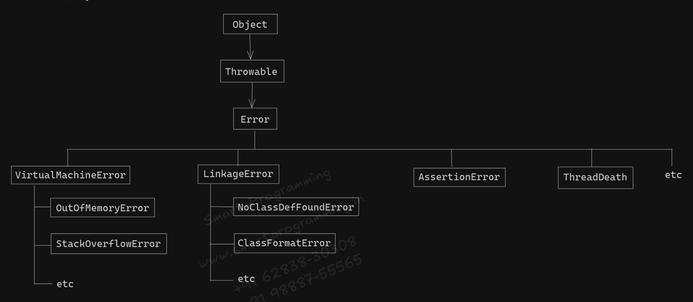
⇒ Error
  -> An error is a problem which is occured at runtime
  -> It is occured because of lack of system resources or JVM
  -> Programmer cannot control / recover the errors
  -> For example :-
     = OutOfMemoryError
     = StackOverflowError

  -> Types of Errors :-
     1. Compile Time Errors
        - Which occurs at compile time
        - Program will not execute in case of compile time errors
        - For eg
           >> Syntax Error
           >> Lexical Error
           etc
     2. Runtime Errors
        - Which occurs at runtime
        - For eg.
           = StackOverflowError
           = OutOfMemoryError
     3. Logical Errors
        - Program is executed but output is not proper
        - These errors cannot be deduct by compiler or JVM

  -> Hierarchy of Error Class :



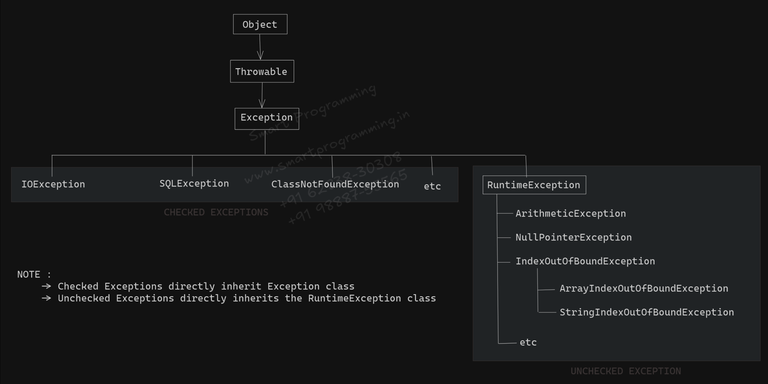
⇒ Exception
  -> Exception is any "unwanted event" that occurs at execution time which disturbs the normal flow of program.
  -> Real World Example :

  -> Exception is usually occur due to problem in program logic
  -> For eg.
     >> ArithmeticException
     >> NullPointerException
     >> IOException
     etc

  -> Types of Exceptions :-
     1. Checked Exceptions
        = Exceptions which compiler can check
        = Program will not compile
        = Its important to handle the checked exceptions
     2. Unchecked Exceptions
        = Exceptions which compiler ignores
        = Program will compile but not execute properly
        = Its not important to handle the unchecked exceptions but its recommended to handle them

  -> NOTE : All the exceptions are occured only at runtime, not at compile-time.

  -> Hierarchy of Exception Class:



NOTE :
  -> Checked Exceptions directly inherit Exception class
  -> Unchecked Exceptions directly inherits the RuntimeException class

Internal Working of Exception

```java
import java.util.Scanner;

public class ExceptionDemo2
{
    public static void main(String[] args)
    {
        System.out.println("-----App started-----");

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter no 1");
        int no1 = scanner.nextInt();

        System.out.println("Enter no 2");
        int no2 = scanner.nextInt();

        int res = no1 / no2; 1. if no2 is 0, then it will provide an exception

        System.out.println("Result is : "+res);

        System.out.println("-----App Finished Successfully-----");
    }
}
```

2. exception is occured in main method, so main method will create an exception object

exception object
1. Exception Class Name
2. Exception Description
3. Stack Trace or Location
→ Default Exception handler

```java
import java.util.Scanner;

public class ExceptionDemo2
{
    void m1()
    {
        m2();
    }

    void m2()
    {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter no 1");
        int no1 = scanner.nextInt();

        System.out.println("Enter no 2");
        int no2 = scanner.nextInt();

        int res = no1 / no2;

        System.out.println("Result is : "+res);
    }
    public static void main(String[] args)
    {
        System.out.println("-----App started-----");

        new ExceptionDemo2().m1();

        System.out.println("-----App Finished Successfully-----");
    }
}
```

exception object → DEH → Print

Exception Handling

-> It is the mechanism by which we handle all types of exceptions and our program will execute in proper flow
-> We use 5 keywords in Exception Handling mechanism
   1. try
   2. catch (actually handle the exception)
   3. finally
   4. throw
   5. throws

Interview Questions

⇒ Errors & Exceptions
1. What is the difference between an Error and an Exception in Java?
2. What are the different categories of exceptions in Java?
3. What is the difference between checked and unchecked exceptions?
4. Can you give examples of checked and unchecked exceptions?
5. Why are some exceptions checked and others unchecked?
6. What is the parent class of all exceptions in Java?
7. What is the parent class of all errors in Java?
8. Can we catch Errors in Java? Should we? Why or why not?
9. What is the difference between Throwable, Exception, and RuntimeException?
10. What is the difference between compile-time exceptions and runtime exceptions?
11. Why is RuntimeException not checked by the compiler?
12. Can we convert a checked exception into an unchecked exception? How?
13. Is it possible for a program to throw both an Error and an Exception? Give an example.
14. What are some common unchecked exceptions in Java?
15. What are some common checked exceptions in Java?

⇒ Internal Working of Exceptions
16. How does exception handling work internally in Java?
17. What is the role of the JVM when an exception is thrown?
18. What happens in the call stack when an exception occurs?
19. What is the process of exception propagation in Java?
20. How does the JVM search for the matching exception handler?
21. What happens if no matching exception handler is found?
22. What is the difference between stack unwinding and exception propagation?
23. What is the impact of exceptions on performance in Java?
24. What is the difference between getMessage() and printStackTrace() methods in exceptions?
25. What information is stored in the stack trace when an exception is thrown?
26. How does the JVM create and fill the exception object internally?
27. How are suppressed exceptions handled in Java internally?
28. What is the role of the Throwable class in exception handling?