Roostify Core Automation Documentation

(Automation using BDD Cucumber Framework)

Issue 1

-Authors

Bhairavi Mayee

Vivek Wagh

Table of Contents

PREFACE

1) <u>Purpose of this Document</u>

   The purpose of this document is to give a brief idea of the Automation Framework which is proposed as a part of R! Core Application. This document consists of:-

- Description of the framework used for R! Core Automation
- Description of all the Technical terms related to the framework
- Description of the Installation part required for the framework set up and Automation related stuff

This document focuses on giving a complete elaboration of the Automation functionality which is being used in Roostify Project for testing. This document provides guidance and material which is intended to assist the QA members involved. It will also prove useful for the background reading for every stakeholder involved in the project

INTRODUCTION/PRE-REQUISITES

Introduction to BDD Cucumber Framework

What is BDD?

In software engineering, **Behavior-Driven Development** (**BDD**) is an agile software development process that encourages collaboration between developers, QA and non-technical or business participants in a software project

Behavior Driven testing is an extension of TDD. Like in TDD in BDD also we write tests first and the add application code. The major difference that we get to see here are

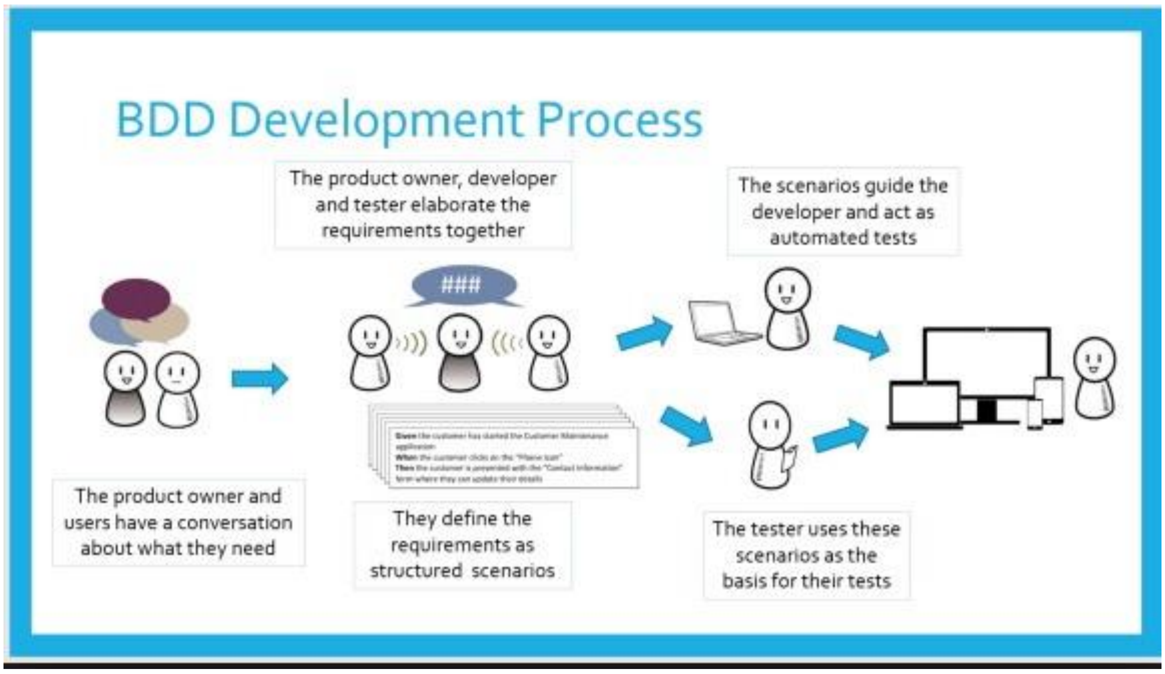
- *Tests are written in plain descriptive English type grammar*
- *Tests are explained as behavior of application and are more user focused*
- *Using examples to clarify requirements*

FEATURES OF BDD:-

1. *Shifting from thinking in “tests” to thinking in “behavior”*
2. *Collaboration between Business stakeholders, Business Analysts, QA Team and developers*
3. *Ubiquitous language, it is easy to describe*
4. *Driven by Business Value*
5. *Extends Test Driven Development (TDD) by utilizing natural language that non technical stakeholders can understand*
6. *BDD frameworks such as Cucumber act a “bridge” between Business & Technical Language*

Why BDD?

- Most software projects involve teams of several people working collaboratively together, so high-quality communication is critical to their success.
- There is a need to solicit feedback to ensure you've been understood correctly.
- That is why,agile software teams have learned to work in small increments, using the software that's built incrementally as the feedback that says to the stakeholders "Is this what you mean?"


BDD Development Process
The product owner, developer and tester elaborate the requirements together
The scenarios guide the developer and act as automated tests
###
The product owner and users have a conversation about what they need
They define the requirements as structured scenarios
The tester uses these scenarios as the basis for their tests

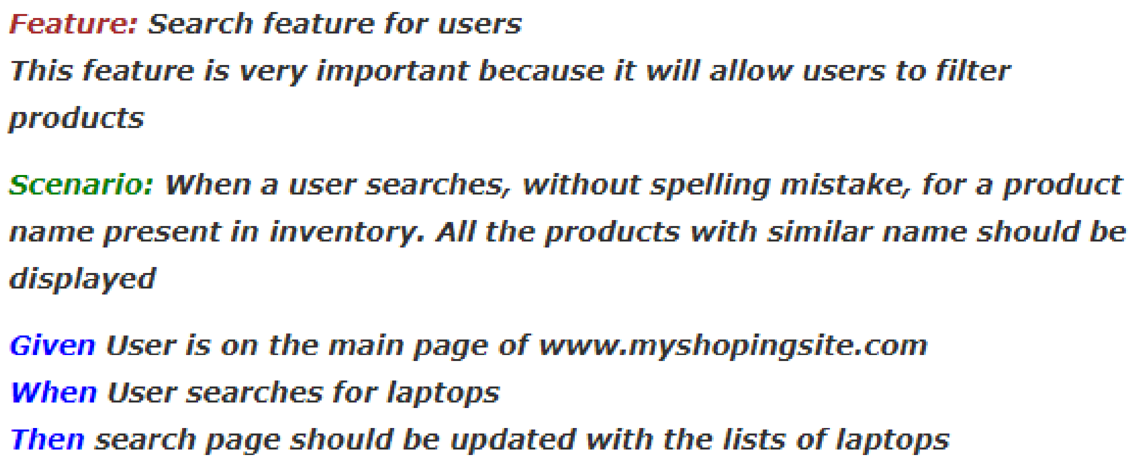
Cucumber as BDD

- ***Cucumber*** is a testing framework which supports ***Behavior Driven Development (BDD).***
- It lets us define application behavior in plain meaningful English text using a simple grammar defined by a language called ***Gherkin***.
- The main feature of the Cucumber is that it focuses on Acceptance testing.
- It makes it easy for anyone in the team to read and write test and with this feature it brings business users in to the test process, helping teams to explore and understand requirements.

Gherkin Language

- There is a need to understand the importance of a common language across different domains of project i.e Clients, Developers, Testers, Business Analysts and the managerial team
- This language should be simple enough to be understood by Business team members and should be explicit enough to remove most of the ambiguities for developers and testers.
- *Gherkin* is a simple, lightweight and structured language which uses regular spoken language to describe requirements and scenarios.

Example of Gherkin:-



```
Feature: Search feature for users
This feature is very important because it will allow users to filter
products

Scenario: When a user searches, without spelling mistake, for a product
name present in inventory. All the products with similar name should be
displayed

Given User is on the main page of www.myshopingsite.com
When User searches for laptops
Then search page should be updated with the lists of laptops
```

- Gherkin contains a set of keywords which define different premise of the scenario.
- The test is written in plain English which is common to all the domains of your project team.
- This test is structured that makes it capable of being read in an automated way. There by creating automation tests at the same time while describing the scenario.

Cucumber Framework components:-

1)Feature File:-

- Feature files are the essential part of cucumber which is used to write test automation steps or acceptance tests.
- The steps are the application specification.
- All the feature files end with .feature extension.
- Features file contain high level description of the Test Scenario in simple language known as Gherkin
- Feature File consists of following components:-

  1)Feature: A feature would describe the current test script which has to be executed.

  2)Scenario: Scenario describes the steps and expected outcome for a particular test case.

  3)Scenario Outline: Same scenario can be executed for multiple sets of data using scenario outline. The data is provided by a tabular structure separated by (I I).

  4)Given: It specifies the context of the text to be executed. By using datatables "Given", step can also be parameterized.

  5)When: "When" specifies the test action that has to performed

  6)Then: The expected outcome of the test can be represented by ‘Then’

Sample Feature File:-

(Example of Scenario)

Feature: Login Functionality Feature
In order to ensure Login Functionality works,
I want to run the cucumber test to verify it is working

Scenario: Login Functionality
Given user navigates to Gmail.com
When user logs in using Username as “USER” and Password “PASSWORD”
Then login should be successful

(Example of Scenario Outline)

Feature: Login Functionality Feature
In order to ensure Login Functionality works,
I want to run the cucumber test to verify it is working

Scenario Outline: Login Functionality
Given user navigates to Gmail.com
When user logs in using Username as <**username**> and Password <**password**>
Then login should be successful
Examples:

```
|username        |password         |
|Tom                |password1     |
|Harry              |password2     |
|Jerry              |password3     |
```

Use of Tags:-

- Cucumber by default runs all scenarios in all the feature files.
- In real time projects, there could be hundreds of feature file which are not required to run at all times.
- Cucumber will run only those feature files specific to given tags.
- Multiple tags can be specified in one feature file.

Example of Tags:-

@SmokeTest

Feature: Login Functionality Feature
In order to ensure Login Functionality works,
I want to run the cucumber test to verify it is working

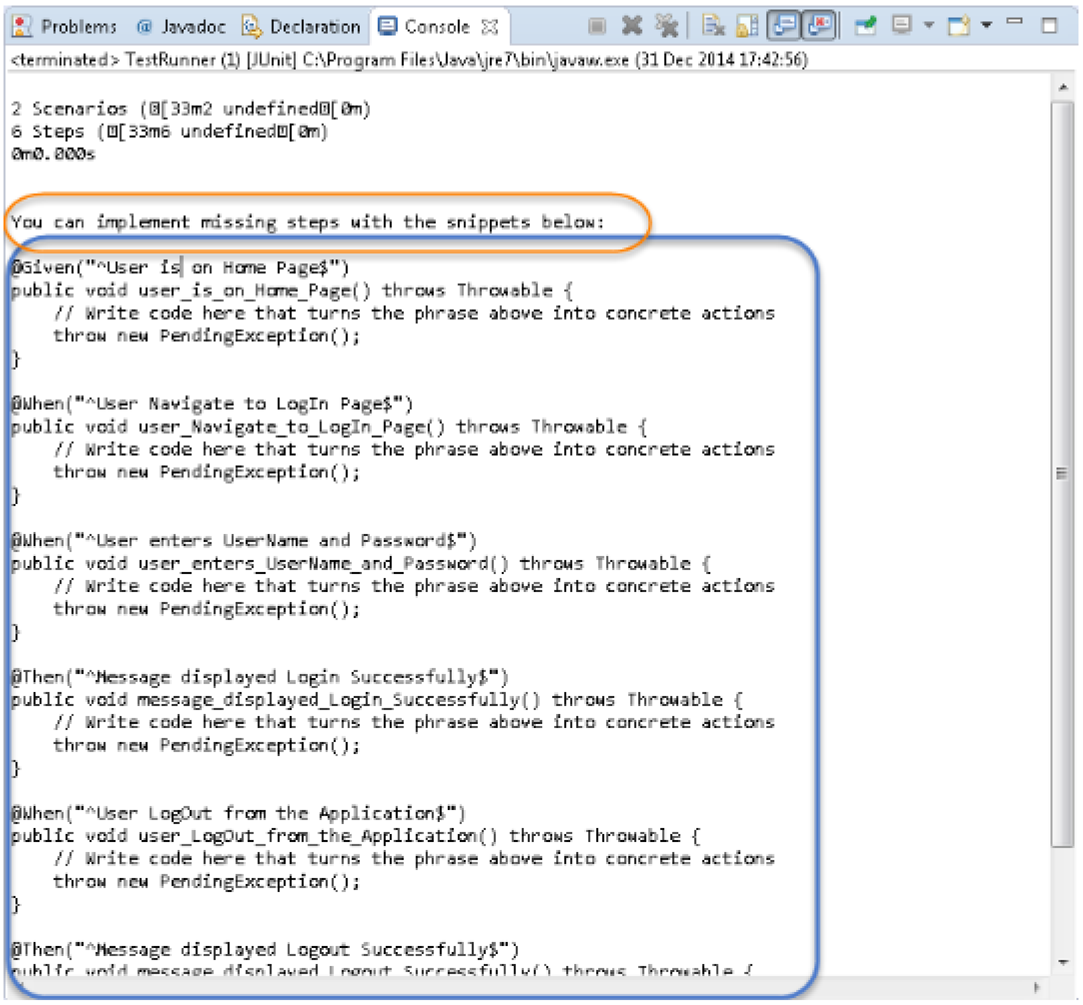
Scenario Outline: Login Functionality
Given user navigates to gmail.com
When user logs in using Username as <**username**> and Password <**password**>
Then login should be successful
Examples:

```
|username        |password|
|Tom     |password1        |
|Harry   |password2        |
|Jerry    |password3        |
```

Step Definition:-

- Step definition maps the Test Case Steps in the feature files(introduced by Given/When/Then) to code.
- It which executes the steps on Application Under Test and checks the outcomes against expected results.
- For a step definition to be executed, it must match the given component in a feature.
- Steps Definition file contains the actual code to execute the Test Scenario in the Features file.
- *Cucumber* finds the *Step Definition* file with the help of Glue code in ***Cucumber Options***

Example of Step Definitions:-

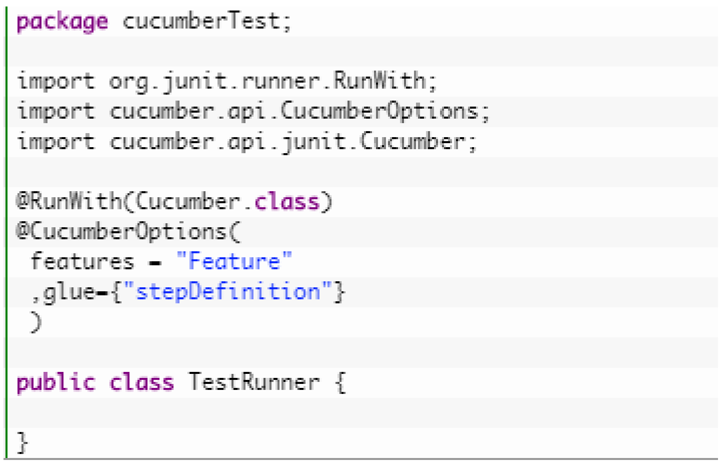


```
<terminated> TestRunner (1) [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (31 Dec 2014 17:42:56)

2 Scenarios (□[33m2 undefined□[0m)
6 Steps (□[33m6 undefined□[0m)
0m0.000s

You can implement missing steps with the snippets below:

@Given("^User is on Home Page$")
public void user_is_on_Home_Page() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@When("^User Navigate to LogIn Page$")
public void user_Navigate_to_LogIn_Page() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@When("^User enters UserName and Password$")
public void user_enters_UserName_and_Password() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Then("^Message displayed Login Successfully$")
public void message_displayed_Login_Successfully() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@When("^User LogOut from the Application$")
public void user_LogOut_from_the_Application() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Then("^Message displayed Logout Successfully$")
public void message_displayed_Logout_Successfully() throws Throwable {
```

Junit Runner Class:-

- *Cucumber* uses ***Junit framework*** to run.
- TestRunner Classuses the *Junit annotation* **@RunWith(),** which tells *JUnit* what is the *test runner class*.
- It more like a starting point for Junit to start executing your tests.

Junit Runner Class Code:-



```java
package cucumberTest;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
 features = "Feature"
 ,glue={"stepDefinition"}
 )

public class TestRunner {

}
```

2)BDD Cucumber Integration with Page Object Model

- The main objective of using Page Object model in the BDD cucumber framework if the UI or any HTML object changes for any page, the test does not need any fix.
- Only the code within the page objects will be impacted but it does not have any impact to the test.
- The project code becomes manageable and maintainable
- To better manage the code and to improve the re-usability, this pattern suggests us to divided an application in different pages or a single page in to sub pages.
- Page Object Pattern technique provides a solution for working with multiple web pages and prevents unwanted code duplication and enables an uncomplicated solution for code maintenance.

Selenium PageFactory

- Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver and it is very optimized.
- ***PageFactory*** is used to ***Initialize Elements*** of a Page class without having to use '***FindElement***' or '***FindElements***'.
- Annotations can be used to supply descriptive names of target objects to improve code readability

@FindBy Annotation

- It helps to ***find*** the elements in the page using ***By*** strategy.
- ***@FindBy*** can accept ***TagName, PartialLinkText, Name, LinkText, Id, Css, ClassName, XPath*** as attributes.
- It is an alternative mechanism for locating the element or a list of elements. This allows users to quickly and easily create PageObjects.

InitElements

- This ***Instantiates an Instance*** of the given class.
- This method will attempt to instantiate the class given to it, preferably using a constructor which takes a WebDriver instance as its only argument or falling back on a no-arg constructor.
- An exception will be thrown if the class cannot be instantiated.

PageFactory.initElements(WebDriver, PageObject.Class);
Parameters:

- ***WebDriver*** *– The driver that will be used to look up the elements*
- ***PageObjects*** *– A class which will be initialised*

Returns: *An instantiated instance of the class with WebElement and List<WebElement> fields proxied*

For Detailed documentation, please visit

https://www.toolsqa.com/selenium-cucumber-framework/page-object-design-pattern-with-selenium-pagefactory-in-cucumber/

3)JUnit

- **JUnit** is an open source unit testing framework for the Java programming language.
- JUnit allows the developer to incrementally build test suites to measure progress and detect unintended side effects.
- Test can be run continuously.
- JUnit shows test progress in a bar that is normally green but turns red when a test fails.
- Cucumber provides a way for non-technical person to define test cases for a product, and on the other hand, our expectation is for smooth and timely execution of such test cases.
- JUnit acts as a bridge between these two.
- The flow is as follows:-
- Stakeholders write down the feature file.
- Step definition file will be created accordingly.
- Specify the JUnit runner class to run the series of test cases.
- Once we run the JUnit runner class −
    - It will parse the Gherkin feature file.
    - It will execute the functions written in the step definition file according to feature file statements.
    - JUnit will combine the test case result.
    - It will build the test report in the specified format (which can be html/JSON).

4)Maven

- **Maven** is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation.
- Maven allows the developer to automate the process of the creation of the initial folder structure, performing the compilation and testing and the packaging and deployment of the final product.
- It handles compilation, distribution, documentation, team collaboration and other tasks seamlessly.
- Maven increases reusability and takes care of most of build related tasks. It helps in bypassing my steps like adding jars to the project library, building reports, executing Junits test cases, creating Jar, War Ear files for the project deployment and many more
- A very significant aspect of Maven is the use of repositories to manage jar files

- Maven Components:-

1)Maven Local Repository- Maven stores all the project jars files or libraries or dependencies. By deault the folder name is ***'.m2'***

2)Maven Central Repository:- For any library required in the project, Maven first look in to the .m2 folderof Local Repository, if it does not find the required libarary then it looks in Central Repository and download the libabry in to local repository.

3)Dependency:-Dependencies are the libararies, which are required by the project. For example Log4j jars, Apache Poi jars, Selenium Jars etc.

Example of Dependency:-

```
<dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>2.43.1</version>
</dependency>
```

4)<u>Surefire Plugin:- </u>The Surefire Plugin is used during the test phase of the build lifecycle to execute the unit tests of an application.

It generates reports in 2 different file formats like plain text file, xml files and html files as well.

5)Maven POM:- POM is Project Object Model XML file that contains information about the project and configuration details used by Maven to build the project.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed

FRAMEWORK

Why BDD Framework for R! Automation ?

The benefits which are observed using BDD Framework are as follows:-

- **Strong collaboration.** With BDD, all the involved parties have a strong understanding of the project and they can all have a role in the communication and actually have constructive discussions. BDD increases and improves collaboration. It enables everyone involved in the project to easily engage in the product development cycle. And by using plain language, all are able to write behavior scenarios.
- **High visibility.** By using a language understood by all, everyone gets a strong visibility into the project's progression.
- **The software design follows business value**. In fact, BDD puts a great importance to the Business value & needs. By setting priorities with the client, based on the value it provides, developers are able to provide better result because they have a strong understanding of how the client thinks. By focusing on the value, no useless features are built.
- **The ubiquitous language**. As mentioned earlier, the ubiquitous language is understandable by all the members of the team, which reduces misconceptions and misunderstanding and makes it easier for new members to join the working process.
- **Software development meets user need.** By focusing on the business' needs, you get satisfied users, and that implies a happy business of course. Actually with BDD, as its name says it, you focus on the behavior, which has a stronger impact than the implementation itself.

BDD Framework used in the Roostify Automation has proved Advantageous in many ways:-

<u>1)Re-Usable Features and Step Definitions:-</u>

- In R!, basically there are three ways through which a loan application can be filled and submitted namely:-
  a)Copy Referral
  b)Invite Link
  c)Add Loan Manually

- The only difference in these scenarios is the way we navigate to the Loan Application. So the same features and Step Definitions are re-usable in terms of coding

<u>2)Time and Coding Efforts</u>

- Since a lot of Code re-usability is involved , there is a less utilization of efforts in terms of Time and Coding

<u>3)Code Maintability</u>

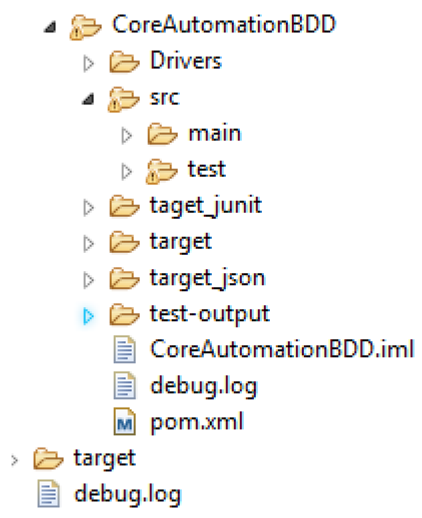
- If new Features get added, we just need to update the feature files and update the Object Repositories which are created separately for every section

<u>4)Regression</u>

- Though 100% Automation is not possible, considerable flows and test cases are included from the Regression point of view
- This Automation suite can be used for testing in every regression cycle and hence reducing human efforts
- End-to-End flows can be run covering all the field level testing ensuring the main functionalities are not getting broken

FRAMEWORK STRUCTURE

This is the screenshot of the Framework Structure:-

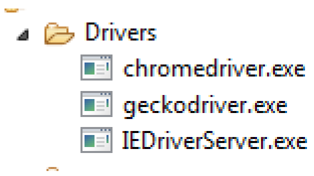


```
CoreAutomationBDD
    Drivers
    src
        main
        test
    taget_junit
    target
    target_json
    test-output
    CoreAutomationBDD.iml
    debug.log
    pom.xml
target
debug.log
```

Project Name:- CoreAutomationBDD

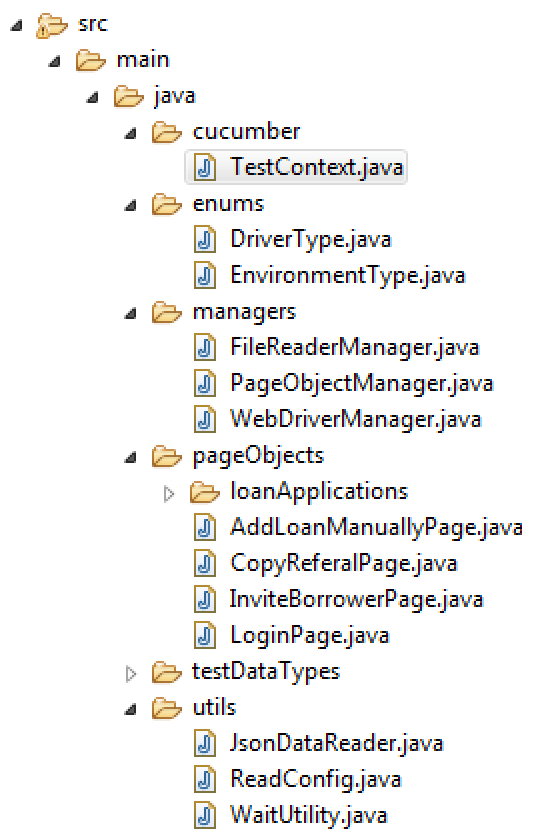
The project consists of folders:-

- <u>Drivers:- </u>This folder consists of the drivers which can be used in order to run the Automation Project.
- Keeping in consideration that we might need different browsers for Browser compatibility testing, these 3 drivers are included in the project structure so that generic project path for drivers can be given



```
Drivers
    chromedriver.exe
    geckodriver.exe
    IEDriverServer.exe
```

- Src- This Folder consists of two main folders in hierarchy namely:-
- Src->Main->Java:-
  In this folder structure, following packages are included:-



```
src
  main
    java
      cucumber
        TestContext.java
      enums
        DriverType.java
        EnvironmentType.java
      managers
        FileReaderManager.java
        PageObjectManager.java
        WebDriverManager.java
      pageObjects
        loanApplications
        AddLoanManuallyPage.java
        CopyReferalPage.java
        InviteBorrowerPage.java
        LoginPage.java
      testDataTypes
      utils
        JsonDataReader.java
        ReadConfig.java
        WaitUtility.java
```

Describing every Java class included in the above folders,

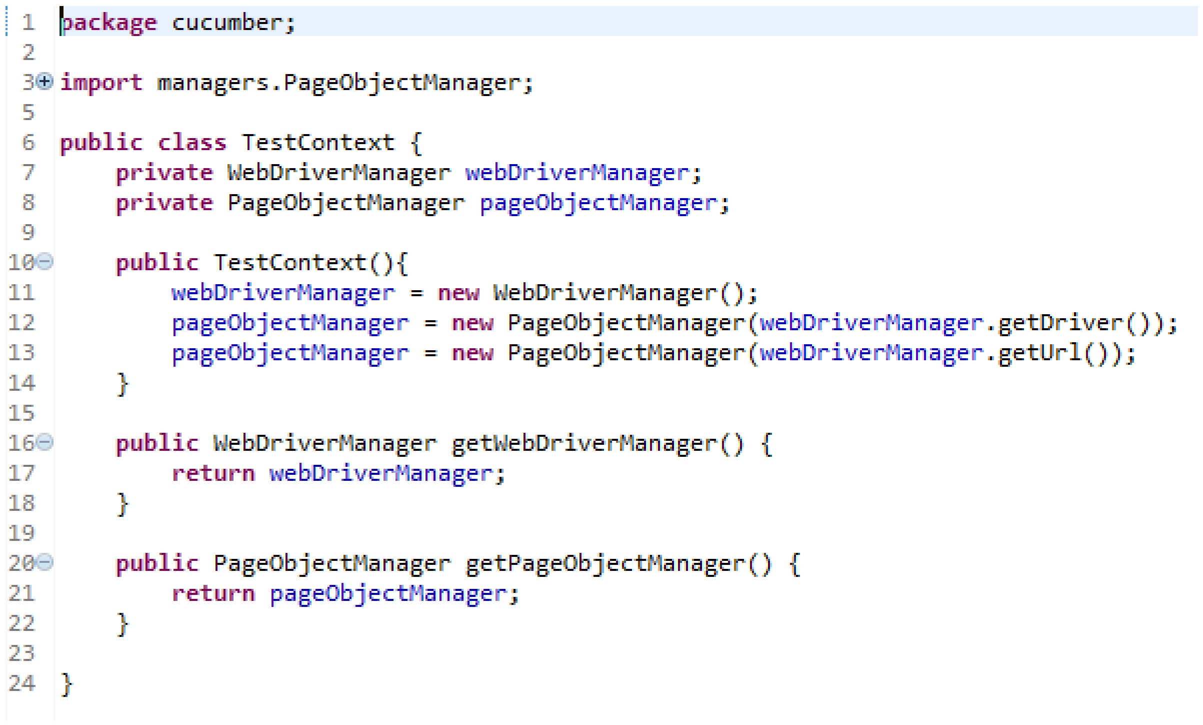
Cucumber:- This folder consist of TestContext.java

- TestContext.java:-
- ***TestContext*** is the parent class and the medium to share the information between the different steps in a test. It can have many class objects in it.

- This class is used for initialization of objects which the Steps file will use namely:-

- **PageObjects** : Provided by **PageObjectManager**
- **WebDriver** : Provided by **WebDriverManager**

The TestContext.java will look like:-

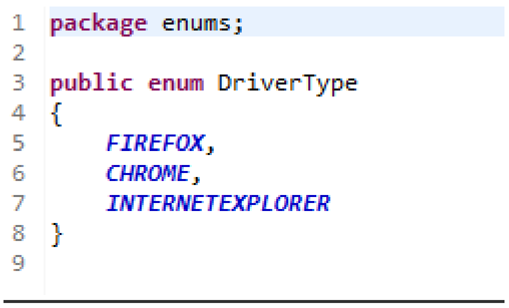


```java
package cucumber;

import managers.PageObjectManager;

public class TestContext {
    private WebDriverManager webDriverManager;
    private PageObjectManager pageObjectManager;

    public TestContext(){
        webDriverManager = new WebDriverManager();
        pageObjectManager = new PageObjectManager(webDriverManager.getDriver());
        pageObjectManager = new PageObjectManager(webDriverManager.getUrl());
    }

    public WebDriverManager getWebDriverManager() {
        return webDriverManager;
    }

    public PageObjectManager getPageObjectManager() {
        return pageObjectManager;
    }

}
```

Enums:-

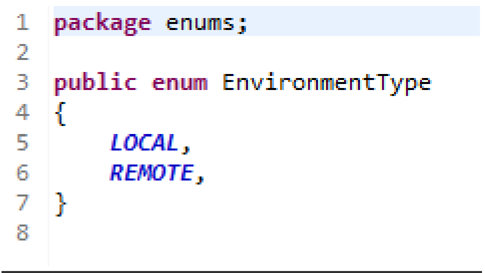
Enums are used to set pre-defined values

- DriverType.java:- In this class, enums are used to specify the Driver types.



```java
package enums;

public enum DriverType
{
    FIREFOX,
    CHROME,
    INTERNETEXPLORER
}

```

- EnvironmentType.java:- In this class, enums are used to specify the Environment types

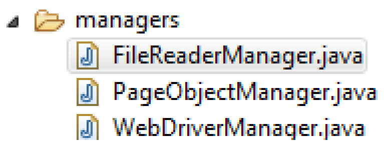


```
package enums;

public enum EnvironmentType
{
    LOCAL,
    REMOTE,
}
```

-

Managers:-

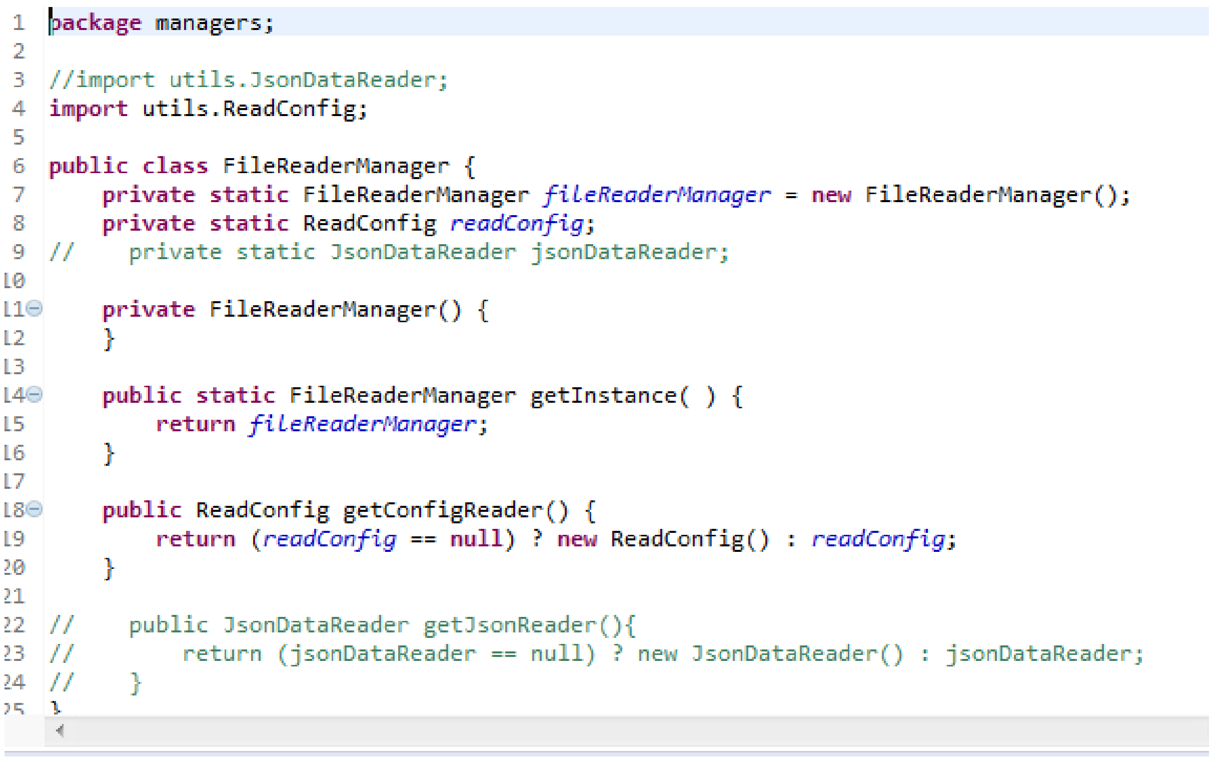
This package consists of the following classes:-



```
managers
    FileReaderManager.java
    PageObjectManager.java
    WebDriverManager.java
```

FileReaderManager.java

- The FileReaderManager class maintains a static reference to its own instance and returns that reference from the **static getInstance()** method.
- FileReaderManager implements a **private constructor** so clients cannot instantiate FileReaderManager instances.
- **getConfigReader()** *method returns the instance of the ConfigFileReader but this method is not static. So that client has to use the getInstance() method to access other public methods of the FileReaderManager*



```java
package managers;

//import utils.JsonDataReader;
import utils.ReadConfig;

public class FileReaderManager {
    private static FileReaderManager fileReaderManager = new FileReaderManager();
    private static ReadConfig readConfig;
//    private static JsonDataReader jsonDataReader;

    private FileReaderManager() {
    }

    public static FileReaderManager getInstance( ) {
        return fileReaderManager;
    }

    public ReadConfig getConfigReader() {
        return (readConfig == null) ? new ReadConfig() : readConfig;
    }

//    public JsonDataReader getJsonReader(){
//        return (jsonDataReader == null) ? new JsonDataReader() : jsonDataReader;
//    }
}
```

PageObjectManager.java

Constructor

```java
public PageObjectManager(WebDriver driver) {
    this.driver = driver;
}
```

This constructor is asking for parameter of type WebDriver. As to create an object of the Pages, this class requires a driver. Now who so ever will create the object of this class needs to provide the driver like :

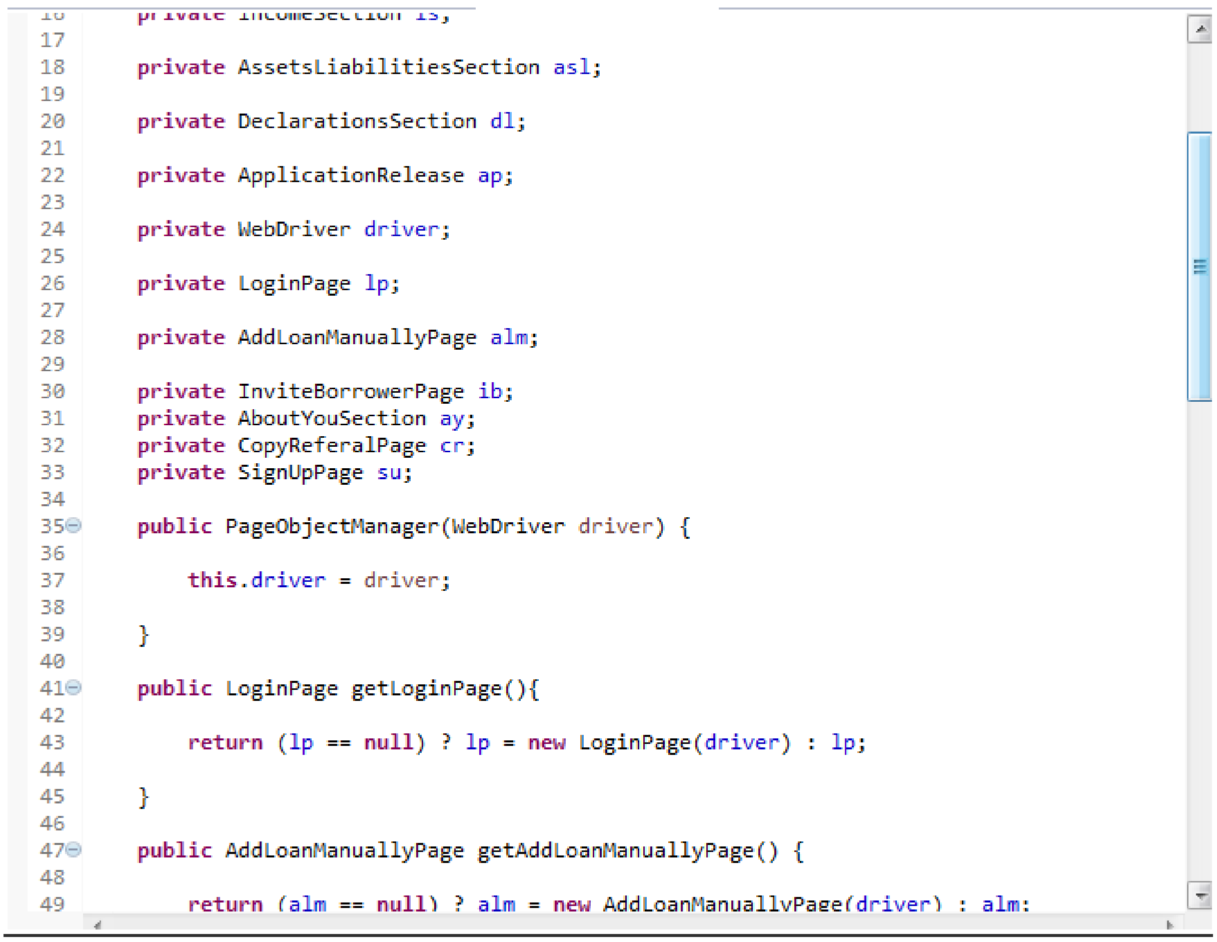
```java
PageObjectManager pageObjectManager = new
PageObjectManager(driver);
```

Page Object Creation Method

```java
public HomePage getHomePage() {
    return (homePage == null) ? new HomePage(driver) : homePage;
}
```

This method has two responsibilities:

1. To create an Object of Page Class only if the object is null.
2. To supply the already created object if it is not null



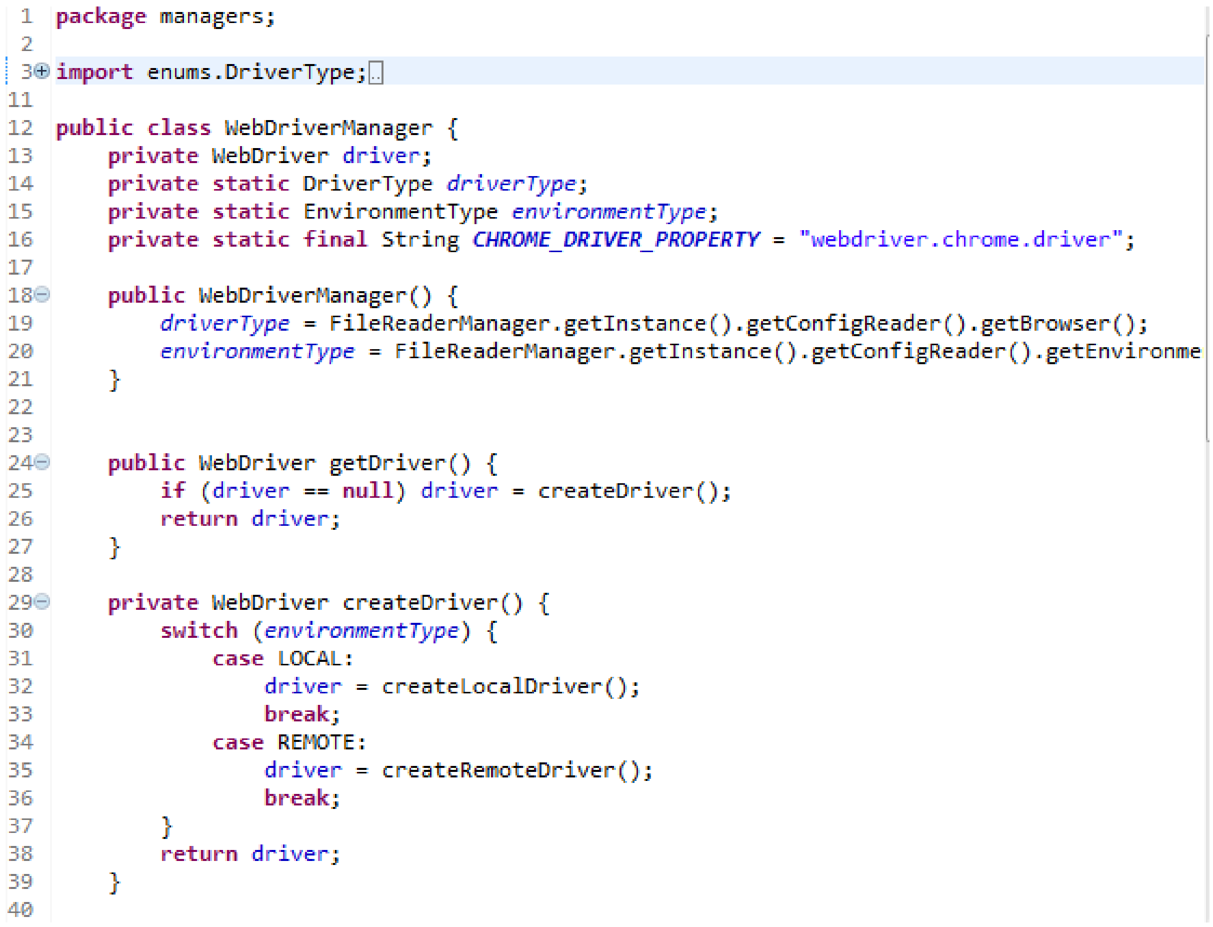
```
    private IncomeSection is;

    private AssetsLiabilitiesSection asl;

    private DeclarationsSection dl;

    private ApplicationRelease ap;

    private WebDriver driver;

    private LoginPage lp;

    private AddLoanManuallyPage alm;

    private InviteBorrowerPage ib;
    private AboutYouSection ay;
    private CopyReferalPage cr;
    private SignUpPage su;

    public PageObjectManager(WebDriver driver) {

        this.driver = driver;

    }

    public LoginPage getLoginPage(){

        return (lp == null) ? lp = new LoginPage(driver) : lp;

    }

    public AddLoanManuallyPage getAddLoanManuallyPage() {

        return (alm == null) ? alm = new AddLoanManuallyPage(driver) : alm;
```

WebDriverManager.java

- In this class,two methods for now which are ***getDriver()*** and ***closeDriver()***.
- *GetDriver* method will decide if the driver is already created or needs to be created.

GetDriver method further call the method ***createDriver()***, which will decide that the remote driver is needed or local driver for the execution.

Accordingly *CreateDriver* method would make a call let’s say to ***createLocalDriver(). CreateLocalDriver*** method will further decide which type of driver needs to be created.



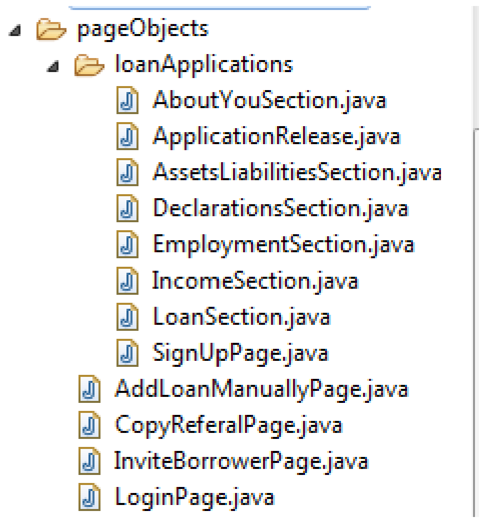
```java
package managers;

import enums.DriverType;

public class WebDriverManager {
    private WebDriver driver;
    private static DriverType driverType;
    private static EnvironmentType environmentType;
    private static final String CHROME_DRIVER_PROPERTY = "webdriver.chrome.driver";

    public WebDriverManager() {
        driverType = FileReaderManager.getInstance().getConfigReader().getBrowser();
        environmentType = FileReaderManager.getInstance().getConfigReader().getEnvironme
    }


    public WebDriver getDriver() {
        if (driver == null) driver = createDriver();
        return driver;
    }

    private WebDriver createDriver() {
        switch (environmentType) {
            case LOCAL:
                driver = createLocalDriver();
                break;
            case REMOTE:
                driver = createRemoteDriver();
                break;
        }
        return driver;
    }

```

PageObjects

This package consists of all the Java classes which contain the locators and the member functions which are specific to the particular section

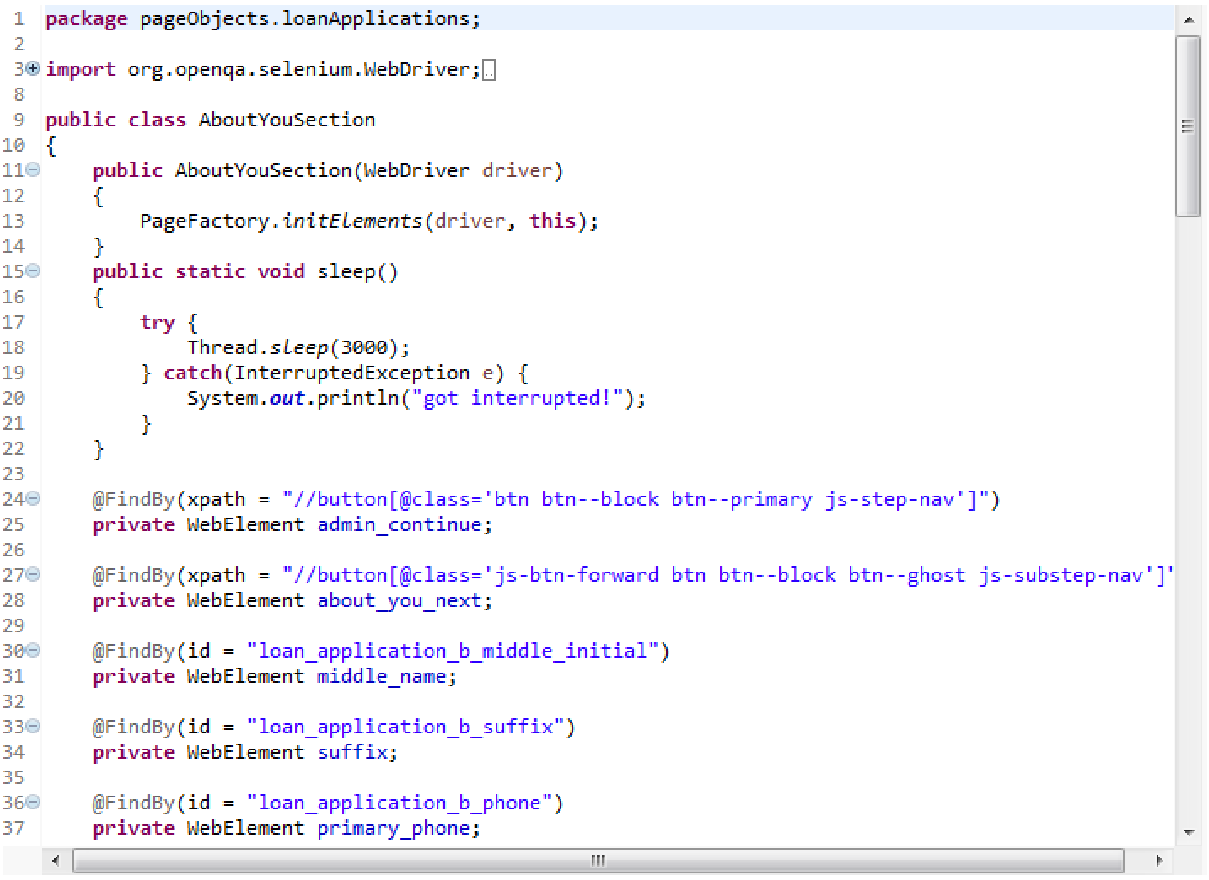
This basically acts as Object Repositories.

Maintened Object Repositories in a separate package so that the locators can be changed if there occurs any change in the structure of the Document Object Model



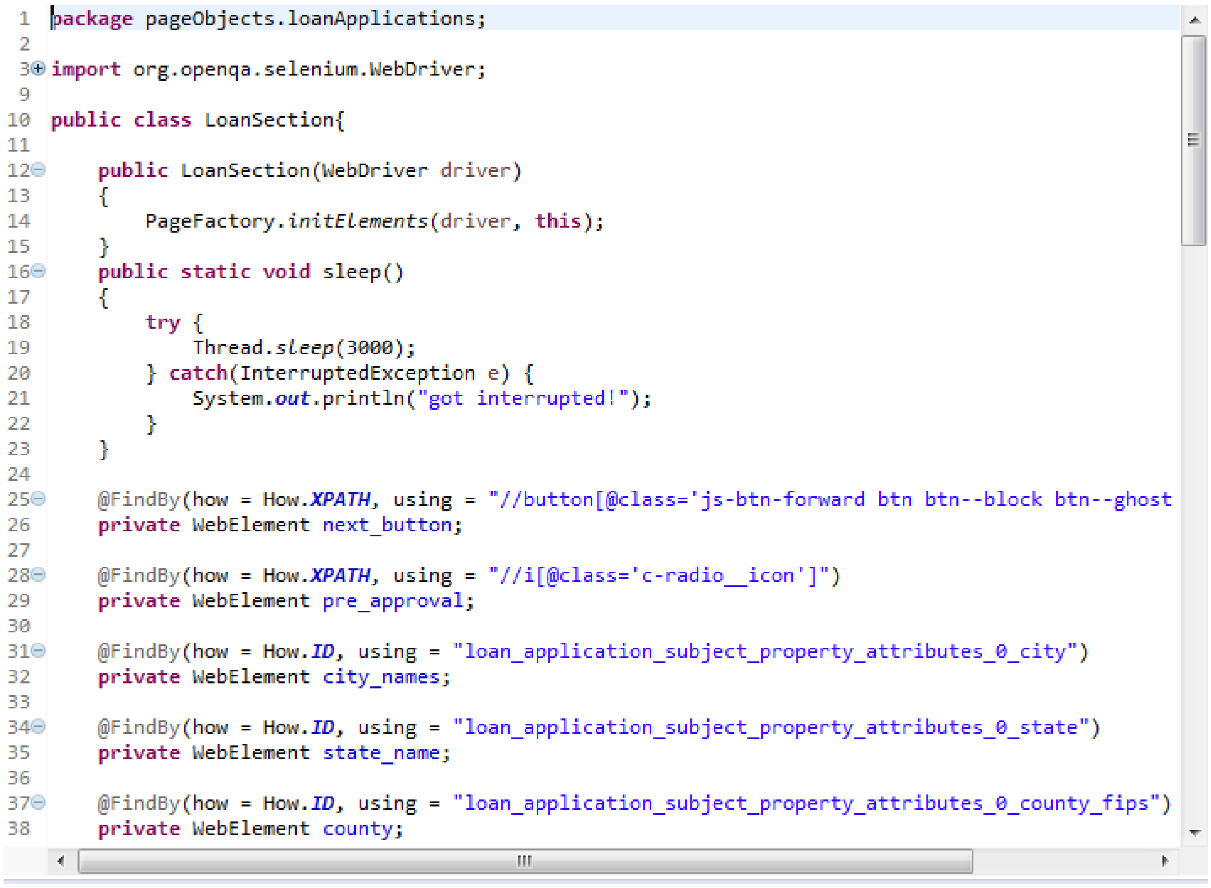
```
pageObjects
  loanApplications
    AboutYouSection.java
    ApplicationRelease.java
    AssetsLiabilitiesSection.java
    DeclarationsSection.java
    EmploymentSection.java
    IncomeSection.java
    LoanSection.java
    SignUpPage.java
  AddLoanManuallyPage.java
  CopyReferalPage.java
  InviteBorrowerPage.java
  LoginPage.java
```

- AboutYouSection.java



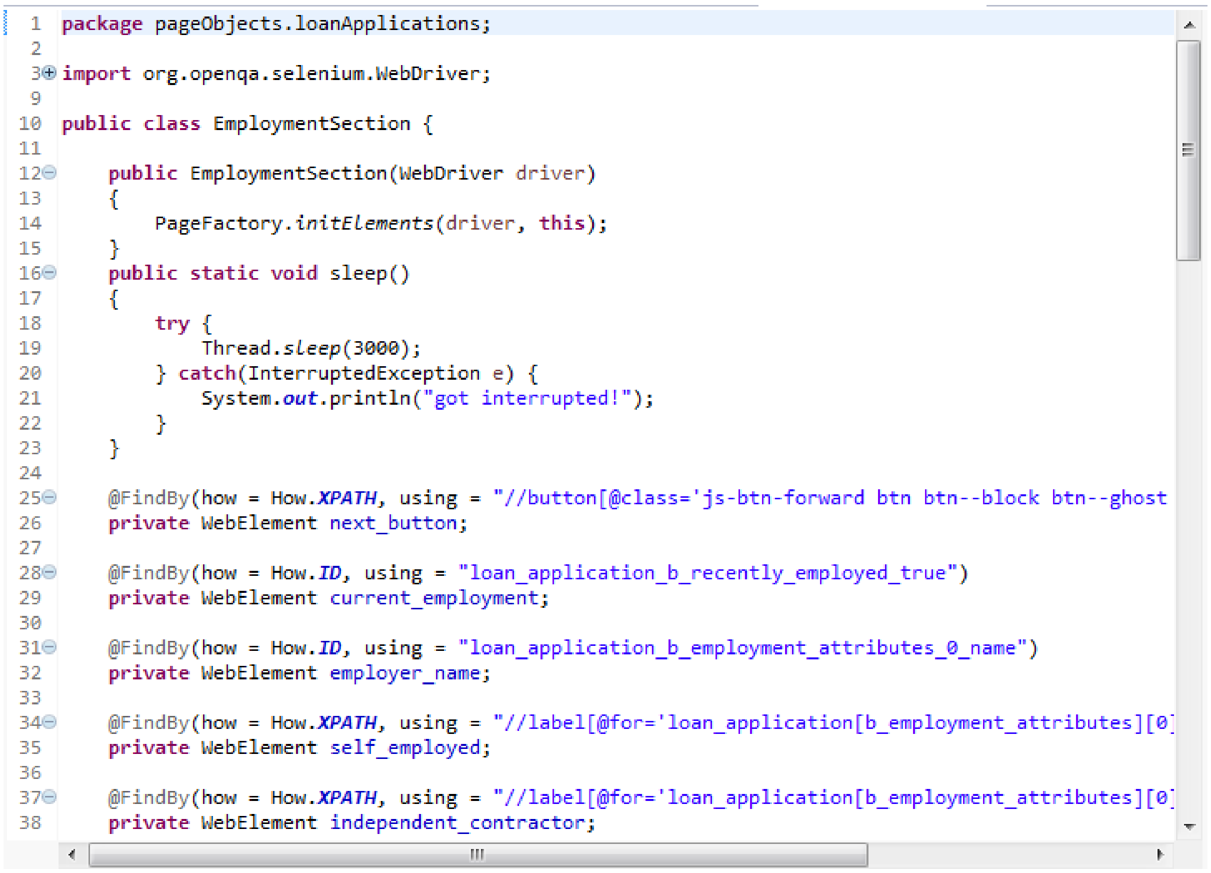
```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class AboutYouSection
{
    public AboutYouSection(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }

    @FindBy(xpath = "//button[@class='btn btn--block btn--primary js-step-nav']")
    private WebElement admin_continue;

    @FindBy(xpath = "//button[@class='js-btn-forward btn btn--block btn--ghost js-substep-nav']'
    private WebElement about_you_next;

    @FindBy(id = "loan_application_b_middle_initial")
    private WebElement middle_name;

    @FindBy(id = "loan_application_b_suffix")
    private WebElement suffix;

    @FindBy(id = "loan_application_b_phone")
    private WebElement primary_phone;
```

- LoanSection.java



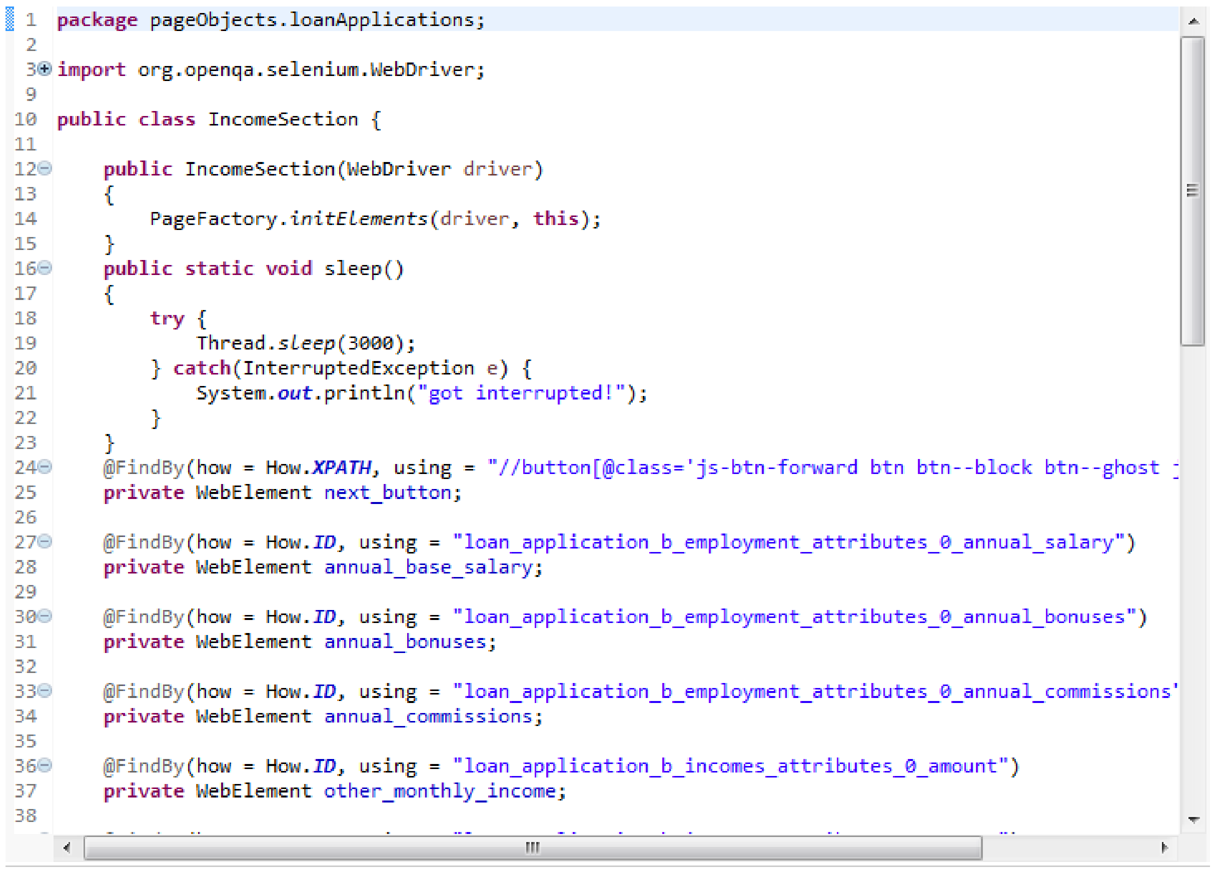
```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class LoanSection{

    public LoanSection(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }

    @FindBy(how = How.XPATH, using = "//button[@class='js-btn-forward btn btn--block btn--ghost
    private WebElement next_button;

    @FindBy(how = How.XPATH, using = "//i[@class='c-radio__icon']")
    private WebElement pre_approval;

    @FindBy(how = How.ID, using = "loan_application_subject_property_attributes_0_city")
    private WebElement city_names;

    @FindBy(how = How.ID, using = "loan_application_subject_property_attributes_0_state")
    private WebElement state_name;

    @FindBy(how = How.ID, using = "loan_application_subject_property_attributes_0_county_fips")
    private WebElement county;
```

- EmploymentSection.java



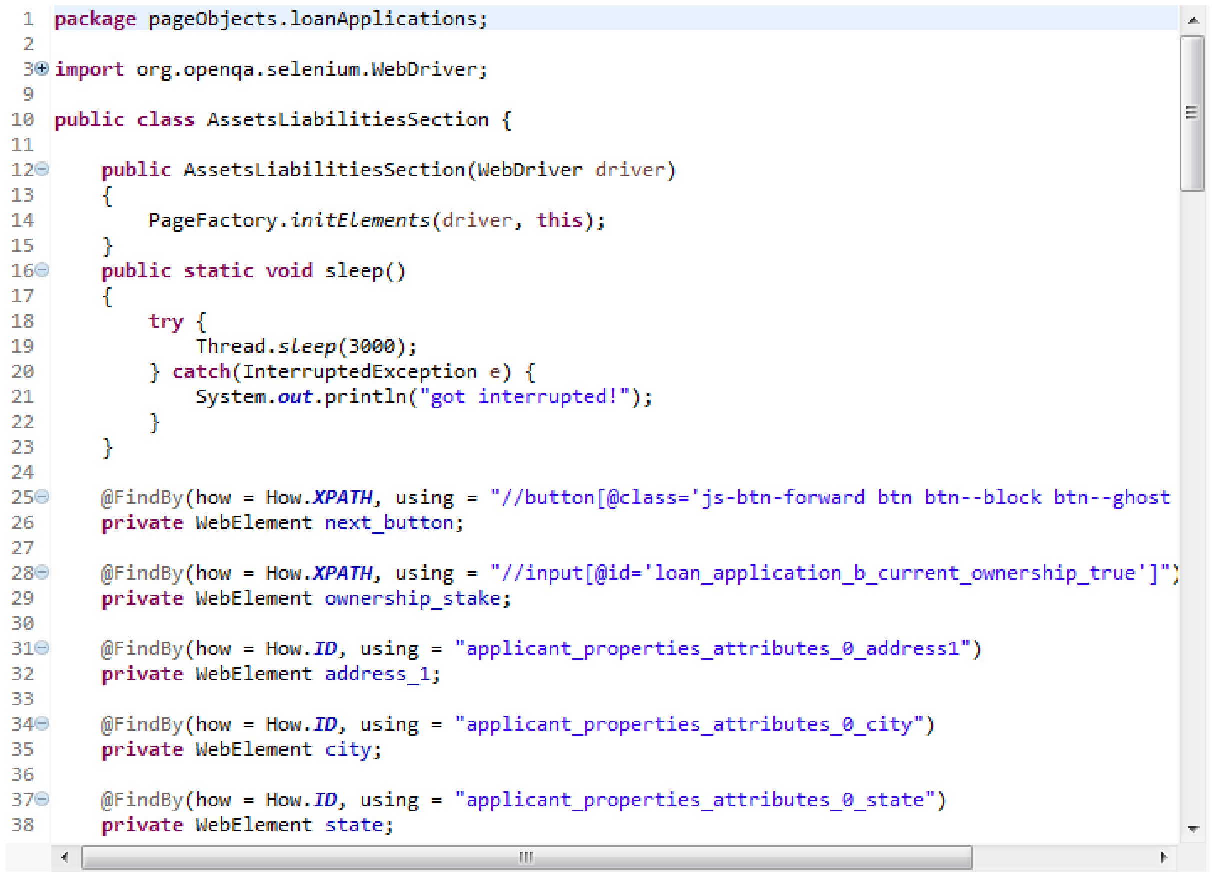
```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class EmploymentSection {

    public EmploymentSection(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }

    @FindBy(how = How.XPATH, using = "//button[@class='js-btn-forward btn btn--block btn--ghost
    private WebElement next_button;

    @FindBy(how = How.ID, using = "loan_application_b_recently_employed_true")
    private WebElement current_employment;

    @FindBy(how = How.ID, using = "loan_application_b_employment_attributes_0_name")
    private WebElement employer_name;

    @FindBy(how = How.XPATH, using = "//label[@for='loan_application[b_employment_attributes][0]
    private WebElement self_employed;

    @FindBy(how = How.XPATH, using = "//label[@for='loan_application[b_employment_attributes][0]
    private WebElement independent_contractor;
```

- IncomeSection.java



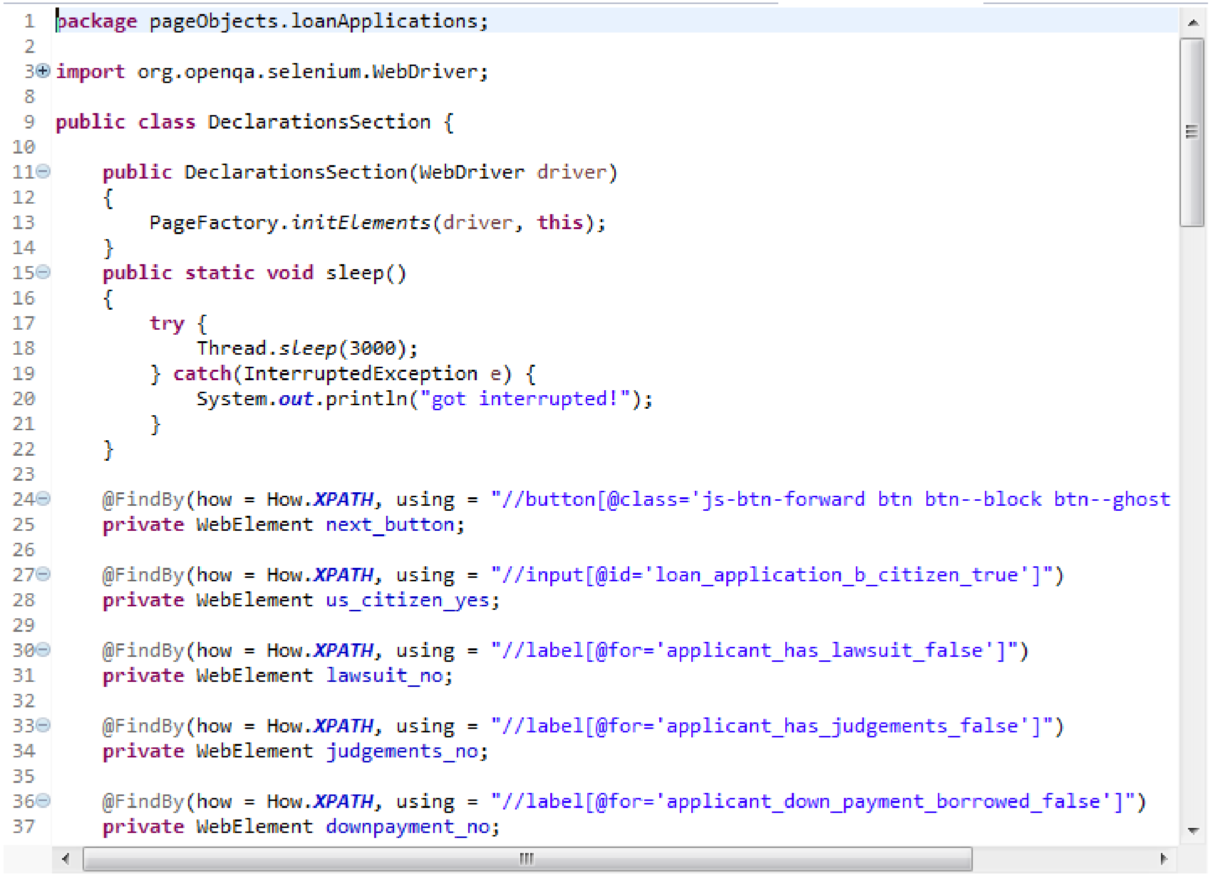
```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class IncomeSection {

    public IncomeSection(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }
    @FindBy(how = How.XPATH, using = "//button[@class='js-btn-forward btn btn--block btn--ghost
    private WebElement next_button;

    @FindBy(how = How.ID, using = "loan_application_b_employment_attributes_0_annual_salary")
    private WebElement annual_base_salary;

    @FindBy(how = How.ID, using = "loan_application_b_employment_attributes_0_annual_bonuses")
    private WebElement annual_bonuses;

    @FindBy(how = How.ID, using = "loan_application_b_employment_attributes_0_annual_commissions"
    private WebElement annual_commissions;

    @FindBy(how = How.ID, using = "loan_application_b_incomes_attributes_0_amount")
    private WebElement other_monthly_income;

```

- AssetsLiabilitiesSection.java



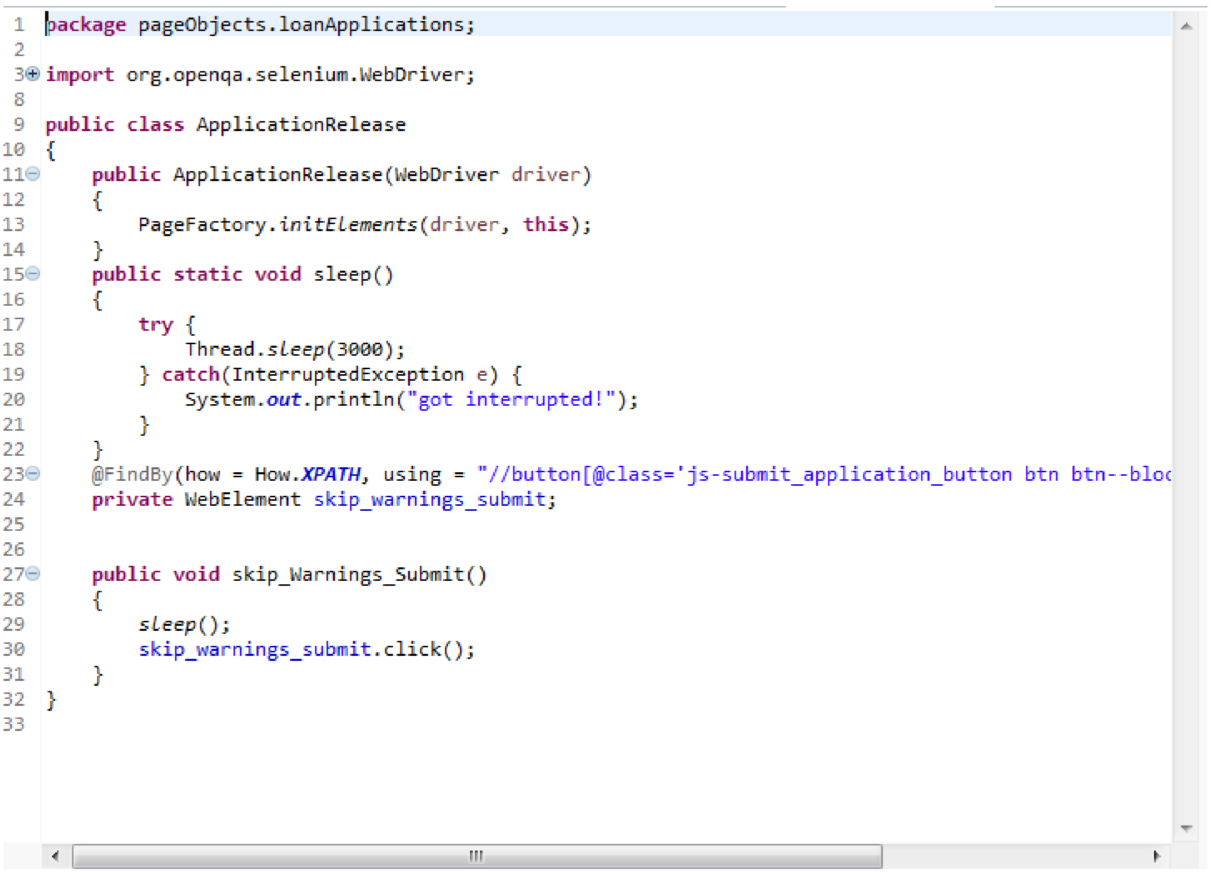
```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class AssetsLiabilitiesSection {

    public AssetsLiabilitiesSection(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }

    @FindBy(how = How.XPATH, using = "//button[@class='js-btn-forward btn btn--block btn--ghost
    private WebElement next_button;

    @FindBy(how = How.XPATH, using = "//input[@id='loan_application_b_current_ownership_true']")
    private WebElement ownership_stake;

    @FindBy(how = How.ID, using = "applicant_properties_attributes_0_address1")
    private WebElement address_1;

    @FindBy(how = How.ID, using = "applicant_properties_attributes_0_city")
    private WebElement city;

    @FindBy(how = How.ID, using = "applicant_properties_attributes_0_state")
    private WebElement state;
```

- DeclarationSection.java



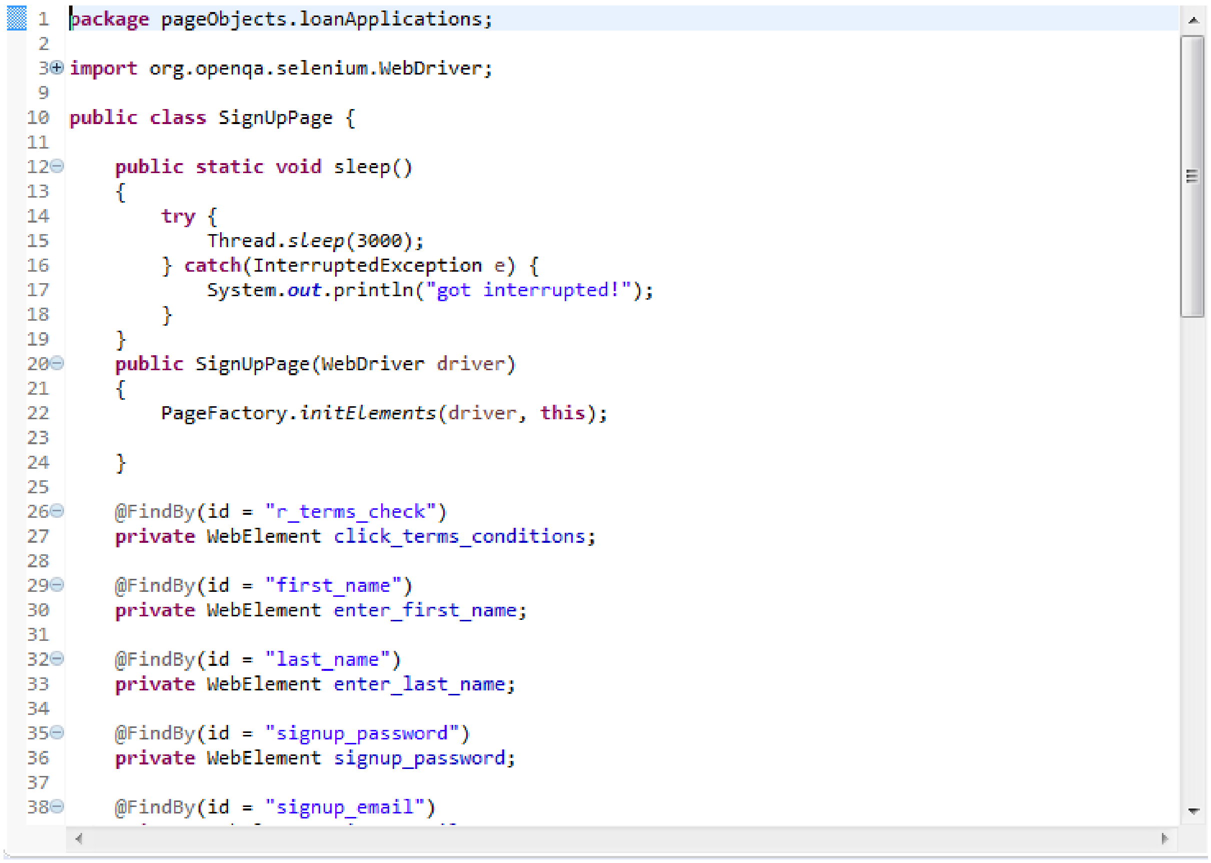
```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class DeclarationsSection {

    public DeclarationsSection(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }

    @FindBy(how = How.XPATH, using = "//button[@class='js-btn-forward btn btn--block btn--ghost
    private WebElement next_button;

    @FindBy(how = How.XPATH, using = "//input[@id='loan_application_b_citizen_true']")
    private WebElement us_citizen_yes;

    @FindBy(how = How.XPATH, using = "//label[@for='applicant_has_lawsuit_false']")
    private WebElement lawsuit_no;

    @FindBy(how = How.XPATH, using = "//label[@for='applicant_has_judgements_false']")
    private WebElement judgements_no;

    @FindBy(how = How.XPATH, using = "//label[@for='applicant_down_payment_borrowed_false']")
    private WebElement downpayment_no;
```

- ApplicationReleaseSection.java



```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class ApplicationRelease
{
    public ApplicationRelease(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }
    @FindBy(how = How.XPATH, using = "//button[@class='js-submit_application_button btn btn--bloc
    private WebElement skip_warnings_submit;


    public void skip_Warnings_Submit()
    {
        sleep();
        skip_warnings_submit.click();
    }
}
```

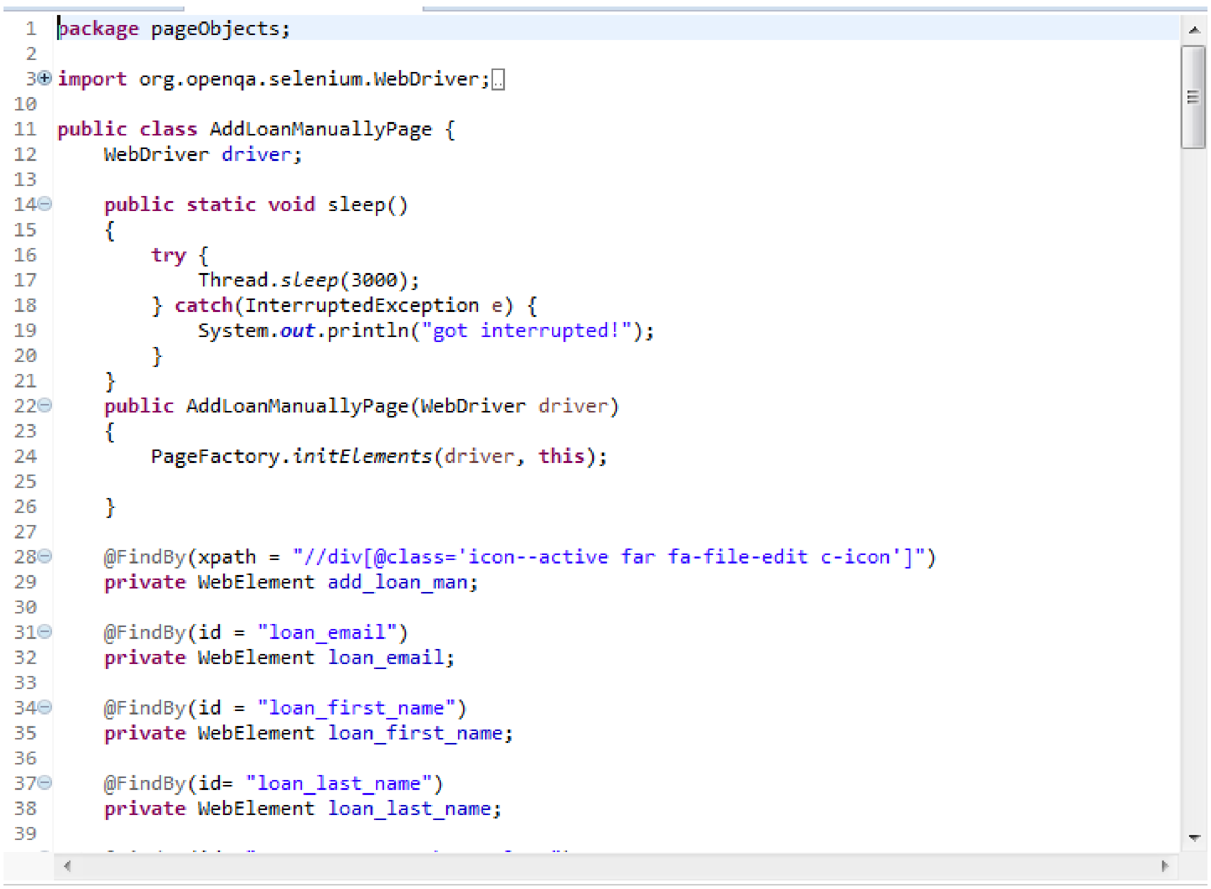
- SignUpPage.java



```java
package pageObjects.loanApplications;

import org.openqa.selenium.WebDriver;

public class SignUpPage {

    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }
    public SignUpPage(WebDriver driver)
    {
        PageFactory.initElements(driver, this);

    }

    @FindBy(id = "r_terms_check")
    private WebElement click_terms_conditions;

    @FindBy(id = "first_name")
    private WebElement enter_first_name;

    @FindBy(id = "last_name")
    private WebElement enter_last_name;

    @FindBy(id = "signup_password")
    private WebElement signup_password;

    @FindBy(id = "signup_email")
```

AddLoanManuallyPage.java

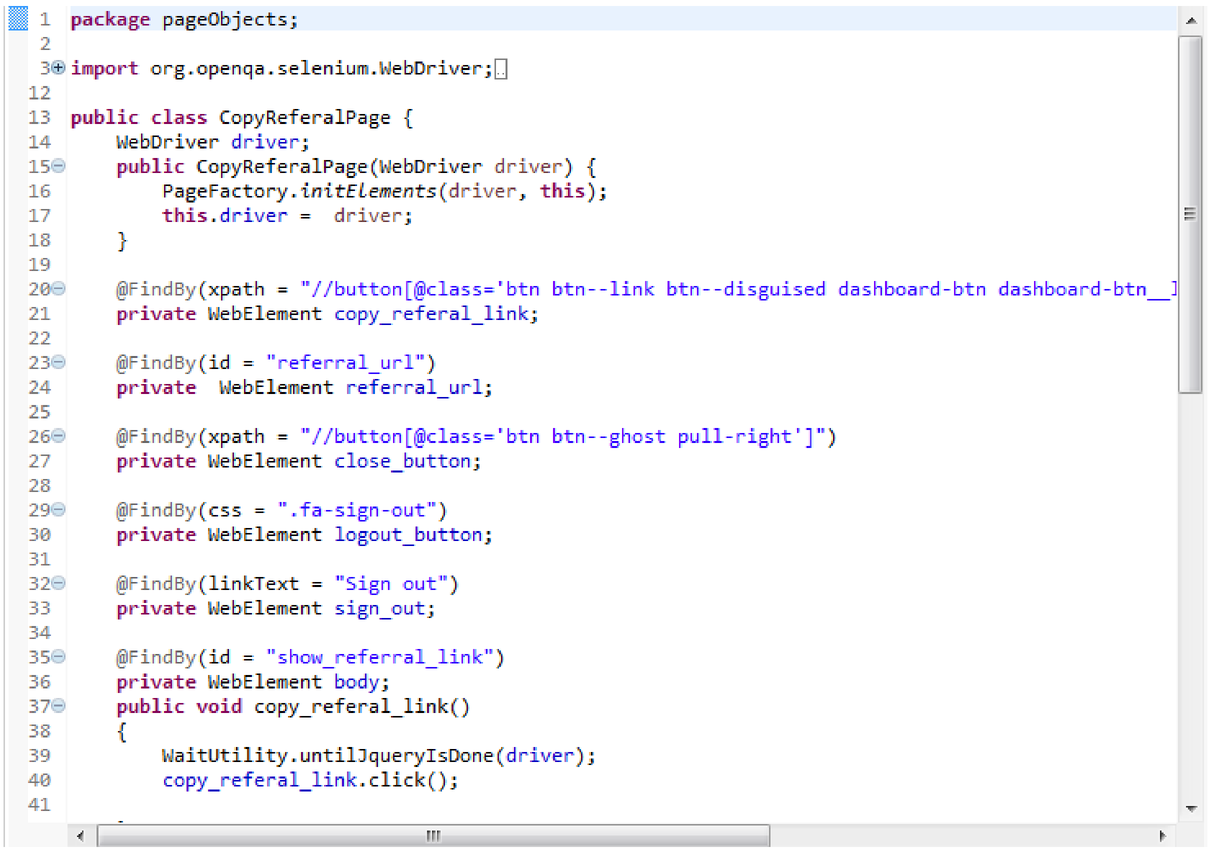
This page is an Object repository for Add Loan Manually Feature since the navigation steps to actual loan application is different



```java
package pageObjects;

import org.openqa.selenium.WebDriver;

public class AddLoanManuallyPage {
    WebDriver driver;

    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }
    public AddLoanManuallyPage(WebDriver driver)
    {
        PageFactory.initElements(driver, this);

    }

    @FindBy(xpath = "//div[@class='icon--active far fa-file-edit c-icon']")
    private WebElement add_loan_man;

    @FindBy(id = "loan_email")
    private WebElement loan_email;

    @FindBy(id = "loan_first_name")
    private WebElement loan_first_name;

    @FindBy(id= "loan_last_name")
    private WebElement loan_last_name;

```

CopyReferalPage.java

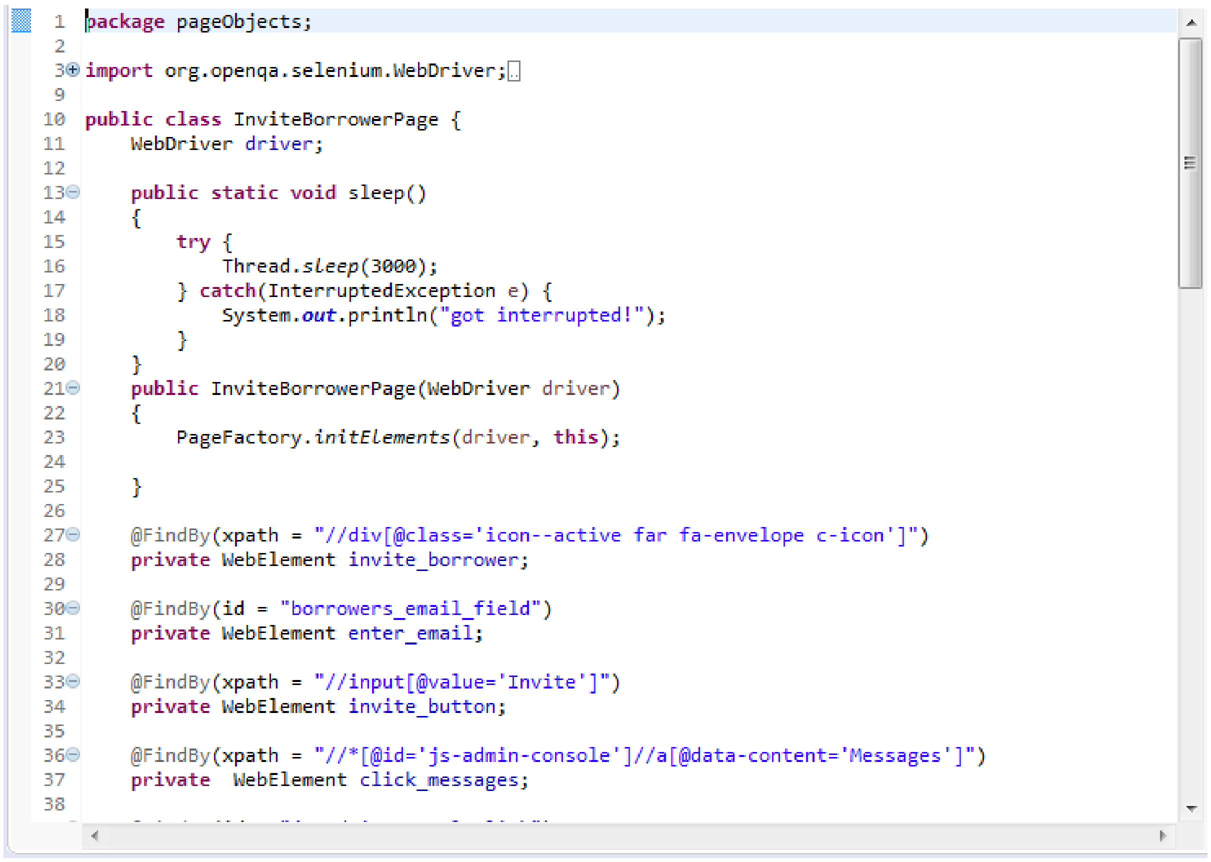
This class stores all the object repositories of the elements which use the method of Copying the referral link from the Admin dashboard



```
package pageObjects;

import org.openqa.selenium.WebDriver;

public class CopyReferalPage {
    WebDriver driver;
    public CopyReferalPage(WebDriver driver) {
        PageFactory.initElements(driver, this);
        this.driver =  driver;
    }

    @FindBy(xpath = "//button[@class='btn btn--link btn--disguised dashboard-btn dashboard-btn__]
    private WebElement copy_referal_link;

    @FindBy(id = "referral_url")
    private  WebElement referral_url;

    @FindBy(xpath = "//button[@class='btn btn--ghost pull-right']")
    private WebElement close_button;

    @FindBy(css = ".fa-sign-out")
    private WebElement logout_button;

    @FindBy(linkText = "Sign out")
    private WebElement sign_out;

    @FindBy(id = "show_referral_link")
    private WebElement body;
    public void copy_referal_link()
    {
        WaitUtility.untilJqueryIsDone(driver);
        copy_referal_link.click();

```

InviteBorrowerPage.java

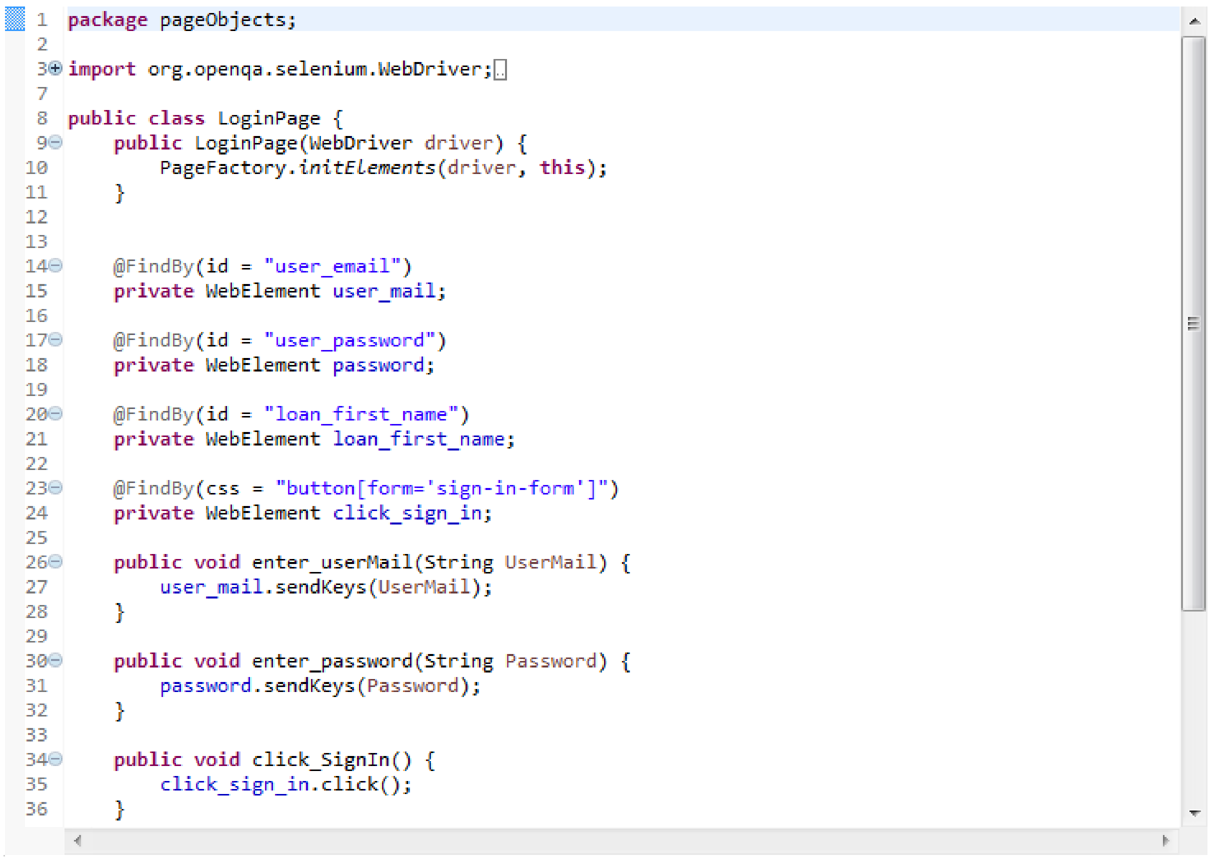
This class stores all the object repositories of the elements which use the method of Inviting Borrower from the Admin Dashboard



```
package pageObjects;

import org.openqa.selenium.WebDriver;

public class InviteBorrowerPage {
    WebDriver driver;

    public static void sleep()
    {
        try {
            Thread.sleep(3000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }
    }
    public InviteBorrowerPage(WebDriver driver)
    {
        PageFactory.initElements(driver, this);

    }

    @FindBy(xpath = "//div[@class='icon--active far fa-envelope c-icon']")
    private WebElement invite_borrower;

    @FindBy(id = "borrowers_email_field")
    private WebElement enter_email;

    @FindBy(xpath = "//input[@value='Invite']")
    private WebElement invite_button;

    @FindBy(xpath = "//*[@id='js-admin-console']//a[@data-content='Messages']")
    private  WebElement click_messages;

```

LoginPage.java

This page stores the Object repositories of the login details for sign in purpose of the user



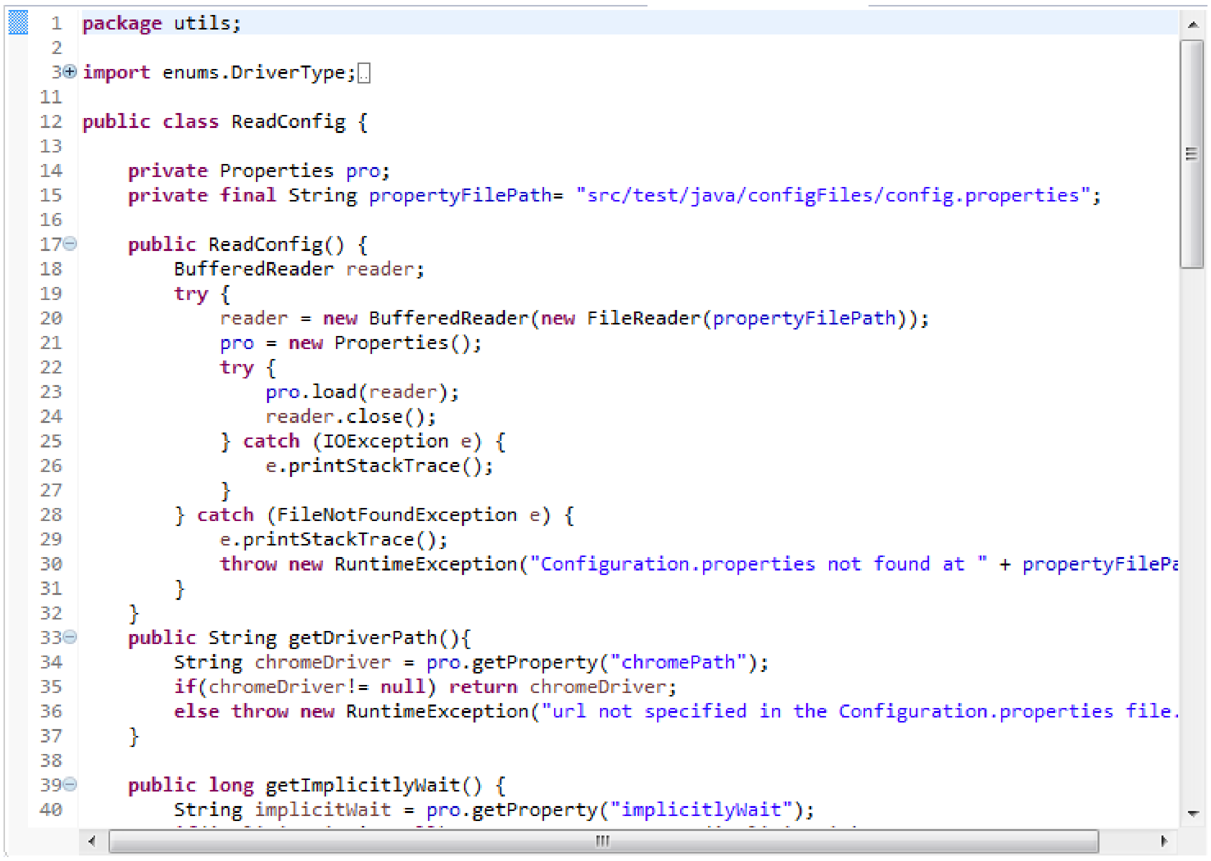
```java
package pageObjects;

import org.openqa.selenium.WebDriver;

public class LoginPage {
    public LoginPage(WebDriver driver) {
        PageFactory.initElements(driver, this);
    }


    @FindBy(id = "user_email")
    private WebElement user_mail;

    @FindBy(id = "user_password")
    private WebElement password;

    @FindBy(id = "loan_first_name")
    private WebElement loan_first_name;

    @FindBy(css = "button[form='sign-in-form']")
    private WebElement click_sign_in;

    public void enter_userMail(String UserMail) {
        user_mail.sendKeys(UserMail);
    }

    public void enter_password(String Password) {
        password.sendKeys(Password);
    }

    public void click_SignIn() {
        click_sign_in.click();
    }
```

Utils

This package is used to write some of the commonly used utilities in script. The functions are written in the respective utilities and hence just called whenever required in the entire application script

ReadConfig.java

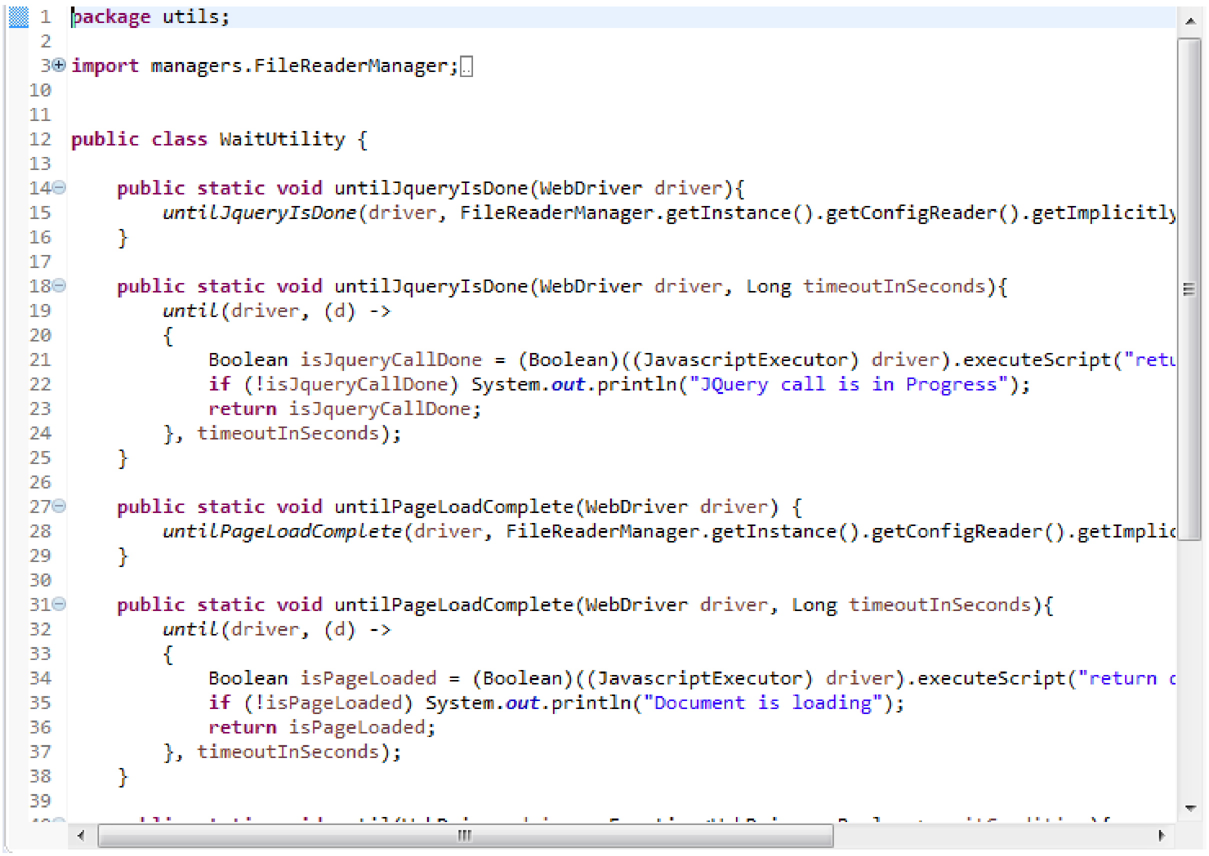
This file is a configuration file reader where the path of the config file is specified



```java
package utils;

import enums.DriverType;

public class ReadConfig {

    private Properties pro;
    private final String propertyFilePath= "src/test/java/configFiles/config.properties";

    public ReadConfig() {
        BufferedReader reader;
        try {
            reader = new BufferedReader(new FileReader(propertyFilePath));
            pro = new Properties();
            try {
                pro.load(reader);
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            throw new RuntimeException("Configuration.properties not found at " + propertyFileP
        }
    }
    public String getDriverPath(){
        String chromeDriver = pro.getProperty("chromePath");
        if(chromeDriver!= null) return chromeDriver;
        else throw new RuntimeException("url not specified in the Configuration.properties file.
    }

    public long getImplicitlyWait() {
        String implicitWait = pro.getProperty("implicitlyWait");
```

WaitUtility.java

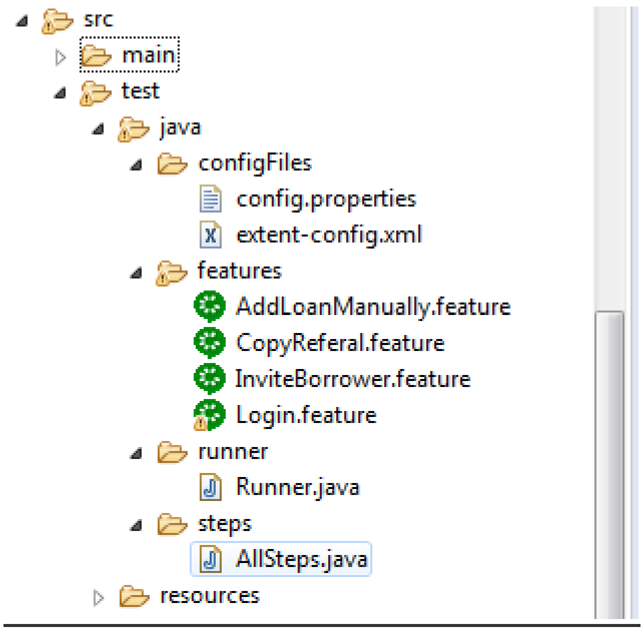
This Utility is used to call the Explicit wait functions we use in order to wait for page to load or either an element to be visible.



```java
package utils;

import managers.FileReaderManager;


public class WaitUtility {

    public static void untilJqueryIsDone(WebDriver driver){
        untilJqueryIsDone(driver, FileReaderManager.getInstance().getConfigReader().getImplicitly
    }

    public static void untilJqueryIsDone(WebDriver driver, Long timeoutInSeconds){
        until(driver, (d) ->
        {
            Boolean isJqueryCallDone = (Boolean)((JavascriptExecutor) driver).executeScript("retu
            if (!isJqueryCallDone) System.out.println("JQuery call is in Progress");
            return isJqueryCallDone;
        }, timeoutInSeconds);
    }

    public static void untilPageLoadComplete(WebDriver driver) {
        untilPageLoadComplete(driver, FileReaderManager.getInstance().getConfigReader().getImplic
    }

    public static void untilPageLoadComplete(WebDriver driver, Long timeoutInSeconds){
        until(driver, (d) ->
        {
            Boolean isPageLoaded = (Boolean)((JavascriptExecutor) driver).executeScript("return d
            if (!isPageLoaded) System.out.println("Document is loading");
            return isPageLoaded;
        }, timeoutInSeconds);
    }

```

Src->Test->Java

This folder mainly consists of the test cases written in form of Cucumber Feature files, the step definitions and the runner class

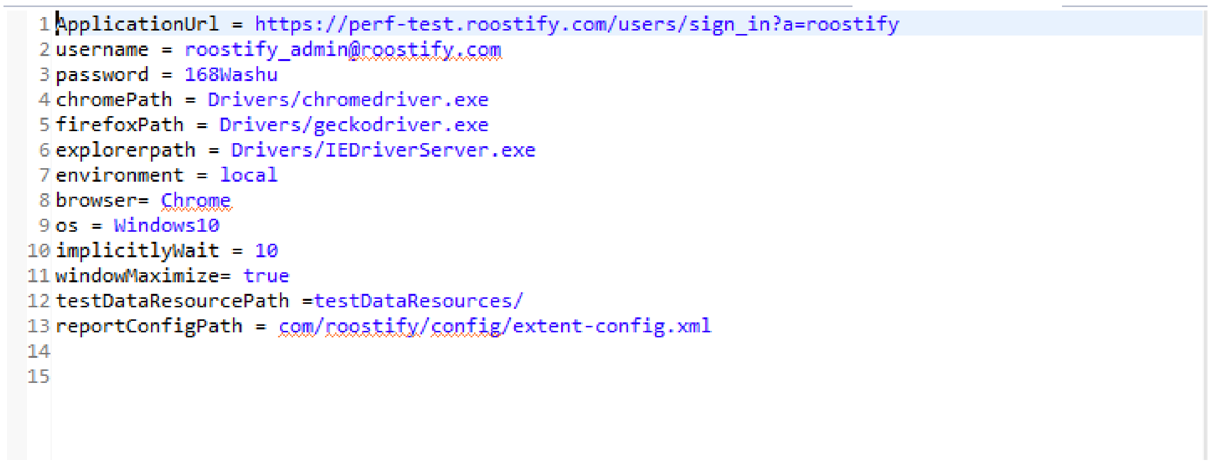


```
src
  main
  test
    java
      configFiles
        config.properties
        extent-config.xml
      features
        AddLoanManually.feature
        CopyReferal.feature
        InviteBorrower.feature
        Login.feature
      runner
        Runner.java
      steps
        AllSteps.java
    resources
```

configFiles

This package consists of a file config.properties which fetches values runtime while running an automation script

We can pass runtime parameters including the environment on which we want to run the script, the user credentials, the type of wait we require etc

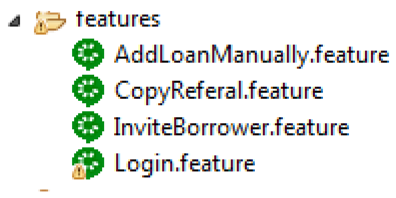


```
ApplicationUrl = https://perf-test.roostify.com/users/sign_in?a=roostify
username = roostify_admin@roostify.com
password = 168Washu
chromePath = Drivers/chromedriver.exe
firefoxPath = Drivers/geckodriver.exe
explorerpath = Drivers/IEDriverServer.exe
environment = local
browser= Chrome
os = Windows10
implicitlyWait = 10
windowMaximize= true
testDataResourcePath =testDataResources/
reportConfigPath = com/roostify/config/extent-config.xml
```

Features

This package consists of all the feature files which we will use through the entire script and flow of the application.

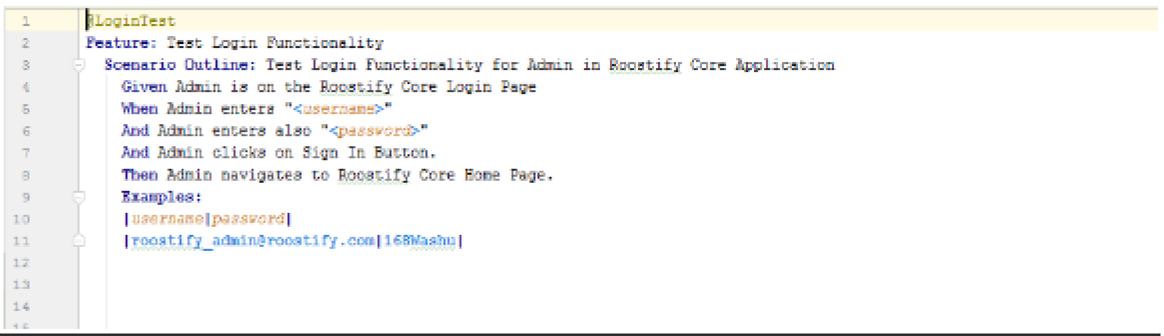
For now, there are 4 feature files namely:-



```
features
    AddLoanManually.feature
    CopyReferal.feature
    InviteBorrower.feature
    Login.feature
```

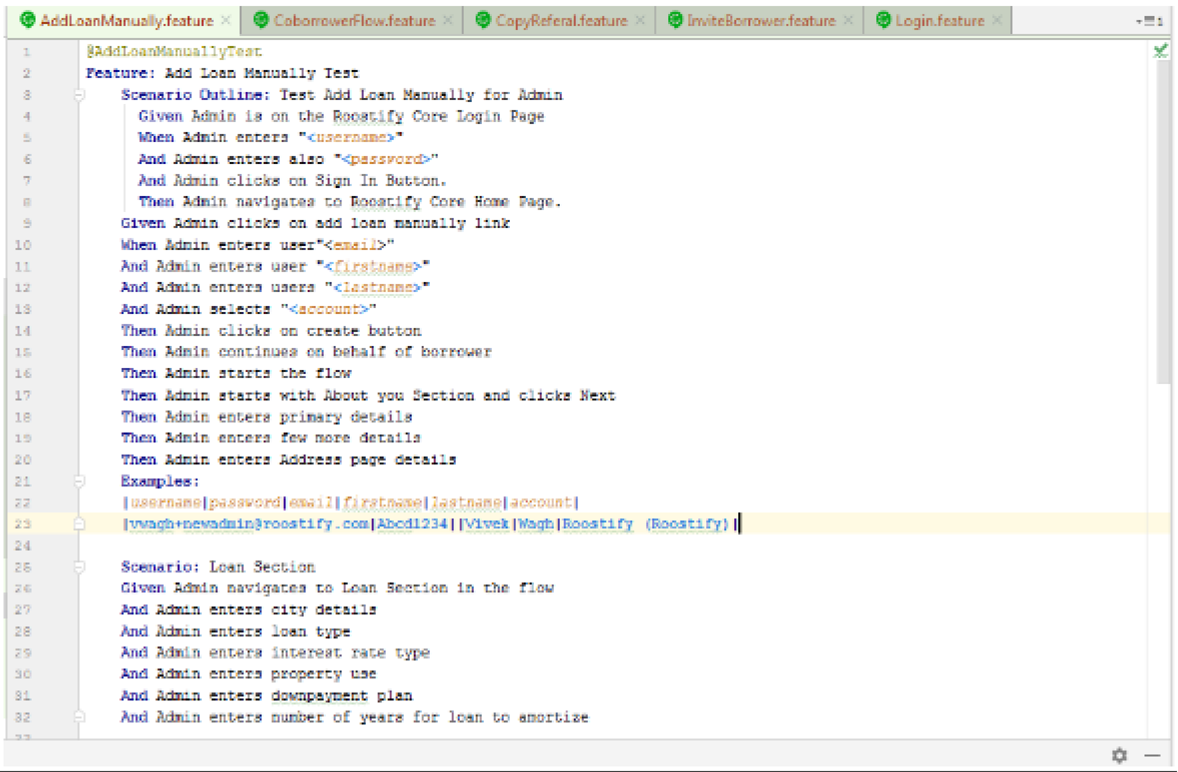
These files define the scenarios and the steps which are followed for the application navigation flows

Login.feature



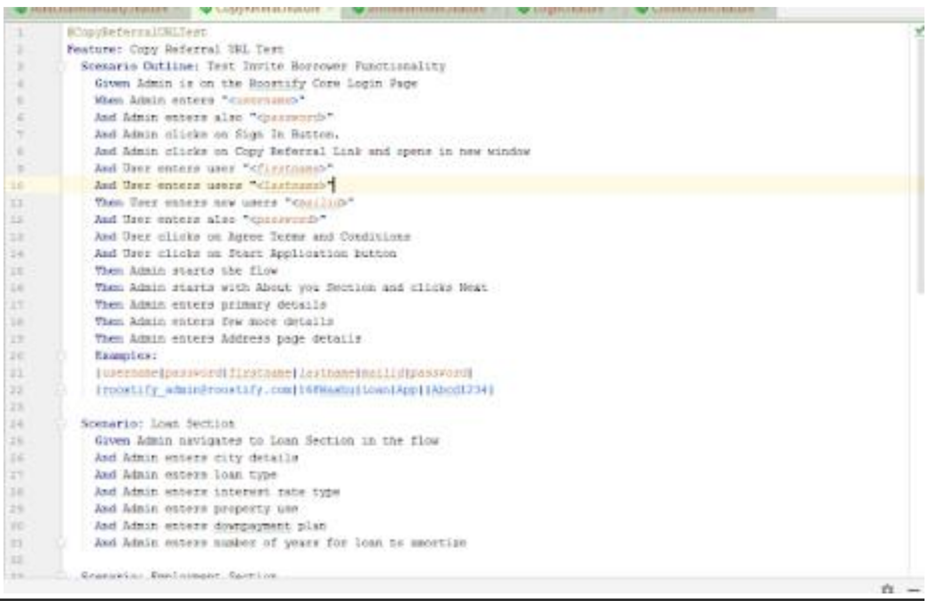
```
@LoginTest
Feature: Test Login Functionality
  Scenario Outline: Test Login Functionality for Admin in Roostify Core Application
    Given Admin is on the Roostify Core Login Page
    When Admin enters "<username>"
    And Admin enters also "<password>"
    And Admin clicks on Sign In Button.
    Then Admin navigates to Roostify Core Home Page.
    Examples:
    |username|password|
    |roostify_admin@roostify.com|168Mashu|
```

AddLoanManually.feature



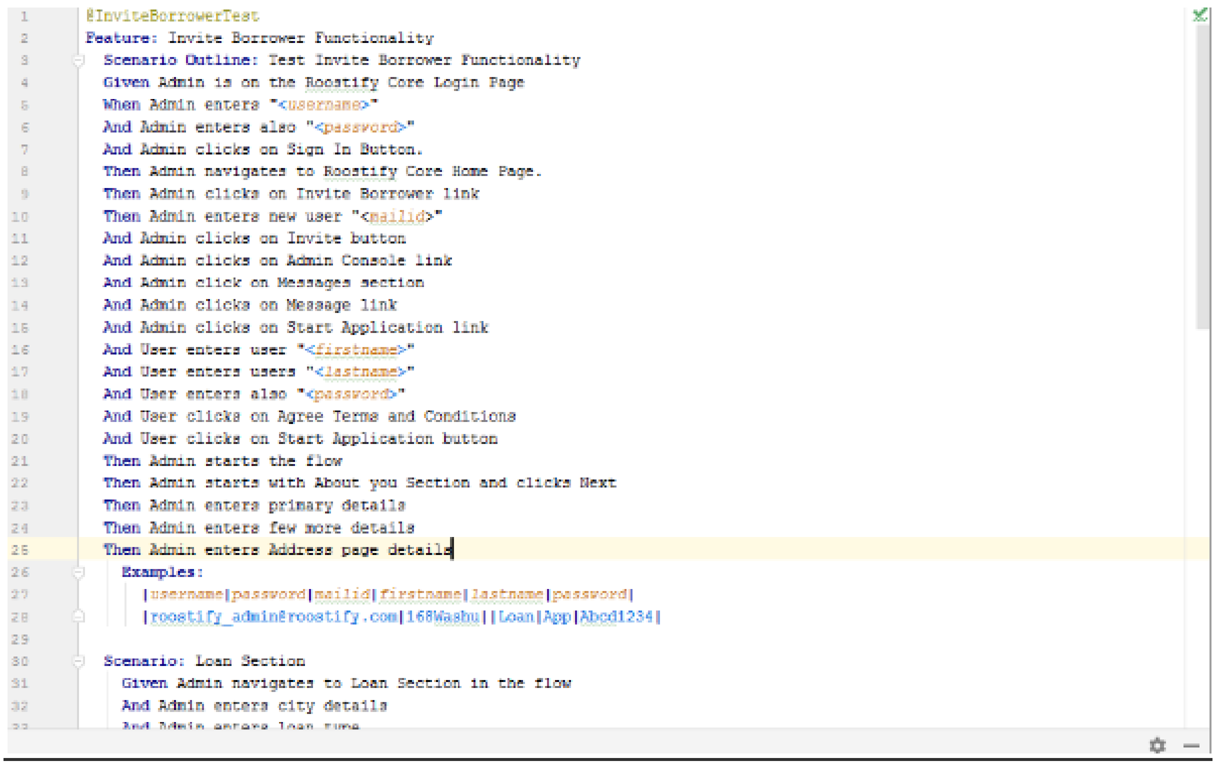
```
@AddLoanManuallyTest
Feature: Add Loan Manually Test
    Scenario Outline: Test Add Loan Manually for Admin
      Given Admin is on the Roostify Core Login Page
      When Admin enters "<username>"
      And Admin enters also "<password>"
      And Admin clicks on Sign In Button.
      Then Admin navigates to Roostify Core Home Page.
    Given Admin clicks on add loan manually link
    When Admin enters user"<email>"
    And Admin enters user "<firstname>"
    And Admin enters users "<lastname>"
    And Admin selects "<account>"
    Then Admin clicks on create button
    Then Admin continues on behalf of borrower
    Then Admin starts the flow
    Then Admin starts with About you Section and clicks Next
    Then Admin enters primary details
    Then Admin enters few more details
    Then Admin enters Address page details
    Examples:
    |username|password|email|firstname|lastname|account|
    |vwagh+newadmin@roostify.com|Abcd1234||Vivek|Wagh|Roostify (Roostify)|

    Scenario: Loan Section
    Given Admin navigates to Loan Section in the flow
    And Admin enters city details
    And Admin enters loan type
    And Admin enters interest rate type
    And Admin enters property use
    And Admin enters downpayment plan
    And Admin enters number of years for loan to amortize
```

CopyReferal.feature



```
@CopyReferralURLTest
Feature: Copy Referral URL Test
  Scenario Outline: Test Invite Borrower Functionality
    Given Admin is on the Roostify Core Login Page
    When Admin enters "<username>"
    And Admin enters also "<password>"
    And Admin clicks on Sign In Button.
    And Admin clicks on Copy Referral Link and opens in new window
    And User enters user "<firstname>"
    And User enters users "<lastname>"
    Then User enters new users "<emailid>"
    And User enters also "<password>"
    And User clicks on Agree Terms and Conditions
    And User clicks on Start Application button
    Then Admin starts the flow
    Then Admin starts with About you Section and clicks Next
    Then Admin enters primary details
    Then Admin enters few more details
    Then Admin enters Address page details
    Examples:
    |username|password|firstname|lastname|emailid|password|
    |roostify_admin@roostify.com|[illegible]|Loan|App||Abcd1234|

  Scenario: Loan Section
    Given Admin navigates to Loan Section in the flow
    And Admin enters city details
    And Admin enters loan type
    And Admin enters interest rate type
    And Admin enters property use
    And Admin enters downpayment plan
    And Admin enters number of years for loan to amortize

  Scenario: Employment Section
```

InviteBorrower.feature



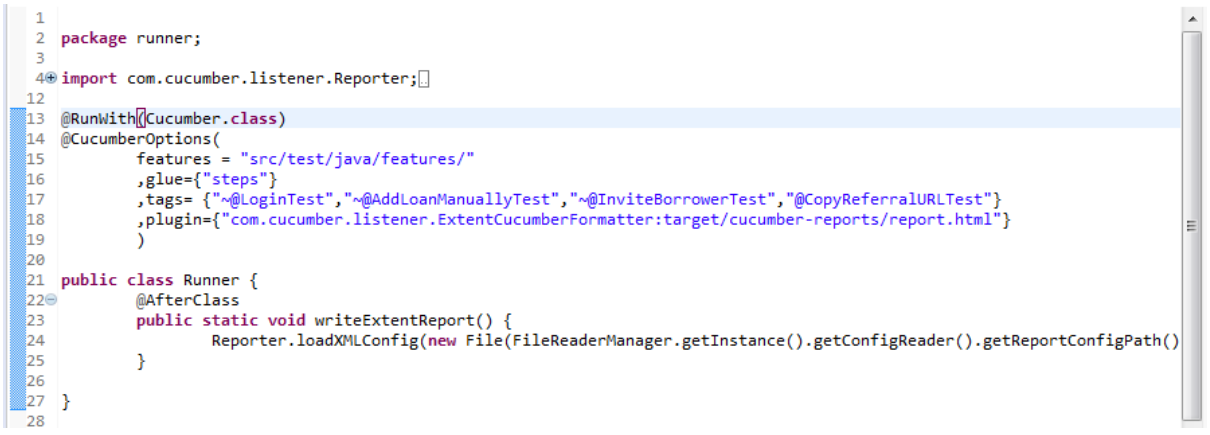
```
@InviteBorrowerTest
Feature: Invite Borrower Functionality
  Scenario Outline: Test Invite Borrower Functionality
  Given Admin is on the Roostify Core Login Page
  When Admin enters "<username>"
  And Admin enters also "<password>"
  And Admin clicks on Sign In Button.
  Then Admin navigates to Roostify Core Home Page.
  Then Admin clicks on Invite Borrower link
  Then Admin enters new user "<mailid>"
  And Admin clicks on Invite button
  And Admin clicks on Admin Console link
  And Admin click on Messages section
  And Admin clicks on Message link
  And Admin clicks on Start Application link
  And User enters user "<firstname>"
  And User enters users "<lastname>"
  And User enters also "<password>"
  And User clicks on Agree Terms and Conditions
  And User clicks on Start Application button
  Then Admin starts the flow
  Then Admin starts with About you Section and clicks Next
  Then Admin enters primary details
  Then Admin enters few more details
  Then Admin enters Address page details
    Examples:
      |username|password|mailid|firstname|lastname|password|
      |roostify_admin@roostify.com|168Washu||Loan|App|Abcd1234|

  Scenario: Loan Section
    Given Admin navigates to Loan Section in the flow
    And Admin enters city details
    And Admin enters loan type
```

Runner

This package consists of the Junit runner class which integrates the step definition as well the feature files

The generic path of the feature files and step definitions is given in the runner file

Using this file we can also specify the tags using which we can custom run the scenario we need to perform

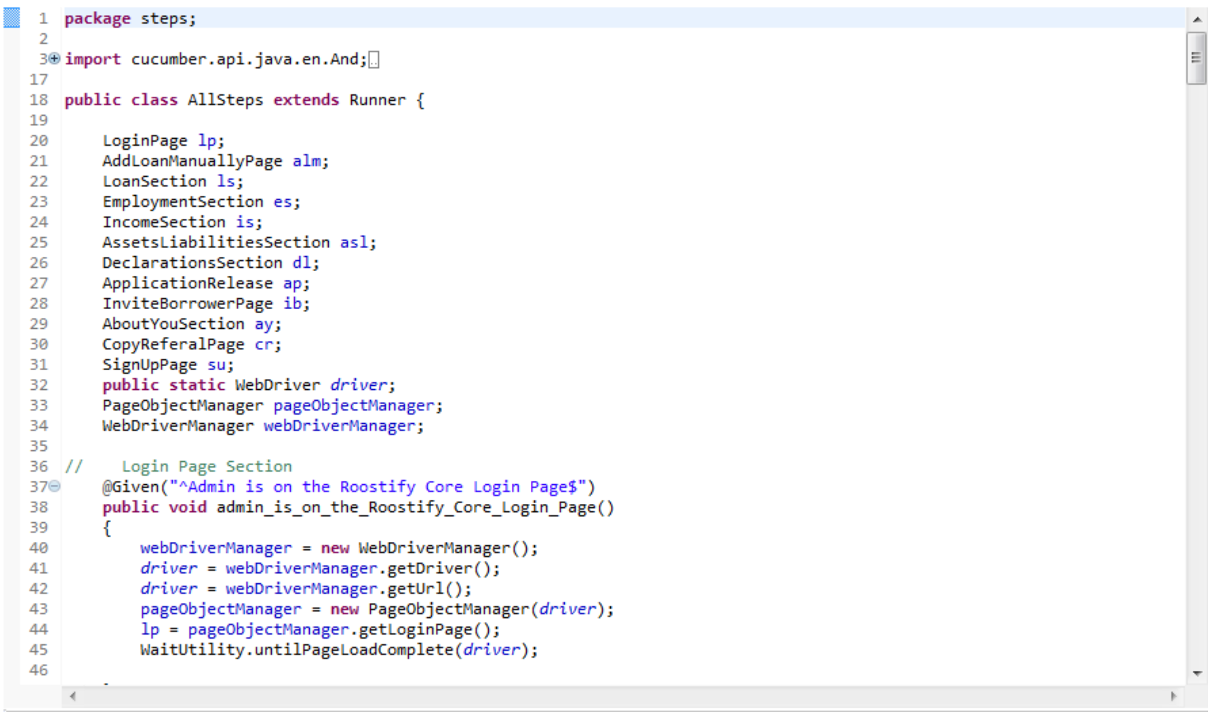


```

package runner;

import com.cucumber.listener.Reporter;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/features/"
        ,glue={"steps"}
        ,tags= {"~@LoginTest","~@AddLoanManuallyTest","~@InviteBorrowerTest","@CopyReferralURLTest"}
        ,plugin={"com.cucumber.listener.ExtentCucumberFormatter:target/cucumber-reports/report.html"}
        )

public class Runner {
        @AfterClass
        public static void writeExtentReport() {
                Reporter.loadXMLConfig(new File(FileReaderManager.getInstance().getConfigReader().getReportConfigPath()
        }

}

```

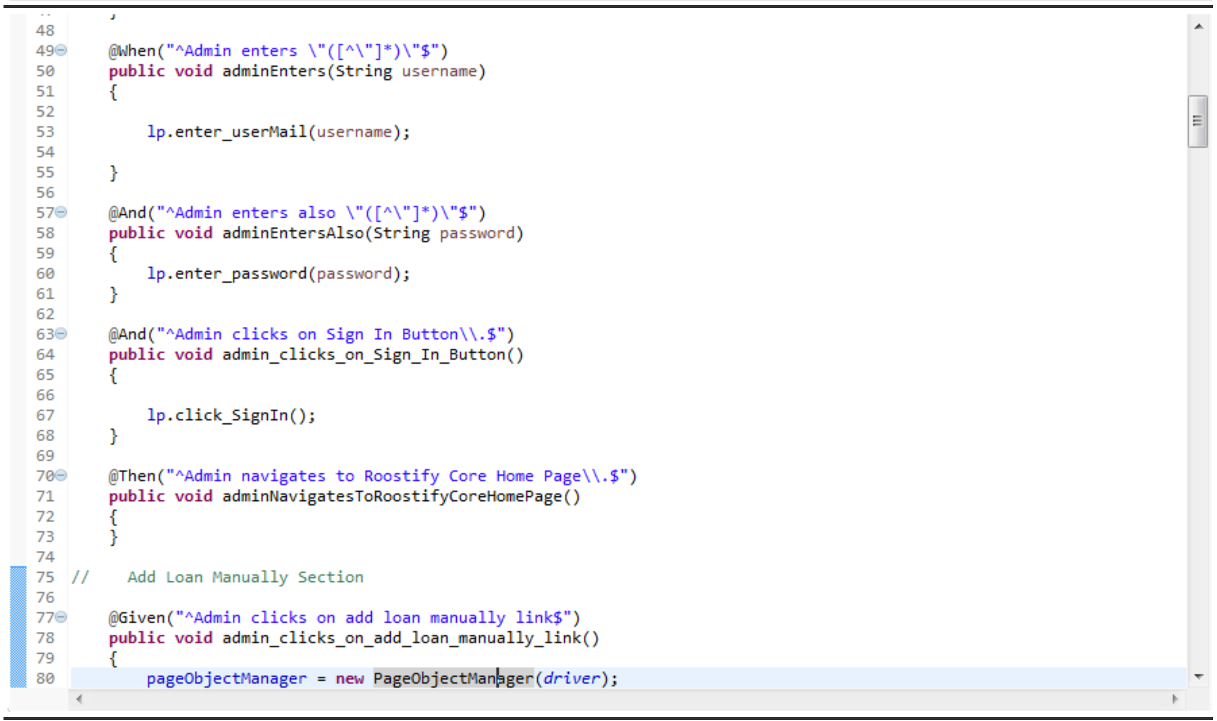
Steps

This package consists of the step definition file .The page objects are called as per the control passes from section to section.

AllSteps.java



```java
package steps;

import cucumber.api.java.en.And;

public class AllSteps extends Runner {

    LoginPage lp;
    AddLoanManuallyPage alm;
    LoanSection ls;
    EmploymentSection es;
    IncomeSection is;
    AssetsLiabilitiesSection asl;
    DeclarationsSection dl;
    ApplicationRelease ap;
    InviteBorrowerPage ib;
    AboutYouSection ay;
    CopyReferalPage cr;
    SignUpPage su;
    public static WebDriver driver;
    PageObjectManager pageObjectManager;
    WebDriverManager webDriverManager;

//    Login Page Section
    @Given("^Admin is on the Roostify Core Login Page$")
    public void admin_is_on_the_Roostify_Core_Login_Page()
    {
        webDriverManager = new WebDriverManager();
        driver = webDriverManager.getDriver();
        driver = webDriverManager.getUrl();
        pageObjectManager = new PageObjectManager(driver);
        lp = pageObjectManager.getLoginPage();
        WaitUtility.untilPageLoadComplete(driver);

```



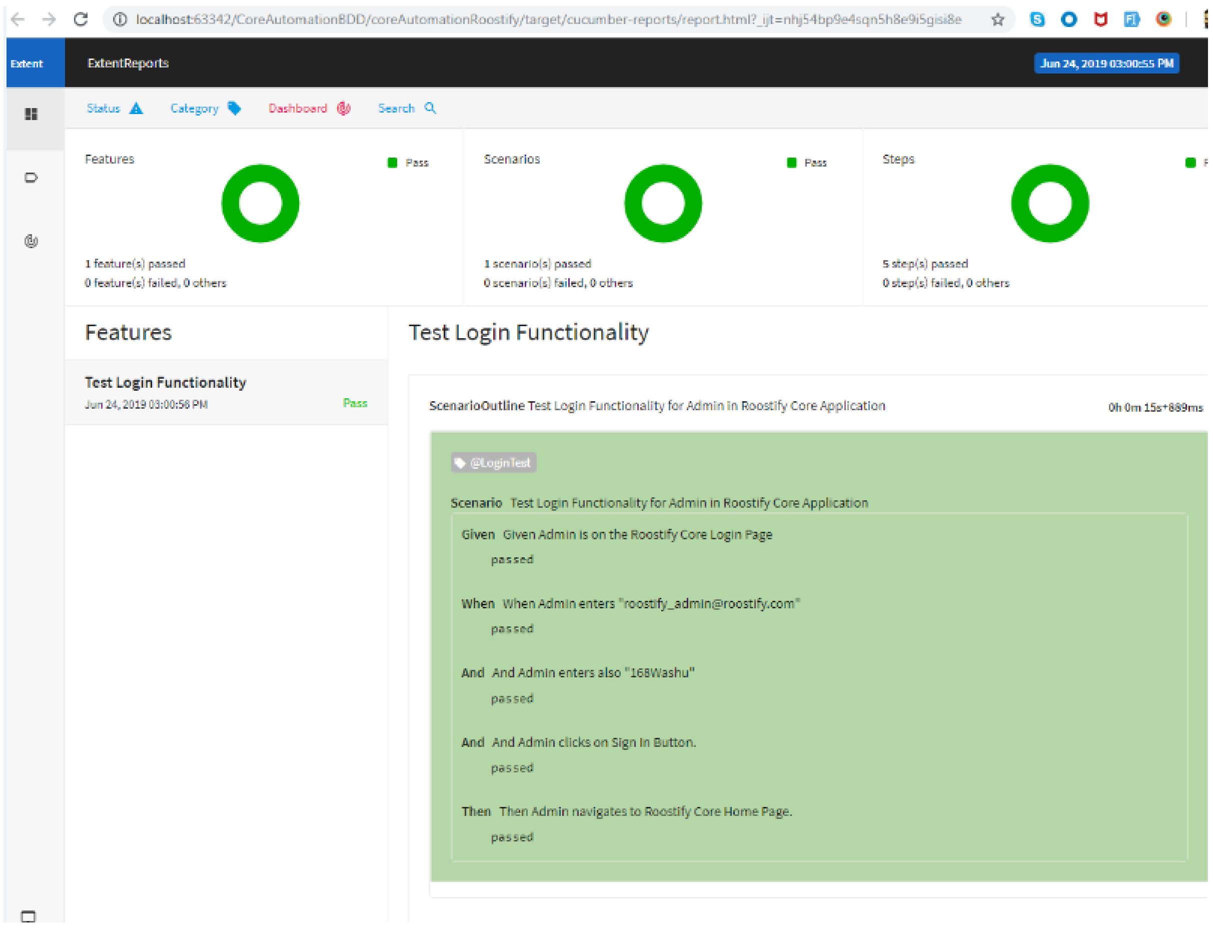
```java
    }

    @When("^Admin enters \"([^\"]*)\"$")
    public void adminEnters(String username)
    {

        lp.enter_userMail(username);

    }

    @And("^Admin enters also \"([^\"]*)\"$")
    public void adminEntersAlso(String password)
    {
        lp.enter_password(password);
    }

    @And("^Admin clicks on Sign In Button\\.$")
    public void admin_clicks_on_Sign_In_Button()
    {

        lp.click_SignIn();
    }

    @Then("^Admin navigates to Roostify Core Home Page\\.$")
    public void adminNavigatesToRoostifyCoreHomePage()
    {
    }

//    Add Loan Manually Section

    @Given("^Admin clicks on add loan manually link$")
    public void admin_clicks_on_add_loan_manually_link()
    {
        pageObjectManager = new PageObjectManager(driver);
```

Test-output

This folder gets generated at the time of default reporting of Cucumber.

Cucumber Reports are generated here in this folder.

For this project, We have used Extent Reports for generating the Reports which will give a clear picture of the test cases which are passed and failed.



localhost:63342/CoreAutomationBDD/coreAutomationRoostify/target/cucumber-reports/report.html?_ijt=nhj54bp9e4sqn5h8e9i5gisi8e

Extent ExtentReports Jun 24, 2019 03:00:55 PM

Status Category Dashboard Search

Features 🟩 Pass
1 feature(s) passed
0 feature(s) failed, 0 others

Scenarios 🟩 Pass
1 scenario(s) passed
0 scenario(s) failed, 0 others

Steps 🟩
5 step(s) passed
0 step(s) failed, 0 others

Features

Test Login Functionality
Jun 24, 2019 03:00:56 PM Pass

Test Login Functionality

ScenarioOutline Test Login Functionality for Admin in Roostify Core Application 0h 0m 15s+889ms

@LoginTest

Scenario Test Login Functionality for Admin in Roostify Core Application

Given Given Admin is on the Roostify Core Login Page
passed

When When Admin enters "roostify_admin@roostify.com"
passed

And And Admin enters also "168Washu"
passed

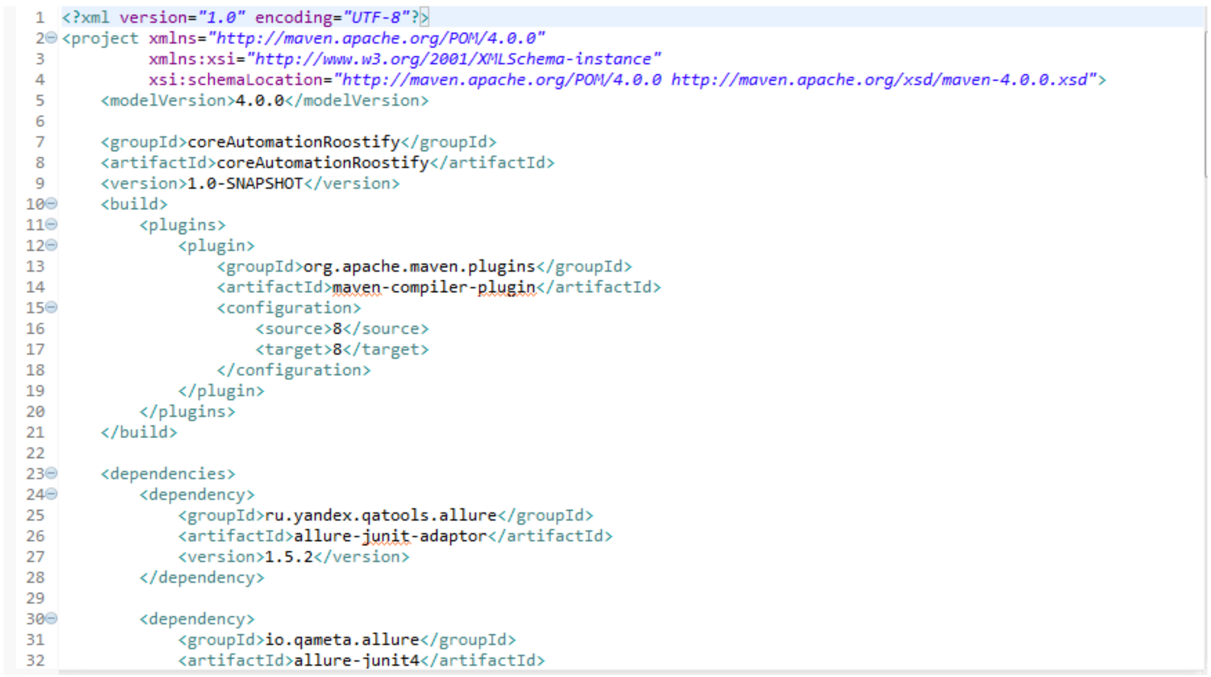
And And Admin clicks on Sign In Button.
passed

Then Then Admin navigates to Roostify Core Home Page.
passed

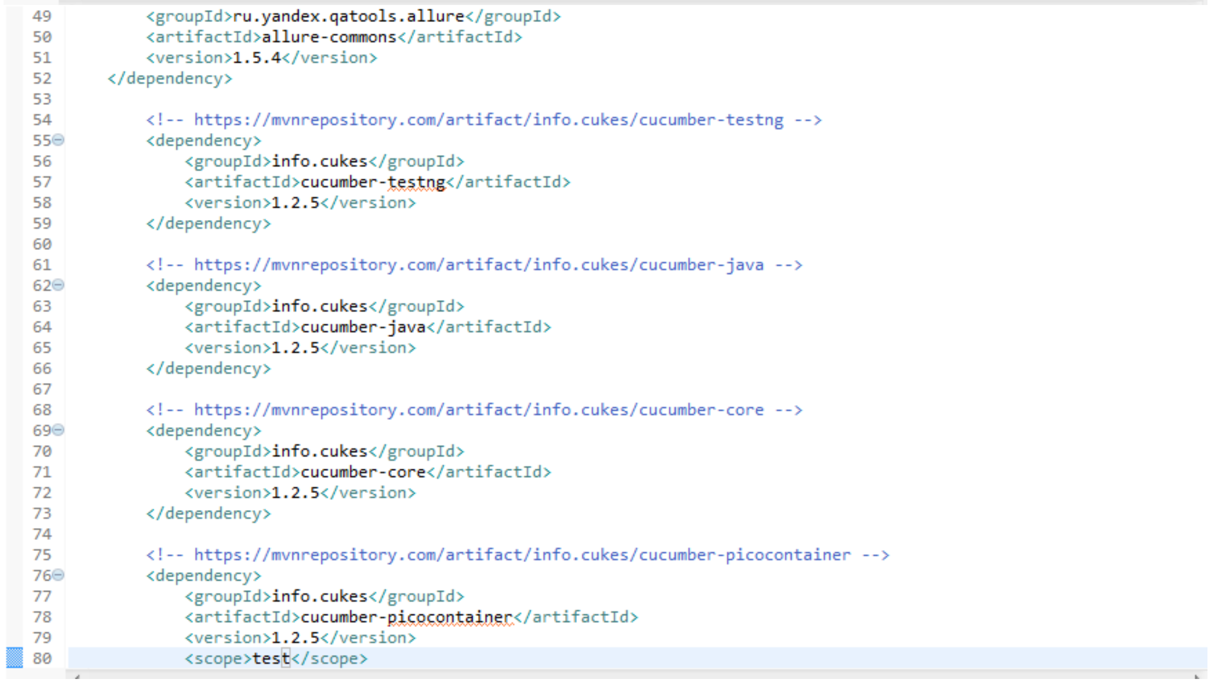
Pom.xml

Pom.xml file consists of all the plugins and dependencies which are needed to execute a cucumber-selenium project

These dependencies can be downloaded from the Maven Central Repository.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>coreAutomationRoostify</groupId>
    <artifactId>coreAutomationRoostify</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>8</source>
                    <target>8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <dependency>
            <groupId>ru.yandex.qatools.allure</groupId>
            <artifactId>allure-junit-adaptor</artifactId>
            <version>1.5.2</version>
        </dependency>

        <dependency>
            <groupId>io.qameta.allure</groupId>
            <artifactId>allure-junit4</artifactId>
```



```xml
        <groupId>ru.yandex.qatools.allure</groupId>
        <artifactId>allure-commons</artifactId>
        <version>1.5.4</version>
    </dependency>

        <!-- https://mvnrepository.com/artifact/info.cukes/cucumber-testng -->
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-testng</artifactId>
            <version>1.2.5</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/info.cukes/cucumber-java -->
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-java</artifactId>
            <version>1.2.5</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/info.cukes/cucumber-core -->
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-core</artifactId>
            <version>1.2.5</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/info.cukes/cucumber-picocontainer -->
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-picocontainer</artifactId>
            <version>1.2.5</version>
            <scope>test</scope>
```

References

- https://www.toolsqa.com/cucumber-automation-framework/
- https://www.softwaretestinghelp.com/cucumber-bdd-tool-selenium-tutorial-30/