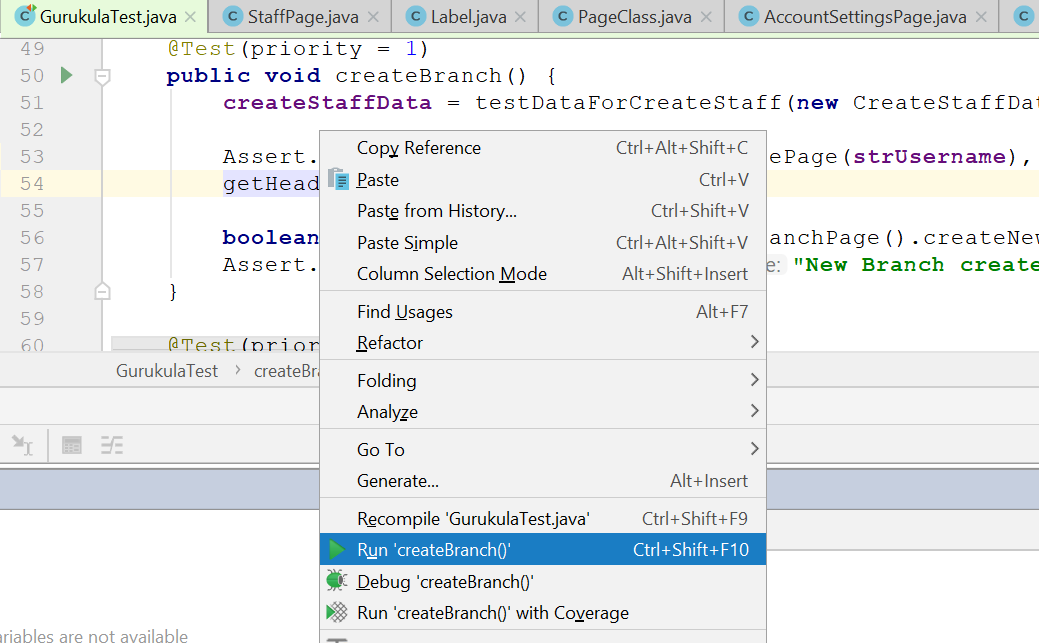
**How To Run Tests:**

**Java- Test File Location :** . \src\test\java\com\snow\gk\GurukulaTest

**Steps to run**

1. Navigate to the above mentioned test file.
2. Right click on the test you want to run, select Run ‘createBranch’. This will run the createBranch() test. Similar way you can run the test displayed in the above file

****

**LogReports**

Log file are generated in the given folder according to timestamp

**Scenarios Automated:**

**Create Staff:** Creates staff with the bean value which has been created using the method(testDataForCreateStaff)

**Create Branch:** Creates branch with the bean value which has been created using the method(testDataForCreateStaff)

**View/Edit/Delete Staff/Branch:** TC creates the Branch/Staff and tests the following conditions as View/Edit/Delete buttons which is displayed in table

**Search Staff/Branch:** TC which create branch/staff and tries to search for the created and validates the search functionality is working as expected.

Note: Searching Staff based on the Branch name is not giving search results

**Cancel Branch & Cancel Staff**: This method tests the cancel button functionality

1. While creating Branch/Staff in Create and Edit Branch pop up’s Cancel button
2. When editing Branch/Staff we have Cancel button
3. When deleting the Branch/Staff, in delete pop up window validates the Cancel button

**Login with Invalid Credentials**: Login the app with invalid credentials and test whether the appropriate error message are displayed.

**Update Password:** This test try to update the password for the admin, but this functionality is not working. Since the test will fail

**Edit Account Settings:** Test tries to update the settings of the user which by updating the email, firstname, lastname etc. Since this is not working for admin. This tc also fails

**About Framework:**

**Elements:**

Elements are the segregated based on the ui, which user see in the webpage (e.g. Button, Table, TextField etc.). Each element has their own interface and implementation part, which will be discussed below. All these elements has a super interface and its implementation class which are IElement and Element.

Any interface which is to be created should extend IElement, and implementation class should extend Element class.

**IElements & Element**

IElement is an interface which holds all common methods which can be possible, for all elements. So that user need not want to duplicate the methods to all elements.

Element class holds the implementation for the methods described in the IElement interface.

**DriverSetup:**

Class is static class which helps to initialize driver (e.g.: chrome). For getting the present driver, user can call getDriver method to get the current driver from the thread

**FrameworkExceptions:**

It’s a customized exception for the framework, use this exception to be thrown from the methods, classes.

**Constants:**

Class contains the constants which are used in the framework

**LogEventListener:**

Class contains code for printing the step by step logs in the log reports with the help of extending the TestListenerAdapter and implementing the WebDriverEventListener.

**Logger:**

For logging some info, error, warning this class is used. The message user passes will be logged in the generated logs.

**AllPages:**

Class has a static method getPage, with page class name as a parameter to it. Then this class returns a page class object of the class passed. It just calls the PageFactory.initElements class of selenium core.

**Components:**

Every pages been created in the application, should extend this class. It converts the web element into framework elements (e.g. Button, Table, DropDown), and it also helps in getting the element names in the logs human readable by getting the web element names into it.

**Waits:**

Class contains the different kinds of wait, can be used throughout framework. If user defines custom wait, then it should be added to the class

**FileUtility**

File related operations are to be added to this class.