Poster Presentation: Gamma Function: Team I

Amit Sachdeva, 40084627

Master of Software Engineering, Concordia University

Objectives

This project is about the functionality of gamma function as well as proper documentation. User can insert any real value and expect real value except on boundary conditions. Firstly, requirements and assumptions are made for our function and best algorithm is decided out of two different algorithms. Target is not limited to functionality but to improve user experience but keeping in mind about the proper coding style. Code reviewing is also done to get know the weak points.

Introduction

Gamma Function: It is commonly referred as factorial function for complex numbers. It is derived by Daniel Bernoulli. The gamma function $\Gamma(z)$ is defined for all complex values of z larger than zero. Complex number can be consist of real and imaginary number, like $z = a + ib$ in which a and b can real numbers. A complex number is typically written in the form where sigma a is the real part and it is the imaginary part. **Range** of the function is $(0, \inf)$ and **Domain** is from $[0, 109]$

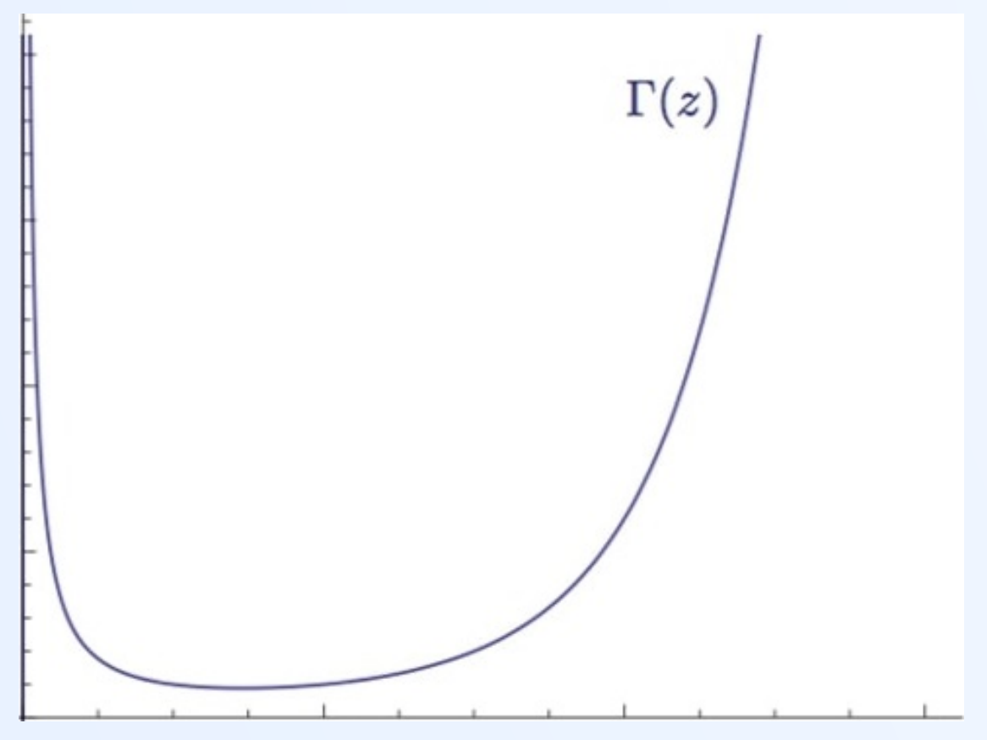
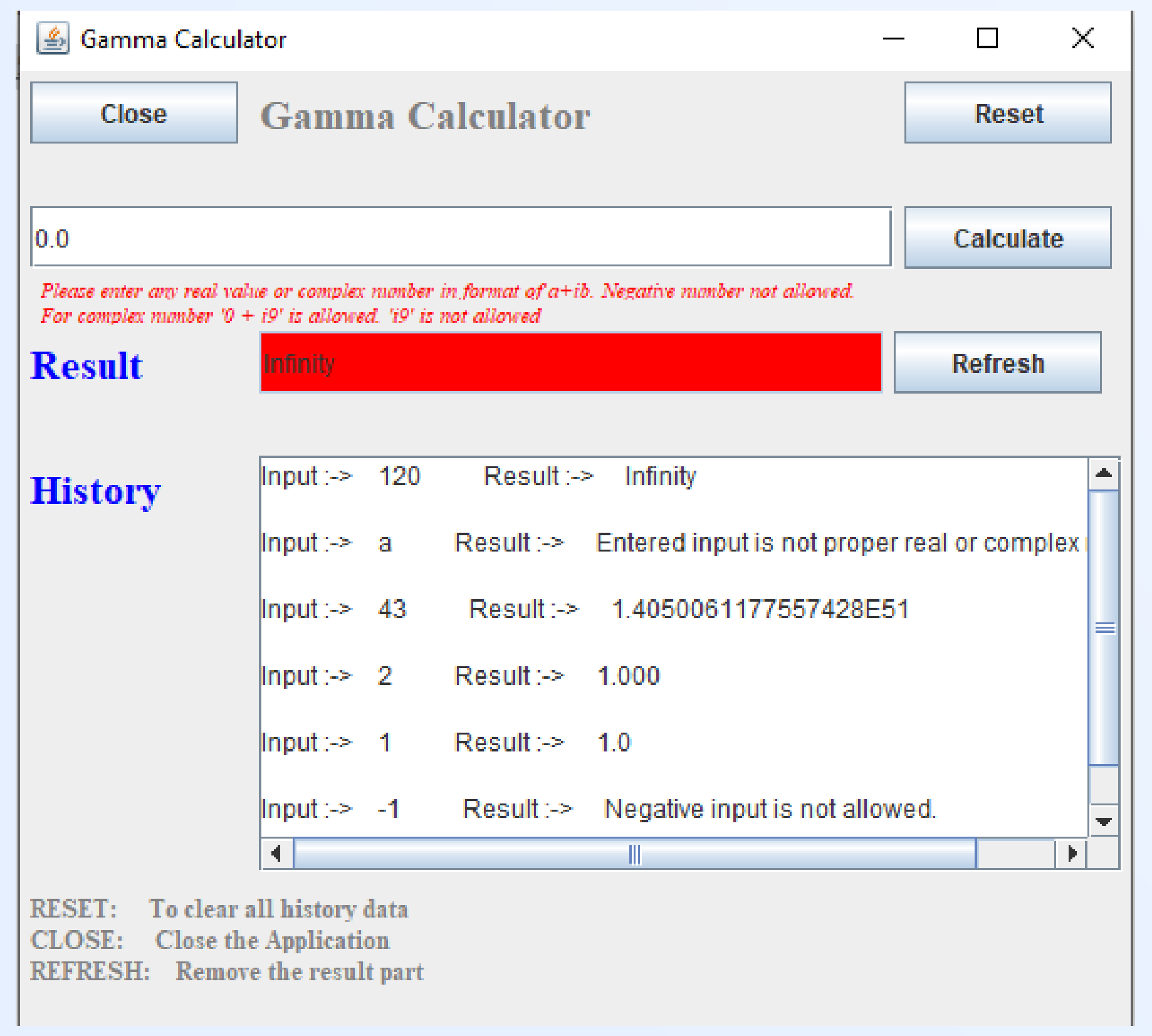


Figure: Gamma Function



0.0

Please enter any real value or complex number in format of a+ib. Negative number not allowed.
For complex number '0 + i9' is allowed. 'i9' is not allowed

Infinity

```
Input :->   120     Result :->   Infinity
Input :->   a       Result :->   Entered input is not proper real or complex
Input :->   43      Result :->   1.4050061177557428E51
Input :->   2       Result :->   1.000
Input :->   1       Result :->   1.0
Input :->   -1      Result :->   Negative input is not allowed.
```

RESET: To clear all history data
CLOSE: Close the Application
REFRESH: Remove the result part

Figure: GUI of Gamma Function

Pseudo code

This algorithm is based on calculating based on using core integral using graph like dividing whole graph in small parts and calculating each part using formula of trapezium $(1/2 * (base1 + base2) * height)$ and combining it at the end. We can get more precise results using this algorithm.

```
function yAxisValue(Argument x, Argument s) {
  calculate the value using $value = s^{x-1}e^{-s}$
  return value
end
}
function gammaFunction (Argument x) {
  if x < 0
    then raise Negative Input Error
  if x > 110
    then return "Infinity"
  Initialize finalData with 0
  Set Interval for gap = $10^{-3}$
  while loop i for range (0,Infinity)
    add the finalData by using formula of trapezium using
    $1/2 * gap * (yAxisValue(i) + yAxisValue(i - gap))$
    increment i with gap value
  return finalData
end
} {
In main function
  Take a input of x
  Call gammaFunction with input x-1 as a argument
end }
```

Requirements and Assumptions

- For **Large input** in positive real value, it will return infinity $\forall$ values > 110.
- For **negative input** $\forall x < 0$, **Function** will return **negative input error**.
- For $x = 0$, **Function** will return **1**
- For $Re(x) > 0$, **Function** will return positive real value, keeping in mind values belongs to real value
- For invalid string like: any string, or wrong complex number like i9, **Function** will return **Wrong input error**

Coding Conventions and Additional things

Coding Convention

1. Methods and variables should be in Camel Case.
2. Class name should be start with capital letter.
3. Package should be in capital letter.
4. Function length should be less than 30 words.
5. Constant should be of capital letters.

Additional Things

1. **JUnit 4** For Testing the functionality of project.
2. **Eclipse** is used as debugger.
3. **Check style** is used to improve code quality.
4. **Java Docs** for documentation and working of methods

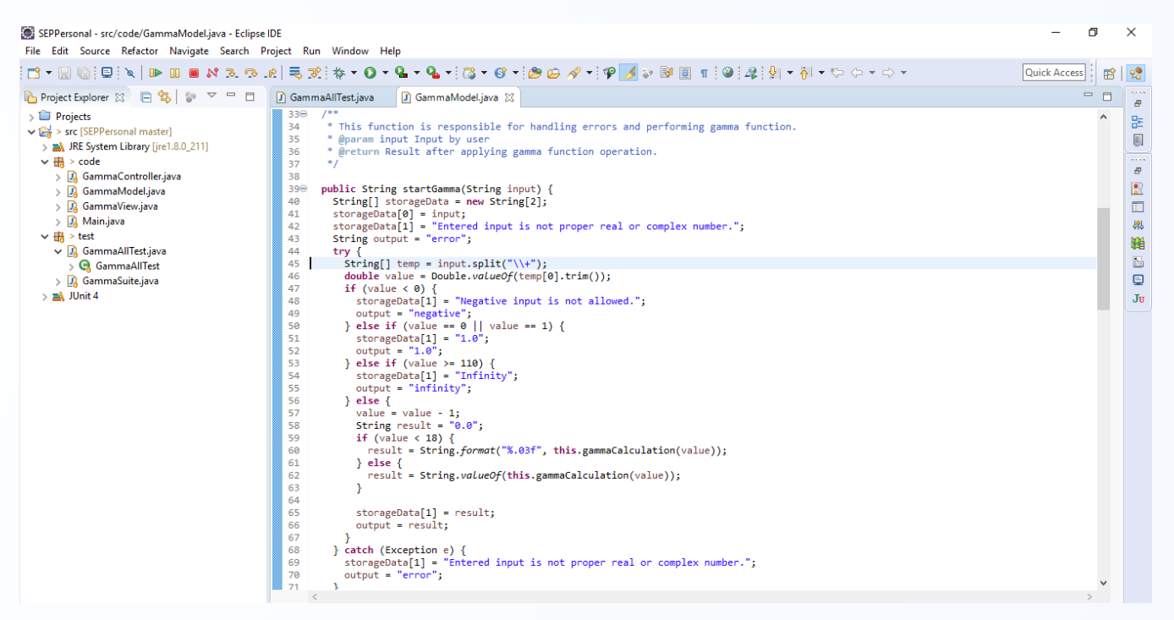


Figure: All Test Cases

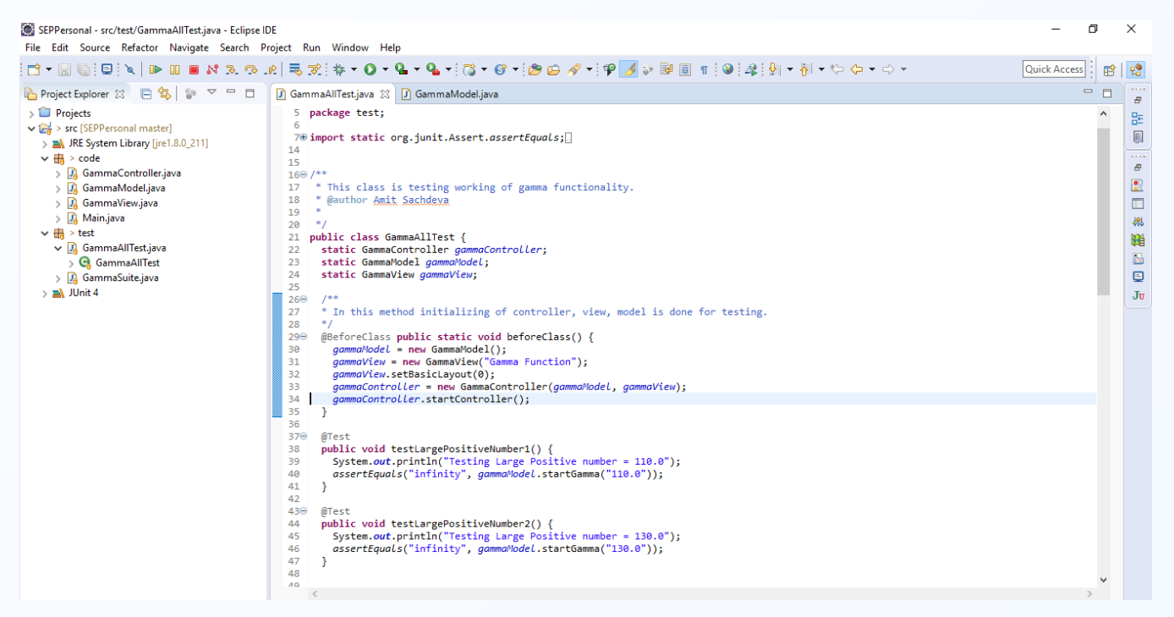


Figure: Coding style and Java docs

Results

Input	Code output	Real Output
6	120.000	120
99	8.13936**E153**	9.4268904**E153**
0	1.0	1
3+i9	2.000	2
i9	Wrong Input Error	Error
-1	Negative Input Error	Error
abc	Wrong Input Error	Error
120	Infinity	Infinity

Explicit Efforts and Critical Decisions

For this project, GUI is used to increase usability with proper exception handling to increase robustness. To enhance maintainability, Design pattern is used. Even each method has different functionality to decrease dependability. Core method of integration is implemented for gamma function which enhances efficiency and correctness. **Critical Decisions**

- **Taylor series** is used instead of **Math.log** = $\sum_{n=1}^{+\infty}(((x-1)/(x+1))^{2n-1})/(2n-1)$
- **Taylor series** is used instead of **Math.exp** = $\sum_{n=0}^{+\infty} x^n/n!$
- For Math.round, String.format function is used.

All this implementations of taylor series instead of internal math functions help to understand and learn the way of implementing every thing from scratch.

Suggested Improvements and Conclusion

Suggested Improvements

- Java docs can be used for test cases to enhance better readability of test cases. Functions length in few cases can be reduced to 20 to 30 lines to enhance the maintainability.
- More arithmetic functions like addition, subtraction can be added to cover all aspects of calculator.

Concluding that, this project has increased the way of handling project along with proper documentation and proper coding style. It also helps in working in a team. Not only that, it also helps to maintain track of working using git, and requirement gathering at initial stage is mandatory.

References

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4247832/
- https://medium.com/cantors-paradise/the-riemann-hypothesis-explained-fa01c1f75d3f
- https://www.eclipse.org/forums/index.php/t/206312/

Contact Information

- Github: `https://github.com/amitsachdeva45/SEPPersonal/tree/master/Documentation/Deliverable4`
- Professor: P. Kamthan