A Novel Framework Of Exception Handling For Semantic Errors Detected In Mathematical And Computational Tools

Zulfiqar Ali Shar, Noor Ahmed Shaikh, Samina Rajper, Aurangzeb Magsi

Abstract: **Exceptional handling is a mechanism provide by programming languages to handle some unexpected scenarios/ errors during the program compilation and continue to precede program execution. Some semantic errors which can't be handled by exception handling produce inappropriate results after program compilation. However, in order to deal with this issue and obtain accurate results, programmers must develop a code to reduce error ration in likewise scenarios. In this paper we introduce an extended exceptional handling framework developed in Java programming language and define the flow of the developed model. Five classes namely Plus, Minus, Division, Multiplication and Remainder are addedin the extended model to handle respective arithmetic exceptions with eight data types such as Byte, Short, Int, Long, Float, Double, Char and String. For experimentation, the developed package i.e. ZAB is imported on two software programs. To measure the complexity of the developed package, we use Cyclomatic Complexity (CC) method and calculated it for all the five classes. The calculation show that CC for Class Plus, Minus and Multi is V(G)=6.ForClassDivision and Reminder,V(G)=2 and respectively. Comparative analysis is done on the calculated results of each class and acquired output of the CC during experimentation show less complexity of Division Class. The Compatibility of Package ZAB isolates and handles semantic errors to reduce compile/run time errors .commotion**

Keywords— **Exceptional Handling, Computational Tools, Semantic Error**

INTRODUCTION

Exception handling is one of the key components of every software which makes it reliable and robust product. Developers during the software design take it in to serious consideration as exceptions are the backbone of any Object Oriented Language (OOL). Numerous programming languages support exception handling as an essential part [1]. Java is an OOL uses exceptions during program execution to avoid run time / compile time errors. A group of classes is encapsulated in the form of package in Java. Hence to consolidate a class file into a package provides the advantage of reusability feature to reduce time and other system resources. User-Defined and Built-In are two types of packages used in Java where user-defined as name suggested are the packages developed by the programmers while the built-in packages are predefined functions like java.io etc [2]. A package can be used by two ways i.e. by calling it via its name or by importing package where second method reduces time complexity[3].

Semantic and Syntax are the two types of exceptions which can interrupt the flow of a program and it may be terminated due to the occurrence of an exception. In mostof the time after the program termination system generates an error message to programmer which shows the exception type and urges programmer to change the logic of the programming code. Exception is somehow simple in the comparison of error. Errors like syntax error, JVM errors etc. are basically abnormal conditions and are unable to be identified and resolved in ordinary conditions [4]. While exceptions unlike errors are the conditions within a programming code and can be handled by the programmer by taking some necessary actions/changin aprogramming logic. DivideByZeroexception, NullPointerException,ArithmeticExceptioand Array Index Out Of Bounds Exception are some examples of exceptions in Java language[5].

If an exception can't be handled by the programmer it may cause the termination of the flow of whole program after which programmer gets an error message which is unwieldy and leads to the change of programming logic. Changing a programming logic is not suitable in every condition as in most of the scenarios it require lot of resources regarding the respective domain of the program [6]. Checked and unchecked are the two type of exceptions reported in literature while some built-in classes supported by the respective exceptions are given in Table 1.

Table 1. Type of Exceptions

Checked exceptions	Class Not Found Exception
	Illegal Access Exception
	No Such Field Exception
	EOF Exception

Department of Computer Science, Shah Abdul Latif University, Khairpur

Unchecked exceptions	ArithmeticException
	ArrayIndexOutOfBoundsEx-ception
	NullPointerException
	NegativeArraySizeException

Both checked as well as unchecked exceptions are occurred in Java programming language [7]. The hierarchical structure along with the graphical representations of these exceptions including Throwable class is depicted in Figure.

1. Checkedexceptions including Throwable class is depicted in Figure 1. Checked exceptions are usually declared in the method's throw clause and are thrown in the particular method while occurring. However, unchecked exceptions are the one which are unexpected and are very difficult to be recovered. Null pointer, any number divided by the '0' are some examples of unchecked exceptions. In order to handle such type of exclusions, an exceptional-handling mechanism is introduced to facilitate by designated the protected code of program [8]. The introduced mechanism is somehow limited with the operations. Any type of failure in exceptional-handling may cause the disappointment in performance analysis and produce false results which ultimately questioned the reliability and efficiency of the software application [9]. Very little work in this regards has been done in the field of software development [10]. Software applications that are based on the numerical methods are the most effecting tools in this concern which with the type of unchecked exceptions may fail to recognize the flow of data control [11]. Keeping in view the requirements, in this study we introduce a completepackage of exceptional handling which with the implementation will handle the arithmetic exceptions and allow the source code to be executed.

Figure 1. Variance among Checked Exception and Unchecked Exception

LITERATURE SURVEY

On exception handling lot of work has been developed and literature available regarding this, packages, and overloadingmethodinprogramminglanguagesbutinobject oriented (OO) system all these are still challenging tasks. Today's Software systems are more complex due to this reason [12]. Exception handling mechanism was proposed for organizational management in multi-agent system [13]. The proposed system enrich the schemes and missions of the functional specification of an organization with the concepts such as Recovery Strategy, Notification Policy, Handling Policy, Throwing Goaland Catchinggoal. Theproposed mechanism is relied on abstractions that are seamlessly integrated with organizational concepts, such as responsibilities, goals and norms. Earlier studies reveal that developers are unwilling or feel it hard to adopt exception handling mechanism, and tend to ignore it until a system failure forces them to do so. To help developers with exception handling, researcher produced recommendations such as code examples and exception types, which still require developers to localize the try blocks and modify the catch block code to fit the context.

Similarly, Jian Zhang [14] proposed a novel neural approach to automated exception handling, which can predict locations of try blocks and automatically generate the complete catch blocks. The author collected a large number of Java methods from GitHub and conduct experiments to evaluate the approach. The evaluation results, including quantitative measurement and human evaluation, show that theproposed approach is highly effective and outperforms allbaselines.

Moreover, Nguyen, T. T [15] introduced two novel methods for the correction of code of exceptional handling namely XRank and XHand. Using both of these techniques author also developed ExAssist tool for code recommendation. From the implementations authorreceived 87% and 96% high accuracy with the XRank and XHnad techniques respectively. Similarly for the ease of developers in case of exception handling, H Melo and others [16] made a qualitative research and came up with the Java guideline. From the undertaken study authors concluded that 70% guideline is available regarding exception handling. Similarly, TT Nguyen [17] performed an empirical study on occurrence and nature of 300 mobile apps based exceptions and related bugs along with the suggestions that how do programmers tackle them. However, several research articles have found certain poor practices in exception handling processes, as well as the predominance of associated anti-patterns. De Pádua [18][19] investigated the relationship among software quality and exceptional handling. During the case study it was found that Exception flow characteristics in Java projects show a substantial association with post-release problems which indicate more practices towards exceptional handling. Using the exception handling standards of sequential code, a challenge to the parallel programming is also resolved. Mehrabi, M [20] discussed the advancements in the parallel computing environment using Javaannotations.

PROPOSED JAVAPACKAGE

Exceptional handling mechanism of Object Oriented languages is unable to handle semantic errors. A case study in this regards is performed on Java programming languages and to encounter this problem a Java Package is developed through which semantic errors occurred in Java programming can be handle [21]. A package is a mechanism in a Java introduced to encapsulate a large number of classes in to a single package. However, in Java language source and class files are managed via hierarchical file system as organizing a class file in to the package is quite easy in Java. There are two types of packages used in Java namely: In-Built Packages and User-Defined Packages. A group of predefined packages are called Java API whereas Figure 2 shown below is the diagrammatical representation of Java API (predefined-packages).

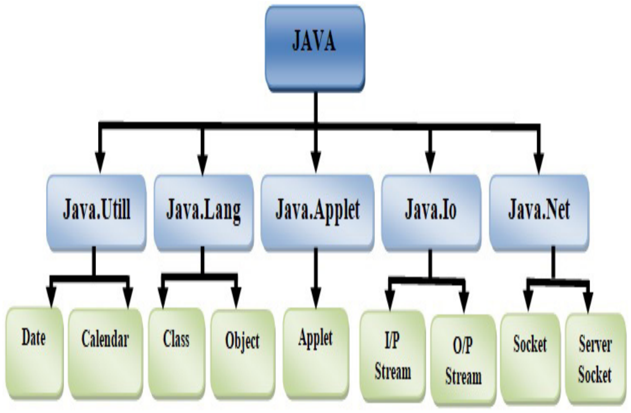


Figure 2. Diagram of Built-in Java Packages

DEVELOPED PACKAGE

A user defined-package is created by user itself and can be import in Java program. To create a new package in Java requires a declaration of directory of the same name as package. Hence in this regards we create a new directory and named it ZAB while it is optional with the Java Development Kit-7 (JDK-7) as using the '–d' option creates directory with the same name during the program compilation. After creating the directory the next step is the creation of class inside the directory with the same name of directory [22]. A ZAClass is also created inside the directory. Furthermore a package is declared and named as ZAB after the class declaration as shown below.

```
package ZAB;
"public class ZAClass
{ public void gNames(String s)
{       System.out.println(s);       }}"
```

Above is the declaration of new package ZAB which contain ZAClass and saved in ZA directory of the program along with the other classes and entities.

Following is the example of a program of string value based on the developed package.

```
"/* import 'ZAClass' class from 'names' ZAB */
importZAB.ZAClass;
public class PName
{ public static void main(String a[])
{ // defining and initializing the string variable namely
//with string Zulfiqar Ali
String name = "Zulfiqar Ali";
// creating the object from ZAClass
// the ZABpackage.
ZAClassobj = new ZAClass();
obj.gNames(name);}} "
```

A.Adding other classes into the defined package:

Using the same package name we created and added a class with the same name as package by making it publically accessible using public access identifier. With this structure we can create and add more than one class in to the developed package. Several in-built exceptions are defined in java libraries [23]. The existing in-built model of exception is extended in this research work. Some arithmetic exceptions are used in this regards such as add, sub, div etc. The developed model is added in to the respective exception handling model and its flow is defined in Figure 3.

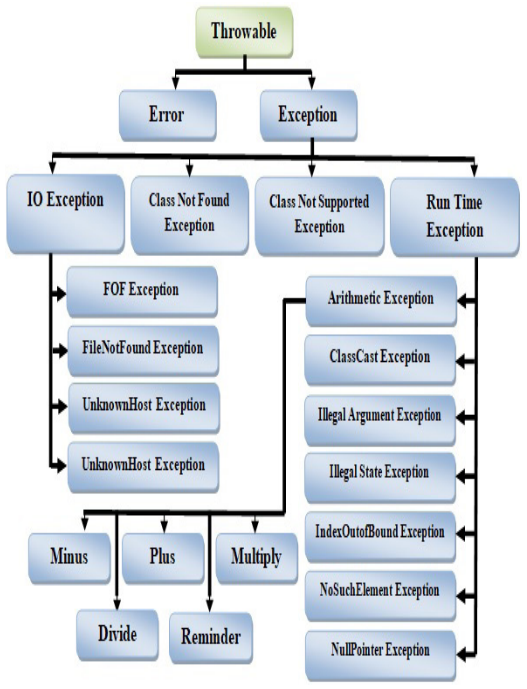


Figure 3. Flow of Extended Model of Exceptional Handling

As shown in Figure 3, we added five arithmetic exceptions in the extended model namely: Plus, Minus, Multiply, Division and Reminder exceptions. The mentioned exceptions are implemented on eight data types i.e. Byte, Short, Int, Long, Float, Double, Char and String in Java Programming language to handle arithmetic exceptions using Method Overloading technique [24]. Experimentations of handling the arithmetic exceptions are performed on the eight selective primitive data types where multiple operations were performed by creating multiple classes in the same package where the class and operator name remained same. Based on the total number of exceptions we developed five different classes with the same name for each exception. As the selective data types for the implementation are eight in number hence, eight different methods/functions are required in each class. Therefore using the method overloading technique we create eight overloaded functions and perform experiments on each of them. The concept of inheritance is used while creating multiple classes [25]. Structurally, we created five classes i.e. plus; minus; multis; divis and rems where class plus is the parent class to the class minus, class minus is parent class to class multis and so on. This technique reduced the workload of creating multiple objects for multiple classes. This concept of inheritance allows us to use a single object for all the created classes [26]. The concept of inheritance used in our methodology is depicted in Figure 4.

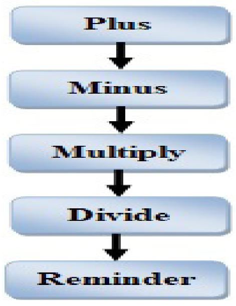


Figure 4. The concept of inheritance used in our methodology.

For each created class we develop eight overloaded functions and stored in the developed package ZAB. While figure 5 is the depiction of the overloaded methods developed for class Plus.

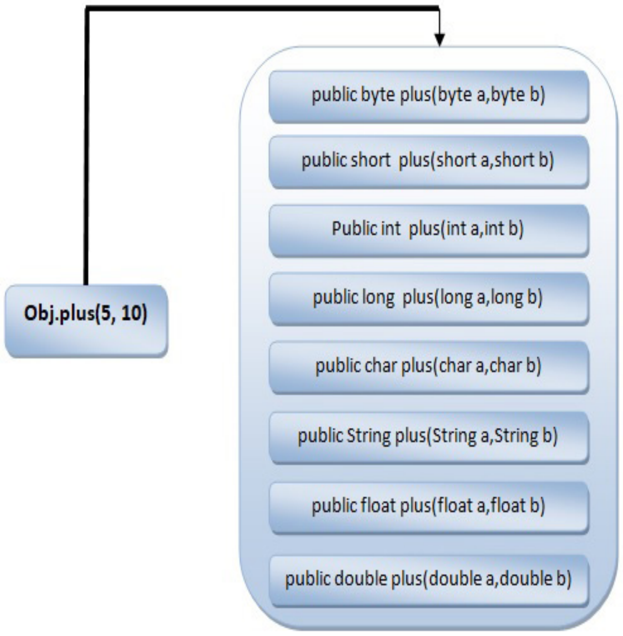

Obj.plus(5, 10)
public byte plus(byte a,byte b)
public short plus(short a,short b)
Public int plus(int a,int b)
public long plus(long a,long b)
public char plus(char a,char b)
public String plus(String a,String b)
public float plus(float a,float b)
public double plus(double a,double b)

Figure 5. Depicted eight overloaded methods of plus class and calling way.

Based on the input data type, program calls the respective function which pass the required data such as if we use byte values as an input data than the program calls the function which pass byte values similarly program calls the function which pass integer data type if the inputted value is of integer type and so on. This mechanism performed on all the data types where program always calls the function which handles the respective data type value.

The problem faced by the programmers when there is a subtraction or addition is performed on the character type or string type values or attempt regarding the division of byte, short, integer and long values or an attempt to reminder the float, double, and long double values results the same problem. An example of the scenario is defined below:

```
za= ”ZA” + ”Baloch”
za= ”ZA” - ”Baloch”
za = 3 / 2,
za = 555 / 3
za = 10.33 % 3,
za = 3234.343 % 23.445,
```

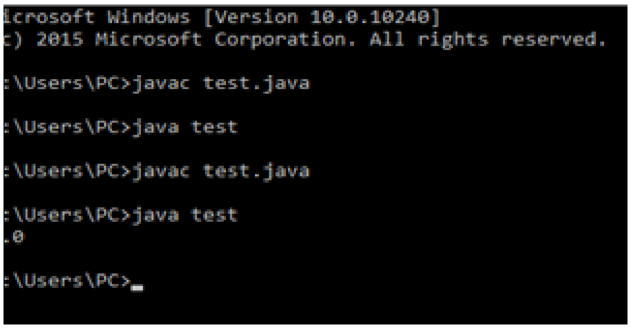
With the above programming code, the system will generate a compile time error which is usually generated when there is a syntax mistake occurs in the programming code. But here it can clearly be seen that there is no any syntax mistake occur in the above programming code. The use of above statements

restricts the whole program to be executed which ultimately sometimes causes the whole software program failure.
To overcome this problem, we extended the existing exceptional model of Java language. Using the exceptional model, the system will somehow manage to avoid the errors. By importing our proposed package, the system will successfully compile the source code and produce results accordingly. However, regarding the exceptional issues, the system will not block the whole program to be executed although it provide the run time error with the error message (written in the body of the function) at the end of the program execution. This method helps the whole program to be executed and provide results of the remaining blocks of the program.

IMPLEMENTATION AND RESULTS

Implementation

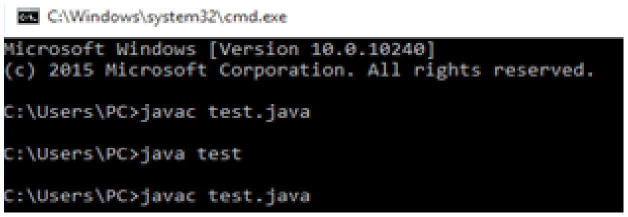
The developed package will be very beneficial for software applications based on the linear systems which use numerical techniques [27]. Hence keeping in view this factor we also select some numerical methods used in the linear systems for implementation of our developed package. In this regards, Bi-Section and Regula-Falsi methods are selected for experimentation. Firstly, we run Regula-Falsi method in Java programming language by using 'Long' data type at some points whereas other data types were also used to check the outcomes of the program. When the program was run without importing the developed package, it was observed that the system produce a compile time error due to the use of invalid data type at some points. Although there is a portion of the program (except the part that uses invalid data type) valid and accurate but due to invalid use of data type at some points the system block the rest of the program to be executed. A second attempt was made with the same code of program just by importing the developed package in to the program. The system successfully compiled the program and showed the results of the program. However, runtime error is occurred after the compilation of the program for that particular block of code where we use invalid data type [28]. Overcompensate of the developed package is that it allows the rest of the program to be executed successfully and provide results accordingly whereas the invalid part of the program shows error message. Figure 6 is the depiction of Regula-Falsi method of with and without importing of the developed package.



```
icrosoft Windows [Version 10.0.10240]
:) 2015 Microsoft Corporation. All rights reserved.

:\Users\PC>javac test.java

:\Users\PC>java test

:\Users\PC>javac test.java

:\Users\PC>java test
.0

:\Users\PC>_
```

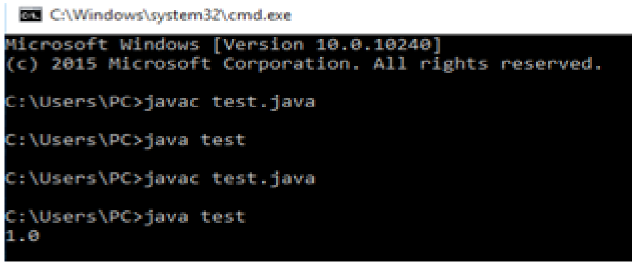
Figure 6. Results of FPM.java and Regula-Falsi program, with/ without import ZAB Package.

Similarly, an experiment is also performed on another numerical method which is used in most of the digital systems for scientific computation tasks known as Bisection method. A program based on bisection is run without importing the developed package and in figure 7(a) it can clearly be seen that the system terminated the program with an error message and didn't allow further compilation. Hence after importing our developed package, the system somehow complete the compilation process and show the output of the program whereas system also show a runtime error after the compilation [29]. Figure 7(b) is the depiction of Bisection method with the implementation of developed method.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\PC>javac test.java

C:\Users\PC>java test

C:\Users\PC>javac test.java
```

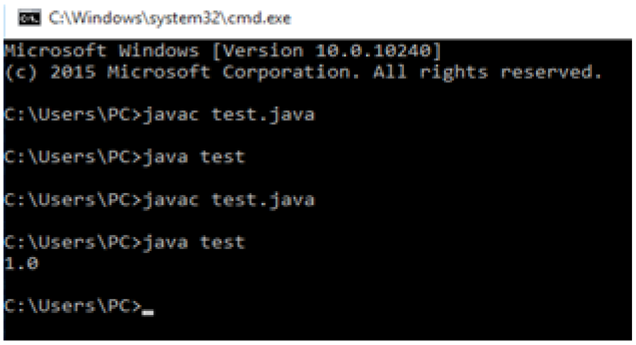
Figure 7(a). Bi-Section.java program, without implementation developed Package



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\PC>javac test.java

C:\Users\PC>java test

C:\Users\PC>javac test.java

C:\Users\PC>java test
1.0
```

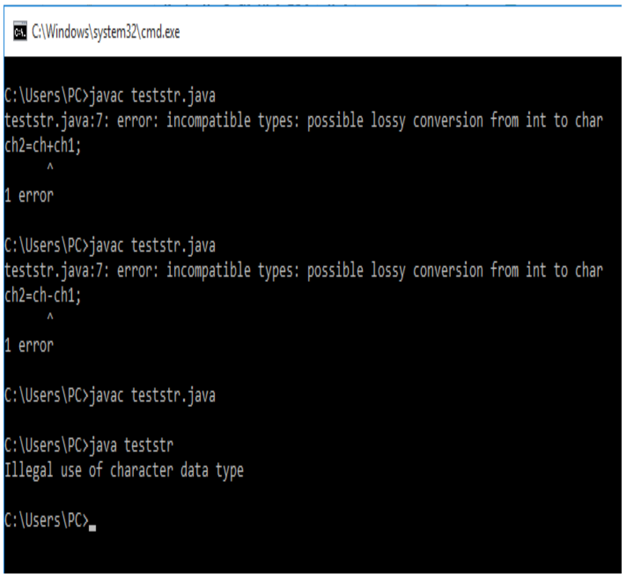
Figure 7(b). Bi-Section.java program, with the implementation developed Package

Likewise an attempt to add, minus, multiply, divide, and reminder the character/string value without implementation of the developed package, the system holds the program to be executed and generated compile time error. Whereas, to successfully run the program we import our developed package after which the system displayed the runtime error after the program compilation. Figure 8(a) and 8(b) are the representation of a Java programs with and without importing the developed package respectively where mathematical operations are performed on the character/string values.



```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\PC>javac test.java

C:\Users\PC>java test

C:\Users\PC>javac test.java

C:\Users\PC>java test
1.0

C:\Users\PC>_
```

Figure 8(a). Results of Test.Java Program without Import ZAB Package

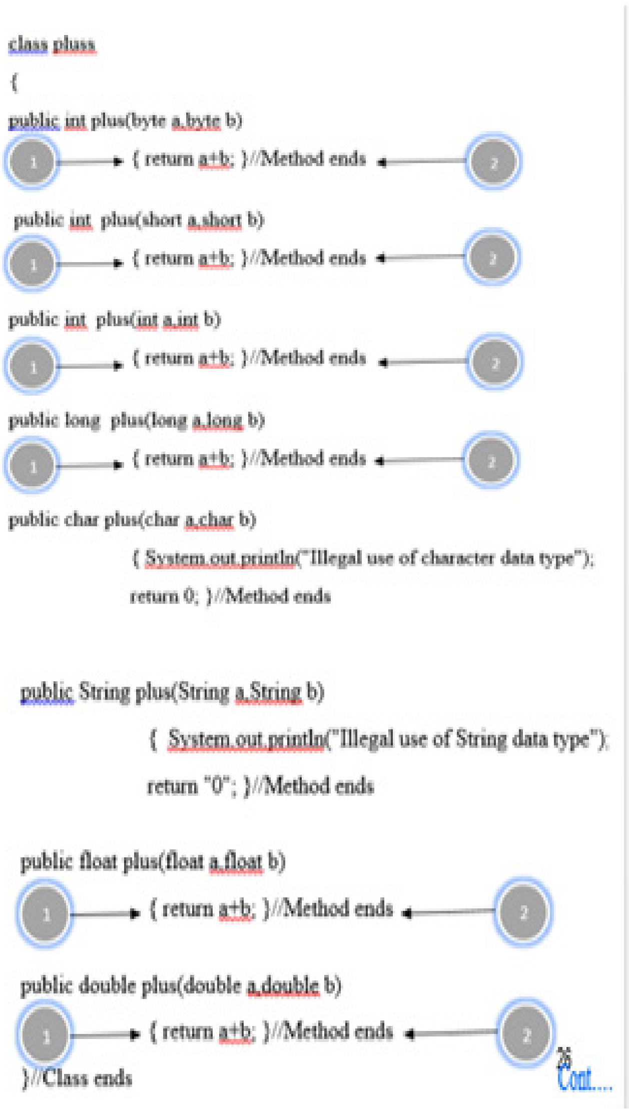


```
C:\Users\PC>javac teststr.java
teststr.java:7: error: incompatible types: possible lossy conversion from int to char
ch2=ch+ch1;
     ^
1 error

C:\Users\PC>javac teststr.java
teststr.java:7: error: incompatible types: possible lossy conversion from int to char
ch2=ch-ch1;
     ^
1 error

C:\Users\PC>javac teststr.java

C:\Users\PC>java teststr
Illegal use of character data type

C:\Users\PC>_
```

Figure 8(b). Result of Teststr.Java program with Import ZAB Package

RESULTS

The eventual impacts of the implementation influence the overall software complexity [30]. Hence, we use MCabCyclomatic complexity (CC) tool which is considered as one of the greatest metric used to measure the complexity of the software after importing the proposed package. This complexity measurement tool uses number of independent paths or flows over the graph. Throughout the graph, modules with the high complexitiy are considered as the modules with low cohesion and vice versa. The CC of the developed class is measured based on the labeled number and matrix connections. At the initial step of CC, numbers and labeled are assigned to Plus class on the basis of its control flow. Figure 9 is demonstration of code of Plus class which show that the numbers are assigned accordingly.

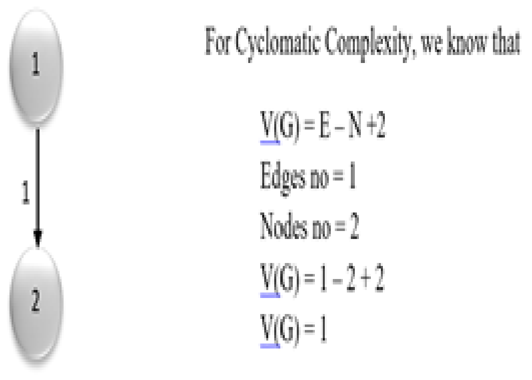


```
class pluss
{
public int plus(byte a,byte b)
        { return a+b; }//Method ends
 public int  plus(short a,short b)
        { return a+b; }//Method ends
public int  plus(int a,int b)
        { return a+b; }//Method ends
public long  plus(long a,long b)
        { return a+b; }//Method ends
public char plus(char a,char b)
        { System.out.println("Illegal use of character data type");
        return 0; }//Method ends

 public String plus(String a,String b)
        {  System.out.println("Illegal use of String data type");
        return "0"; }//Method ends

 public float plus(float a,float b)
        { return a+b; }//Method ends
 public double plus(double a,double b)
        { return a+b; }//Method ends
 }//Class ends
```

Figure 9. show that step 1 has applied to Class Pluss

From above figure it can be seen that labeled class contain 8 overloaded functions i.e. Plus. Each method is set with different types of arguments which perform 8 different arithmetic operations accordingly. Separate CPU processing time is taken by each operation of the method. Number assigned to each method of class Plus separately with the help of CC.

Secondly, according to the labeled number a flow graph is generated for class Plus and calculated CC for each of the method discretely. Figure 10 is the depiction of Flow Graph and CC for Plus method.



For Cyclomatic Complexity, we know that

V(G) = E – N +2

Edges no = 1

Nodes no = 2

V(G) = 1 – 2 + 2

V(G) = 1

Figure 10. Flow Graph and CC of Plus Method that Add Byte type Values

The flow graph and CC of the remaining methods of same calss are also measured accordingly. From the Figure 9 it is observed that due to the equal number of processing instructions and owing the same control flow, CC calculations six out of eight methods are quite similar. Whereas, CC of the program with multiple methods can be measured with the following equation.

V(G) = E-N+2p (i)

Where 'V(G)' is the function used to measure Cyclomatic Complexity. 'E' represents total number of edges, 'N' is the total number of nodes while 'p' represents predictor or module or no of connected regions in G.

For the above program, the complexity can be measure as follows:

V(G) = E-N+2p

Total Edges = 6

Total Nodes = 12

Modules = 6

V(G) = 6 – 12 + 2(6) = 6 (complexity of Class Plus) (ii)

Connection Matrix of Class 'Plus'

The 'Plus' class of Java programming language contain eight overloaded methods. For understanding we produce connection matrix (CM) of six out of the total eight methods as an example in Table 2.

	Connected to Nodes			
Nodes	1	2	3	Connections
1		1		1-1=0
2				0-0=0
				0+1=1

e

Comparative Analysis of Developed Classes

The process of flow graph and connection matrices is done for all the classes such as Class Plus, Minus, Multi, Div and Rem. CC is also measured for all the classes. For Class Pluss V(G)=6, Class Minus V(G)=6, Class Mutlis V(G)=6, Class Divis V(G)=2 and Class Rems V(G)=4. During the comparative analysis it is observed that CC for Class Div is V(G)=2. Table 3. Providesa detailed analysis of five mentioned classes. Hence, the CC of 5 classes i.e Pluss, Minus, Mutlis, Divis and Rems is summarized in Table 3.

Table 3. Comparative Analysis of Developed Classes

Class Name	V(G)	V(G)	V(G)	V(G)	V(G)	V(G)	V(G)	V(G)	Total
Class Plus	1	1	1	1	1	1	0	0	6
Class Minus	1	1	1	1	1	1	0	0	6
Class Multi	1	1	1	1	1	1	0	0	6
Class Divis	1	1	0	0	0	0	0	0	2
Class Rems	1	1	1	1	0	0	0	0	4

CONCLUSION

A language control structure that allows software engineers to express the program's behavior when an exceptional (surprising) event occurs is known as an exemption handling system [31]. When an exception occurs, the program stops preparing and ends, and the framework makes a mistake. This lacking in the software development immensely required a logical solution to tackle the said issue. Lacking during the literature survey provides a way to develop such a framework which can handle arithmetic exceptions during the program execution. Hence, in this study we develop a mechanism in order to tackle the said issue. We examine two exceptions that occurred at run-time as well as assembly time, as well as their instrument handling. Apart from various modules and capacities provided by a specific language, it has been discovered that the entire course of dealing with special cases is the same in OO language [32]. When it comes to getting syntactic and semantic exemptions, there is no difference in how the attempt and catch blocks are used. Logically, if semantic exemptions aren't handled at the right time and in the right place, the framework won't be able to produce the desired result. There are five administrators (+, -, *,/, %) classes that are not available in exemption, thus getting the semantic special situations is very important, especially in iterative numerical problems. In this regard, an Extended Exception Handling Model is formulated in this specialty, which described the evolution of the exemption handling model and we included a model with in-related exemption.

Through the strategy overburdening component of the Java programming language, we appended five Arithmetic Exceptions, namely Plus Exception, Minus Exception, Multiply Exception, Divide Exception, and Reminder Exception, to deal with the special cases of arithmetic exceptions on (Byte, Short, Int, Long, Float, Double, long twofold Char, String) data types. Experimentation is carried out in the Java programming language, with the developed model efficiently handling the five referred activities with eight distinct data types. However, more testing is necessary for other operations. The Cyclomatic Complexity (CC) is calculated for each of the five classes individually throughout the experimentation. For example, the CC of Class Pluss is V(G)=6, V(G)=6 of Minss Class, V(G)=6 of Multis Class, V(G)=6 of Divis Class, V(G)=2 of Divis Class, and V(G)=4 of Rems Class. At the point when analyzed CC of the each evolved class, we discovered less intricacy of Divis class among every one of the classes. Structure the determined outcomes, it is proven that all of the classes, with the exception of Divis, are more perplexing. The findings of this research will be highly beneficial, especially for mathematics numerical problems, as well as for understudies, engineers, and scientists in software engineering, mathematics, measures, and other fields.

REFERENCES

[1] M. Baldoni, C. Baroglio, O. Boissier, R Micalizio, S Tedeschi, “Distributing Responsibilities for Exception Handling in JaCaMo”, AAMAS ‘21: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgentSystemsages, pp. 1752–1754, 2021.

[2] E. Robbins, A. King, J. M. Howe, “Backjumping is Exception Handling”, Theory and Practice of Logic Programming, 21(2), pp. 125-144, 2021.

[3] X. Jia, S. Chen, X. Zhou, X. Li, R. Yu, X. Chen, J. Xuan, “Where to Handle an Exception? Recommending Exception Handling Locations from a Global Perspective.”, In 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC), pp. 369-380, 2021.

[4] T. Nguyen, P. Vu, T. Nguyen, “Recommending exception handling code.”, In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 390-393, Sep. 2019.

[5] D. Marcilio, C.A. Furia, “How Java Programmers Test Exceptional Behavior”, IEEE/ACM 18th International, 2021.

[6] E.S. Najumudheen, R. Mall, D. Samanta, “Modeling and coverage analysis of programs with exception handling.”, In Proceedings of the 12th Innovations on Software Engineering Conference (formerly known as India Software Engineering Conference),pp. 1-11, Feb. 2019.

[7] F .Ebert, F. Castor, A. Serebrenik, “An Exploratory Study on Exception Handling Bugs in Java Programs”, IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 552-556, 2020.

[8] Y. Wang, S. Wang, B. Yang, B. Gao, S. Wang, “An effective adaptive adjustment method for service composition exception handling in cloud manufacturing.”, Journal of Intelligent Manufacturing, pp. 1-17, 2020.

[9] L.P. Lima, S. R. Licoln, M. B. Carla, P. Matheus, “Assessing exception handling testing practices in open-source libraries” Empirical Software Engineering, vol. 26 issue. 85, June. 2021.

[10] K. Lin, C. Tao, Z. Huang, “Exception Handling Recommendation Based on Self-Attention Network”In the proceedings of IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Oct. 2021, Wuhan, China.

[11] S. Priyadarshan, H. Nguyen, R. Sekar, “On the Impact of Exception Handling Compatibility on Binary Instrumentation”, In Proceedings of the 2020 ACM Workshop on Forming an Ecosystem Around Software Transformation, pp. 23-28, Nov. 2020

[12] T. Nguyen, P. Vu, T. Nguyen, T, “Code recommendation for exception handling”, In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1027-1038, 2020.

[13] M. Baldoni, C. Baroglio, O. Boissier, R. Micalizio, S. Tedeschi, “Exception Handling in Multiagent Organizations: Playing with JaCaMo.”, In 9th International Workshop on Engineering Multi-Agent Systems, EMAS, 2021.

[14] J. Zhang, X. Wang, H. Zhang, H. Sun, “Learning to handle exceptions”, In 35th IEEE/ACM, 2020.

[15] T.T. Nguyen, P.M. Vu, T. Nguyen, “Recommendation of exception handling code in mobile app development”, 2019, arXiv preprint arXiv:1908.06567.

[16] H. Melo, R. Coelho, C. Treude, “Unveiling Exception Handling Guidelines Adopted by Java Developers”, IEEE 26th International, pp. 128-139, 2019.

[17] T.T. Nguyen, P.M. Vu, T.T. Nguyen, “An Empirical Study of Exception Handling Bugs and Fixes”, ACM Southeast Conference – ACMSE – Session 2: Short Papers – ISBN: 978-1-4503-6251-1, pp. 257-260, 2019.

[18] G. B. De Pádua, “Studying and assisting the practice of java and c# exception handling”, Doctoral dissertation, Concordia University, 2018

[19] G. B. De Pádua, W. Shang, “Studying the relationship between exception handling practices and post-release defects”, In Proceedings of the 15th International Conference on Mining Software Repositories, pp. 564-575, May. 2018.

[20] M. Mehrabi, N. Giacaman, O. Sinnen, “Unobtrusive Asynchronous Exception Handling with Standard Java Try/Catch Blocks.”, In IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 855-864, May. 2018.

[21] R. Coelho, J. Rocha, H. Melo, “A Catalogue of Java Exception Handling Bad Smells and Refactoring”, The Hillside Group, hillside.net, pp. 1-23, 2018.

[22] P. Preetesh, V. Tokekar, “An Investigation of Exception Handling Practices in .NET and Java Environments.”, International Journal of Applied Engineering Research, vol. 13 issue. 5, pp. 2130-2140, 2018.

[23] H. Osman, A. Chiş, J. Schaerer, M. Ghafari, “On the Evolution of Exception Usage in Java Project”, IEEE 24th, pp. 1-5, 2018.

[24] L. Qi, S. Meng, X. Zhang, R. Wang, X. Xu, Z. Zhou, W. Dou, “An exception handling approach for privacy-preserving service recommendation failure in a cloud environment.”, In the Journal of Sensors, vol. 18, issue. 7, 2018.

[25] S. Nakshatri, M. Hegde, S. Thandra, ,“Analysis of exception handling patterns in Java projects: An empirical study”, In IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), pp. 500-503, May. 2016.

[26] L. Guo, S. Manglani, X. Li, Y. Jia, “Teaching autonomous vehicles how to drive under sensing exceptions by human driving demonstrations”, SAE Technical Paper. sae.org, 2017.

[27] G. Raupp, G. Pautasso, C. Rapson, W. Treutterer, J. Snipes, P. De Vries, F. Rimini, “Preliminary exception handling analysis for the ITER plasma control system.”, Fusion Engineering and Design, 123, pp.541-545, 2017.

[28] R. Coelho, L. Almeida, G. Gousios, A. V. Deursen, C. Treude, “Exception handling bug hazards in Android.”, Empirical Software Engineering, vol. 22, issue. 3, pp.1264-1304, 2017.

[29] D. McQuillan, “Algorithmic states of exception.”, European Journal of Cultural Studies, vol. 18, issue. 4, pp-564-576, 2015.B after the class declaration as shownbelow.

[30] B. Jakobus, E. A. Barbosa, A. Garcia, C. J. De Lucena, “Contrasting exception handling code across languages: An experience report involving 50 open source projects.”, In IEEE 26th International Symposium on Software Reliability Engineering (ISSRE), pp. 183-193, Nov. 2015.

[31] G.S. Sheikh, “A qualitative study of major programming languages: teaching programming languages to computer science students.”, In the Journal of Information & Communication Technology (JICT), vol. 10, issue. 1, 2016.

[32] F. Ebert, F. Castor, A. Serebrenik, “An exploratory study on exception handling bugs in Java programs.”, In Journal of Systems and Software, vol. 106, pp. 82-101, 2015.