SOEN 6011 Project Function 8 Beta(x, y)

Anqi Wang 40057695 Github Repo: https://github.com/AnqiAngelineWang/SOEN6011

Software Engineering, Concordia University, Montreal, Canada

Introduction and Description

- Beta function B(x,y)is the incomplete gamma functions $\gamma(a, z)$ and $\Gamma(a, z)$. It is one of the important meaningful mathematic functions. Generally speaking, Beta function is related with Euler Integral and is the first kind. It can be considered as the incomplete beta functions. Beta function has the general form:
  $B(x, y) = \Gamma(x)\Gamma(y)/\Gamma(x + y)$
  $B(x, y) = \int_0^\infty t^{x-1} \times (1 - t)^{y-1} dt$
  - Domain: x,y $\subset (0, +\infty)$ for all real value, greater than 0.
    Co-domain: the solution generated by B(x,y) , satisfy with x,y $\subset (0, +\infty)$
- It has the shape:

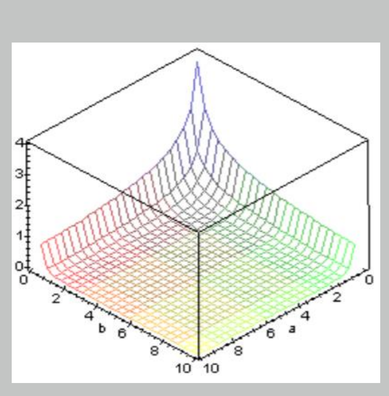
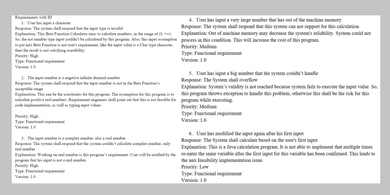


Figure 1. Graph of the Beta Function

Functional and Non-functional Requirement

- These functional and non-functional requirements followed ISO/IEC/IEEE 29148 Standard. Their individual rationale with detailed explanation are in the report.
- Functional Requirement



Requirements with ID
1. User has input a character
Response: The system shall respond that the input type is invalid
Explanation: This Beta Function Calculator aims to calculate numbers in the range of (1, +∞). So, the not number type input couldn't be calculated by this program. Also, this input assumption is put into Beta Function is user-user's requirement, like the input value is a Char type character, then the result is not satisfying readability.
Priority: High
Type: Functional requirement
Version: 1.0

2. The input number is a negative infinite decimal number
Response: The system shall respond that the input number is not in the Beta Function's acceptable range.
Explanation: This can be the constraint for this program. The assumption for this program is to calculate positive real numbers. Requirement engineers shall point out that this is not flexible for code implementation, as well as typing input values.

Priority: High.
Type: Functional requirement
Version: 1.0

3. The input number is a complex number, also a real number
Response: The system shall respond that the system couldn't calculate complex number, only real number
Explanation: Working on real number is this program's requirement. User will be notified by the program that his input is not a real number.
Priority: Low
Type: Functional requirement
Version: 1.0

4. User has input a very large number that has out of the machine memory
Response: The system shall respond that this system can not support for this calculation
Explanation: Out of machine memory may decrease the system's reliability. System could not process in this condition. This will increase the cost of this program.
Priority: Medium
Type: Functional requirement
Version: 1.0

5. User has input a big number that the system couldn't handle
Response: The System shall overflow
Explanation: System's validity is not reached because system fails to execute the input value. So, this program throws exception to handle this problem, otherwise this shall be the risk for this program while executing
Priority: Medium
Type: Functional requirement
Version: 1.0

6. User has modified the input again after his first input
Response: The System shall calculate based on the user's first input
Explanation: This is a Java calculation program. It is not able to implement that multiple times re-enter the same variable after the first input for this variable has been confirmed. This leads to the unit flexibility implementation issue.
Priority: Low
Type: Functional requirement
Version: 1.0

- Non-functional Requirement

Requirements with ID: 1. The system has maintainability	
Response	The system can be maintained in the future
Priority	High
Type	Non-Functional requirement
Version	1.0

Requirements with ID: 2. The system has sustainability	
Response	The system can be reused in the future
Priority	High
Type	Non-Functional requirement
Version	1.0

Algorithm Selection

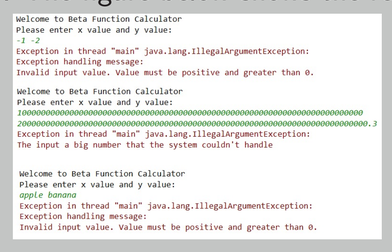
- Algorithm Optional 1
  - Explanation: This algorithm works based on approximation values in the array to estimate beta results.
  - Advantages: Easy to understand the logistic behind.
    Disadvantages: This algorithm can only obtain accurate results for integer inputs, not for decimal inputs. The result has large uncertainties.
- Algorithm Optional 2
  - Explanation:This algorithm generates beta results based on mathematic models.
  - Advantages: The code structure is clear and easy to implement.
    Disadvantages: This algorithm needs to use built-in functions for mathematical processing, like Math.floor, or Math.exp.
- Algorithm Optional 3
  - Explanation:This algorithm calculate Beta solutions based on input value ranges. It has accuracy for almost 15 digits after decimal point.
  - Advantages: It has high accuracy and low errors in results. It can handle both integers and decimals inputs.
    Disadvantage: It needs lightly longer processing time.
- Algorithm Optional 3 has been selected for the implementation.

Reference

- Stewart, J. (2008). Transcendental Functions [Abstract]. Calculus, 6, 71-73. Retrieved August 12, 2019.
- ISO/IEC/IEEE 29148:2018. Systems and Software Engineering – Life Cycle Processes – Requirements Engineering. 2019. ISO, IEC, IEEE. (P.9-P.16)
- Olver, F. W., Lozier, D. W., Boisvert, R. F., Clark, C. W. (Eds.). (2010). NIST handbook of mathematical functions hardback and CD-ROM. Cambridge university press.

Implementation

- This source code has followed the standard Google Java Style Guide, which is corresponding to the whole team's program style. The team has confirmed that source code file structure is correspondence.
- Correctness and Efficiency: For this program, the maximum length allowance number is followed double data type 1.79E308. Double datatype is a primitive datatype and it does not require much system memories.
- Maintainability and Program Style: For this program, coding convention has been regulated. corresponding team's program style has kept with the same format, block indentation and statement has been regulated. The code follows line wrapping and line break strictly. There is no extra whites- pace. Exactly each section has one blank line to separate.
- Debugger: Jetbrains InteliJ Idea IDE build-in debugger is the major tool.
- Checkstyle: This program uses Checkstyle devel- opment tool during software implementation.
- Error handling and User Interface: This program has a clear user interface. It is straight forward for user to un- derstand the logic and instructions. This program also kindly reminds user if he has input a valid value. If error occurs, error handling exceptions shall be invoked. Try and catch blocks keep the program functioning, and exceptions have thrown error messages to remind user. The figure below shows the result.



```
Welcome to Beta Function Calculator
Please enter x value and y value:
-1 -2
Exception in thread "main" java.lang.IllegalArgumentException:
Exception handling message:
Invalid input value. Value must be positive and greater than 0.

Welcome to Beta Function Calculator
Please enter x value and y value:
1000000000000000000000000000000000000000000000000000000000000000000000000
2000000000000000000000000000000000000000000000000000000000000000000000000.3
Exception in thread "main" java.lang.IllegalArgumentException:
The input a big number that the system couldn't handle


Welcome to Beta Function Calculator
Please enter x value and y value:
apple banana
Exception in thread "main" java.lang.IllegalArgumentException:
Exception handling message:
Invalid input value. Value must be positive and greater than 0.
```

Figure 2. User Interface with Exception Handling, Error Messages

- The above run cases shows that functional requirements has been processed, especially for boundary check.

Unit Testing

- This program introduces Junit assert .java file (AssertTests.java). The test cases have satisfied client's requirement, and matches with user assumptions. The program has passed all test cases. Test cases ID and user (assumptions) requirements ID have matched, and explain below:

Figure 3. Each test case contains requirement

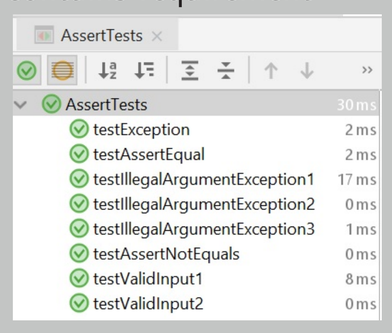


Figure 4. All test cases have passed

- Software testing is believed to be useful to check program accuracy, satisfy requirements, and increase software robustness.

Conclusion and Reflects

- Test cases almost covered majority of source code.

Check List	Result	Comments
Integrity Checking	Good	The source code has been integrated
Consistency Check	Bad	Format problems occurs
Correctness Checking	Good	All coments and variables have defined correctly
Modifiable Check	Good	The source code satisfied the requirement
Predictability Check	Good	The source code doesn't have errors
Robustness Check	Good	The source code has handling functions
Readability Check	Okay	There is one line of code not following coding standard
Verifiability check	Good	The code has no redundant functions

Figure 5. Team member code review result

- Critical decisions: When I decide which algorithm to choose, it is the critical decision. There are algorithms are accurate, but not satisfying the project requirement, like it needs to use math packages to simulate integral calculation. Or, the algorithm is not precise for this project.
- Lesson learnt myself: I need to make clear for functional and non-functional requirement. And test cases need to match with requirement. Also, test cases had better cover most of the codes, which will be more persuaded to demonstrate the program's testability.
- Lesson learnt from team member: I need to pay attention in coding format. The format I followed must be correct and accurate, especially, no extra spaces in the source code. Also, I need to pay attention to naming convention. This is important for readability. While coding, line length had better not exceed 100 characters. Last but not least, writing exception handling error message should be enhanced. This will provide user easier time to understand the problem happened in the code.