Concordia University

SOEN-6011 Software Engineering Processes

SCIENTIFIC CALCULATOR

DELIVERABLE 3 (D3)

Problem 5 & 7

GitHub : `https://github.com/san089/SOEN-6011`

Author: Sanchit Kumar(40081187)

July 31, 2019

1 Problem 5 : Source Code Review

Tool Used : Upsource 2019.1
Link : `https://www.jetbrains.com/upsource/`

1.1 Tool Overview

JetBrains Upsource is a polyglot code review tool. It offers lot of benifits to the users such as saving time on automating routine tasks, locating and fixing errors, taking advantage of intelligent support from the IDE, and increasing their overall productivity. With Upsource, teams can save significant time on locating and fixing errors, so that these hours of saved time per day are used for actual product development and company resources are allocated wisely. Developers can also automate many manual routine tasks, and get useful hints and code insights straight away, not wasting any time on switching to other tools or consulting documentation. In addition, Upsource reliably improves coding practices and eases the handling of multi-language solutions.

Upsource is built on top of IntelliJ Platform developed by JetBrains. That's why Upsource combines code review functionality with IDE-specific capabilities, including static code analysis and code-aware navigation.

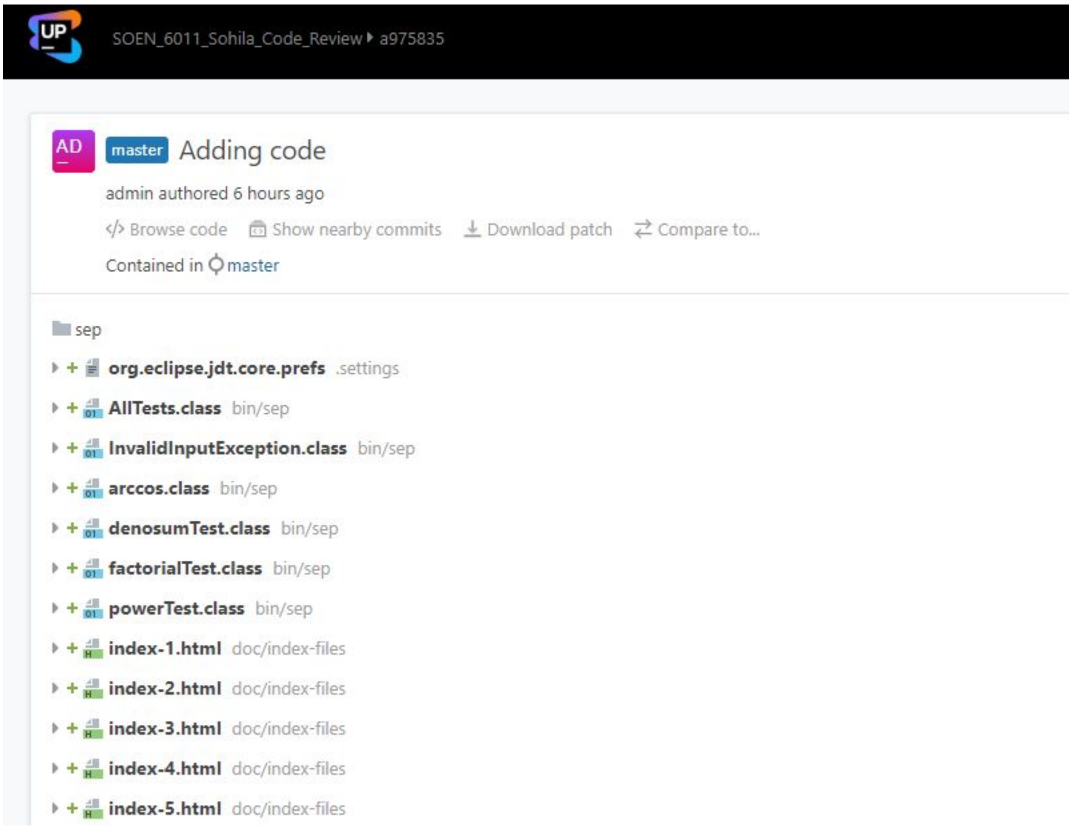
Figure 1: Upsource

1.2 Approach

Upsource provides teams to collaborate on a common portal, but in order to access its portal team first needs to create their account. For accessing source code review reports please follow below steps to gain access to the project on portal.

Below are the steps to start collaborating.

1. Create account on the given link - `http://desktop-j65qh2b/`.
2. Click Login → Sign Up → Fill your details and complete the registration.
3. Email ”sanchitkumar013@gmail.com” to add you to the project group.
4. You will be provided access to the project group and you can start collaborating.



Adding code

admin authored 6 hours ago

Contained in master

sep

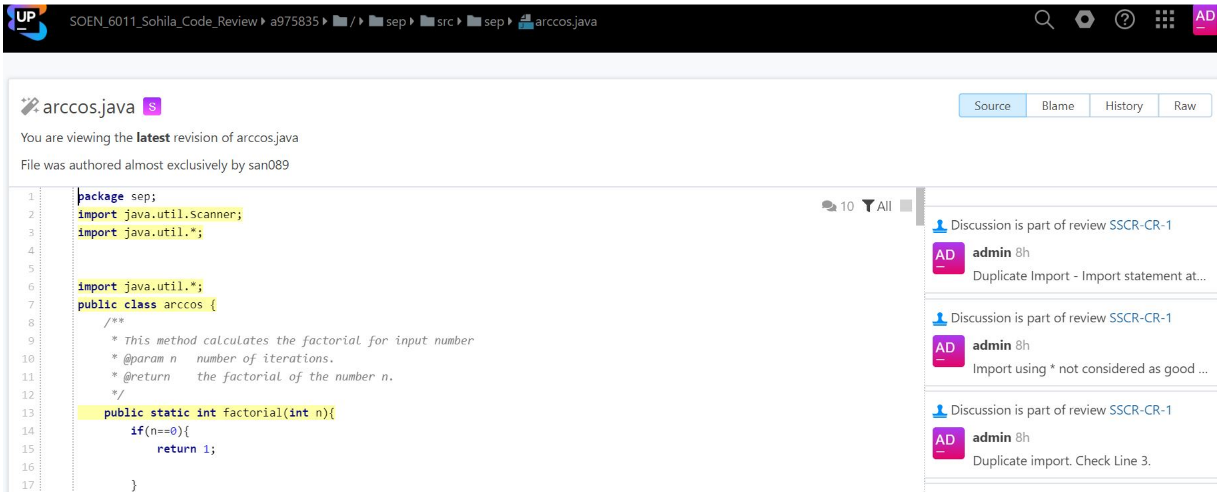
File	Path
org.eclipse.jdt.core.prefs	.settings
AllTests.class	bin/sep
InvalidInputException.class	bin/sep
arccos.class	bin/sep
denosumTest.class	bin/sep
factorialTest.class	bin/sep
powerTest.class	bin/sep
index-1.html	doc/index-files
index-2.html	doc/index-files
index-3.html	doc/index-files
index-4.html	doc/index-files
index-5.html	doc/index-files

Figure 2: Upsource Project View

1.3 Issues Reported For Code Review

- Variables initialized but never used in the code.
- Modules imported but are never used in the code.

- Naming convention not followed for class names.
- Using the ".*" form for importing modules.
- Accessing static method of class via instance variable.
- Unnecessary "continue" statements at the end of while loop.


UP
SOEN_6011_Sohila_Code_Review ▸ a975835 ▸ / ▸ sep ▸ src ▸ sep ▸ arccos.java
AD
arccos.java
S
Source
Blame
History
Raw
10
All

You are viewing the **latest** revision of arccos.java

File was authored almost exclusively by san089

```
1   package sep;
2   import java.util.Scanner;
3   import java.util.*;
4
5
6   import java.util.*;
7   public class arccos {
8       /**
9        * This method calculates the factorial for input number
10       * @param n   number of iterations.
11       * @return    the factorial of the number n.
12       */
13      public static int factorial(int n){
14          if(n==0){
15              return 1;
16
17          }
```

Discussion is part of review SSCR-CR-1
AD **admin** 8h
Duplicate Import - Import statement at...

Discussion is part of review SSCR-CR-1
AD **admin** 8h
Import using * not considered as good ...

Discussion is part of review SSCR-CR-1
AD **admin** 8h
Duplicate import. Check Line 3.

Figure 3: Upsource Code Review Sample

2 Problem 7: Testing

2.1 Project Details

Below are the details about project.

Table 1: Module 2 - tan(x)

Project Name	Scientific Calculator
Module Name	tan(x)
Developed By	Anusha Keralapura Thandavamurthy
Development End	29-July-2019
Tested By	Sanchit Kumar
Testing End	01-August-2019.

2.2 Testing Detailed Report

2.2.1 cos(x) Testing Reports

- **TEST CASE ID** : TC1
  TYPE : Functional
  DESCRIPTION : cos(x) value test.
  INPUT : 0.7853975
  EXPECTED RESULT : 0.7071072502792262
  ACTUAL RESULT : 0.7071072502792262
  TEST RESULT : PASS

- **TEST CASE ID** : TC2
  TYPE : Functional
  DESCRIPTION : cos(x) value test.
  INPUT : 0
  EXPECTED RESULT : 1
  ACTUAL RESULT : 1
  TEST RESULT : PASS

- **TEST CASE ID** : TC3
  TYPE : Functional
  DESCRIPTION : cos(x) value test.
  INPUT : 3.14159
  EXPECTED RESULT : -1.0
  ACTUAL RESULT : -1.0000000035255
  TEST RESULT : FAIL

2.2.2 sin(x) Testing Reports

- **TEST CASE ID** : TC4
  TYPE : Functional
  DESCRIPTION : sin(x) value test.
  INPUT : 0.7853975
  EXPECTED RESULT : 0.7071072502792262
  ACTUAL RESULT : 0.7071072502792262
  TEST RESULT : PASS

- **TEST CASE ID** : TC5
  TYPE : Functional
  DESCRIPTION : sin(x) value test.
  INPUT : 0
  EXPECTED RESULT : 0

ACTUAL RESULT : 0
TEST RESULT : PASS

- **TEST CASE ID** : TC6
  TYPE : Functional
  DESCRIPTION : sin(x) value test.
  INPUT : 3.14159
  EXPECTED RESULT : 0
  ACTUAL RESULT : 2.65306088421985E-6
  TEST RESULT : FAIL

- **TEST CASE ID** : TC7
  TYPE : Functional
  DESCRIPTION : sin(x) value test. Test for accuracy.
  INPUT : 6.2832000000000000001
  EXPECTED RESULT : 0.00001469
  ACTUAL RESULT : -0.0010335410874021851
  TEST RESULT : FAIL

2.2.3 tan(x) Testing Reports

- **TEST CASE ID** : TC8
  TYPE : Functional
  DESCRIPTION : tan(x) value test.
  INPUT : 39
  EXPECTED RESULT : 0.809783081231065
  ACTUAL RESULT : 0.809783081231065
  TEST RESULT : PASS

- **TEST CASE ID** : TC9
  TYPE : Functional
  DESCRIPTION : tan(x) value test.
  INPUT : 0
  EXPECTED RESULT : 0
  ACTUAL RESULT : 0
  TEST RESULT : PASS

- **TEST CASE ID** : TC10
  TYPE : Functional
  DESCRIPTION : tan(x) value test.
  INPUT : 225
  EXPECTED RESULT : 1

ACTUAL RESULT : 0.999993366047522
TEST RESULT : FAIL

- **TEST CASE ID** : TC11
  TYPE : Functional
  DESCRIPTION : tan(x) value test.
  INPUT : 315
  EXPECTED RESULT : -1
  ACTUAL RESULT : -1.0000092876074058
  TEST RESULT : FAIL

2.2.4 tan(x) Negative Values Testing Reports

- **TEST CASE ID** : TC12
  TYPE : Functional
  DESCRIPTION : tan(x) negative value test.
  INPUT : -64
  EXPECTED RESULT : -2.0502989318619114
  ACTUAL RESULT : -2.0502989318619114
  TEST RESULT : PASS

- **TEST CASE ID** : TC13
  TYPE : Functional
  DESCRIPTION : tan(x) negative value test.
  INPUT : -45
  EXPECTED RESULT : -1
  ACTUAL RESULT : -0.9999986732059835
  TEST RESULT : FAIL

- **TEST CASE ID** : TC14
  TYPE : Functional
  DESCRIPTION : tan(x) negative value test.
  INPUT : -180
  EXPECTED RESULT : 0
  ACTUAL RESULT : 0
  TEST RESULT : PASS

2.2.5 tan(x) Decimal Values Testing Reports

- **TEST CASE ID** : TC15
  TYPE : Functional

DESCRIPTION : tan(x) decimal value test.
INPUT : 66.8
EXPECTED RESULT : 2.333168462359474
ACTUAL RESULT : 2.333168462359474
TEST RESULT : PASS

- **TEST CASE ID** : TC16
  TYPE : Functional
  DESCRIPTION : tan(x) decimal value test.
  INPUT : 1111111111111111111111.99999999999999999
  EXPECTED RESULT : 0.62486935
  ACTUAL RESULT : Program Not Responding
  TEST RESULT : FAIL

- **TEST CASE ID** : TC17
  TYPE : Functional
  DESCRIPTION : tan(x) decimal value test.
  INPUT : 89.999999
  EXPECTED RESULT : 57295779.814
  ACTUAL RESULT : 743910.253181
  TEST RESULT : FAIL

2.3 Test Results Summary

Table 2: Test Summary

TEST CASE ID	**STATUS**	**IMPACT**
TC1	PASS	-
TC2	PASS	-
TC3	FAIL	Minor
TC4	PASS	-
TC5	PASS	-
TC6	FAIL	Minor
TC7	FAIL	Minor
TC8	PASS	-
TC9	PASS	-
TC10	FAIL	Major
TC11	FAIL	Major
TC12	PASS	-
TC13	FAIL	Major
TC14	PASS	-
TC15	PASS	-
TC16	FAIL	Major
TC17	FAIL	Major