

Problem 1 - SOEN 6011

Pavit Srivatsan - 40155323

July 25, 2021

F7: Power Function

0.1 Introduction

A power function or exponentiation is of the form:

$$f(x, y) = x^y$$

where x and y are real numbers.

0.2 Domain and Codomain

1. When y is a non-negative integer, the domain is all real numbers: $(-\infty, \infty)$
2. When y is a negative integer, the domain is all real numbers excluding zero $(-\infty, 0) \cup (0, \infty)$
3. When y is an irrational number and $y > 0$, the domain is all non-negative real numbers and when y is an irrational number and $y < 0$, the domain is all positive real numbers
4. The codomain of the function is $[-\infty, \infty]$ and can be indeterminate

0.3 Characteristics of Power Function.

1. The behaviour of the power function depends on whether the value of y is a positive or a negative and an odd or an even number
2. Also, the power function outputs differently for negative and positive fractional powers

References

- [1] "*Exponentiation*" https://en.wikipedia.org/wiki/Exponentiation#Power_functions
- [2] "*Fractional Exponents*" <https://www.mathsisfun.com/algebra/exponent-fractional.html>

Problem 2

The assumptions and requirements for the function

$$x^y$$

as per ISO/IEC/IEEE 29148 standards.

2.1 Assumptions

- Assumption 1
 - ID: ASSUMP1
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 1
 - Difficulty: Easy
 - DESC: fractional inputs are entered as double values
 - Rationale: when input base or exponent value equals 2/3, it must be expressed as 0.67
- Assumption 2
 - ID: ASSUMP2
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 1
 - Difficulty: medium
 - DESC: The output of larger values of exponent and base are expressed in terms of exponents
 - Rationale: when input is 100 raised to 100, the output is expressed as 1.9047931533522278E25
- Assumption 3
 - ID: ASSUMP3
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 2

- Difficulty: High
- DESC: Users enter whole numbers and rational numbers
- Rationale: Irrational numbers are not handled by the code. For Example: π , $\sqrt{2}$
- Assumption 4
 - ID: ASSUMP4
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 3
 - Difficulty: Easy
 - DESC: Mathematical symbols such as infinity, indeterminate are represented in words
 - Rationale: Symbols such ∞ are represented in words - infinity

2.2 Functional Requirements

- Requirement 1
 - ID: FUNR1
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 1
 - Difficulty: Easy
 - DESC: The arguments passed to the function x power y shall be valued within $-\infty$ to $+\infty$ and fractions are expressed as double values.
 - Rationale: when $x = 2.2$, $y = 4.45$ where x expressed in radians.
- Requirement 2
 - ID: FUNR2
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 2
 - Difficulty: Easy

- DESC: The function shall return the value 1 when any base value is raised to the power zero
 - Rationale: 10 raised to the power 0 returns 1
- Requirement 3
 - ID: FUNR3
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 2
 - Difficulty: Easy
 - DESC: The function shall return zero when base value zero is raised to any exponent value
 - Rationale: 0 raised to the power 10 returns 0
- Requirement 4
 - ID: FUNR4
 - Version: 1.0
 - Type: functional
 - Owner: Pavit Srivatsan
 - PRIORITY: 2
 - Difficulty: Medium
 - DESC: The function shall accept only numerical values as specified in the domain
 - Rationale: string values, numbers with special characters, special characters are not allowed as inputs and an appropriate error message is displayed

Problem 3 - SOEN 6011

Pavit Srivatsan

August 2021

Pseudocode and Algorithm

Calculate: $f(x, y) = x^y$

ALGORITHM 1: Iterative algorithm to calculate x^y

```
1. function power_function_iterative(x,y)
in:  double number x,y
out: double number result
2.  $result \leftarrow 1$ 
3.  $temp \leftarrow 1$ 
4. for  $temp \leq y$  do
5.    $result \leftarrow result * x$ 
6.    $temp \leftarrow temp + 1$ 
7. end for
8. return result
```

The output is stored in result, which is initially set to 1. It is then looped from 1 to y, x number of times, incremented by one on each iteration and on each iteration we multiply result by x. At the end of the loop value of result is equal to x^y .

ALGORITHM 2: Recursive Divide and Conquer algorithm to calculate x^y

```
1. function power_function_recursive(x,y)
in: double number x,y
out: double number result
3.  $power \leftarrow exponent\_helper(x, y)$ 
4.  $result = power$ 
5. return result
```

```
1. function exponent_helper(x,y)
in: double number x, y
out: double number sum
2. if  $x < 0$  then
3.    $x \leftarrow 1.0/x$ 
4.    $y \leftarrow -y$ 
5.   return  $exponent\_helper(x, y)$ 
6. else if  $y = 0$  then
7.   return 1.0
8. else if  $y = 1$  then
9.   return x
10. else if  $y \bmod 2 = 0$  then
11.    $y \leftarrow y/2$ 
12.    $y \leftarrow y/2$ 
13.   return  $exponent\_helper(x, y)$ 
14. else
15.    $x \leftarrow x * x$ 
16.    $y \leftarrow y - 1$ 
17.    $y \leftarrow y/2$ 
18.   return  $exponent\_helper(x, y)$ 
19. end if
```

A helper function called `exponent_helper` is defined which calculates x^y . In the base case when $y = 0$, we return 1, otherwise when $y = 1$ we return x . When x is even we recurse on $x = x * x$ and $y = y/2$. In case when x is odd we recurse on $x = x * x$ and $y = (y - 1)/2$. In the end, in our main function `power_function_recursive` we multiply the result of `exponent_helper` to the value of a and return our result.

Advantages and Disadvantages

Algorithm 1:

Advantages:

1. In terms of space complexity, iterative algorithms don't suffer from stack overflow because all operations are done on the heap.
2. They are easy to comprehend by humans and have better readability.

Disadvantage:

1. The time complexity of the iterative algorithm is $O(n)$, hence it is not very efficient for larger inputs in terms of time.
2. Proper terminating condition for loop is important or else it might lead to infinite looping.

Algorithm 2:

Advantages:

1. The time complexity of the recursive algorithm is $O(\log n)$. This recursive algorithm is optimized (tail recursive) so that we don't get stack overflow error and it handles large inputs better.
2. Recursion has higher maintainability than looping. Handling the base case properly requires little or no modifications.

Disadvantages:

1. Due to the continuous allocation of memory space leading to a stack overflow, efficiency is significantly affected.
2. It is difficult to comprehend. A certain level of expertise is required for understanding it vividly.

Conclusion

Recursive algorithm which is based on Divide and Conquer Algorithmic Strategy is preferred over iterative algorithm

References

- [1] TutorialsPoint,
https://www.tutorialspoint.com/java/lang/math_pow.htm
- [2] GeeksforGeeks,
<https://www.geeksforgeeks.org/write-a-c-program-to-calculate-powxn/>
- [3] MathBitsNotebook,
<https://mathbitsnotebook.com/Algebra1/FunctionGraphs/FNGTypeExponential.html>

SOEN6011- P4

Pavit Srivatsan
40155323

August 2021

1 Debugging

Debugging is the process of detecting and removing of existing and potential errors in a software code that can cause it to behave unexpectedly or crash.

1.1 Debugging the program in Eclipse IDE

A.Setting up breakpoints

Suitable breakpoints are chosen. We must be careful in choosing breakpoints to understand the program flow. There are certain cases in which breakpoints can be skipped based on the program logic.

B.Debug Configurations

A Java program can be debugged simply by right clicking on the Java editor class file from Package explorer. Select Debug As → Java Application or use the shortcut Alt + Shift + D, J instead.

C.Debug perspective

In debug perspective we can skip breakpoints, step into, step over or resume statement executions. We can modify variable values in the variables tab in debug perspective.

1.2 Advantages

- we understand program flow and logic better
- Identify errors and bugs better instead of printing variable values
- Modify program to be more efficient

2 Quality Attributes

2.1 Correctness

The values are more precise. Since the calculator is scientific in nature, correct values are mandatory. Results are accurate as possible.

2.2 Efficiency

The algorithm chosen is Divide and Conquer Algorithm which is more efficient than Iterative algorithm. The time complexity of this algorithm ($O(\log(n))$) is better than Iterative ($O(n)$).

2.3 Maintainability

The function is developed from scratch and hence it requires helper functions. These functions are placed appropriately. Necessary comments are added to make it more maintainable.

2.4 Robustness

All relevant errors are added so that users are directed to enter proper inputs. It handles various errors and exception.

2.5 Usability

The program is run in a menu driven model. Menu driven model is easy to use and comprehend.

3 CheckStyle

Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task. This makes it ideal for projects that want to enforce a coding standard [1].

3.1 Features

Checkstyle can check many aspects of your source code. It can find class design problems, method design problems. It also has the ability to check code layout and formatting issues[1].

3.2 Advantages

- portable between IDEs. If you decide to use IntelliJ later, or you have a team using a variety of IDEs, you still have a way to enforce consistency [3].

- better external tooling. It's much easier to integrate checkstyle with your external tools since it was really designed as a standalone framework. You can plug into your SCM as a pre-commit hook, or into your build tool, quite easily. Using Eclipse style convention you would need to write or locate a plugin to do the same thing [3].
- ability of creating your own rules. Eclipse defines a large set of styles, but checkstyle has more, and you can add your own custom rules [3].

3.3 Limitations

There are basically only a few limits for Checkstyle:

- Java tokens (identifiers, keywords) should be written with ASCII characters ONLY, no Unicode escape support in keywords and no Unicode support in identifiers.[1]
- To get valid violations, code have to be compilable, in other case you can get not easy to understand parse errors.[1]
- You cannot determine the type of an expression. Example: "getValue() + getValue2()".[1]
- You cannot determine the full inheritance hierarchy of type.[1]
- You cannot see the content of other files. You have content of one file only during all Checks execution. All files are processed one by one. [1]

References

- [1] Checkstyle,
<https://checkstyle.sourceforge.io/index.html>
- [2] Debugging,
https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php
- [3] Stackoverflow,
<https://stackoverflow.com/questions/13644624/advantage-of-using-checkstyle-rather-than-using-eclipse-built-in-code-formatter>

P5 - Source Code Review

Pavit Srivatsasn

August 2021

1 Introduction

The source code review for function F2: (tanx) is performed using the tool SonarLint. SonarLint is an IDE extension that helps you detect and fix quality issues as you write code. Like a spell checker, SonarLint squiggles flaws so that they can be fixed before committing code.

2 SonarLint Features

SonarLint highlights code issues with markers on open files. It also provides an issues summary table for a selected component in the IDE, including the creation time of the issue.

2.1 Code Reliability

Catch tricky bugs to prevent undefined behaviour from impacting end-users.

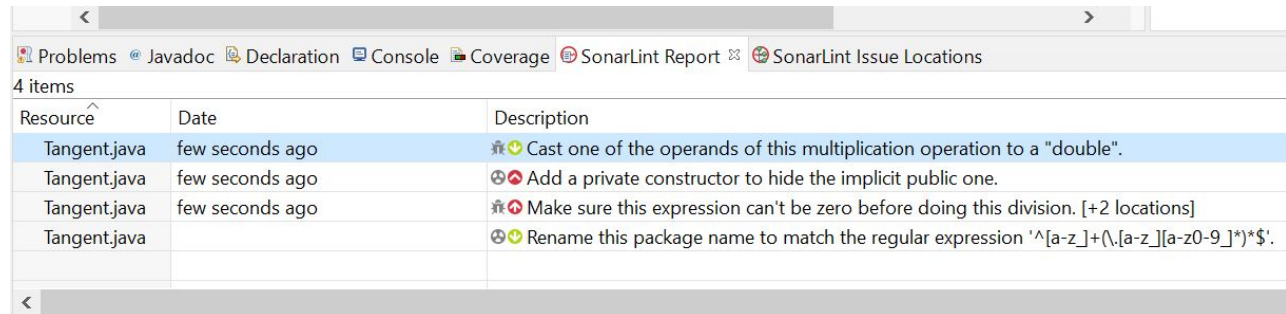
2.2 Application Security

Fix vulnerabilities that compromise the app, and learn AppSec along the way with Security Hotspots.

2.3 Technical Debt

Make sure your codebase is clean and maintainable, to increase developer velocity!

3 SonarLint Result



4 items		
Resource	Date	Description
Tangent.java	few seconds ago	Cast one of the operands of this multiplication operation to a "double".
Tangent.java	few seconds ago	Add a private constructor to hide the implicit public one.
Tangent.java	few seconds ago	Make sure this expression can't be zero before doing this division. [+2 locations]
Tangent.java		Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*\$'.

Figure 1: Sonar Lint Result for Tangent Function F2

Google Checkstyle is used for implementation.

3.1 Review Result 1

Cast one of the operands of this multiplication operation to a "double".

Code:

```
number = wholePart * 10000000; [line : 46]
```

Description:

When arithmetic is performed on integers, the result will always be an integer. You can assign that result to a long, double, or float with automatic type conversion, but having started as an int or long, the result will likely not be what you expect. For instance, if the result of int division is assigned to a floating-point variable, precision will have been lost before the assignment. Likewise, if the result of multiplication is assigned to a long, it may have already overflowed before the assignment. In either case, the result will not be what was expected. Instead, at least one operand should be cast or promoted to the final type before the operation takes place.

3.2 Review Result 2

Add a private constructor to hide the implicit public one.

Description:

Utility classes, which are collections of static members, are not meant to be instantiated. Even abstract utility classes, which can be extended, should not have public constructors. Java adds an implicit public constructor to every class which does not define at least one explicitly. Hence, at least one non-public constructor should be defined.

3.3 Review Result 3

Make sure this expression can't be zero before doing this division. [+2 locations]

.

Code:

```
double result = calculateSin(angle) / calculateCos(angle); [line : 159]
```

Description:

If the denominator to a division or modulo operation is zero it would result in a fatal error. When working with double or float, no fatal error will be raised, but it will lead to unusual result and should be avoided anyway. This rule supports primitive int, long, double, float as well as BigDecimal and BigInteger.

3.4 Review Result 4 (enviornment - ignored)

Rename this package name to match the regular expression. .

Problem 6 Unit Testing

Pavit Srivatsan
40155323

1 General Guidelines

- Test cases are assigned ID for better tracking and readability
- Test cases should be small and precise.
- Each test case is followed by one line statement justifying its description.
- Each test case contains the function name used in the unit test case.
- Test case shall cover all the cases expected.
- Test Case should mention an expected value and actual value.
- Delta values are added for functions dealing with double values.
- Test cases should be self-contained.
- Test case should map to the functional requirement and assumptions.

2 Unit Test case Implementation

- JUnit Testing Framework is used
- Sample Implementation:

ID : TC1

Test Case : description about the test case in JavaDoc format

Association : Associated assumption or Functional Requirement

References

- [1] Unit Testing ,
<https://phauer.com/2019/modern-best-practices-testing-java/>

Problem 7: Testing

Pavit Srivatsan
Student ID: 40155323

August 2021

Testing of Function F4: Γ

1 Introduction

The function provided for the testing purpose is Γ gamma function. The team member has provided all the required documents to perform the testing of a function. Two tests are performed:

- A. Manual Testing
- B. Running Unit test cases

1.1 Test Step

1. The source code is imported into Eclipse and compiled.
2. **Manual Testing:** The program is run for different inputs based on requirements and results were verified manually by entering inputs by running the program code.
3. **Unit Testing:** The test program is run and test results are verified.

1.2 Test Results: Manual Testing

Test Case	Description	Result
TC1	Input cannot be zero	PASS
TC2	Input cannot be a negative integer	PASS
TC3	A valid positive integer	PASS
TC4	Input is not a half-integer value	PASS
TC5	Input is half-integer value	PASS

1.3 Test Results: Unit Testing

Test Case	Description	Result
TC1	Input cannot be zero	PASS
TC2	Input cannot be a negative integer	PASS
TC3	A valid positive integer	PASS
TC4	Input is not a half-integer value	PASS
TC5	Input is half-integer value	PASS

1.4 Conclusion

It is evident from the above tables that the program is bug free. The comments provided adequate information for the tester. The test cases have passed in both unit testing and manual testing.