

# ENTERPRISE APPLICATION DEVELOPMENT-1

# Exception Handling

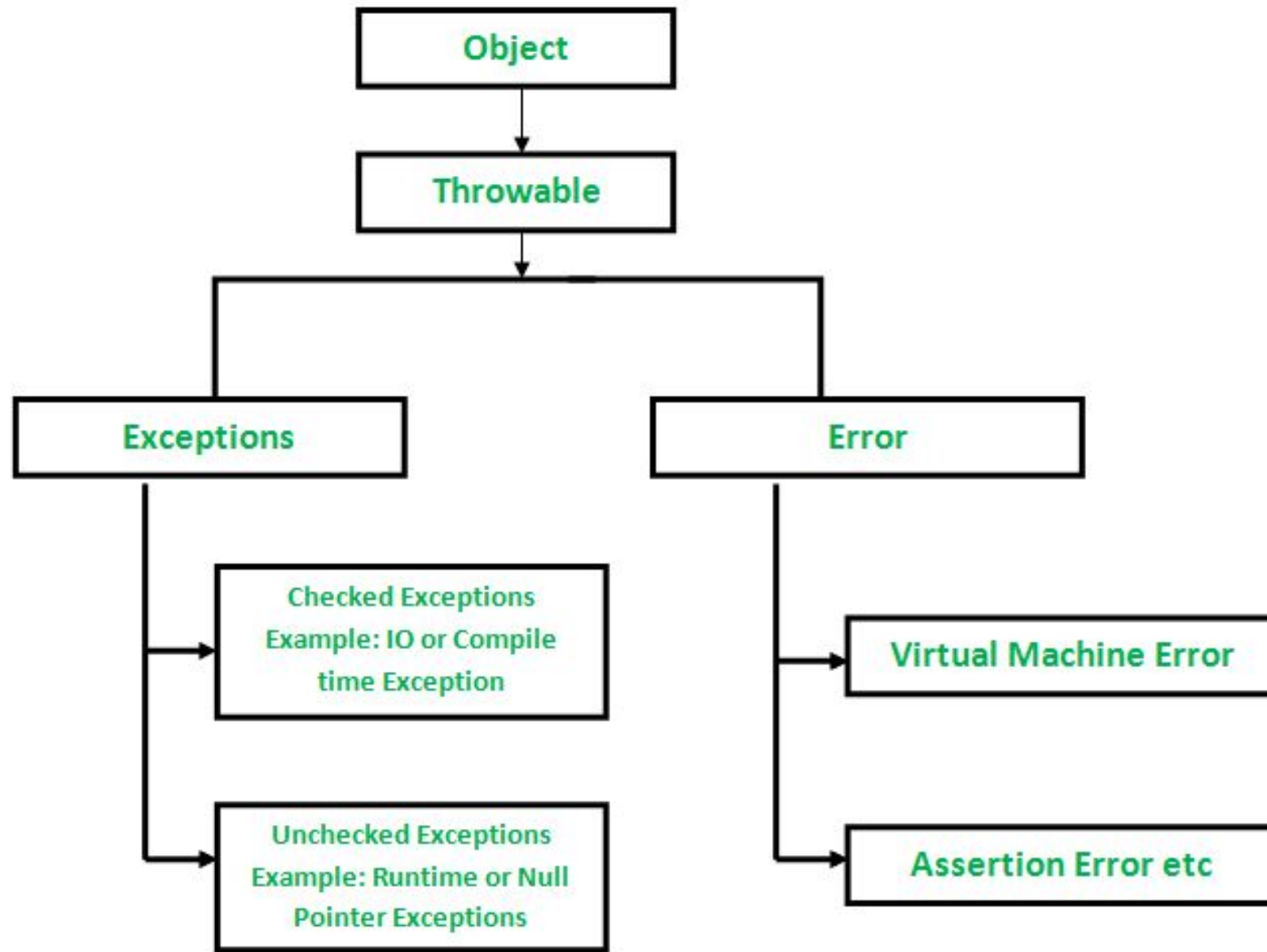
*“Exception is an abnormal condition.”*

*“Exception is an event that interrupts the normal flow of execution.”*

*“An exception (or exceptional event) is a problem that arises during the execution of a program.”*

*“Exception is an error event that can happen during the execution of a program and disrupts its normal flow.”*

*“An Exception is an unwanted event that interrupts the normal flow of the program.”*



# Compile time vs run time errors

Compile-time	Runtime
The compile-time errors are the errors which are produced at the compile-time, and they are detected by the compiler.	The runtime errors are the errors which are not generated by the compiler and produce an unpredictable result at the execution time.
In this case, the compiler prevents the code from execution if it detects an error in the program.	In this case, the compiler does not detect the error, so it cannot prevent the code from the execution.
It contains the syntax and semantic errors such as missing semicolon at the end of the statement.	It contains the errors such as division by zero, determining the square root of a negative number.

# Why an exception occurs?

- There can be several reasons that can cause a program to throw exception. For example: Opening a non-existing file in your program, Network connection problem, bad input data provided by user etc.

```
import java.io.File;
import java.io.FileReader;

public class FileNotFound_Demo {

    public static void main(String args[]) {
        File file = new File("E://NIBM.txt");
        FileReader fr = new FileReader(file);
    }
}
```



```
C:\>javac FileNotFound_Demo.java
FileNotFound_Demo.java:8: error: unreported exception FileNotFoundException;
must be caught or declared to be thrown
        FileReader fr = new FileReader(file);
                        ^
1 error
```

# Java Exception Keywords

## try

The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.

## catch

The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.

## Finally

The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.

## Throw

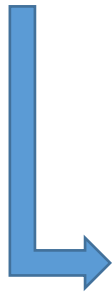
The "throw" keyword is used to throw an exception.

## Throws

The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

# Example..

```
public class testClass {  
    public static void main(String args[])  
    {  
        try{  
            //code that may raise exception  
            int data=100/0;  
        }catch(ArithmeticException e){System.out.println(e);}  
        //rest code of the program  
        System.out.println("rest of the code...");  
    }  
}
```

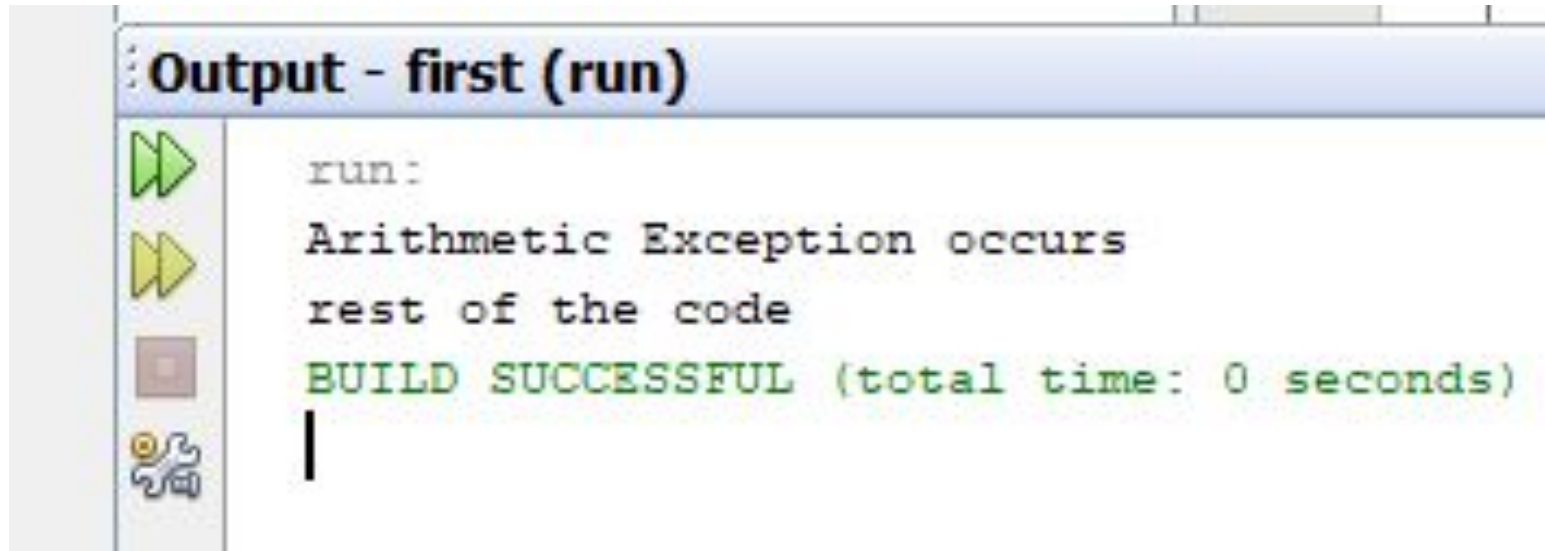


```
run:  
java.lang.ArithmeticException: / by zero  
rest of the code...  
BUILD SUCCESSFUL (total time: 0 seconds)  
|
```

# Catch multiple exceptions



```
public class testClass {  
    public static void main(String args[])  
    {  
        try{  
            int a[]=new int[5];  
            a[5]=30/0;  
        }  
        catch(ArithmeticException e)  
        {  
            System.out.println("Arithmetic Exception occurs");  
        }  
        catch(ArrayIndexOutOfBoundsException e) |  
        {  
            System.out.println("ArrayIndexOutOfBoundsException Exception occurs");  
        }  
        catch(Exception e)  
        {  
            System.out.println("Parent Exception occurs");  
        }  
        System.out.println("rest of the code");  
    }  
}
```

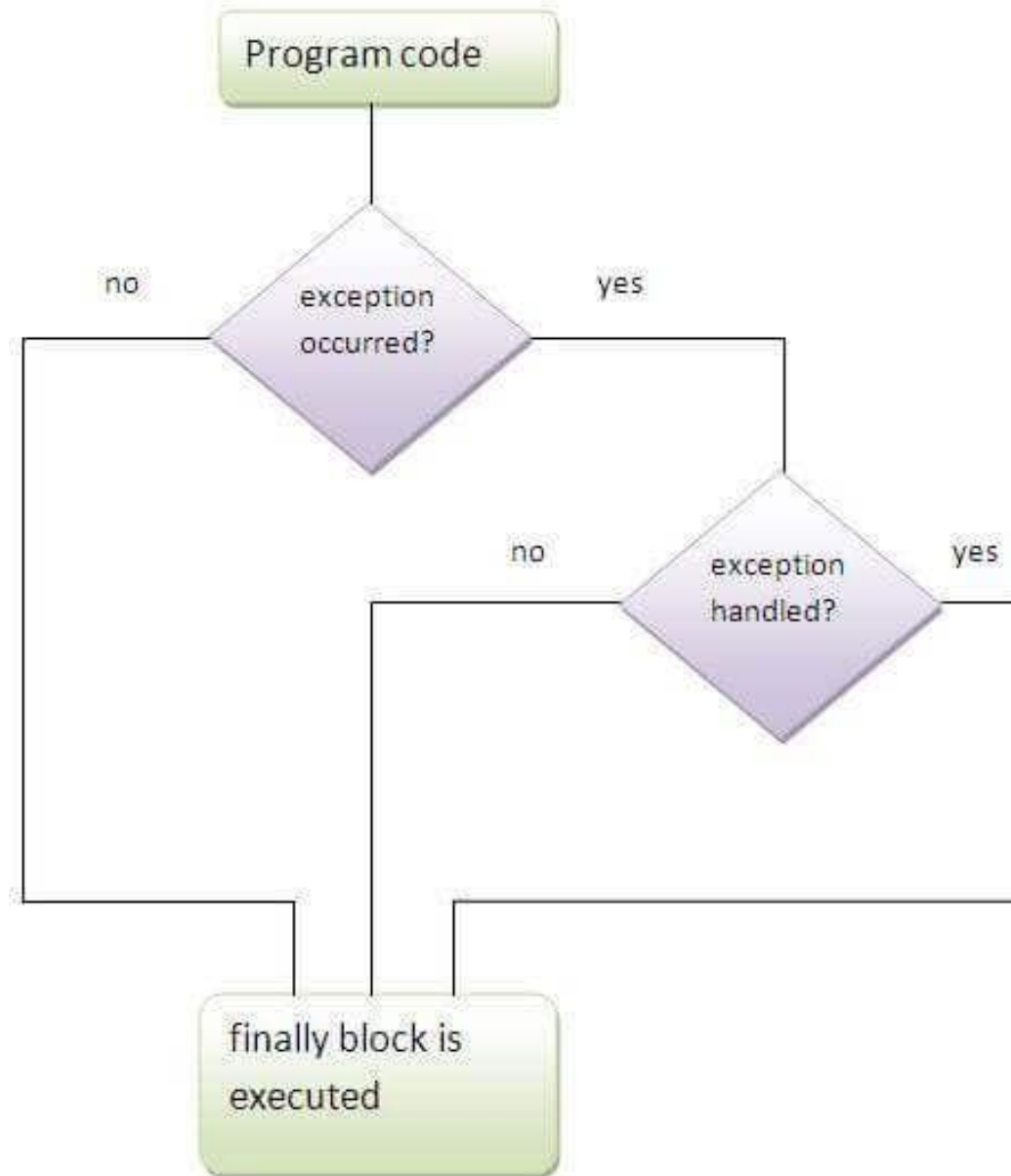


```
run:
Arithmetic Exception occurs
rest of the code
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

But!! We have more catch statements??

# Java finally block

- **Java finally block** is a block that is used *to execute important code* such as closing connection, stream etc.
- Java finally block is always executed whether exception is handled or not.



# throw

- It is possible for your program to throw an exception explicitly  
throw *ThrowableInstance*
- Here, *ThrowableInstance* must be an object of type **Throwable** or a subclass **Throwable**
- There are two ways to obtain a **Throwable** objects:
  - Using a parameter into a catch clause
  - Creating one with the **new** operator

**Throws clause** is used to declare an exception, which means it works similar to the try-catch block.

```
public class test
{
    void checkAge(int age)
    {
        if(age<18)
            throw new ArithmeticException("Not Eligible for voting");
        else
            System.out.println("Eligible for voting");
    }

    public static void main(String args[])
    {
        test obj = new test();
        try
        {
            obj.checkAge(13);
        }catch(Exception e)
        {
            System.out.println(e.toString());
        }
        System.out.println("End Of Program");
    }
}
```

# throws

- Java throws keyword is used to declare an exception.
- Checked exception can be propagated with throws.
- Throws is used with the method signature.
- You can declare multiple exceptions e.g.  
`public void method()throws IOException,SQLException.`

```
public class Example1{
    int division(int a, int b) throws ArithmeticException{
        int t = a/b;
        return t;
    }
    public static void main(String args[]){
        Example1 obj = new Example1();
        try{
            System.out.println(obj.division(15,0));
        }
        catch(ArithmeticException e){
            System.out.println("You shouldn't divide number by zero");
        }
    }
}
```





```
// Demonstrate finally.
class FinallyDemo {
    // Through an exception out of the method.
    static void procA() {
        try {
            System.out.println("inside procA");
            throw new RuntimeException("demo");
        } finally {
            System.out.println("procA's finally");
        }
    }

    // Return from within a try block.
    static void procB() {
        try {
            System.out.println("inside procB");
            return;
        } finally {
            System.out.println("procB's finally");
        }
    }

    // Execute a try block normally.
    static void procC() {
        try {
            System.out.println("inside procC");
        } finally {
            System.out.println("procC's finally");
        }
    }

    public static void main(String args[]) {
        try {
            procA();
        } catch (Exception e) {
            System.out.println("Exception caught");
        }
        procB();
        procC();
    }
}
```

# Output

inside procA

procA's finally

Exception caught

inside procB

procB's finally

inside procC

procC's finally

# Example

```
public class ExceptionHandling
{
    public static void main(String[] args)
    {
        try
        {
            int i = Integer.parseInt("abc");
        }

        catch(Exception ex)
        {
            System.out.println("Exception occurred: " + ex.getMessage());
        }

        catch(NumberFormatException ex)
        {
            System.out.println("Invalid input: " + ex.getMessage());
        }
    }
}
```



```
public class ExceptionHandling
{
    public static void main(String[] args)
    {
        try
        {
            int i = Integer.parseInt("abc");    //This statement throws NumberFormatException
        }

        catch(Exception ex)
        {
            System.out.println("This block handles all exception types");
        }

        catch(NumberFormatException ex)
        {
            //Compile time error
            //This block becomes unreachable as
            //exception is already caught by above catch block
        }
    }
}
```

### Unreachable catch block error

When you are keeping multiple catch blocks, the order of catch blocks must be from most specific to most general ones. i.e sub classes of Exception must come first and super classes later. If you keep super classes first and sub classes later, compiler will show unreachable catch block error.

# Example