

Name: Gautam K K

Department: Impact Trainee

Task: Java

Date: 25/11/24

## Exception Handling

### Exceptions

- Java language supports the exception handling mechanism
- Exception are the unwanted error or bugs that interrupt the normal execution of program during the runtime and displays the error message
- The parent class of all the exceptions are “java.lang.Exception” class
- The super class of the exception is Throwable class it has subclasses --> Exception and error

There are mainly two types of exceptions in Java as follows:

- Checked exception
- Unchecked exception

### Checked exception

Checked exception are also called as compile time exception, these exceptions are checked by the compiler during the compilation time to check whether programmer is handled the exception are not, if not it shows compilation error

Some of Checked exceptions are

- SQL Exception
- IO Exception
- ClassNotFoundException
- FileNotFoundException
- InterruptedException

### Unchecked Exceptions

The exceptions that occur during the runtime of the program is called as unchecked or Runtime exceptions , these are not checked during the compilation process

Some of Unchecked exceptions are

- ArithmeticException
- ArrayIndexOutOfBoundsException
- IllegalArgumentException
- NullPointerException
- NumberFormatException

### Number of Predefined Exception

In java there are more than 50 predefined exceptions in **java.lang**, **java.io**, **java.sql**, **java.util**

The number will be changing as the new version got updated.

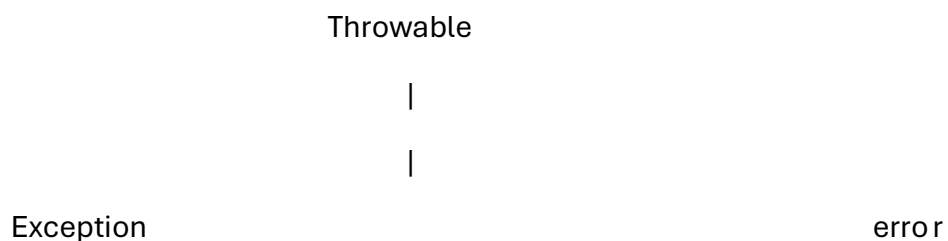
### Finally Block

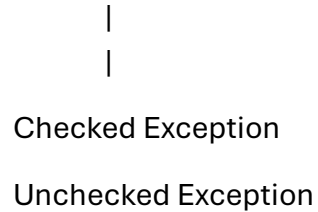
- The finally block is followed by the try-catch block
- It is used to execute the code that run regardless of exceotion is occured or not
- Finally block is executed even if the try/catch block has the return, break or continue is present
- Mainly used for cleaning up the resources like closing the databse connections, closing the file, releasing the locks if using thread safety.

Finally not execute

- JVM crash or termination – example if the code is forcefully terminated like `system.exit()` method
- Thread Interruption – if the exception handling is done by specific thread that the thread is Interrupt using the `threadname.interrupt()`

### Diff between Exception & Error





In java, exceptions and error are the subclass of throwable class

**Exception:** It is the condition that can be handled programmatically, exceptions are the abnormal conditions that can except and handled

**Error:** Error is the serious problem that cannot be handled and it is due to system failures

- `OutOfMemoryError`: Raised when the JVM runs out of memory.
- `StackOverflowError`: Happens when the stack overflows due to deep recursion.
- `VirtualMachineError`: Represents errors related to the JVM itself

Feature	Exception	Error
Inheritance	Inherits from Throwable → Exception	Inherits from Throwable → Error
Recoverability	Generally recoverable	Generally not recoverable
Handling	Handled using try-catch or throws	Not typically handled by the program
Examples	<code>IOException</code> , <code>SQLException</code> , <code>NullPointerException</code>	<code>OutOfMemoryError</code> , <code>StackOverflowError</code>
Occurrence	Caused by abnormal conditions that can be fixed (e.g., user input errors, I/O issues)	Caused by JVM or system issues that cannot be fixed (e.g., memory issues, stack overflow)
Checked/Unchecked	Can be checked or unchecked	Always unchecked

## Catch Blocks

- The number of catch blocks in a single try block is not strictly limited,
- One try Block can Have Multiple catch Blocks:
- You can have multiple catch blocks after a single try block to handle different types of exceptions.

## Order of catch Blocks:

- catch blocks should be ordered from **specific to general**. This is because Java checks catch blocks from top to bottom, and once it finds a match, it will not check the remaining catch blocks.
- Use General exception only in the last catch block

## Example:

```
public class ExceptionsDemo {

    ExceptionsDemo(){
        System.out.println("Instance is initailaized");
    }
    public int divideNumbers(int number1, int number2)
    {
        if(number2==0) {
            throw new ArithmeticException("Cannot be divide by zero");
        }
        return number1/number2;
    }
    public static void main(String[] args) {

        try {
            ExceptionsDemo exdemo = new ExceptionsDemo();
            System.out.println("Exception causing Method is called");
            int result = exdemo.divideNumbers(10,0);
        }
        catch(ArithmeticException e) {
            System.out.println("Caught ArithmeticException: " + e.getMessage());
        }finally {
            System.out.println("Cleanup code in finally block.");
        }
    }
}
```

}

}

```
<terminated> ExceptionsDemo [Java Application] C:\Users\gautam.krishnasamy\.p2\pool\plugins\org.eclipse.justj.openjdk.hot
Instance is initailaized
Excption causing Method is called
Caught ArithmeticException: Cannot be divide by zero
Cleanup code in finally block.
```

## Custom Exception

```
class InvalidAgeException extends Exception{
    static String msg = "You age is not egilible from Invalid Age Exception";
    InvalidAgeException(){
        super(msg);
    }
}

public class CustomException {

    public static void checkAge(int age) {
        try {
            if(age < 18) {
                throw new InvalidAgeException();
            }
            else if(age >=18 && age<100){
                System.out.println("You are eligible");
            }
            else {
                throw new Exception();
            }
        }
        catch(InvalidAgeException e) {
            System.out.println("Exception Error :"+ e.getMessage());
        }
        catch(Exception e) {
            System.out.println("UnExcepted Exception Error :"+ e.getMessage());
        }
        finally {
```

```

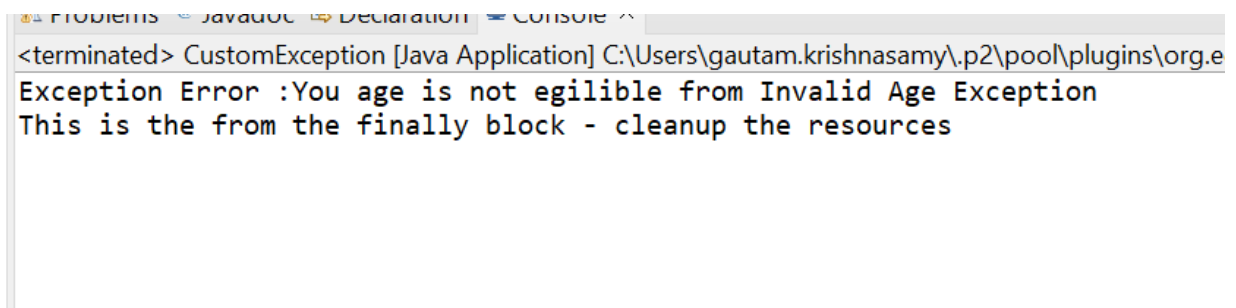
System.out.println("This is the from the finally block - cleanup the resources");
}
}
public static void main(String[] args) {

    //          calling the static method to check
    checkAge(0011);
}

}

```

Output:



```

<terminated> CustomException [Java Application] C:\Users\gautam.krishnasamy\.p2\pool\plugins\org.e
Exception Error :You age is not egilible from Invalid Age Exception
This is the from the finally block - cleanup the resources

```

### Default capacity of collection class:

Collection class	Capacity
ArrayList	10
LinkedList	N/A
Vector	10
Stack	10
Priority Queue	11
Hash map	16
Linkedhashmap	16
Hash Set	16
Linked Hash Set	16
TreeSet	N/A

### Collection class

ArrayList:

Accessing elements: Fast

Shifting elements: Slow

LinkedList:

Accessing elements: Slow

Shifting elements: fast

HashSet: (it compares with specific hash code)

Accessing elements: fast

Shifting elements: fast

TreeMap/TreeSet:

Accessing elements: Slow

Shifting elements: NA

PriorityQueue:

Accessing elements: Slow except first element(fast)

Shifting elements: Slow