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:

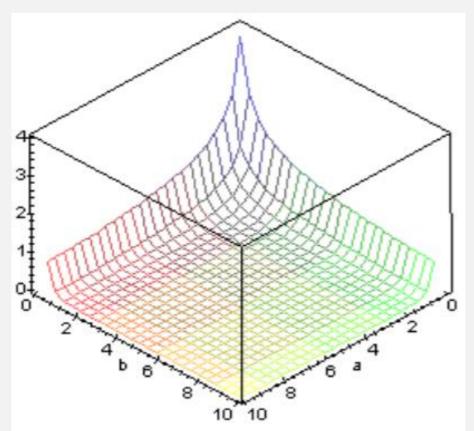


Figure 1. Graph of the Beta Function

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	A Maria in the second and the second
User has input a character	User has input a very large number that has out of the machine memory
Response: The system shall respond that the input type is invalid	Response: The system shall respond that this system can not support for this calculation
Explanation: This Beta Function Calculator aims to calculate numbers, in the range of (0, +∞).	Explanation: Out of machine memory may decrease the system's reliability. System could it
So, the not number type input couldn't be calculated by this program. Also, this input assumption	process in this condition. This will increase the cost of this program.
to put into Beta Function is not user's requirement, like the input value is a Char type character,	Priority: Medium
then the result is not satisfying userability.	Type: Functional requirement
Priority: High	**
Type: Functional requirement	Version: 1.0
Version: 1.0	
	User has input a big number that the system couldn't handle
 The input number is a negative infinite decimal number 	Response: The System shall overflow
Response: The system shall respond that the input number is not in the Beta Function's	Explanation: System's validity is not reached because system fails to execute the input val-
acceptable range	
Explanation: This can be the constraints for this program. The assumption for this program is to	this program throws exception to handle this problem, otherwise this shall be the risk for the
calculate positive real numbers. Requirement engineers shall point out that this is not feasible for	Fr-S
code implementation, as well as typing input values.	Priority: Medium
Priority: High	Type: Functional requirement
Type: Functional requirement	Version: 1.0
Version: 1.0	
Version: 1.0	6. User has modified the input again after his first input
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	Response: The System shall calculate based on the user's first input
real number	Explanation: This is a Java calculation program. It is not able to implement that multiple tin
Explanation: Working on real number is this program's requirement. User will be notified by the	re-enter the same variable after the first input for this variable has been confirmed. This lea
program that his input is not a real number.	the anti feasibility implementation issue.
Priority: High	Priority: Low
Type: Functional requirement	Type: Functional requirement
Version: 1.0	rype. runcuonar requirement

Figure 2. Functional Requirement List Non-functional Requirement

Requiremen	its with ID: 1. The system has maintainability	
Response	The system can be maintained in the future	
Priority	High	
Туре	Non-Functional requirement	
Version	1.0	
Requireme	nts with ID: 2. The system has sustainability	
Requireme Response	nts with ID: 2. The system has sustainability The system can be reused in the future	
•		
Response	The system can be reused in the future	

Figure 3. Non-Functional Requirement List

Algorithm Selection

Algorithm Optional 1

- Explanation: This algorithm works based on approximation values in the array to estimate beta results. It calculates Beta solutions based on input value ranges.
- Advantages: Easy to understand the logistic behind. The code structure is clear and easy to implement.
- 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. 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.
- Disadvantages: It needs lightly longer processing time.
- This algorithm has been selected for the implementation.

Implementation

- This source code has followed the standard Google Java Style Guide, which is corresponding to the whole team's program style.
- Correctness and Efficiency: 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: Coding convention has been regulated.
- Debugger: Jetbrains InteliJ Idea IDE build-in debugger is the major tool.
- Checkstyle: This program uses Checkstyle development tool during software implementation.
- Error handling and User Interface: This program has a clear user interface. It is straight forward for user to understand the logic and instructions. It also 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.

Implementation Continue

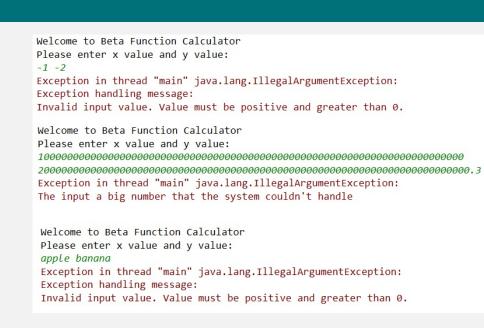


Figure 4. User Interface with Exception Handling, Error Messages

• 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 5. Each test case contains requirement

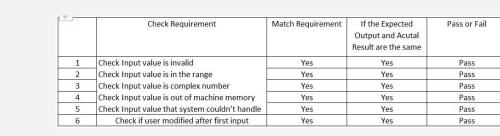


Figure 6. Test cases and requirements

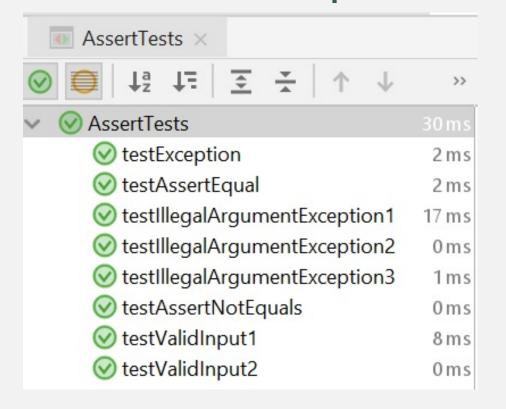


Figure 7. All test cases have passed

 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.

	15.
Result	Comments
Good	The source code has been integrated
Bad	Format problems occurs
	All coments and variables have defined
Good	correctly
	The source code satisfied the
Good	requirement
Good	The source code doesn't have errors
Good	The source code has handling functions
	There is one line of code not following
Okay	coding standard
Good	The code has no redundant functions
	Good Bad Good Good Good Good Good Okay

Figure 8. 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.

References

- 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)