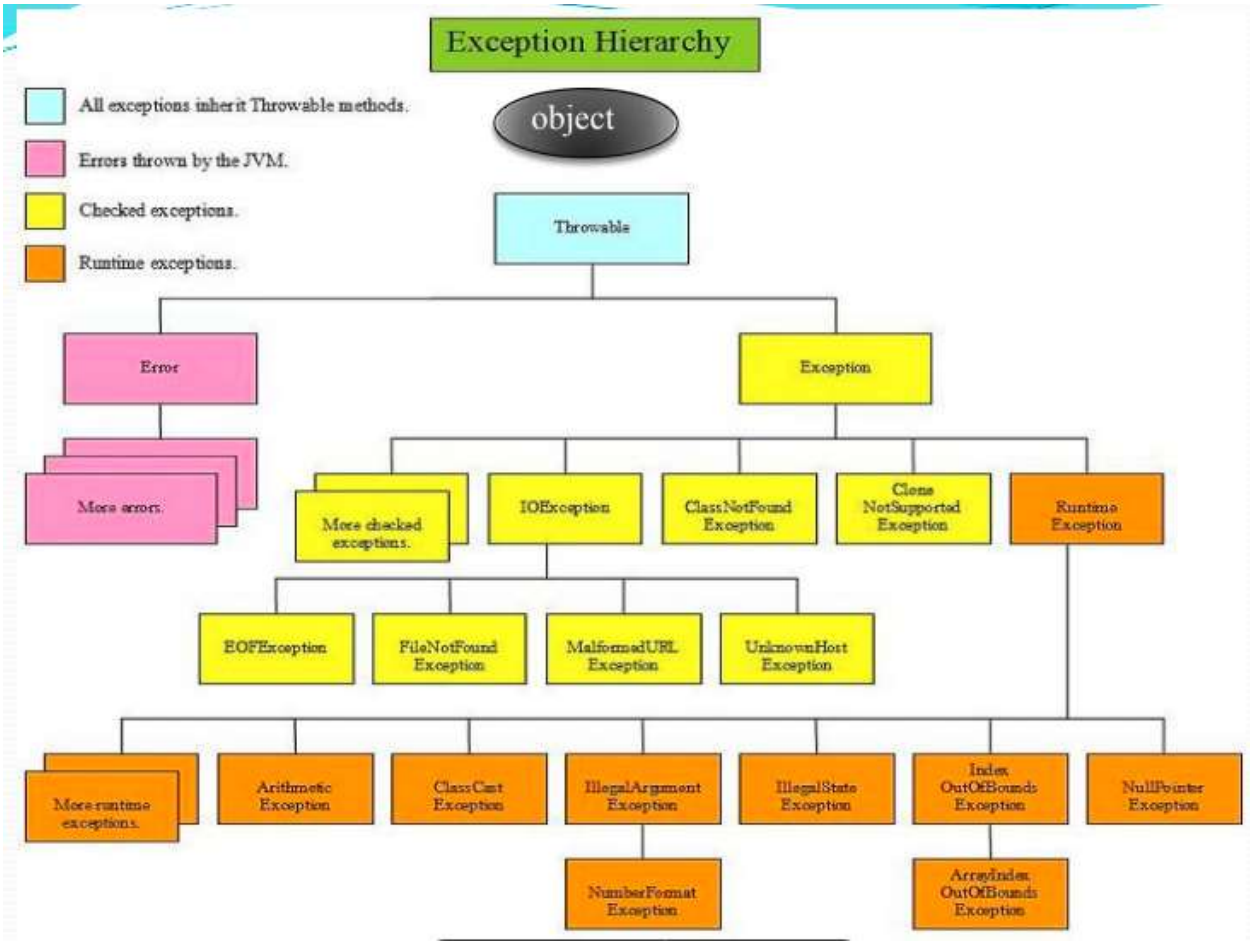
Exception:

- An abnormal event in a program is called Exception.
- All Exceptions occur at runtime only but some are detected at compile time and some are detected at runtime.
- Exceptions that are checked at compile time by the java compiler are called "Checked exceptions".

eg: ClassNotFoundException, NoSuchMethodException, NoSuchFieldException etc.

- Exceptions that are checked at run time by the JVM are called "Unchecked exceptions".

eg: ArrayIndexOutOfBoundsException, ArithmeticException, NumberFormatException etc.



Some common exceptions scenarios

- **Scenario where ArithmeticException occurs :**

  If we divide any number by zero, there occurs an ArithmeticException.

```
int a=50/0;//ArithmeticException
```

- **Scenario where NullPointerException occurs :**

  If we have null value in any variable, obtaining the length of that variable occurs an NullPointerException.

```
String s=null;
System.out.println(s.length());//NullPointerException
```

- **Scenario where ArrayIndexOutOfBoundsException occurs**

  If you are inserting any value in the wrong index, it would result ArrayIndexOutOfBoundsException as shown below:

```
int a[]=new int[5];
a[10]=50; //ArrayIndexOutOfBoundsException
```

List of important built-in exceptions

Exception Class	Meaning
ArithmeticException	Thrown when an exceptional condition has occurred in an arithmetic operation
ArrayIndexOutOfBoundsException	Thrown to indicate that an array has been accessed with an illegal index
ClassNotFoundException	Thrown when we try to access a class whose definition is not found
FileNotFoundException	Raised when a file is not accessible or does not open
IOException	Thrown when an input-output operation failed or interrupted
NoSuchFieldException	Thrown when a class does not contain the field(or variable) specified
NullpointerException	Raised when referring to the members of a null object

List of important built-in exceptions

Exception Class	Meaning
NumberFormatException	Raised when a method could not convert a string in to a numeric format
RuntimeException	This represents any exceptions which occurs during runtime
StringIndexOutOfBoundsException	Thrown by String class methods to indicate that an index is either negative or greater than the size of the string

Exception Handling

- An exception can be handled by the programmer where as an error cannot be handled by the programmer.
- Exception handling doesn't mean fixing an exception, We need to provide an alternative solution for the free flow of program.

What happens when Exception has occurred?

- Exception occurs only either inside the block or a method.
- When exception has raised, that block or method creates an exception object which contains the complete information of that exception including.
  - Name of the Exception
  - Explanation of the Exception
  - State of the Exception (Stack Trace)

- Once object has been created, it passes to JVM, then JVM handles that exception with the help of Default Exception Handler.
- Default Exception Handler checks whether the method contains any exception handling code or not. If method won't contain any handling code then JVM terminates that method abnormally and removes corresponding entry form the stack.
- Default Exception Handler just prints the basic information about the exception which has occurred and terminates the method abruptly.

- When there is an exception the programmer should do the following tasks:
- If the programmer suspects any exception in program statements, he should write them inside try block.

```
try
  {
    statements;
  }
```

- Within the try block if anywhere an exception raised then rest of the try block won't be executed even though we handled that exception.
- When there is an exception in try block JVM will not terminate the program abnormally.
- JVM stores exception details in an exception stack and then JVM jumps into catch block.
- The programmer should display exception details and any message to the user in catch block.

```
catch ( ExceptionClass obj)
    {
        statements;
    }
```

- Programmer should close all the files and databases by writing them inside finally block.
- Finally block is executed whether there is an exception or not.

```
finally
    {
        statements;
    }
```

Performing above tasks is called Exception Handling.

Error: If there is an error occured in our program, our program will be terminated. We cant save our program. Errors ***cant*** be handled

- Errors
  - Compile time Errors ➡ These errors occurs due to the **wrong** syntax. (Syntactical Errors)
  - Runtime Errors ➡ These errors occurs due to the Ineffiency of the machine
  - Logical Errors ➡ These errors occurs due to the WRONG logic given by the programmers

Exception: If there is an Exception occured in our program, our program will be terminated , but we can save our program by handling that Exception. ***Exceptions can be handled***

- Exception
  - Checked Exceptions ➡ The exceptions which are detected at compilation time by the *Java compiler* are known as Checked exceptions
  - Un-Checked Exceptions ➡ The exceptions which are detected at Run-time by the ***JVM*** are known as Un-Checked exceptions

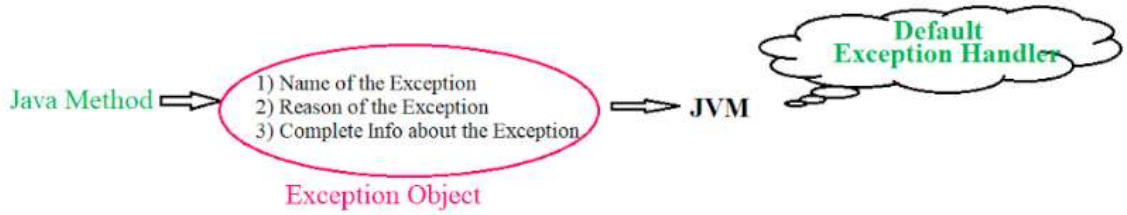
NOTE

1) Every Exception always occurs **ONLY** inside a method (or) inside a block.

2) Every Exception always occurs only during **Runtime**. (Some Exceptions are detected at Compilation time & Some Exceptions are detected at Runtime)

Q) What happens if there is an Exception occured in our Program?

A)



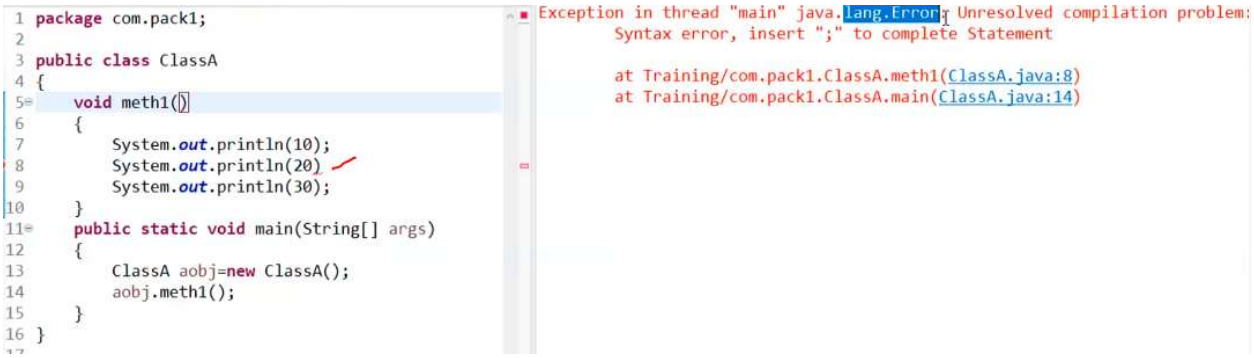
Exception Handling: Exception handling means not resolving the exception (or) not removing the exception, we are providing an alternative way to continue the program execution.

Q) How to handle an Exception?

A) We can handle an Exception by using **try-catch-finally** BLOCKS

```
try
{
    // Inside the try block we need to write the suspecious code
}
catch(Exception e)
{
    // Inside the catch block we need to catch (or) handle the Exception which occured in the try block
}
finally
{
    // finally block will be executed always irrespective of the Exception
}
```

Error



```
1  package com.pack1;
2
3  public class ClassA
4  {
5      void meth1()
6      {
7          System.out.println(10);
8          System.out.println(20)
9          System.out.println(30);
10     }
11     public static void main(String[] args)
12     {
13         ClassA aobj=new ClassA();
14         aobj.meth1();
15     }
16 }
```

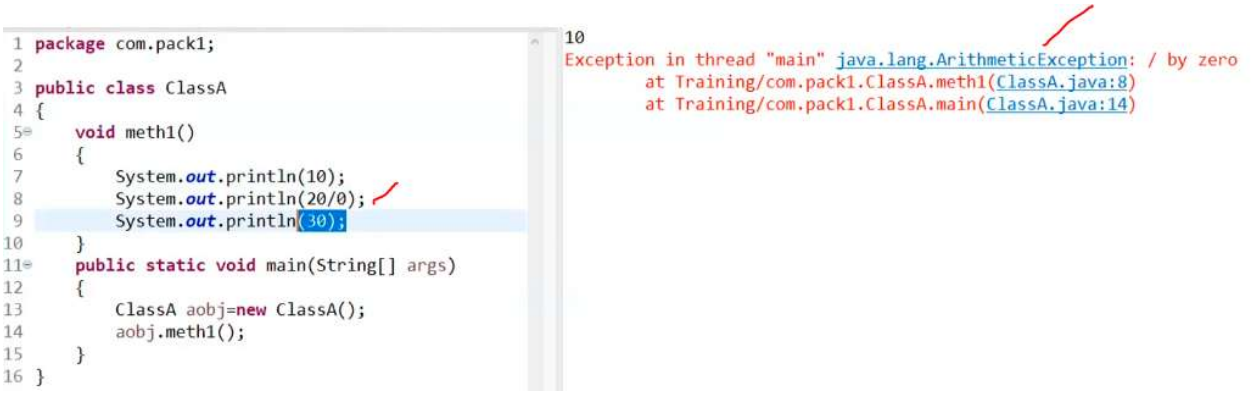
```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Syntax error, insert ";" to complete Statement

        at Training/com.pack1.ClassA.meth1(ClassA.java:8)
        at Training/com.pack1.ClassA.main(ClassA.java:14)
```

If it is an error the program will terminate.

We cannot save our program, which means we cannot handle the errors.

Exception



```
1  package com.pack1;
2
3  public class ClassA
4  {
5      void meth1()
6      {
7          System.out.println(10);
8          System.out.println(20/0);
9          System.out.println(30);
10     }
11     public static void main(String[] args)
12     {
13         ClassA aobj=new ClassA();
14         aobj.meth1();
15     }
16 }
```

```
10
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Training/com.pack1.ClassA.meth1(ClassA.java:8)
        at Training/com.pack1.ClassA.main(ClassA.java:14)
```

If it is an exception the program will terminate. But we can save our program by handling the exception.

Note:

Basically exception is a Class.

Parent class is Object Class

From object class child class is throwable class

From throwable class the child's classes are Error Class and Exception Class