© October 2015 | IJIRT | Volume 2 Issue 5 | ISSN: 2349-6002

Exception Handling

Gourav Agghi

***Abstract-* This paper proposes about the exception handling and what is its role in java. When an error occur within a method, the method creates an object and hands it off to the runtime system. this object called an exception object,contains information about the error and including its types.**

***Definition-* An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.**

I. INTRODUCTION

After a method throws an exception ,the runtime system attempts to find something to handle it. The set of possible "somethings"to handle the exception is the ordered list of methods that had been called to get to the method where the error occurred. The list of method is also known as stack. The runtime system searches the call stack for a method that contain a block of code that can handle the exception. This block of code is called an exception handler. The search begin with the method in which the error occur and proceeds through the call stack in the reverse order in which the method were called.when an appropriate handler is found, the runtime system passes the exception to the handler. An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler.

1.1. The Catch or Specify Requirement

Valid Java programming language code must honor the Catch or Specify Requirement. This means that code that might throw certain exception must be enclosed by either of the following:

- ⇨ A try statement that catches the exception. The try must provide a handler for the exception, as described in Catching and Handling Exceptions.
- ⇨ A method that specifies that it can throw the exception. The method must provide a throw clause that lists the exception, as described in specifying the Exception Thrown by a method.

Code that fails to honor the catch or specify Requirement will not compile. Not all exceptions are subject to the Catch or Specify Requirement. To understand why, we need to look at the three basic categories of exceptions, only one of which is subject to the requirement.

Exception Handling has three kind of exceptions.

1.) Checked Exception
2.) Error Exception
3.) Runtime Exception.

- ⇨ The first kind of exception is the checked exception. These are exceptional conditions that a well written application should anticipate and recover from. For example, suppose an application prompts a user for an input file name ,then opens the file by passing the name to the constructor for java.io.FileReader object succeeds, and the execution of the application proceeds normally.But sometime the user supplies the name of nonexistent file, and the constructor throws java.io.FileNotFoundException. A well written programwill catch this exception and notify the user of this mistake,possibly prompting for a corrected file name.
- ⇨ The second kind of exception is the error. These are exceptional conditions that are external to the application, and that the application usually cannot anticipate or recover from.For example, suppose that an application successfully opens a file for input, but is unable to read the filebecause of hardware or system malfunction. The unsuccessful read will throw java.io.IOError. An application might

choose to catch the exception, in order to notify the user of the problem but it also might make sense for the program to print a stack trace and exit.

Errors are not subject to Catch or Specify Requirement. Errors are those exceptions indicated by error and its subclasses.

⇨ The third kind of exception is the runtime exception. These are exceptional conditions that are internal to the application,and that the application usually cannot anticipate or recover from.These usually indicate programming bugs,such as logic errors or improper use of an API. For example, consider the application described previously that passes a file name to the constructor for File Reader. If a logic errors causes a null to be passed to the constructor, the constructor will throw NullPointerException. The application can catch this exception, but it probably makes more sense to eliminate the bugs that caused the exception to occur.

Runtime exceptions are not subject to catch or specify requirement. Runtime exception are those indicated by RuntimeException and its subclasses.

Errors and runtime exceptions are collectively known as unchecked exceptions.

1.2. Catching and Handling Exceptions

The section describes how to use the three exception handler components-the try,catch,and finally blocks-to write an exception handler. then, the try-with resources statement, introduced in java SE 7,is explained. The try-with resources statement is particularly suited to situations that use closeable resources such as streams.

1.3. The Try Block

The first step in constructing an exception handler is to enclose the code that might throw an exception within a try block. In general, a try looks like the following:

```
try{
    code
}
Catch and finally blocks…
```

The segment in the example labeled code contain one or more legal line of code that could throw an exception.

1.4. The Catch Block

You associate exception handlers with a try block by providing one or more catch blocks directly after the try block. No code can be between the end of try block and the beginning of the first catch block.

```
try{
} catch(ExceptionType name) {
}catch (ExceptionType name) {
}
```

Each catch block is an exvception handler that handles the type of exception indicated bu its argument. The argument type, ExceptionType, declares the type of exception that the handler can handle and ust be the name of a class that inherits from the Throwable class.The handler can refer to the exception with name.

II. CONCLUSION

This paper concludes about the various topics of Exception Handling. It also discuss about the different types of blocks.

III. REFERENCES

[1]. https://en.wikipedia.org/wiki/Exception_handling

[2]. https://en.wikipedia.org/wiki/Exception_handling_syntax

[3] https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html