SOEN 6011 PROJECT DELIVERY 2: CALCULATOR OF FUNCTION 3(CF3) APPLICATION

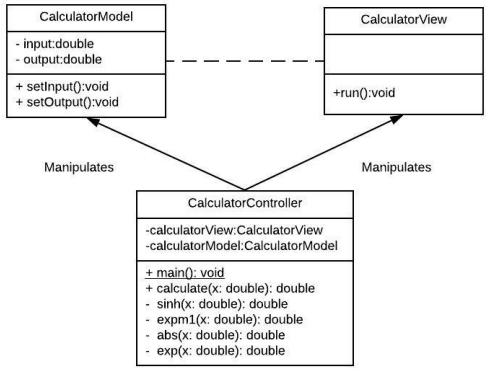
July 30, 2019

Xueying Li 40036265
https://github.com/raphealshirley/SOEN6011-PROJECT

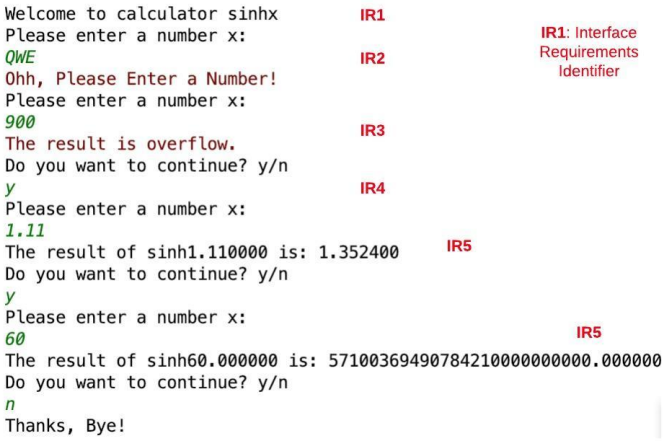
1 CF3 Application Implementation

1.1 User Interface

User Interface(UI) is designed as Text-Based User Interface(TUI). Model-View-Controller(MVC) design pattern is implemented to decrease coupling and increase cohesion[1]. a class diagram is shown as following(figure 1a).
As it is shown, `CalculatorController` class is the entrance of the calculator and `CalculatorController` not only manipulates `CalculatorModel` and `CalculatorView` but also provide calculation service to user while user will only interact with `CalculatorView`.
Figure 1b shows typical messages(Msgs) from TUI of the calculator application. Each Msg is related to a requirement regarding to UI. As it is shown in figure 1b, identifier is marked at the end of each msg.



(a) MVC Class Diagram of UI



(b) TUI Typical Msgs

Figure 1: UI Design Description and Implementation

1.2 Error Handling

Following table 1 gives errors and the explanations. All of this EXCEPTION/ERROR HANDLING ensures the input is validate which meats Functional Requirement: FR4.

Table 1: Errors/Exceptions and Descriptions

Errors/Exceptions	Description
Unexpected Input	Answer should be y/n when asked "Do you want to continue?"
Number Format	Input is not a number
Out of Bound	Output number(sinhx) is out of bound
Assertion Error	Absolute param n should less than 2048 when calculate 2^n

1.3 Debugger Description

Debugging is an essential technical process helps to detect and remove bugs. A debugger is a computer program that allows the programmer to control how a program executes and examine the program state while the program is running[2].

Traditional debugging technique is putting "System.out.print" to each logical segmentation to trace the output on the console. Other common debugging techniques are stepping, breakpoints and exceptions[3]. To debug the application, debuggers used during implementation are combination of various techniques including "traditional print", stepping, and breakpoints.

Considering about advantages such as simple execution and easy detection of "System.out.print", this traditional technique is used to debug several output format problems. However, it also has disadvantages of the complexity when it comes to debug statements with complicate logic, such as while or for loop.

Another debugger used in the implementation is stepping. Basically, Stepping is the name for a set of related debugger features that let us execute (step through) our code statement by statement.Related Stepping commands are Step Into, Step Over etc. Advantages of stepping is that it is more effective to step through code and inspect variables. However, it is still ineffective considering that it could not automatically generate logging, programmer have to inspect it step by step manually. It is even more irritating if the file is large and debugging has to begin from the first statement.

So, breakpoints are also used for debugging. Advantages of breakpoints are it allowed the execution stops at targeted statement. One of the disadvantages is that it usually have to be used with other techniques such as stepping.

Table 2 conclude advantages and disadvantages of the three debuggers used in CF3 Application implementation process.

Table 2: Advantages and Disadvantages of Debuggers

Debugger Name	Advantages	Disadvantages
System Print	Simple	Inefficient when logic is complex
Stepping	Efficient tracking Variables and stepping though statements	Always start from beginning
Breakpoints	Start at every required statement	Have to combine with others

1.4 Quality Assurance

Main concerned quality attributes are correctness, efficiency, maintainability, robustness and usability. In order to ensure those quality attributes, following design and implementation decisions are made.

1.4.1 Correctness

First, the problem had been defined completely as user input was validated each time and the range of output was defined as R covering all input domain. Second, algorithms had been proved to be correct during design phrase. Third, program was kept simple and clear to implement algorithms.

1.4.2 Efficiency

Binary is the primary language for computer. In calculation algorithms, Binary calculations were used, which ensured that calculation speed was very fast.All the calculations were within million seconds.

1.4.3 Robustness

First,Input validates ensured the input was valid not only referring to its format but also took its range into consideration, and input errors were all handled. Second, the MVC architecture is used. MVC separates concerns which make the architect is very robust. Moreover, all calculations were implemented

though binary calculation and binary is machine language, which ensures the robustness of the calculation.

1.4.4 Maintainability

First, the algorithms had been verified to be optimized with time complexity of O(1), before implementing. Second, design pattern such as model-view-controller was used to separate logic, view and data thus decreasing coupling and enhancing maintainability. Also, Same google java code style was used and checkstyle was used to ensure the code quality.

1.4.5 Usability

First, UI was designed as TUI and user just input a number to get result, which is simple to learn to use. Then, input validates and related error messages clearly showed input errors and the required input, which largely decreases the complexity of the usage of the software.

1.5 Checkstyle: Source code quality check

In this application, Checkstyle[5] is used for course code quality checking. Checktyle is a static code analysis tool used in software development for checking if Java source code complies with coding rules[5].Coding standards for checking is google coding standard according to Google Java Style Guide[6]. It generated a quality report with each warning showing the position and message of the warning. The final result is correct with result from *Checkstyle found no problems in the file(s).*

Using of checkstyle could ensure the same programming style in order to improve the quality, readability, re-usability, and maintainability. It could also decrease the cost of development and maintenance. However, the disadvantage is that it does not confirm the correctness and complement of the code.

2 Unit Testing

2.1 UI Test Cases

The tool used in CF3 Application is JUnit. Following gives UI test cases.

Test Case ID:	TIR1	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorView	Requirement ID:	IR1
Test Title:	Input number	Description:	Testing input massage
Pre-condition:	Program is executed.		
Post-condition:	User input is obtained.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1	Input message is shown and user input is obtained.		

Test Case ID:	TIR2	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorView	Requirement ID:	IR2
Test Title:	Not-a-number	Description:	Testing error message if input is not valid.
Pre-condition:	Input is obtained.		
Post-condition:	Input is validated.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a non-number	Error message is shown. And asked new input.		
2. Input a number	New input is validated.		

Test Case ID:	TIR3	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorView	Requirement ID:	IR3
Test Title:	out-of-bound	Description:	Testing error message if output is out-of-bound.
Pre-condition:	Input is obtained.		
Post-condition:	Output is validated.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a number (>700)	Overflow message is shown. And ask new input.		
2. Input a number	New input is validated.		

Test Case ID:	TIR4	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorView	Requirement ID:	IR4
Test Title:	Repeat Input	Description:	Testing new input message shown after error messages.
Pre-condition:	Error message was shown,		
Post-condition:	New input is obtained.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a large number (>700)	Overflow message is shown. And ask new input.		
2. Input a non-number	Error message is shown. And ask new input.		
3. Input a number	New input is validated.		

Test Case ID:	TIR5	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorView	Requirement ID:	IR5
Test Title:	Output the result	Description:	Testing output is shown.
Pre-condition:	Valid input is obtained.		
Post-condition:	Output is shown.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a valid number	Calculated result is shown.		

2.2 Functional Requirements Test Cases

Following shows the test cases for testing functional requirements. Related testing data is attached.

Test Case ID:	TFR1	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorController	Requirement ID:	FR1
Test Title:	Execute Application	Description:	Testing executation application.
Pre-condition:			
Post-condition:	Calculator is running.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. command line: java -jar <dir to file>	Calculator is running.		

Test Case ID:	TFR2	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorController	Requirement ID:	FR2
Test Title:	Input a number (input data: 0)	Description:	Testing user input
Pre-condition:	Calculator is running to ask an input.		
Post-condition:	Input is obtained.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a number	Input is obtained.		

Test Case ID:	TFR3	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorController	Requirement ID:	FR3
Test Title:	Obtain an output	Description:	Testing result output
Pre-condition:	Input is obtained.		
Post-condition:	Output is calculated.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a number(Input data: 0)	Input is obtained.		
2.	Result is obtained.		

Test Case ID:	TFR4	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorController	Requirement ID:	FR4
Test Title:	Input is validated	Description:	Testing input validating
Pre-condition:	Input is obtained.		
Post-condition:	Input is validated.		
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a number (input data: 0)	Input is validated.		

2.3 Performance Requirements

Fowling gives the test cases of performance requirements.

Test Case ID:	TQR1	Test Designed Date:	27.JUL.2019
Module Name:	All	Requirement ID:	QR1
Test Title:	Accuracy	Description:	Testing accuracy
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Input a number(Test data: data.txt)	Input is validated.		
2	Actual Result/Expected Result		

Test Case ID:	TQR2	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorController	Requirement ID:	QR2
Test Title:	Testability	Description:	Testing testability
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Run Junit	Junit could run test cases.		

Test Case ID:	TQR3	Test Designed Date:	27.JUL.2019
Module Name:	CalculatorController	Requirement ID:	QR3
Test Title:	System Reliability	Description:	Testing system reliability
Test Steps	Expected Result	Actual Result	Status(Pass/Fail)
1. Run test data with 100 input (Test data: data.txt).	100 results were obtained.		

2.4 Test data

Test data is attached with source code. Which are data.txt and result.txt. See Appendix.

References

[1] Robert M. Kline, *https://www.cs.wcupa.edu/rkline/java/mvc-design.html*

[2] wikipedia, Debugger *https://en.wikipedia.org/wiki/Debugger*

[3] Sanjeev Kumar Aggarwal and M. Sarath Kumar (2003). *"Debuggers for Programming Languages". In Y.N. Srikant and Priti Shankar (eds.).*

[4] tutorials point, Tutorials Point India Limited *https://www.tutorialspoint.com/software$_t$esting$_d$ictionary*

[5] checkstyle *https://checkstyle.sourceforge.io/*

[6] Google Java Style Guide *https://google.github.io/styleguide/javaguide.html*

Appendices

Test data - data.txt 274.55486212372483
365.3194167910182
979.6688667326436
597.6267579001179
173.239263168962
153.44906841845162
401.0176504769912
991.6029936397139
857.0807533516161
745.2457661496765
235.62051166692189
724.2714500039492
271.751230385772
368.06722386682986
459.55405710983064
753.5954018390726
682.8196955988969
663.6285224546323
621.8694572326466
622.0826113938813
946.1238251773175
165.77905927485014
557.5703589698621
480.7491108101446
808.9003571809282
750.6352515512824
428.9116155285292
87.22418540255904
297.0028089800897
509.172949689386
621.5379509075659
953.1978836861508
99.98005204710803
919.1409865753433
331.5531554627147
821.2952675950561
878.4847755181044
26.16516544234615
645.4301517028761
562.5944707606858
587.4262258785564

583.1702570559939
536.7422885350691
403.38843917822476
233.47662380220524
742.7544933201087
482.0173965124461
113.42297218710928
966.9865375824623
948.9098722932597
332.5853966434197
417.72661574326463
46.05919427879901
132.4884994364327
670.1118329198126
430.2805629278631
4.479186710013572
179.0737132587925
120.57830400579716
350.1598527865064
305.3868993315155
908.4221683800099
4.793152723275851
9.409635831164053
869.0342469804397
481.4007909944984
263.08416956827364
116.80325515547251
125.65382705890981
233.35078942136067
807.6465118449639
202.7617800656837
663.4176936909864
422.0680567261703
328.08924157470045
338.0971710948932
116.65835069737629
274.09781689965604
104.04373719654726
672.8745837354384
670.9691608668392
14.798692675352942
563.2677793045857
721.5711384147037
449.4969237780656
275.639180693397
164.4997085157749
106.84284926885523
88.68475085235971
386.44777856852994
681.3870483792888
166.5924728820577
11.918564395755006
580.8452879855956
117.13415288440076
427.66520295365706
265.97595454939824
69.84601067279739

289.0842087328328
454.381256932028
Related result-result.txt 8.642345102326037E118
2.2655666797980295E158
Infinity
1.757815129364873E259
8.626329593827038E74
2.1930759467666924E66
7.223085444208477E173
Infinity
Infinity
Infinity
1.0657566335748682E102
Infinity
5.236363793481831E117
3.536179475794933E159
1.9088031516109936E199
Infinity
1.7530565985528105E296
8.11288537522069E287
5.935314614538213E269
7.345399492906739E269
Infinity
4.964787473830235E71
7.058301047652903E241
3.05953910954076E208
Infinity
Infinity
9.395455773829308E185
3.8014774021760717E37
4.848987383736382E128
6.760399933978215E220
4.260615833159146E269
Infinity
1.3175129990075787E43
Infinity
4.905416407974919E143
Infinity
Infinity
1.1544017435772688E11
1.0132657029200254E280
1.0731099086516932E244
6.530379627494191E254
9.259651759027757E252
6.356003376410119E232
7.7329144044686965E174
1.2490487095807301E101
Infinity
1.0875907144963274E209
9.076970965361688E48
Infinity
Infinity
1.3771223545992385E144
1.3041698795950901E181
5.037602786598594E19
1.72979335813754E57
5.306904831131746E290

3.693551459766589E186
44.075799419370775
2.949140929760305 5E77
1.1626850045855314E52
5.908290675884789E151
2.1223413204840485E132
Infinity
60.336478901474564
6102.712624716608
Infinity
5.870527368856112E208
9.015153461228371E113
2.666730859968787E50
1.860945806629089E54
1.1013622783082121E101
Infinity
5.718633021878132E87
6.570729235824939E287
1.0018392169885769E183
1.5357382176546646E142
3.4096182579126503E146
2.3070020484379927E50
5.471915542949874E118
7.66639064854543E44
8.407923151537874E291
1.2507603232650016E291
1336474.1793791542
2.1040613085142367E244
Infinity
8.184716360596138E194
2.5559077557021028E119
1.3812926071832983E71
1.2595919389944752E46
1.6378266311263465E38
3.397095690713508E167
4.184129868031408E295
1.1198547685107727E72
75013.03783051597
9.054686450653105E251
3.71266910826248E50
2.7015172758686136E185
1.625022520610729E115
1.0782191041272566E30
1.7646015743967076E125
1.0820381424888303E197