



L-Università ta' Malta
Faculty of Information &
Communication Technology

Department of
Computer Information
Systems

Software Testing

Nicole Abdilla (0142902L)
B.Sc. (Hons) Computing and Business

Study-unit: **Fundamentals of Software Testing**
Code: **CPS3230**
Lecturer: **Dr Mark Micallef**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I, the undersigned, declare that the assignment submitted is my work, except where acknowledged and referenced.

I understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected and will be given zero marks.

Nicole Abdilla



Student Name

Signature

Student Name

Signature

Student Name

Signature

Student Name

Signature

CPS3230

Course Code

Software Testing

Title of work submitted

29/10/2022

Date

Table of Contents

Project	4
Setting up the Project	4
The First Task	5
Introduction	5
Unit Testing and Test-Driven Development.....	5
Coverage Analysis	7
Test Doubles.....	7
Assumptions.....	7
The Second Task	8
The Feature File	8
The Step Definition File	9
Design Decisions	10
Testability of website and Recommendation for Market Alert UM	11

Project

The project presented in this documentation can be accessed from the below links:

Google Drive:

<https://drive.google.com/drive/folders/1d73YGt7UtHX63jKp1WamfeBeSxhfzAqv?usp=sharing>

GitHub:

https://github.com/nicoleabdilla20/cps3230_softwaretesting/

Setting up the Project

The below are instructions to load and run the tests of the project.

1. Download Folder from GitHub or Google Drive
2. Unzip Folder
3. Launch IntelliJ
4. Open the unzipped folder
5. In src\test\java\com\softwaretesting\tests\tdd\EbayTest update the chromedriver location to the destination stored on the PC. If the user does not have a chromedriver installed can install from <https://chromedriver.chromium.org/downloads>.
6. Right click on the test folder and run all tests

The First Task

This section provides a brief overview of the website that was used to screen scrape data.

Introduction

The website chosen for the first part of the assignment was eBay. It is accessible via the following link: <https://www.ebay.com/>. This is a customer-to-customer (C2C) and business-to-business (B2B) e-commerce website. It is known as a shopping and auction website where users can view and purchase items from all over the world. The website was selected because of its well-designed structure.

Unit Testing and Test-Driven Development

In the first part of the assignment, the classes in interest are Ebay and EbayTest. The hierarchy of the project folder can be viewed from Figure 1.

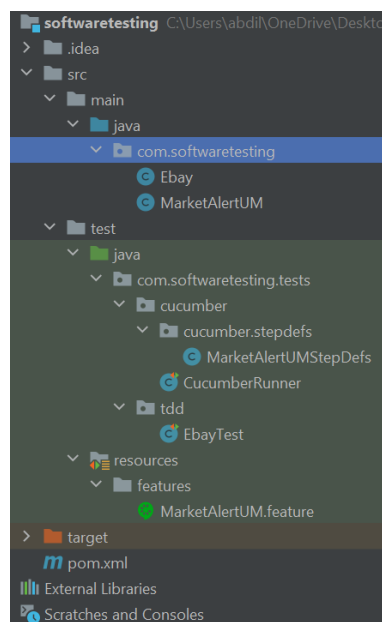


Figure 1 – Hierarchy of Project Folder

In the assignment guidelines, there were four primary objectives. The program needs to be able to do the following tasks:

1. Visit the Website,
2. Search for a Product,
3. Screen Scrape the Results, and
4. Upload 5 alerts to marketalertum.com

In the pom.xml file, multiple technologies were added from the Maven Repository which can be accessed from <https://mvnrepository.com/>. For the first part of the assignment, the below technologies were used:

- Webdriver was used to perform tests on the browser,

- Junit used as a testing framework for Java,
- Selenium used for automating tests,
- Jsoup was used for Screen Scrapping, and
- Unirest was used to POST and DELETE alerts

In the video presented with this submission, the tests found in the EbayTest class were run. The program can be seen visiting the website eBay. Then, searchers for iPad where data is being retrieved by the screen scraper method. The data retrieved from the website is outputted. Then, it uploads 5 alerts to the marketalertum.com with a post request using REST API framework. In the below snippet, the results are displayed.

Latest alerts for Nicole Abdilla




Apple iPad mini 4 128GB, Wi-Fi + Cellular (Unlocked), 7.9in - Space Grey **(just now)**




Pre-Owned - 128 GB - 7.9 in - Wi-Fi + Cellular

Price: €155.33

[Visit Item](#)




Apple - iPad 9 (Latest Model) with Wi-Fi - 64GB - Factory Sealed! **(just now)**




Free Fast Shipping - Factory Sealed - Factory Warranty Brand New

Price: €309.00

[Visit Item](#)




Apple iPad Pro 12.9-inch, Space Gray, 128GB, Wi-Fi Only, Original Box and Bundle **(just now)**




Fast Free 2-Day Shipping + Bundle Offer Included! Pre-Owned - 128 GB - 12.9 in - Wi-Fi

Price: €299.00

[Visit Item](#)



Apple iPad 6 (6th Gen) - (2018 Model) - 32GB - 128GB - Wi-Fi - Cellular - Good **(just now)**



30 Day Warranty - Free Returns - Top Rated Plus Seller Good - Refurbished

Price: €139.00

[Visit Item](#)



Apple iPad Pro (9.7 inch)- Wi-Fi - Cellular - All Colors - Good **(just now)**



30 Day Warranty - Free Returns - Top Rated Plus Seller Good - Refurbished

Price: €159.00

Figure 2 – Five alerts posted to marketalertum.com

Coverage Analysis

The unit tests were implemented in the EbayTest class. The class has 14 tests which have 100% code coverage as displayed in Figure 3.

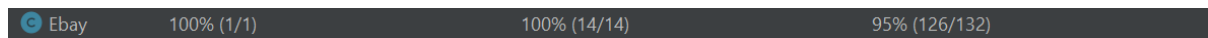


Figure 3 – Code Coverage

However, a 95%-line coverage was obtained, the reason for the 6 lines not being covered is because of the Try-catch block in three methods. The methods are:

- ScreenScraper (int num),
- SendRequest (int a, String h, String d, String u, String i, int p), and
- CheckforNulls (int a, String h, String d, String u, String i, int p).

No unit tests were created to check whether a class would throw an exception. As below snippet the red shows the line of code not covered by the tests.

```
64 // Method to pass non-null values to post alerts to website
    3 usages
65 public void SendRequest(int a, String h, String d, String u, String i, int p) {
66     try {
67         sendPOST(a, h, d, u, i, p);
68     } catch (UnirestException e) {
69         throw new RuntimeException(e);
70     }
71 }
```

Figure 4 – Line of Codes not Tested

Test Doubles

Maximum coverage was obtained by using test doubles. Figure 5 shows testing a method that converts a String to an Integer. The tests are testing for a String that contains integer, string, and null value. The object *ebay* was defined in the setup method.

Assumptions

The below are some assumptions that were made:

- The alert type is fixed as 6 for eBay items in the Screen Scraper.
- Assuming there are always results for the keyword iPad.

```
@Test
public void teststringToint_ToPass() {
    //Exercise
    ebay.stringToint( "123");
    //Verify
    Assertions.assertTrue( condition: true);
}

@Test
public void teststringToint_ToFail() {
    //Exercise
    ebay.stringToint( "abc");
    //Verify
    Assertions.assertFalse( condition: false);
}

@Test
public void teststringToint_ToNull() {
    //Exercise
    ebay.stringToint( "");
    //Verify
    Assertions.assertEquals( expected: 0, actual: 0);
}
```

Figure 5 – Tests on stringToint() method

The Second Task

The requirements for this section included writing several tests using Cucumber and Webdriver technologies.

The Feature File

```
Feature: MarketAlertUM

  Scenario: Test 1: Valid Login
    Given I am a user of marketalertum
    When I login using valid credentials
    Then I should see my alerts

  Scenario: Test 2: Invalid Login
    Given I am a user of marketalertum
    When I login using invalid credentials
    Then I should see the login screen again

  Scenario: Test 3: Alert Layout
    Given I am an administrator of the website and I upload 3 alerts
    And I am a user of marketalertum
    When I view a list of alerts
    Then each alert should contain an icon
    And each alert should contain a heading
    And each alert should contain a description
    And each alert should contain an image
    And each alert should contain a price
    And each alert should contain a link to the original product website

  Scenario: Test 4: Alert limit
    Given I am an administrator of the website and I upload more than 5
alerts
    Given I am a user of marketalertum
    When I view a list of alerts
    Then I should see 5 alerts

  Scenario Outline: Test 5: Icon check
    Given I am an administrator of the website and I upload an alert of
type <alert-type>
    Given I am a user of marketalertum
    When I view a list of alerts
    Then I should see 1 alerts
    And the icon displayed should be <icon file name>
  Examples:
    | alert-type | icon file name |
    | 1          | icon-car.png   |
    | 2          | icon-boat.png  |
    | 3          | icon-property-rent.png |
    | 4          | icon-property-sale.png |
    | 5          | icon-toys.png  |
    | 6          | icon-electronics.png |
```

Figure 6 – Feature File

The Step Definition File

The step definition file is named as MarketAlertUMStepDefs it uses methods from MarketAlertUM class. There is no conditional logic in this class. The logic occurs inside MarketAlertUM class. The purpose of the sleep() method is in case a user has a poor internet connection and it is used for the loading of the webpage for the video.

```
public class MarketAlertUMStepDefs {

    WebDriver browser = null;
    Ebay ebay;
    public MarketAlertUM marketAlertUMPageObject;

    public final String userId = "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44";
    public final String invaliduserId = "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44TEST";

    @Before
    public void setup() {
        System.setProperty("webdriver.chrome.driver",
"/Users/abdil/webtesting/chromedriver.exe");
        browser = new ChromeDriver();
        ebay = new Ebay();
        marketAlertUMPageObject = new MarketAlertUM(browser);
        browser.manage().window().maximize();
    }

    @After
    public void teardown() {
        browser.quit();
        browser = null;
        marketAlertUMPageObject = null;
        ebay = null;
    }

    //----- Scenario 1 -----
    @Given("I am a user of marketalertum")
    public void i_am_a_user_on_the_marketalertum_website() {
        marketAlertUMPageObject.getWebPage();
    }

    @When("I login using valid credentials")
    public void i_log_in_using_valid_credentials_using_userID() {
        marketAlertUMPageObject.login(userId);
    }

    @Then("I should see my alerts")
    public void i_should_be_logged_in() {
        marketAlertUMPageObject.testLoginSuccess();
    }
}
```

Figure 7 – Snippet of Step Definition File


Design Decisions

In the assignment guidelines, there was a list of steps to each scenario. Some steps were altered in the feature file such as the step “When I login using valid credentials”. In the beginning, of writing the tests, the string "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44" was added at the end of the step. However, this had to be removed for two main reasons. Firstly, it was not an efficient way to implement the step. Secondly, in the long run if there would be modification need to update every step where the user ID would be present. Now, if the user ID is changed, all that needs to be altered is the variable declared in the MarketAlertUMStepDefs. By this modification, duplicate code was reduced, which made scenario steps easier to read.

In the feature file, scenario 4 last step has no keyword to perform the step. Therefore, the keyword *Then* was added to perform the step. In scenario 5, there were <alert-type> and <icon file name> in the steps. Therefore, it was changed to a scenario outline that accepts multiple values. The value in the examples is being passed in the methods found in the MarketAlertUMStepDefs class. Overall, scenario 5 is telling the cucumber to run the scenario for alert type from 1 to 6. The file name is compared to the result icon from the alert.


The below snippet is the result of the first example:

Latest alerts for Nicole Abdilla



Apple iPad Air 16GB Wi-Fi 9.7 in Space Gray MD785LL/A Very Good 1

year warranty (just now)



Very Good - Refurbished · 16 GB · 9.7 in · Wi-Fi

Price: €100.20

[Visit item](#)

Figure 8 – First Example of Scenario Outline 5

The below is the result of the last example:

Latest alerts for Nicole Abdilla



Apple iPad Air 16GB Wi-Fi 9.7 in Space Gray MD785LL/A Very Good 1
year warranty (2 minutes ago)



Very Good - Refurbished · 16 GB · 9.7 in · Wi-Fi

Price: €100.20

[Visit item](#)

Figure 9 - Last Example of Scenario Outline 5

There were no other adjustments to the steps in the scenarios.

Testability of website and Recommendation for Market Alert UM

In the beginning of the assignment the website <https://www.scanmalta.com/shop/> was used. However, the website does not allow for web testers to load their pictures on another website. So, since the pictures were displayed as icons in the alerts, it did not look that neat. Therefore, had to find a better website to perform screen scrapping. Therefore, the recommendation with regard to testing a website would be to check whether a website would allow a user to perform a range of tests on it.

Testing marketalerum.com was not that troubling once understood how to retrieve data using Selenium from a table within HTML. The below are some suggestions for the marketalertum.com:

1. Each attribute would have either a class name or id as it would be easier to identify, and
2. User interface more similar to a marketplace
3. Adding a type of list to store items the user is interested in.

Overall, this assignment has taught me a lot about web testing.