Test
public void testSortingItems() {
    // Add unsorted products
```

```
        Electronics electronics1 = new Electronics("789", "TestElectronic1", 3, 40.0, "Manufacturer1",
"Brand1", "24 months", "Electronics");
        Electronics electronics2 = new Electronics("456", "TestElectronic2", 5, 30.0, "Manufacturer2",
"Brand2", "12 months", "Electronics");
        shoppingManager.addElectronicProduct(electronics1);
        shoppingManager.addElectronicProduct(electronics2);

        // Sort products
        shoppingManager.sortingItems(shoppingManager.productMap.get("All"));

        // Check if the products are sorted by ID
        assertEquals("456", shoppingManager.productMap.get("All").get(0).getProductID());
        assertEquals("789", shoppingManager.productMap.get("All").get(1).getProductID());
    }
}
```

UserSignIn Class

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import static org.junit.Assert.assertTrue;

public class UserSignInSeleniumTest {

    private WebDriver driver;

    @Before
```

```
public void setUp() {
    // Set the path to the ChromeDriver executable
    System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

    // Initialize the ChromeDriver instance
    driver = new ChromeDriver();

    // Open the UserSignIn application
    driver.get("file:///path/to/UserSignIn.html");
}

@Test
public void testSignInAndLogIn() {
    // Find the username, password, and log in button elements
    WebElement usernameField = driver.findElement(By.id("usernameField"));
    WebElement passwordField = driver.findElement(By.id("passwordField"));
    WebElement logInButton = driver.findElement(By.id("logInButton"));

    // Enter valid username and password
    usernameField.sendKeys("testuser");
    passwordField.sendKeys("testpassword");

    // Click the log in button
    logInButton.click();

    // Check if the online shop page is displayed
    WebElement onlineShopTitle = driver.findElement(By.id("onlineShopTitle"));
    assertTrue(onlineShopTitle.isDisplayed());
}
```

```
    @Test
    public void testInvalidCredentials() {
        // Find the username, password, and log in button elements
        WebElement usernameField = driver.findElement(By.id("usernameField"));
        WebElement passwordField = driver.findElement(By.id("passwordField"));
        WebElement logInButton = driver.findElement(By.id("logInButton"));

        // Enter invalid username and password
        usernameField.sendKeys("invaliduser");
        passwordField.sendKeys("invalidpassword");

        // Click the log in button
        logInButton.click();

        // Check if an error message is displayed
        WebElement errorMessage = driver.findElement(By.id("errorMessage"));
        assertTrue(errorMessage.isDisplayed());
    }

    @After
    public void tearDown() {
        // Close the browser window
        driver.quit();
    }
}
```

OnlineShop Class

```
import org.junit.jupiter.api.Test;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

public class OnlineShopTest {

    @Test
    public void testVisualizeItems() {
        // Mock data or prepare a test file for products
        WestminsterShoppingManager mockManager = new WestminsterShoppingManager();
        List<Product> mockProducts = List.of(
                new Electronics("E1", "TV", 500.0, "Samsung", "Smart", "1 year"),
                new Clothing("C1", "T-Shirt", 20.0, "Nike", "M", "Blue")
                // Add more mock products as needed
        );
        mockManager.setMockProducts(mockProducts);

        OnlineShop onlineShop = new OnlineShop();
```

```
        onlineShop.setShoppingManager(mockManager);

        // Assume the "All Products.txt" file contains the mock products
        onlineShop.visualizeItems("All");

        // Assert that the table model in OnlineShop is updated with mock product data
        DefaultTableModel tableModel = onlineShop.getTableModel();
        assertNotNull(tableModel);
        assertEquals(2, tableModel.getRowCount());

        // Assert that the first row contains the expected data
        assertEquals("E1", tableModel.getValueAt(0, 0));
        assertEquals("TV", tableModel.getValueAt(0, 1));
        assertEquals("Electronics", tableModel.getValueAt(0, 2));
        assertEquals(500.0, tableModel.getValueAt(0, 3));
        assertEquals("Number of Available Items: 1,Manufacturer: Samsung, Brand: Smart, Warranty: 1
year", tableModel.getValueAt(0, 4));

        // Add more assertions based on your expected behavior
    }

    @Test
    public void testAddToCart() {
        // Mock data or prepare a test file for products
        WestminsterShoppingManager mockManager = new WestminsterShoppingManager();
        List<Product> mockProducts = List.of(
                new Electronics("E1", "TV", 500.0, "Samsung", "Smart", "1 year")
                // Add more mock products as needed
        );
        mockManager.setMockProducts(mockProducts);

        OnlineShop onlineShop = new OnlineShop();
        onlineShop.setShoppingManager(mockManager);
```

```
        // Set up the UI components needed for the test
        onlineShop.setQuantityTextField(new JTextField("2"));
        onlineShop.setProductDetailsArea(new JTextArea("Product ID: E1\nName: TV\nCategory: Electronics\nPrice: $500.0\nInfo: Number of Available Items: 1,Manufacturer: Samsung, Brand: Smart, Warranty: 1 year"));

        // Assume the user clicks the "Add to Cart" button
        onlineShop.getAddToCartButton().doClick();

        // Assert that the shopping cart is updated with the correct data
        ShoppingCart shoppingCart = onlineShop.getShoppingCart();
        assertNotNull(shoppingCart);
        assertEquals(1, shoppingCart.getItems().size());
        ShoppingCartItem cartItem = shoppingCart.getItems().get(0);
        assertEquals("E1", cartItem.getProductID());
        assertEquals("TV", cartItem.getProductName());
        assertEquals(2, cartItem.getQuantity());

        // Add more assertions based on your expected behavior
    }

public void setShoppingManager(WestminsterShoppingManager shoppingManager) {
        this.shoppingManager = shoppingManager;
    }


    // New method to set the "Add to Cart" button (for simulation in testing)
    public void setAddToCartButton(JButton addToCartButton) {
        this.addToCartButton = addToCartButton;
    }
```

```
    // Modified method to allow injecting products for testing
    public void visualizeItems(String selectedType, List<Product> loadedProducts) {
        tableModel.setRowCount(0);

        for (Product product : loadedProducts) {
            if ("All".equals(selectedType) || selectedType.equals(product.getProductType())) {
                Object[] rowData = createRowData(product);
                tableModel.addRow(rowData);
            }
        }

        // Set the renderer for the "Info" column after adding rows
        if (productTable.getColumnCount() > 4) {
            CustomCellRenderer customCellRenderer = new CustomCellRenderer();
            TableColumn infoColumn = productTable.getColumnModel().getColumn(4);
            infoColumn.setCellRenderer(customCellRenderer);
            infoColumn.setPreferredWidth(500);
        }
    }

    // Modified method to allow injecting dependencies for testing
    public double updateShoppingCart(String selectedData, int quantityToAdd,
WestminsterShoppingManager shoppingManager) {
        // Remaining code...
    }



    // New method to get the table model (for testing)
    public DefaultTableModel getTableModel() {
        return tableModel;
    }
```

```
    // New method to get the "Add to Cart" button (for testing)
    public JButton getAddToCartButton() {
        return addToCartButton;
    }
    // Helper methods for setting mock objects in OnlineShop class
    private static class WestminsterShoppingManagerMock extends WestminsterShoppingManager {
        private List<Product> mockProducts;

        public void setMockProducts(List<Product> mockProducts) {
            this.mockProducts = mockProducts;
        }

        @Override
        public List<Product> loadAllFromFile(String filename) {
            return mockProducts;
        }
    }

}
```

Main Class

```
import org.junit.jupiter.api.Test;
import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.Scanner;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class MainTest {

    @Test
```

```
    public void testShoppingManagerMenu() {
        // Redirect System.in for testing user input
        String input = "1\n5\n"; // Example: Select Shopping Manager (1) and exit (5)
        InputStream in = new ByteArrayInputStream(input.getBytes());
        System.setIn(in);

        // Capture console output for assertion
        InputStream originalSystemIn = System.in;
        System.setIn(in);
        Main.main(new String[]{});
        System.setIn(originalSystemIn);
    }

    @Test
    public void testCustomerMenu() {
        // Redirect System.in for testing user input
        String input = "2\n"; // Example: Select Customer (2)
        InputStream in = new ByteArrayInputStream(input.getBytes());
        System.setIn(in);

        // Capture console output for assertion
        InputStream originalSystemIn = System.in;
        System.setIn(in);
        Main.main(new String[]{});
        System.setIn(originalSystemIn);
    }

    @Test
    public void testInvalidUserType() {
        // Redirect System.in for testing user input
        String input = "3\n1\n5\n"; // Example: Invalid input (3), then Shopping Manager (1), and exit
(5)
        InputStream in = new ByteArrayInputStream(input.getBytes());
```

```
        System.setIn(in);

        // Capture console output for assertion
        InputStream originalSystemIn = System.in;
        System.setIn(in);
        Main.main(new String[]{});
        System.setIn(originalSystemIn);
    }

    // Add more test methods based on your specific use cases and requirements
}
```

02.TEST PLAN

WestminsterShoppingCart Class

SCREEN NAME		WestminsterShoppingManager Class	
PREPARED BY		H.M.S.Nilukshi	
DESIGNATION			
DATE		2024/01/08	
SCENARIO ID	1	SCENARIO DESCRIPTION	Testing the functionalities of WestministerShoppingManager class

S.NO	TESTCASE ID	TEST CASE DESCRIPTION	PRECONDITION	TEST DATA	EXPECTED RESULTS	POST CONDITION	ACTUAL RESULTS	STATUS	DEFECT ID	COMMENTS
S.NO 1	WSMCT001	Adding product to system.	The system is functioning normally right now.	Product ID = AHA123,Product Name = CropTop,Number of available Items = 34,Price = 2000.00,Manufacturer = Celiene,Product Type = Clothing,Color = red,Size = s	Adding product to the HashMap and prompt "New product is added."	The product should be added to the Hashmap	Adding product to the HashMap and prompt "New product is added."	Pass	-	-
S.No 2	WSMCT 002	Removing product from system.	The system is functioning normally right now.	Product ID = AHA123	Removing the product from Hashmap and prompt "The product with ID wsw is removed.	Product should be removed from both hashmap and file	System display "The product with ID AHA123 is removed. Product with ID AHA123 removed from the file."	Pass	-	-

					Product with ID wsw removed from the file."					
S.NO 3	WSMCT 003	Getting the current products in the system	There are products in the system	-	Prompt all the products in the system.	All the products in system should be shown to the user.	Prompt all the products in the system according to the alphabet method.	Pas s	-	-
S.No 4	WSMCT 004	Saving all the products entered to the file.	There are products in the hashmap .	-	Save all the products entered without deleting previous products.	Saving all the products to file.	Save all the products entered without deleting previous products and prompt "Saved to the Files."	Pas s	-	-
S.NO 5	WSMCT 005	Loading all the products from file.	There are products in the saved file.	-	Load all the products from file to hashmap.	Loading all products.	Loads all the products from file and prompts "Loaded From Files."	Pas s	-	-
S.No 6	WsMCT 006	Existing from the system.	The system is functioning normally right now.	-	Exit from programme.	Exit programme.	Exit from the programme.	pass		

Online Shop class

SCREEN NAME			OnlineShop Class							
PREPARED BY			H.M.S.Nilukshi							
DESIGNATION										
DATE			2024/01/12							
SCENARIO ID	1	SCENARIO DESCRIPTION	Testing the functionalities of OnlineShop class							
S.NO	TESTCASE ID	TEST CASE DESCRIPTION	PRECONDITION	TEST DATA	EXPECTED RESULTS	POST CONDITION	ACTUAL RESULTS	STATUS	DEFECT ID	COMMENTS
S.NO 1	OST001	Showing All products on the table.	The system is functioning normally right now.	Click Drop down “ALL”. Click “Show” button.	Show all the products on the table.	All Products should be there on the table.	Show all the products on the table.	Pass	-	-
S.No 2	OST 002	Showing Electronic products on the table.	The system is functioning normally right now.	Click Drop down “Electronics”. Click “Show” button	Show Electronic the products on the table.	Electronic Products should be there on the table.	Show Electronic products on the table.	Pass	-	-
S.NO 3	OST 003	Showing Clothing products on the table.	The system is functioning normally right now.	Click Drop down “Clothing”	Show Clothing the products on the table.	Clothing Products should be there on the table.	Show Clothing products on the table.	Pass	-	-

S.No 4	OST 004	Show the selected item.	Products should be loaded to table.	Click on a table row.	Show the selected rows data on the Jtext panel.	Show the details of selected item.	Show the selected product in the Jtextpanel .	Pas s	-	-
S.NO 5	OST 005	Add selected item to shopping cart.	Product is selected.	Click on the add to cart button.	Add the selected item to shopping cart table and give calculations.	Add selected item to shopping cart table calculate total.	Selected product added to the shopping cart table showed calculated prices.	Pas s	-	-
S.No 6	OST 006	Purchasing the shopping cart.	Shopping cart has products.	Click the purchase button.	Add user name to file and clear the shopping cart.	Clear the shoppingcart.	User name saved to a file called _purchase_history and cleared the shopping cart.	pass		
S.No7	OST 007	Removing the product from shopping cart.	Shopping cart has products.	Click on the row.Click the remove button.	The selected row will be deleted and show total according to that.	Row will be remove and total is calculated.	The selected row is deleted and show the correact total.	pass		

UserLogin Class

SCREEN NAME		UserLogin Class	
PREPARED BY		H.M.S.Nilukshi	
DESIGNATION			
DATE		2024/01/12	
SCENARIO ID	1	SCENARIO DESCRIPTION	Testing the functionalities of UserLoginclass

S.NO	TESTCASE ID	TEST CASE DESCRIPTION	PRECONDITION	TEST DATA	EXPECTED RESULTS	POST CONDITION	ACTUAL RESULTS	STATUS	DEFECT ID	COMMENTS
S.NO 1	ULCT001	Registering to the system.	The system is functioning normally right now.	Click on register button. Input username=Hello Password=Kitty Click on register button.	User is registered to the system and data is saved in a file.	User should be able to register.	User is registered to the system and data is saved in a file name user credentials.	Pass	-	-
S.No 2	ULCT 002	Login to the system.	The system is functioning normally right now.	Input Input username=Hello Password=Kitty Click on log in button.	User is logged in to the system and shows online shop.	User should be able to access online shop.	User is logged in and user is in the inline shop.	Pass	-	-

Console Menu

	Test case	Expected result	Actual result	Pass/Fail
01	Add a product	“To add new Electronic Product, Press 01 To add new Clothing Product, Press 02”	“To add new Electronic Product, Press 01 To add new Clothing Product, Press 02”	pass
	Enter invalid input	“Invalid input. Please enter a number.”	“Invalid input. Please enter a number.”	Pass
	Enter valid input	“Insert the Product Id:”	“Insert the Product Id:”	Pass
	Enter valid product Id (String inputs)	“Insert the Product Name:”	Insert the Product Id:AABC “Insert the Product Name:”	Pass
	Enter valid product name(String inputs)	“Insert the Number of items:”	“Insert the Number of items:”	Pass
	Enter valid number of items	Insert the price:	Insert the price:	Pass
	Enter Invalid number of items	Invalid input. Please enter a valid integer.	Invalid input. Please enter a valid integer.	Pass
	Enter valid price	Insert the Manufacturer:	Insert the Manufacturer:	Pass
	Enter invalid price	Invalid input. Please enter a valid double.	Invalid input. Please enter a valid double.	Pass
	Enter valid manufacturer(String inputs)	Insert the Brand:	Insert the Brand:	Pass
	Enter valid brand(String inputs)	Insert warranty Period:	Insert warranty Period:	Pass
	Enter valid warranty period(StringInputs)	Electronic product is added.	Electronic product is added.	Pass
	Enter valid size(String inputs)	Insert the color:	Insert the color:	Pass

	Enter valid color(String inputs)	Clothing product is added.	Clothing product is added.	Pass
02	Remove a product	Enter 1 if you want to remove Electronic item or Enter 2 If you want to remove clothing item	Enter 1 if you want to remove Electronic item or Enter 2 If you want to remove clothing item	pass
	Valid input	Enter the Product ID to remove Electronic item: or Enter the Product ID to remove Clothing item:	Enter the Product ID to remove Electronic item: or Enter the Product ID to remove Clothing item:	Pass
	Invalid input	Invalid Input. Enter 1 or 2:	Invalid Input. Enter 1 or 2:	Pass
	Enter valid product id	The product with ID ww is removed. Product with ID ww removed from the file.	The product with ID ww is removed. Product with ID ww removed from the file.	Pass
	Enter invalid product id	The product with ID w is not found.	The product with ID w is not found.	pass
03	View all products	View all savedproduct from file	View all savedproduct from file	pass
04	Save to file	Save product details to file	Save product details to file	pass
05	Load from file	Load product details from file.	Load product details from file.	pass
06.	Exit	Exit from the program.	Exit from the program.	pass