Is it Possible to Develop a Language without Exception Handling?

Richa Baijal
Student, M.Tech,
Computer Science and Engineering
Career Point University,
Alaniya, Jhalawar Road,
Kota-325003 (Rajasthan)

ABSTRACT
In this paper, a knowledge of library files that are used to handle exceptions in different languages is provided. The languages taken C,C++ and Java are taken as the reference languages. This paper provides the knowledge about library files used to handle exceptions in these languages which are very important from a programmer/researchers'point of view.

General Terms
Library files in C, C++ and Java ; error handling in C, Exceptions.

Keywords
Exceptions, Library files in C ,C++,Java; error handling, exception handling.

1. INTRODUCTION
Abnormal condition which occurs during runtime during the execution of a program is called an exception/runtime error. Runtime error is in a way different from exception. If there is a runtime error, it is due to some hardware malfunction whereas an exception is caused due to improper logic provided in the program. When such a condition is encountered, it has to be resolved by the language constructs/compilers so that reliability of the language is maintained.

2. STUDY OF LANGUAGES TO GET AN IDEA ABOUT THE LIBRARY FILES THAT HANDLE EXCEPTIONS/ERRORS

2.1 Error Handling In C :
C does not provide direct support for error handling . C functions return -1 or NULL when a error occurs in the program. They also set an error code **errno** which makes a programmer understand that error has occurred in the program . Let us consider an example and then understand the functions and header files associated with it.

Example :

```
#include <stdio.h>

#include <errno.h>   // Various Error Codes Are Defined in This Header File

#include <string.h>

extern int errno ;

int main ()
{
        FILE * p;

        int errnum;

        p = fopen ("alpahbet.txt", "rb");


                if (p == NULL)

        {

  errnum = errno;

  fprintf(stderr, "Value of errno: %d\n", errno);

  perror("Error printed by perror");

 fprintf(stderr, "Error opening file: %s\n", strerror( errnum ));

        }

        Else

        {

         fclose (p);

        }

        return 0;

}
```

Well, now just have a quick look at the program and let us understand it simply in one line :

This program will return a '0' if everything goes well i.e the file which i am looking for is found.

But,unfortunately things don't go well for me. I get the following Output for this program :

Output :

```
Value of errno : 2

Error printed by perror : No such file or directory

Error opening file: No such file or directory
```

Pretty Cool ! Now, let us understand how is it achieved in C.

Well, C has a header file to handle the errors <errno.h> the same we included while writing our program. This is a library file in C and contains macros which identify what type of error has occurred by displaying the no. In integer format.Let us have a look at this macros list :

Macros	Error No.	Type of Error
#define EPERM	1	Operation Not permitted
# define ENOENT	2	No such file or directory
# define ESRCH	3	No such process
# define EINTR	4	Interrupted system Call
# define EIO	5	I/O Error
# define ENXIO	6	No such device or address
# define E2BIG	7	Argument list too long
# define ENOEXEC	8	Exec format error
# define EBADF	9	Bad file number
# define ECHILD	10	No child processes
# define EAGAIN	11	Try again
# define ENOMEM	12	Out of memory
# define EACCES	13	Permission denied
# define EFAULT	14	Bad address
# define ENOTBLK	15	Block device required
# define EBUSY	16	Device or resource busy
# define EEXIST	17	File exists
# define EXDEV	18	Cross device link
# define ENODEV	19	No such device
# define ENOTDIR	20	Not a directory
# define ISDIR	21	Is a directory
# define INVAL	22	Invalid argument
# define ENFILE	23	File table overflow
# define EMFILE	24	Too many open files
# define ENOTTY	25	Not a typewriter
# define ETXTBSY	26	Text file busy
# define EFBIG	27	File too large
# define ENOSPC	28	No space left on device
# define ESPIPE	29	Illegal seek
# define EROFS	30	Read only file system
# define EMLINK	31	Too many links
# define EPIPE	32	Broken pipe

The macro list clearly defines our error code no.2 i.e. it is unable to find the file we specified.

It says : No such file or directory.

There are 131 such error codes in C defined under <error.h> library.

Let us quickly examine two more functions and their work in error handling :

1. **perror()** : It returns the textual value of current errno.
   As in our example, error code is 2. Since,we have used the perror() function it is returning the string value for the corresponding error code.
2. **Strerror()** : It returns the pointer to the textual representation of current errno value.

Stderr is a file stream also called as Standand Error Stream which is used to output all the errors.

2.2 Exception Handling in C++ :

Exception is a condition that occurs during runtime. In C++ ,an exception is handled using three keywords or block namely try catch and throw.[2]

1. **throw** : A program throws an exception when it first encounters it. It shows that the program has encountered a problem.
2. **catch** : catch represents the catching of an exception. In other words,it specifies the place in the program where you want to handle the problem.
3. **try** :A try block identifies the block of code for which particular exceptions will be activated.

It is followed by one or more catch blocks.

Let us take a n example to understand Exception Handling in C++ :

Example:

```
#include <iostream>

using  namespace std;

double division(int  p, int q)

{

                if( q == 0 )

                {

                 throw "Division by zero condition!";

                }

                return (p/q);

                }

int main ()

{

  int  k = 50;

  int  l  = 0;

  double  m= 0;

  try

                 {

                 m = division(k, l);

                cout << m << endl;

                }

                catch (const char* msg)

                {

                 cerr << msg << endl;

                  }


  return 0;

}
```

Output : Division by Zero condition !

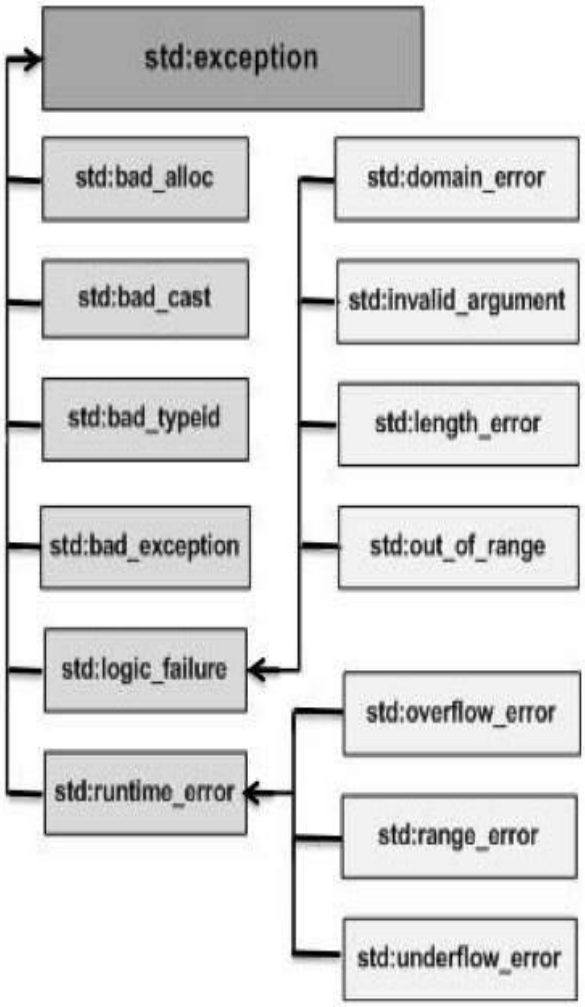
Since, an exception of type const char* is raised, therefore while catching the exception it is mentioned in catch block .

A short note about using namespace standard routine here :

The std namespace is special; it is short for the word "standard." The built in C++ library routines are kept in the standard namespace. That includes stuff like cout, cin, string, vector, map, etc. Because these tools are used so commonly, it's popular to add "using namespace std" at the top of your source code so that you won't have to type the std:: prefix constantly. And because these functions are kept in a namespace, if you really want to use "vector" as a variable name, you still can. Namespaces give you more freedom to use short, accurate names.

Let us now study what type of exceptions are handled by C++ :[3]

There are standard exceptions which are defined in <exception> header file in C++. They are arranged in parent-child class hierarchy as shown :



The table provides a definition of these exceptions in brief :

Exception	Description
std::exception	An exception and parent class of all the standard C++ exceptions.
std::bad_alloc	This can be thrown by **new**.
std::bad_cast	This can be thrown by **dynamic_cast**.

std::bad_exception	This is useful device to handle unexpected exceptions in a C++ program
std::bad_typeid	This can be thrown by **typeid**.
std::logic_error	An exception that theoretically can be detected by reading the code.
std::domain_error	This is an exception thrown when a mathematically invalid domain is used
std::invalid_argument	This is thrown due to invalid arguments.
std::length_error	This is thrown when a too big std::string is created
std::out_of_range	This can be thrown by the at method from for example a std::vector and std::bitset<>::operator[]().
std::runtime_error	An exception that theoretically can not be detected by reading the code.
std::overflow_error	This is thrown if a mathematical overflow occurs.
std::range_error	This is occured when you try to store a value which is out of range.
std::underflow_error	This is thrown if a mathematical underflow occurs.

C++ also provides a feature where user can override the exception class and can define a new exception :

Example: Overriding std :: exception class to create your own exception

```
#include <iostream>
#include <exception>

using  namespace std;

struct MyException : public exception

{

 const char * what () const throw ()

 {

   return "C++ Exception";

 }

};

int main()

{

 try

 {

  throw MyException();

 }

 catch(MyException& e)

 {

 std::cout << "MyException caught" << std::endl;

std::cout << e.what() << std::endl;

 }

 catch(std::exception& e)

 {

  //Other errors

 }

}
```

Output :

```
MyException caught

C++  Exception
```

Use of what () : what() is a public method provided by exception class and it has been overridden by all child exception classes. It returns the cause of an exception.

2.3 Exception Handling in Java

There are two types of exceptions in Java.

1. **Checked Exceptions** :All exceptions other than runtime exceptions are referred to as Checked Exceptions.

Examples of checked Exceptions :

(i) ClassNotFound Exception

(ii) Illegal AccessException

(iii) NoSuchFieldException

(iv) EOFException, etc.

2. **Unchecked Exceptions** : All run-time exceptions are called unchecked exceptions.

Examples of unchecked Exception :

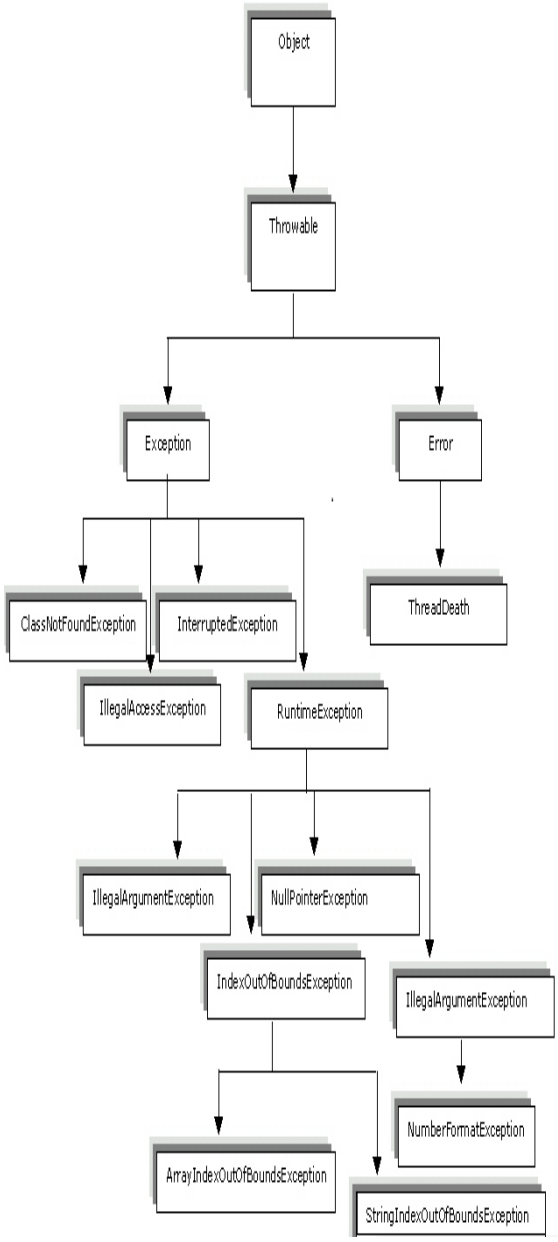
(i) Arithmetic Exception

(ii) ArrayIndexOutOfBoundsException

(iii) NullPointer Exception

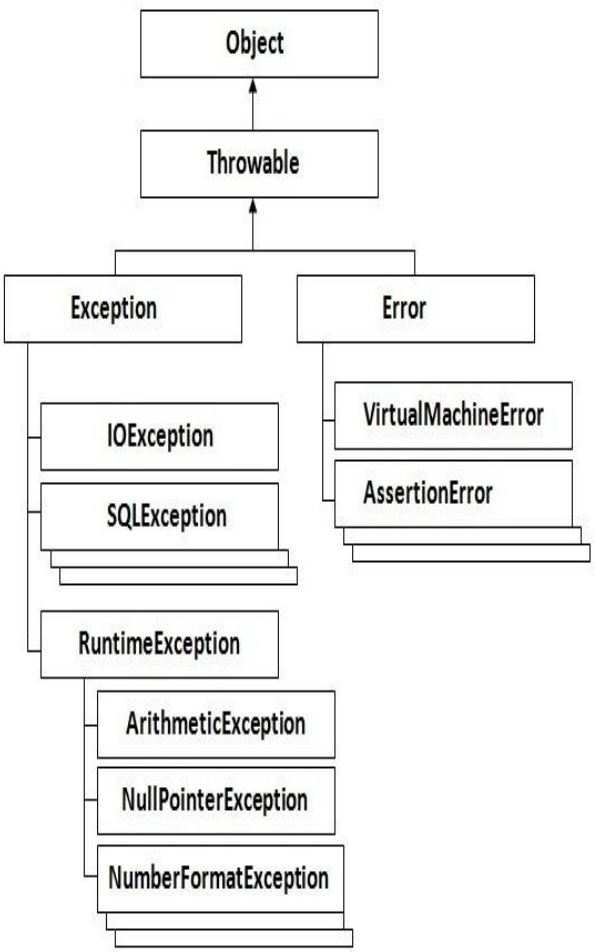
(iv) NegativeArraySizeException,etc.

Exception Heirarchy in Java :[4,6]



Heirarchy of Java exception Classes :[4,5]

So far,we have seen seen exception classes or macros in C++ and C respectively. Java is awidely used language with wide no. Of classes defined to handle exceptions. Here is a look :



Java uses 5 keywords to handle exceptions.

These are :

1.try:-In try block we enclose the code that might throw an exception.Try block is necessary followed by a catch of finally block in Java.

2.catch:Catch block is used after the try block.It is used to handle the exception.

3.finally : finally block is written after throw/throws block. It is always executed whether an exception occurs or not.

4. throw: It is used to throw an exception that is being caught in the program.

5.throws: If a method cannot handle the checked exception, the method must declare it using throws keyword .

Let us make exception handling in Java more clear by this example:

```
public class ExcepTest{

  public static void main(String args[]){
    int a[] = new int[2];
    try{
      System.out.println("Access element three :" + a[3]);
```

```
    }catch(ArrayIndexOutOfBoundsException e){
      System.out.println("Exception thrown  :" + e);
    }
    finally{
      a[0] = 6;
      System.out.println("First element value: " +a[0]);
      System.out.println("The finally statement is executed");
    }
  }
}
```

The result of above program will be :

Output :
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3
First element value: 6
The finally statement is executed

User Defined Exceptions :

Java facilities the programmers to define exception methods themselves. Although the important points to be kept in mind while doing so are :

1. All exceptions must be a child of Throwable class.
2. If you are writing a checked exception,that is automatically enforced by the handle,you should extend the Exception class.
3. If you are writing a runtime Exception, you need to extend the RuntimeException class.

3. FUTURE AREA OF RESEARCH WORK

Many modern languages provide rich exception handling features .Three of them,we have gone through. The curiousity revolves around the concept: “Is it possible to design a language without exception handling feature ?”.If it is done,what will be gained ?

Although, Swift language from Apple does not provides exception handling feature but it does provide assertion and return values which is a kind of error handling mechanism introduced by C.

Why this kind of programming world is desirable ?

Well, a simple answer to this question would be that if processor architecture can be reduced from CISC to RISC ; then why not this feature ? Certain algorithms can definitely enhance this idea of a programming language without exception handling.

4. CONCLUSION

Error handling and exception handling library files of different languages have been studied to gain a new lookout of developing a language that will not handle exceptions in future and will seem an equally reliable language then also.

5. REFERENCES

[1] Yashavant P. Kanetkar . Let Us C ; Infinity Science Press, 2008

[2] Yashavant P. Kanetkar . Let Us C++ ; Infinity Science Press, 2008

[3] Nicolai M. Josuttis,The C++ Standard Library: A Tutorial And Reference

[4] Stephen Stelting,Robust Java: Exception Handling, Testing, and Debugging; Prentice Hall PTR, 2004.

[5] Herbert Schildt,Java2:The Complete Reference.

[6] Bruce Eckel,Thinking In Java;Pearson Education,India.