# Testing summary page

1.Introduction

This document is used to record the test process, summarize the test conditions of each round, analyze the test data, summarize the problems exposed during the test work and the remaining risks, and give corresponding test recommendations.

1.1 Test scope

The test scope includes the software core logic part, user interface and the image part. And in this test summary we mainly focus on the reliability, robustness, availability and security of this system.

1.2 Testing standard

1. The whole testing process automated integration testing, integration testing and system testing.

2. The Junit testing coverage more than 80%

3. Automate testing with well-designed test cases.

2.Testing method

2.1 Unit testing

Unit testing is the inspection and verification of the smallest testable unit in the software. In our java program, the unit means a class. A unit is the smallest measured functional module specified by humans. Unit testing is the lowest level of testing activity to be performed during the software development process, and the individual units of the software are tested in isolation from the rest of the program.

|  |  |  |
| --- | --- | --- |
| **Unit name** | **Test description** | **Test case** |
| add | Verify addition function | 2+2 |
| sub | Verify sub function | 2-2 |
| multiplication | Verify multiplication  function | 2\*2 |
| dev | Verify divide function | 2/2 |
| Dev-dev | Verify the function of continuous division | 44/2/2 |
| Sub-sub | Verify the function of continuous division | 22-2-10 |
| Log1 | Verify the function of continuous division | Log(10)(100) |
| Log2 | Test the logarithm of base 1 | Log(1)(100) |
| Dev0 | Test a number divided by 0 | 1/0 |
| Sin1 | Verify the function of sin | sin(90) |
| Sin2 | Verify the function of sin | Sin(0) |
| Cos1 | Verify the function of cos | Cos(90) |
| Cos2 | Verify the function of cos | Cos(90) |
| Fac1 | Verify the function of  factorial | Fac(0) |
| Fac2 | Verify the function of  factorial | Fac(5) |
| Exp1 | Verify the function of  exponential operation | 4^(1/2) |
| Exp2 | Verify the function of  exponential operation | 4^(0) |
| Exp3 | Verify the function of special  exponential operation | 0^(0) |
| Exp4 | Verify the function of  exponential operation | 4^(2) |
| Tan1 | Verify the function of  tan | tan(0) |
| Tan2 | Verify the function of  tan | tan(90) |
| fun | Verify the function of  Function expression | f(x) = 50 \* SIN(x\_1) |
| show | Verify the function of  The show method | ((((((((((1.0+2.0)-(3.0\*5.0))+(6.0-2.0))  +(cos(2.0)))+(8.0)!)-(sin(2.0)))-(tan(2.0)))  +(2.0^(2.0)))+(3.0/2.0))+(log(10.0)(100.0))) |
| Sin(x) | Verify the function of sin(x) | 1+2-3\*5+(6-2)+cos(2)+fac(8)-sin(2)-tan(2)+2^2 + 3/2 + log(10)(100) |
| Sin(x)2 | Verify the function of sin(x) | f(x)=sin(x\_1) |
| Get result | Verify the function of result | 9 + 1 |
| check | Verify the check function | 9+1 |
| check2 | Verify the check2 function | F(x)=546 |
| Array1 | Verify the function of array (float) | f(x) = x\_1 |
| Array2 | Verify the function of array (double) | f(x) = x\_1 |
| Save and load | Verify the function of save and load | “1” “2” “3” “4” “5” |

**2.2 Integration Testing**

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher levels components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. (Integration testing, 2019.11)

|  |  |  |
| --- | --- | --- |
| Integration parts | Test description | Result |
| Parser part and tokenizer part | Testing the lower level integrated modules | 1 bug, has fixed |
| Parser part and tokenizer part and draw part | The lower level modules integrated with the medium level modules | 2 bugs, has fixed |
| Parser part , tokenizer part , draw part and UI part | Integrated all modules | 1 bug, has fixed |

2.3 System testing

System testing is the test of the whole system. It means that the system will be tested with computer hardware, peripherals, certain supporting software, data and personnel, and other system elements and environments .The purpose of system testing is to find out where the software does not conform to or contradict the system definition by comparing it to the system's requirements definition.

|  |  |  |
| --- | --- | --- |
| Test number | Test action description | Test result |
| 1. | Press the del and ac button | Ac can delete all but when the string is null and you still del, the system break down |
| 2. | Press number button | display numbers properly |
| 3. | Press scientific constant button | Pi is not consist with the Pi in java.math |
| 4. | Press calculation symbol button | Operation normally |
| 5. | Get result | Operation normally |
| 6. | Input function expression | Operation normally |
| 7. | Save and load history | Operation normally |

3. Testing result

3.1 bug list and fixed history

|  |  |  |  |
| --- | --- | --- | --- |
| Bug number | Description | Find time | Fixed time |
| 1. | Fac (0.1) and fac (0) | 2019.10.2 | 2019.10.3 |
| 2. | -(-1) can’t be understand by the parser | 2019.10.3 | 2019.10.3 |
| 3. | Wrong result of parser | 2019.10.5 | 2019.10.7 |
| 4. | Typo error | 2019.10.5 | 2019.10.7 |
| 5. | Input f(x), the parser will crash | 2019.10.7 | 2019.10.8 |
| 6. | Input is empty error | 2019.10.7 | 2019.10.8 |
| 7. | Mismatch with draw | 2019.10.7 | 2019.10.8 |
| 8. | Tan (90) error | 2019.10.8 | 2019.10.11 |
| 9. | Keyboard pop bug | 2019.10.9 | 2019.10.11 |
| 10. | Edit bug | 2019.10.10 | 2019.10.11 |
| 11. | Del bug | 2019.10.11 | 2019.10.13 |
| 12 | Save bug | 2019.10.13 | 2019.10.13 |

3.2 coverage percentage

|  |  |  |
| --- | --- | --- |
| ClassName | Method Coverage | Line Coverage |
| FileUtil | 100% | 92% |
| AddExp | 100% | 100% |
| CalculateCore | 100% | 90% |
| CosExp | 100% | 100% |
| DivExp | 100% | 100% |
| Exp | 100% | 100% |
| ExpExp | 100% | 100% |
| FacExp | 100% | 93% |
| FunctionExp | 100% | 92% |
| FunctionParser | 100% | 100% |
| LogExp | 100% | 100% |
| MulExp | 100% | 100% |
| NumExp | 100% | 100% |
| Parser | 88% | 89% |
| SinExp | 100% | 100% |
| SubExp | 100% | 100% |
| TanExp | 100% | 100% |
| Token | 80% | 90% |
| VarToken | 100% | 100% |
| Tokenizer | 85% | 88% |
| TrigonometricExp | 100% | 100% |
| VarExp | 100% | 100% |

4. summary

Through testing we discovered the vulnerabilities in the code as well as some problem about user interaction issues. We have solved these problems and improved our code structure. The testing also helps us to ensure that our code and project requirements are consistent. After testing and code corrections, we believe that this project has met all user needs and has well user interactions.

5.Reference

1.Integration testing, Retrieved October 16, 2019,

from <https://en.wikipedia.org/wiki/Integration_testing>