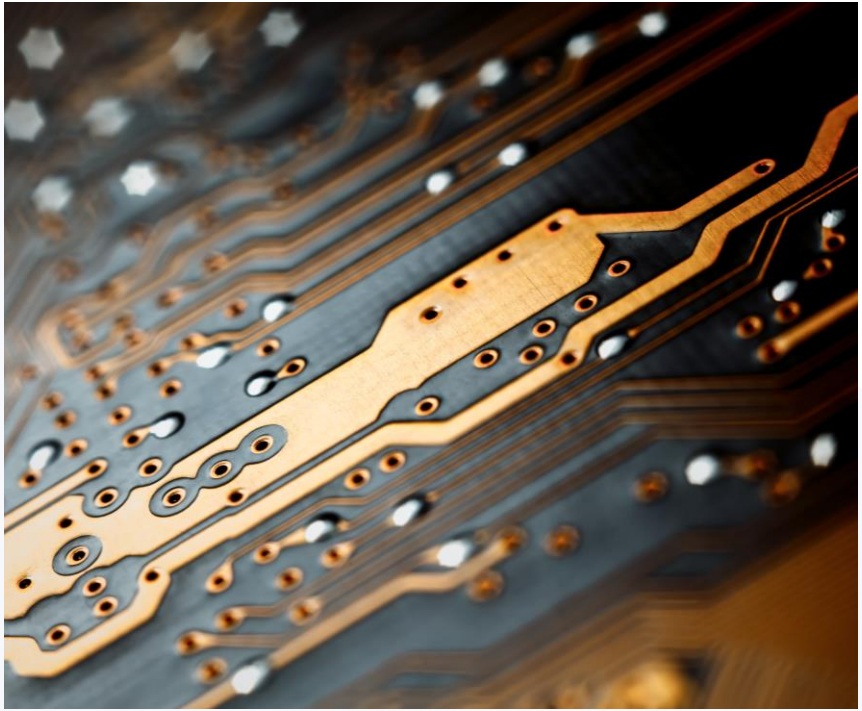
Xebia

Exception Handling

Exception

- An exception is an event or condition that disrupts the normal flow of a program's execution. In Java, exceptions are objects that represent these exceptional situations. When an exception occurs, it can be caught and handled, preventing the program from crashing and allowing for error recovery and graceful termination. Exceptions in Java are categorized into checked and unchecked exceptions. Checked exceptions must be explicitly caught or declared in a method's signature, while unchecked exceptions (usually subclasses of Runtime Exception) do not require explicit handling.



Types of Exception

1) Checked Exception : the Exception checked by the Compiler

- Checked exception should be handle before compilation, with the help of try or throws key word otherwise it will raise Exception on compilation time

2) Unchecked Exception : run time Exception which will not be checked by the compiler

Differences between checked and unchecked Exception

Checked Exception

- Compiler will not check this Exception
- Try catch block are used to handle this Exception

Ex

- Arithmetic Exception

Unchecked Exception

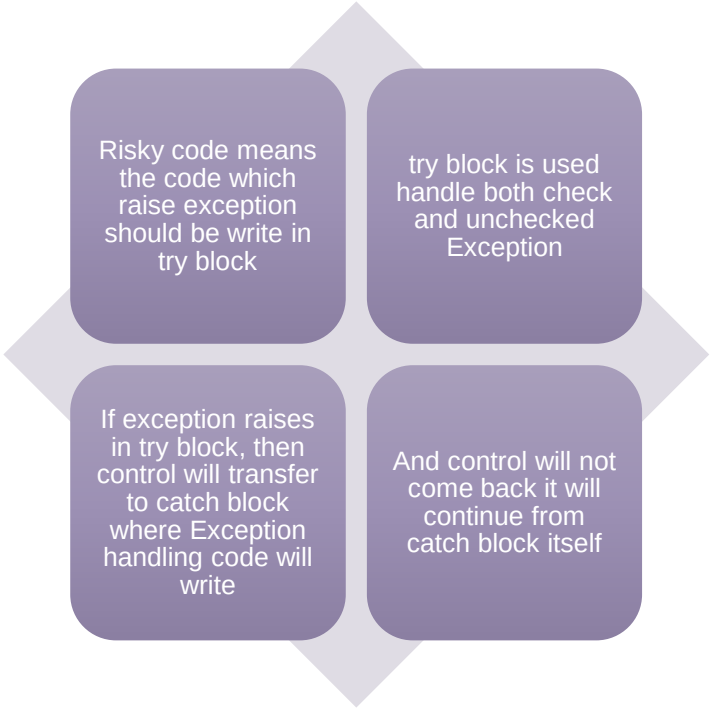
This exception checked by compiler

Throws key word and try catch block is used to handle this Exception

Ex :

- io exception

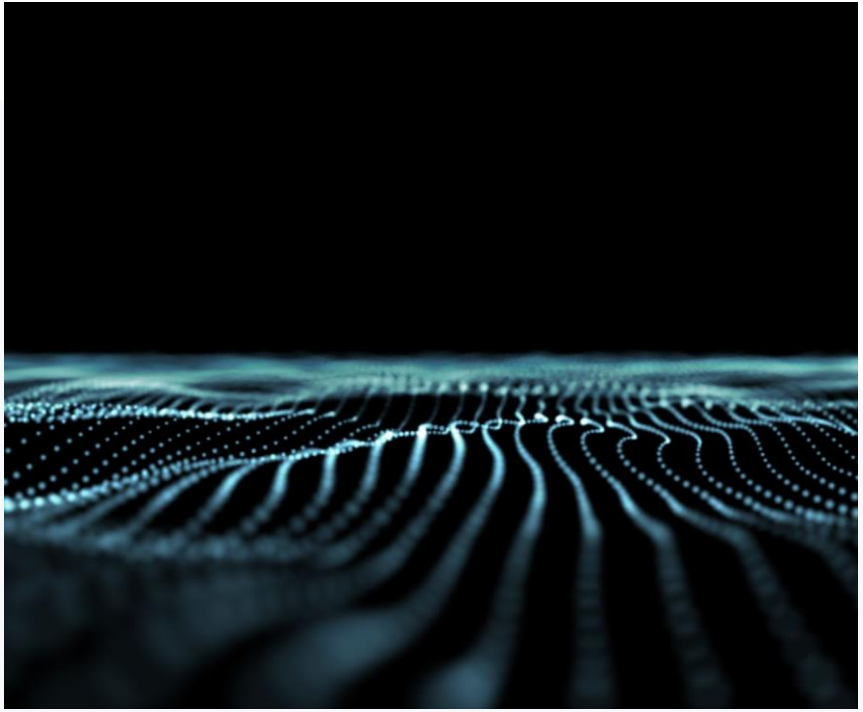
Try Block


Risky code means the code which raise exception should be write in try block
try block is used handle both check and unchecked Exception
If exception raises in try block, then control will transfer to catch block where Exception handling code will write
And control will not come back it will continue from catch block itself

Catch Block

In this block the Exception handling code will written

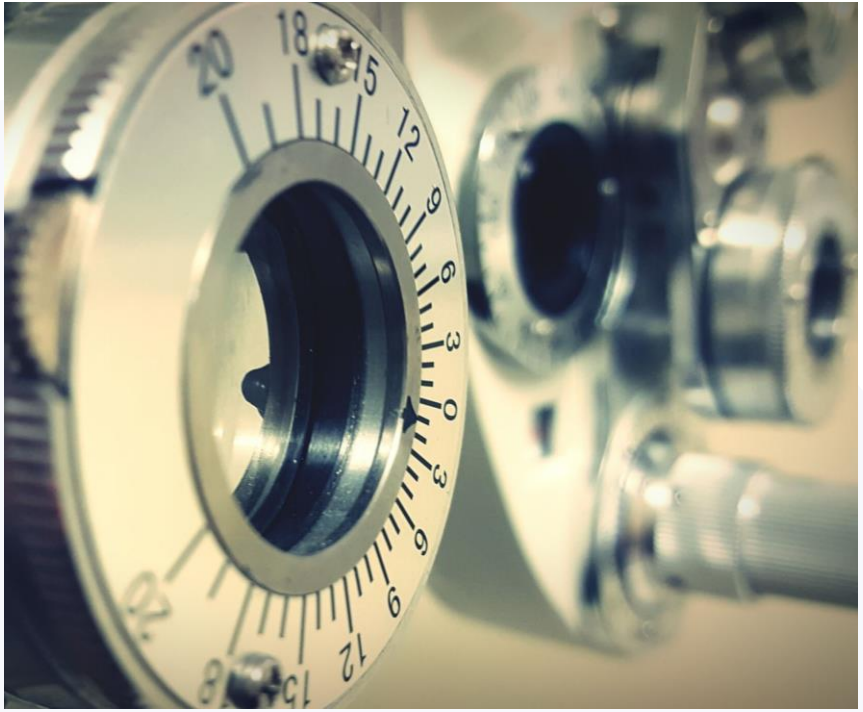
- If we did not handle the Exception, then default Exception handler will handle the Exception and terminate the execution
- Catch Block always followed by try



Finally Block

Finally Block will always execute regarding of raising Exception

- Finally always with be try or try catch block
- Finally block mostly used to close the connection which are open



Differences between final, finally, finalize()

final

- final is a key word applicable to class variable , method
- When you declare variable as final then its value can't be change
- When you declare method as final then you cannot override that method
- When you declare class as final then you cannot inherit that class

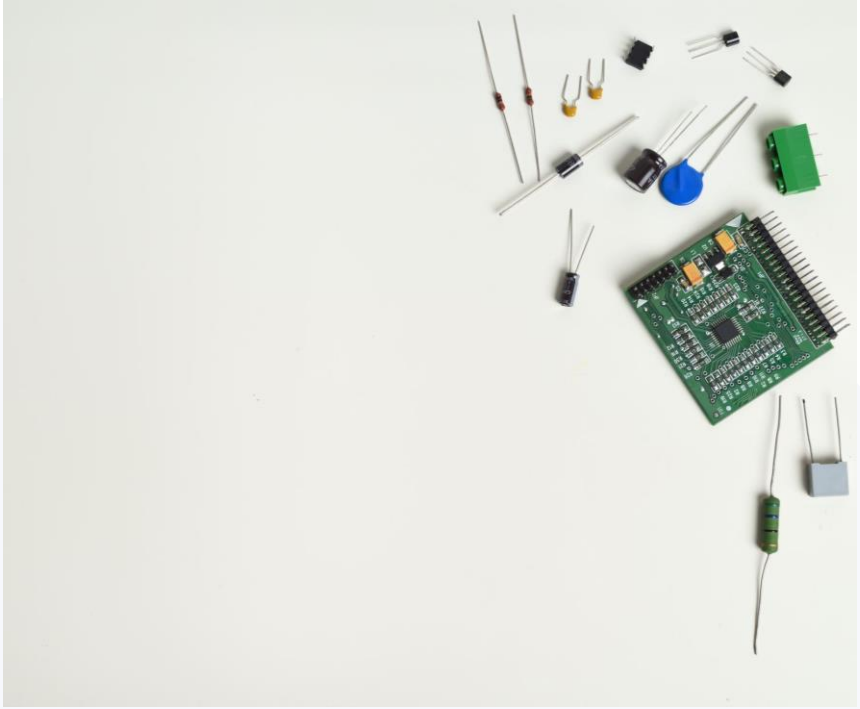
Differences between final, finally, finalize()

finally

- Finally is a block always associated with try catch block
- This block is used to write clean up code for open connection

finalize()

- Is a method of Object whose work is to close the connection.
- finalize() is called after the Object is destroyed.



Exception Hierarchy

- Java has a hierarchical exception class structure, with the Throwable class at the top. Throwable is the base class for all exceptions in Java.
- Exceptions are categorized into two main branches: checked exceptions (subclasses of Exception) and unchecked exceptions (subclasses of Runtime Exception).
- Error is another branch, representing serious, typically unrecoverable errors that should not be caught or handled by application code.
- The Exception class is further divided into various subclasses, such as IO Exception, SQL Exception, and Null Pointer Exception, to represent different types of exceptional situations.
- The exception hierarchy allows for specific exceptions to be caught and handled while letting more general exceptions propagate up the call stack when necessary.

Control Flow in Exception

- Control flow in Java exceptions is managed using try, catch, finally, and optionally, throw and throws statements.
- A try block encloses the code where an exception may occur.
- A catch block follows a try block and contains code to handle specific exceptions. Multiple catch blocks can be used to handle different exception types.
- A finally block, if present, is executed regardless of whether an exception occurred or not. It's typically used for cleanup tasks.
- The throw statement is used to explicitly throw an exception within code.
- The throws keyword in method signatures indicates that a method may throw certain types of exceptions, which must be handled by the caller or propagated to the caller's caller.

VM Reaction to Exception

- When an exception is thrown in Java, the VM (Virtual Machine) initiates an exception-handling mechanism.
- The VM searches for a matching catch block that can handle the thrown exception. If a matching catch block is found, it executes the code within that block.
- If no suitable catch block is found in the current method, the VM proceeds to the calling method and continues the search up the call stack.
- If no appropriate catch block is found at any level in the call stack, the program terminates, and an error message or stack trace is generated, indicating the unhandled exception.
- The finally block, if present, is executed regardless of whether an exception is caught or not. This block is typically used for cleanup tasks.

Try Catch

- The try-catch block is used to handle exceptions in Java.
- Code that might throw an exception is placed within the try block.
- If an exception occurs within the try block, the program execution is transferred to the catch block, where you can specify how to handle the exception.
- Multiple catch blocks can be used to handle different types of exceptions, allowing you to provide specific error-handling logic.
- The catch block is optional, but if used, it must immediately follow the try block.
- The catch block catches and handles exceptions based on their types.
- The finally block (optional) can be used to execute cleanup code that runs regardless of whether an exception occurs or not.
- The try-catch block allows for graceful error handling and prevents program crashes due to unhandled exceptions.

Throws

- The throws keyword is used in a method's declaration to indicate that the method may throw certain types of exceptions.
- When a method includes the throws keyword, it informs callers of the method about the exceptions they need to handle or propagate.
- Methods that declare checked exceptions using throws must either catch and handle those exceptions using a try-catch block or declare the exceptions themselves using throws in their own method signature.
- The throws keyword is typically used for checked exceptions that are not directly handled within the method but are passed up the call stack to be handled by calling methods or by the program's main method.

Try with Resources

- "Try with Resources" is a Java feature introduced in Java 7 that simplifies resource management, such as file handling or database connections, by automatically closing resources at the end of a try block.
- You can use the try block with parentheses to declare and initialize resources that implement the Auto Closeable interface.
- When the try block exits, whether normally or due to an exception, the resources are automatically closed in the reverse order of their creation.
- This feature ensures that resources are properly released, reducing the risk of resource leaks and improving code readability.
- Example resources include FileInputStream, FileOutputStream, and database connections.
- The "Try with Resources" syntax reduces the need for explicit finally blocks to close resources manually.

Custom Exception

- Custom exceptions, also known as user-defined exceptions, are exceptions that you create by extending existing exception classes or creating your own exception class.
- You can define custom exceptions to represent specific error conditions in your application that are not covered by built-in exceptions.
- Custom exceptions are typically derived from the Exception class or its subclasses.
- By creating custom exceptions, you can provide meaningful error messages and handle application-specific error scenarios more effectively.
- To create a custom exception, you define a new class that extends Exception or a subclass of Exception, and you can add custom fields and methods as needed.

xebia.com