Java Mathematical Kit: A Dynamic Mathematical Guide

Nasrullah Memon[1], Abdul Rasool[1], Muhammad Urs Shaikh[2] and Abdul Qadeer Rajput[1]

[1]Department of Computer Systems and Software Engineering
Mehran University of Engineering and Technology Jamshoro Sindh Pakistan
nmemon@acm.org, parqureshi@yahoo.com, aqkrajput@hotmail.com
[2]Department of Basic Sciences and Related Studies
Mehran University of Engineering and Technology Jamshoro Sindh Pakistan
ursshaikh@yahoo.com

ABSTRACT

Mathematics is tough sledding for many. The difficulty is that most students fail to appreciate that mathematics is "just a language." It happens to be a very rigorous language, one with very little ambiguity associated with its symbols. It's also a very abstract one. And it's primarily the latter attribute, abstractness, which causes many students to falter. In this paper an attempt has been made to present design, development and implementation of software, **"***Java Mathematical Kit***"** (JMK). It aims to establish a framework to support solutions of various mathematical problems by using click and done methodology. The JMK is mathematical software, which has some unique features (Step Evaluator, etc.); which are not available in other mathematical software.

Keywords: Java Mathematical Kit (JMK), Step Evaluator, Matrixulator, Dynamic graphs, Dynamic error indicator.

1. INTRODUCTION

During the initial development of computers in the 1940's and early 1950's the chain of intermediate technologists between the computer designer/builder and the computer user was very short. In fact it was not uncommon for one individual to be actively involved in the design, construction, and application of a computer. By 1970 there was a significant amount of infrastructure between the computing machine and the end user - for example, from low-level to high-level one could cite *operating systems*, *compilers*, *general-purpose software*, and *applications-specific software*. Each level of this infrastructure absorbed the creativity and energies of its own community of specialists trying to improve its link in the chain. This frequently took the form of trying to meet the perceived needs of the higher levels in the infrastructure while coping with the limitations and peculiarities of the resources provided by the lower levels.

The *Java Mathematical Kit* is a contribution to the general-purpose mathematical software stratum of the computing infrastructure

2. BACKGROUND AND MOTIVATION

Through the late 1960's and up to about 1971, a number of collections or libraries of mathematical software had been assembled. Typically each of these libraries was developed for, and useable on, just one brand of computer system. Examples would be a library collected and/or developed by a computer vendor for use with their computer, or a library collected and/or developed by a large organization's computing center for use within that organization. For instance, Michael Powell recalls developing a math library at the Harwell Laboratory in the early 1960's.

Even the two organizations, NAG [1] and IMSL [2], that later became the world's main suppliers of math libraries across diverse machine types, each started in 1970 by targeting a single machine type. NAG started as an informal cooperative effort by six institutions in England to develop a math library for their newly acquired ICL 1906A systems. Their first library, in both Algol and Fortran versions, was available to the participating institutions in October 1971, and they immediately started attacking the problem of developing the library for other computer types.

IMSL was founded in 1970 as a company to develop and market mathematical and statistical libraries. A major motivation for the founding of IMSL was a perception of customer dissatisfaction with the math libraries provided by IBM for use on their systems. IMSL's first product, available in 1971, was a Fortran library for the IBM/370-360.

Designing math library software to be portable across diverse systems was particularly difficult in the 1970's because there were significant differences between the arithmetic characteristics of computers from different manufacturers. The state of standardization of programming languages was in its infancy, and operating systems (encompassing file naming and file storage) were quite different on different computer brands. Fortran was the only language widely supported in the U.S., and the only language for which there was an ANSI standard (1966). However, the standardized language had major shortcomings so the programmers tended to use vendor-specific extensions and that worked against portability. Algol and Fortran were both used in Europe.

A project called NATS (National Activity for Testing Software or NSF, Argonne, Texas, Stanford) [3] undertook during 1971-1972, to investigate the problems of achieving Fortran portability for mathematical software by testing, and as necessary modifying, a Fortran version of a set of dense matrix eigenvalue/eigenvector algorithms. These 30 algorithms were originally developed in Algol and published in a series of papers in the 1960's by J. H. Wilkinson and others, and subsequently published together as the *Handbook* ... [4]. The collection was converted to Fortran by Virginia Klema and others at Argonne National Laboratory and then processed for portability by NATS. NATS announced the availability of EISPACK [2, 6, 7] in 1972, the package having been successfully tested on IBM 360-370, CDC 6000-7000, Univac 1108, Honeywell 635, and PDP-10 computers.
The same EISPACK code ran on all these systems, with the exception of one variable that needed to be set to indicate the precision of the arithmetic on the host system. EISPACK was widely requested and distributed. This project demonstrated the great utility of carefully tested portable mathematical software based on state of the art algorithmic research. But did (Fortran) portability mean sacrificing efficiency? Library developers began giving more attention to this issue, and new hardware designs presented new challenges and opportunities.

Nowadays much mathematical software are available in market, for example, *MathCAD*, *Mathematica*, *Matlab*, *TK Solver*, *Adams*, etc. We developed our software (*Java Mathematical Kit*) using Java Technology as our tool of development.

3. IN WHAT SENSE IT IS DIFFERENT FROM OTHER SOFTWARE?

This software holds some features like:

- *Graphical User Interface*
- *Step Evaluator*
- *Matrixulator*
- *Dynamic Graphs (GML)*
- *Dynamic Error Indicator*

3.1 Graphical User Interface

This Software has User Friendly and Self Explanatory Graphical Interface (as shown in figure 1). This means that users do not have to worry about Syntaxes of Commands. One has just to enter the function with proper conventions and then select the required command to cope with problem. For invalid inputs, the software generates dynamic error with exact location, indicating where the user has made the error. Thus by knowing what and where error has occurred, user has a facility to correct it without much botheration.

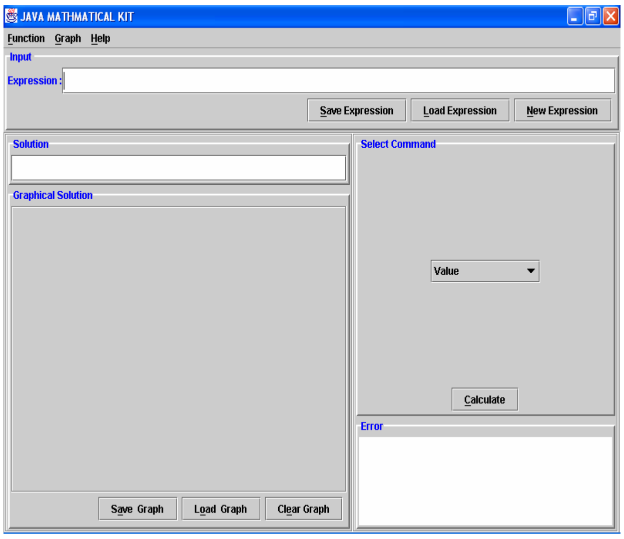
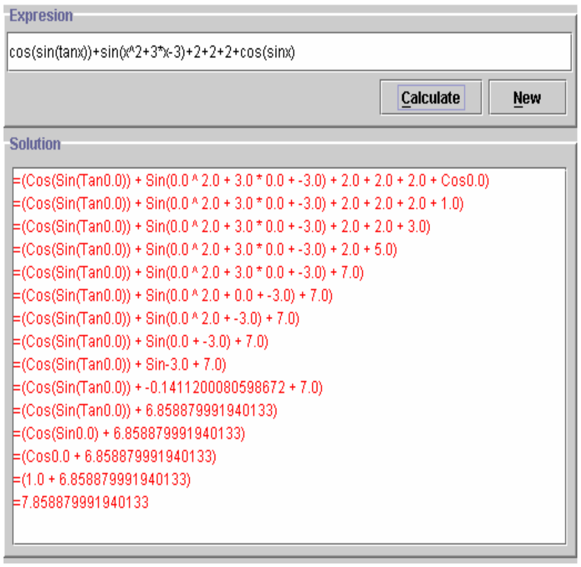


Figure 1: Graphical User Interface

3.2 Step Evaluator

This Software has a very unique quality of solving an expression or function in *Step by Step* fashion. User can view a complete step by step method of solving an expression that *Step Evaluator* performs manually. Thus user may solve an expression with Step Evaluator, save and print it. This feature is not present in any Software, as shown in figure 2.



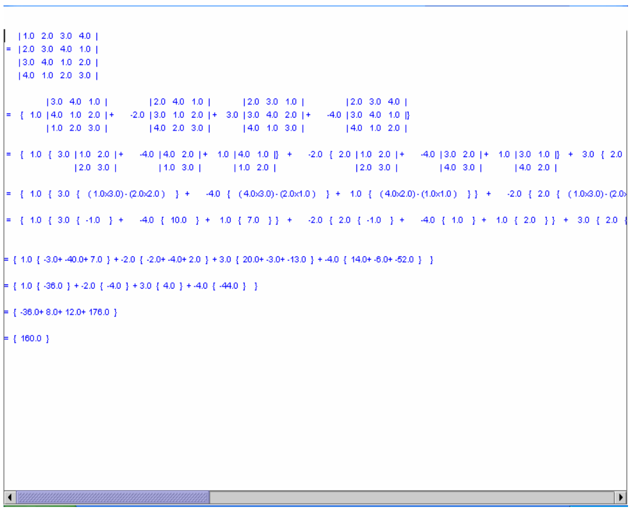
```
cos(sin(tanx))+sin(x^2+3*x-3)+2+2+2+cos(sinx)
```

```
=(Cos(Sin(Tan0.0)) + Sin(0.0 ^ 2.0 + 3.0 * 0.0 + -3.0) + 2.0 + 2.0 + 2.0 + Cos0.0)
=(Cos(Sin(Tan0.0)) + Sin(0.0 ^ 2.0 + 3.0 * 0.0 + -3.0) + 2.0 + 2.0 + 2.0 + 1.0)
=(Cos(Sin(Tan0.0)) + Sin(0.0 ^ 2.0 + 3.0 * 0.0 + -3.0) + 2.0 + 2.0 + 3.0)
=(Cos(Sin(Tan0.0)) + Sin(0.0 ^ 2.0 + 3.0 * 0.0 + -3.0) + 2.0 + 5.0)
=(Cos(Sin(Tan0.0)) + Sin(0.0 ^ 2.0 + 3.0 * 0.0 + -3.0) + 7.0)
=(Cos(Sin(Tan0.0)) + Sin(0.0 ^ 2.0 + 0.0 + -3.0) + 7.0)
=(Cos(Sin(Tan0.0)) + Sin(0.0 ^ 2.0 + -3.0) + 7.0)
=(Cos(Sin(Tan0.0)) + Sin(0.0 + -3.0) + 7.0)
=(Cos(Sin(Tan0.0)) + Sin-3.0 + 7.0)
=(Cos(Sin(Tan0.0)) + -0.1411200080598672 + 7.0)
=(Cos(Sin(Tan0.0)) + 6.858879991940133)
=(Cos(Sin0.0) + 6.858879991940133)
=(Cos0.0 + 6.858879991940133)
=(1.0 + 6.858879991940133)
=7.858879991940133
```

Figure 2: Preview of Step-Evaluator

3.3 Matrixulator

This software has a special feature named as "*Matrixulator*" for matrix manipulation. With this, user may add, subtract, and multiply any two suitable matrices. One can also evaluate determinant of a matrix using "*Matrixulator*" tool. Moreover, all these operations are performed in step by step manner. This is also a unique property of the software.



```
   |1.0  2.0  3.0  4.0 |
=  |2.0  3.0  4.0  1.0 |
   |3.0  4.0  1.0  2.0 |
   |4.0  1.0  2.0  3.0 |

           |3.0  4.0  1.0 |         |2.0  4.0  1.0 |        |2.0  3.0  1.0 |          |2.0  3.0  4.0 |
=  {  1.0 |4.0  1.0  2.0 |+   -2.0 |3.0  1.0  2.0 |+  3.0 |3.0  4.0  2.0 |+   -4.0 |3.0  4.0  1.0 |}
           |1.0  2.0  3.0 |         |4.0  2.0  3.0 |        |4.0  1.0  3.0 |          |4.0  1.0  2.0 |

=  {  1.0  {  3.0 |1.0  2.0 |+   -4.0 |4.0  2.0 |+  1.0 |4.0  1.0 |}  +   -2.0  {  2.0 |1.0  2.0 |+   -4.0 |3.0  2.0 |+  1.0 |3.0  1.0 |}  +  3.0  {  2.0
                  |2.0  3.0 |         |1.0  3.0 |        |1.0  2.0 |                      |2.0  3.0 |         |4.0  3.0 |        |4.0  2.0 |

=  {  1.0  {  3.0  {  (1.0x3.0) - (2.0x2.0)  }  +   -4.0  {  (4.0x3.0) - (2.0x1.0)  }  +  1.0  {  (4.0x2.0) - (1.0x1.0)  } }  +   -2.0  {  2.0  {  (1.0x3.0) - (2.0x

=  {  1.0  {  3.0  {  -1.0  }  +   -4.0  {  10.0  }  +  1.0  {  7.0  } }  +   -2.0  {  2.0  {  -1.0  }  +   -4.0  {  1.0  }  +  1.0  {  2.0  } }  +  3.0  {  2.0  {

= { 1.0 { -3.0+ -40.0+ 7.0 } + -2.0 { -2.0+ -4.0+ 2.0 } + 3.0 { 20.0+ -3.0+ -13.0 } + -4.0 { 14.0+ -6.0+ -52.0 }  }

= { 1.0 { -36.0 } + -2.0 { -4.0 } + 3.0 { 4.0 } + -4.0 { -44.0 }  }

= { -36.0+ 8.0+ 12.0+ 176.0 }

= { 160.0 }
```

Figure 3: Preview of Matrixulator

3.4 Dynamic Graphs

This Software has efficiencies to draw graph of any function in one variable as shown in figure 4. As the name suggests, user may change the scale and view of the graph dynamically at runtime, after drawing the graph. The user may save the current graph or load previously saved graphs. This is done by GML (*Graphic Markup Language*), a special format in which this software saves and retrieves the graphs.

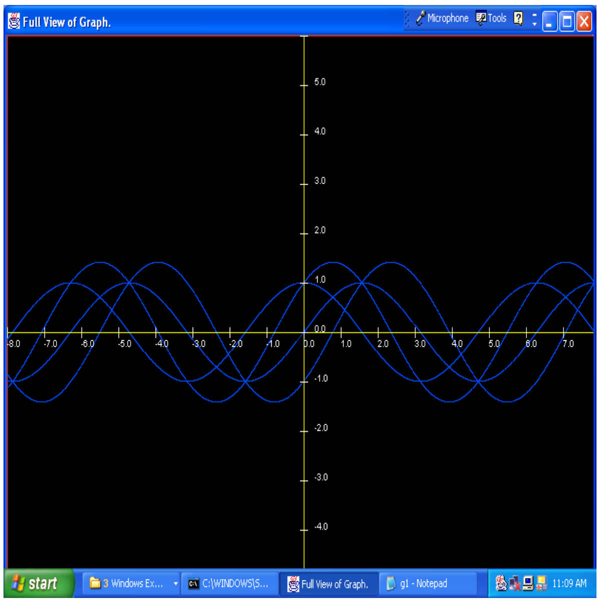
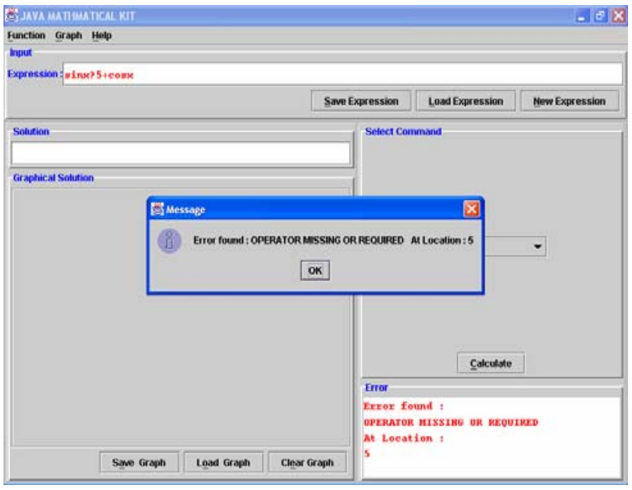


Figure 4: Dynamic Graphs

3.5 Dynamic Error Handling

This Software generates different type of errors with location numbers to help the user to correct it. This property is shown in Figure 5 and discussed in detail in the next section.



Expression : sinx?5+cosx

Error found : OPERATOR MISSING OR REQUIRED At Location : 5

```
Error found :
OPERATOR MISSING OR REQUIRED
At Location :
5
```

Figure 5: Dynamic Error Handling

4. DANYMIC ERROR DETECTION FACILITIES

This Software has some special “Error Trapping Routines” which are designed to locate all possible types of dynamic errors which a user may commit, during input session.

“*Self Detecting Skills*” of this software not only locates the error, but also let the user know the type and exact position/place of the error.

Table 1 shows errors that appear when user enters invalid inputs.

4.1 Ambiguous Operators

This type of error appaers when the user enters empty parenthesis like "()”. It is displayed with the location at which it occurs.

Example: 5+()+4

4.2 Extra "(" in the Expression

This type of error appears when the user enters an extra parenthesis like "(“. It may be noted that this error is not displayed with the location at which it occurs.

Example: 5+((a + b)+4

4.3 Extra ")" in the Expression

This type of error appears when the user enters an extra parenthesis like ")”. It is not also displayed with the location at which it occurs.

Example: 5+(c + d))+4

4.4 Unexpected Decimal Point

This type of error appears when the user enters an extra Decimal point. It is displayed with the location at which it occurs.

Example: 2.3.4

4.5 Unnecessary Decimal Point

This type of error appears when the user enters continuous decimal points. It is displayed with the location at which it occurs.

Example: 2..4

4.6 Unidentified Letter or Symbol

This type of error appears when the user enters any letter that is not used in any *Mathematical Symbol.* This type of error is displayed with the location at which it occurs.

Example 1: sinx+ +cosx
Example 2: sinx+u+cosx

Note: This error may appear due to Space or blank being entered by the user as in first example.

4.7 Operator Missing or Required

This type of error appears when the user enters two operands without operator. It is displayed with the location at which it occurs.

Example: 2x

Note: There is no automatic multiply operation in software, therefore user must write 2*x instead of 2x.

4.8 Operand Missing or Required

This type of error appears when the user enters two operators without operand. It is displayed with the location at which it occurs.

Example: 2+*x

4.9 Undefined Symbol

This type of error appears when the user enters a symbol comprising of letters, which are used in other *Symbols.* It is displayed with the location at which it occurs.

Example: cinx

4.10 Unexpected ")" in Expression

This type of error appears when the user closes parenthesis **")"** without opening it. This error is not displayed with the location at which the error occurs.

Example: 5+)+4

4.11 Incomplete Equation

This type of error appears when an unusual end of expression is made. This error is not displayed with the location at which it occurs.

Example: 5+4+

4.12 Not a number (NAN)

This type of error appears when the result of required operation is a non-real value.

This error is not displayed with the location at which it occurs.

Example: Root of x^2+2

4.13 Infinity

This type of error is displayed when the result of expression is infinity. It is not displayed with the location where the error has occurred.

Example: 5/0

Note: Error may come in form of +Infinity or -Infinity.

5. SYSTEM ARCHITECTURE

The user inputs a string that is send to the *Expression Class*, which in turn converts it to postfix and then evaluates it to find the value of function. *Class Command* and its subclasses, to implement their algorithms, use this value. The result is then converted back in string form. This string is calculated output. The whole process is shown in Figure 6.

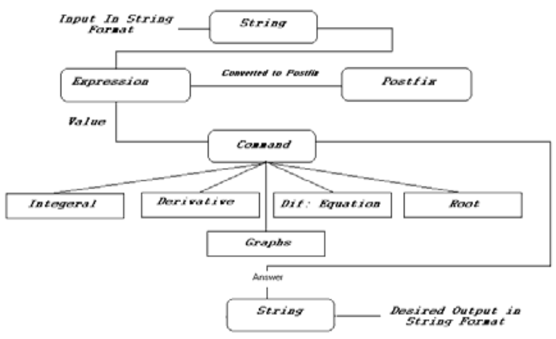


Figure 6: Working of Java Mathematical Kit

6. FUNCTION WRITING CONVENTIONS

The following rules may be applied in writing functions.

- No *White Space* in Expression.
- Function must be in one variable, that is, “x”.
- The Fundamental Binary Operators which a user can use in Function are:

1. “ + “ for Addition
2. “ - “ for Subtraction
3. “ * “ for Multiplication
4. “ / “ for Division
5. “ ^ “ for Power

- The Unary Operators which a user can use in functions are:

1. Sinx for Sine
2. Cosx for Cosine
3. Tanx for Tangent
4. Sin~x for arcSine
5. Cos~x for arcCosine
6. Tan~x for arcTangent
7. Log for Logarithm with base 10
8. Ln for Natural Logarithm
9. Exp for e

- Parenthesis may be used to increase or decrease the precedence of any operation.
- Function is not *Case Sensitive*.

7. CONCLUSION

One of the motivations for developing the *Java Mathematical Kit* is to provide features, which are not available in any other mathematical software. It is hoped that this software will be of great help to both students and teachers.

In this paper, we presented the design, development, and implementation of mathematical software named as *Java Mathematical Kit*. With this software we can:

- Integrate a function and finds its definite value
- Differentiate a function at a point
- Find roots of an equation and solve simultaneous equations
- Find value of function at different instants
- Finding the roots of equations graphically
- Draw, Save and Load graphs of a function

This Software is Robust. It possesses Self-detecting ability for any type of dynamic and input errors. It cannot only detect what type of error is made but also hits where the error has occurred (if the error is input by the user).

This Software not only finds the values of functions but can also show all steps required for performing the task. This functionality is an important feature of this software in the shape of "*Step - Evaluator* "(a special part of this software).

Moreover, this software is also capable of performing matrix arithmetic, means it can add, multiply, divide, and subtract any matrix of any order. Besides, it can also find determinants and inverses of the matrix. It can also show the steps that one performs manually.

This is dynamic software that can save or load users work and graphs. This is done with the help of GML

(*Graphics Markup Language*), a special format, which saves and loads the graph.

8. FUTURE EXTENSIONS

The authors are still working for the extension of the software and hope that following extensions be implemented in very short span of time.

1. Simultaneous Linear Equation Solver
2. Non Linear Equation Solver
3. Step by Step Derivative Calculator
4. Discrete Mathematician
5. 3D graphs (graphing the functions involving two variables)

9. ACKNOWLEDGEMENTS

This work is partially supported by Mehran University of Engineering and Technology Jamshoro, Pakistan.

REFERENCES

[1] B. FORD, *The Nottingham Algorithms Group (NAG) Project*, ACM SIGNUM Newsletter, 8, 2, April 1973, pp. 16-21.

[2] O. G. JOHNSON, *IMSL's ideas on subroutine library problems*, ACM SIGNUM Newsletter, 6, 3, Nov 1971, pp. 10-12.

[3] ANNOUNCEMENT, *NATS Project - Collaborative Research toward the Development of a Certified Sub-routine Library*, ACM SIGNUM Newsletter, 6, 3, Nov 1971, p. 5.

[4] J. H. WILKINSON AND C. REINSCH, eds., *Handbook for Automatic Computation, Vol. 2, Linear Algebra*, Springer-Verlag, New York, 1971.

[5] ANNOUNCEMENT, *The Certified Eigensystem Package*, EISPACK, ACM SIGNUM Newsletter, 7, 2, July 1972, pp. 4-5.

[6] B. S. GARBOW, J. M. BOYLE, J. J. DONGARRA, AND C. B. MOLER, *Matrix Eigensystem Routines -EISPACK Guide Extension*, Vol. 51 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1977.

[7] B. T. SMITH, J. M. BOYLE, B. S. GARBOW, Y. IKEBE, V. C. KLEMA, AND C. B. MOLER, *Matrix Eigensystem Routines - EISPACK Guide*, Vol. 6 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1974 (2nd ed., 1976, with additional author, J. J. DONGARRA).

[8] MathML International Conference 2000, www.mathmlconference.org, UIUC Illinois USA, Oct. 20-21, 2000.

[9] Workshop on *The Future of Mathematical Communication* http://www.msri.org/activities/events/9900/fmc99/, Dec. 1999.

[10] ActiveMath, http://www.mathweb.org/activemath

[11] Calc101, http://www.calc101.com/

[12] Ezmath, http://www.w3.org/People/Raggett/EzMath/

[14] Institute for Computational Mathematics, demos of mathematical computation http://icm.mcs.kent.edu/research/demo.html

Table 1: Errors that appear when user enters invalid inputs.

S.No.	Type	Input	Remarks	Example
1	Ambiguous Operators	Empty parenthesis like "()”	Type of Error is displayed along with Location at which Error has occurred	5+()+4
2	Extra "(" in Expression	Extra parenthesis like "(“	This Type of Error is displayed without the Location at which Error has occurred	5+(()+4
3	Extra ")" in Expression	Extra Parenthesis like ")”	This Type of Error is displayed without the Location at which Error has occurred	5+())+4
4	Unexpected Decimal Point	An extra Decimal point	Type of Error is displayed along with the Location at which Error has occurred	2.3.4
5	Unnecessary Decimal Point	Continuous Decimal points	Type of Error is displayed along with the Location at which Error has occurred	2..3
6	Unidentified Letter or Symbol	Letter which is not used in any symbol	Type of Error is displayed along with the Location at which Error has occurred	1. simx+ +cosx 2. sinx+u+cosx
7	Operator Missing or Required	Two operands without operator	Type of Error is displayed along with the Location at which Error has occurred	2x
8	Operand Missing or Required	Two operators without operand	Type of Error is displayed along with the Location at which Error has occurred	2+*x

9	Undefined Symbol	A symbol comprising of letters which are used in other Symbols	Type of Error is displayed along with the Location at which Error has occurred	cinx
10	Incomplete Equation	An unusual end of expression is made	This Type of Error is displayed without the Location at which Error has occurred	5+4+
11	Nan	Type of Error comes when the result of required operation comes as a non real value	This Type of Error is displayed without the Location at which Error has occurred	Root of x^2+2
12	Infinity	Type of Error comes when the result of expression is infinity	This Type of Error is displayed without the Location at which Error has occurred	8/0