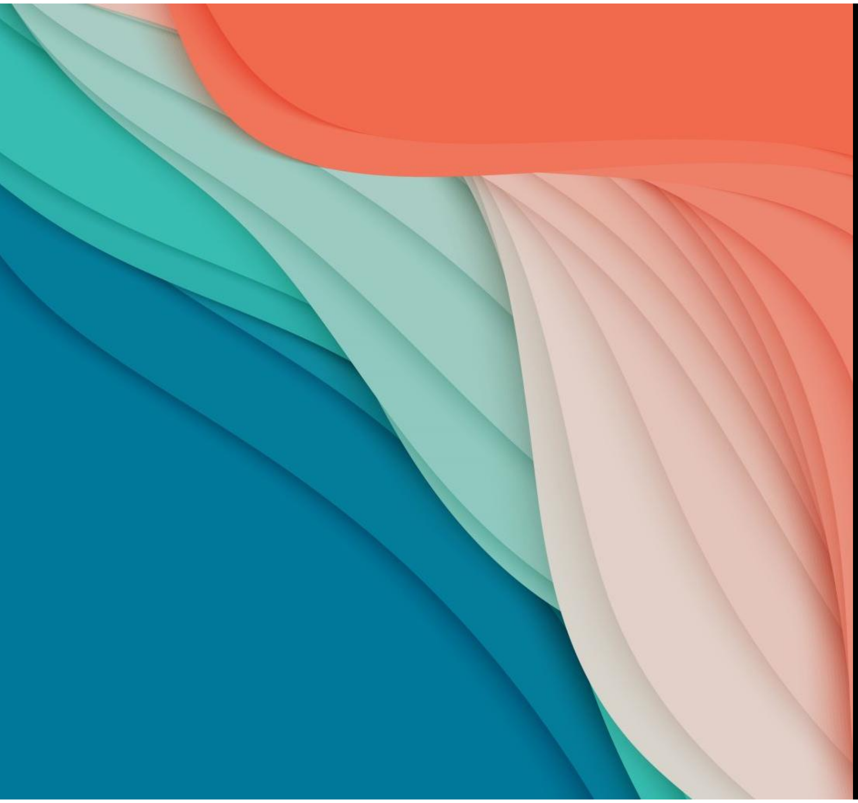


UNIT III

Exception handling

EXAMPLE OF EXCEPTION

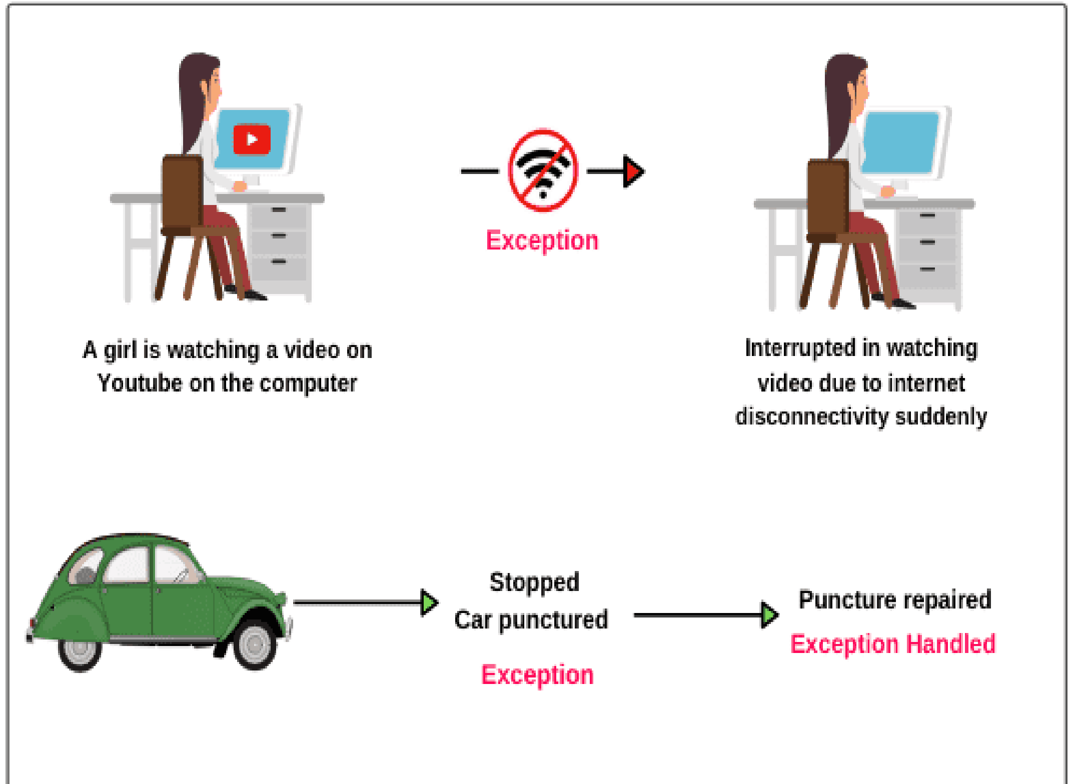
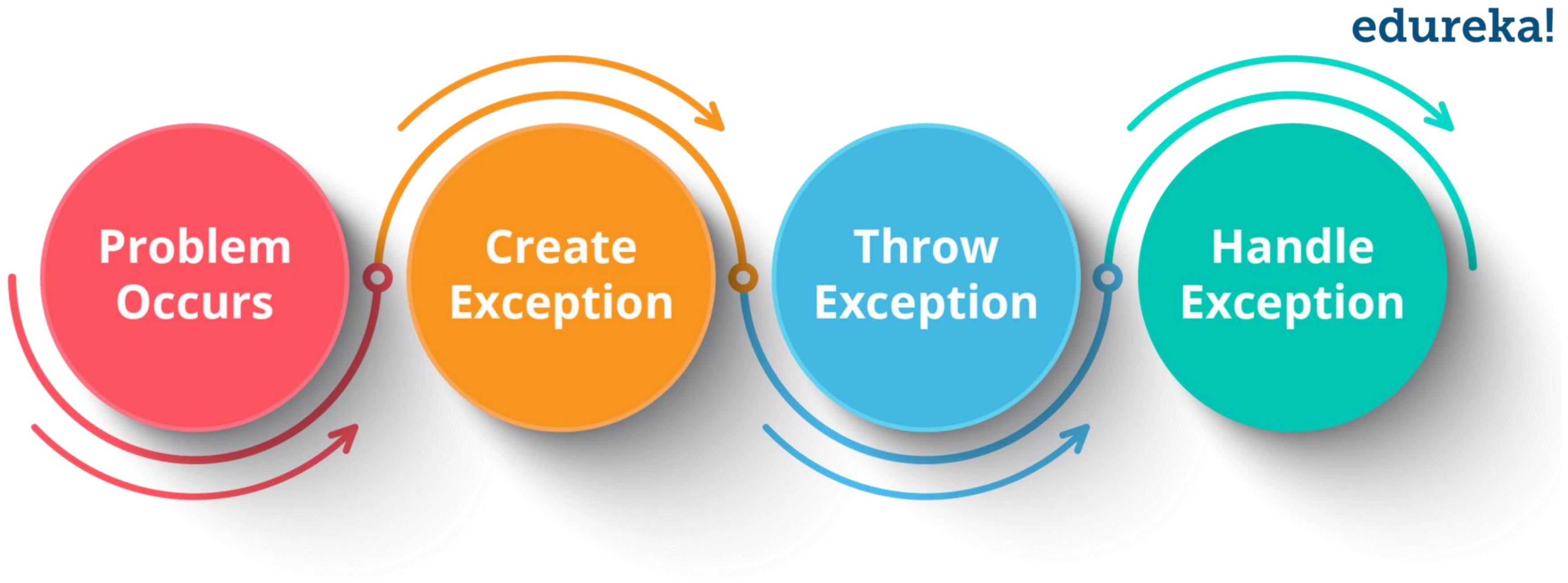

A girl is watching a video on Youtube on the computer
Exception
Interrupted in watching video due to internet disconnectivity suddenly
Stopped
Car punctured
Exception
Puncture repaired
Exception Handled

Fig: Realtime Example of Exception Handling



JAVA EXCEPTIONS

WHAT IS EXCEPTION?

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

When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally

When an exception occurs, and if you don’t handle it, the program will terminate abruptly (the piece of code after the line causing the exception will not get executed).

WHY EXCEPTIONS OCCURS?

- A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

CATEGORIES OF JAVA EXCEPTIONS

Checked exceptions – A checked exception is an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions.

These exceptions cannot simply be ignored, the programmer should take care of (handle) these exceptions.

Checked Exception
Unchecked Exception
Error
Exception Handling

CATEGORIES OF JAVA EXCEPTIONS

Unchecked exceptions − An unchecked exception is an exception that occurs at the time of execution.

These are also called as **Runtime Exceptions.**

These include programming bugs, such as logic errors or improper use of an API.

Runtime exceptions are ignored at the time of compilation.

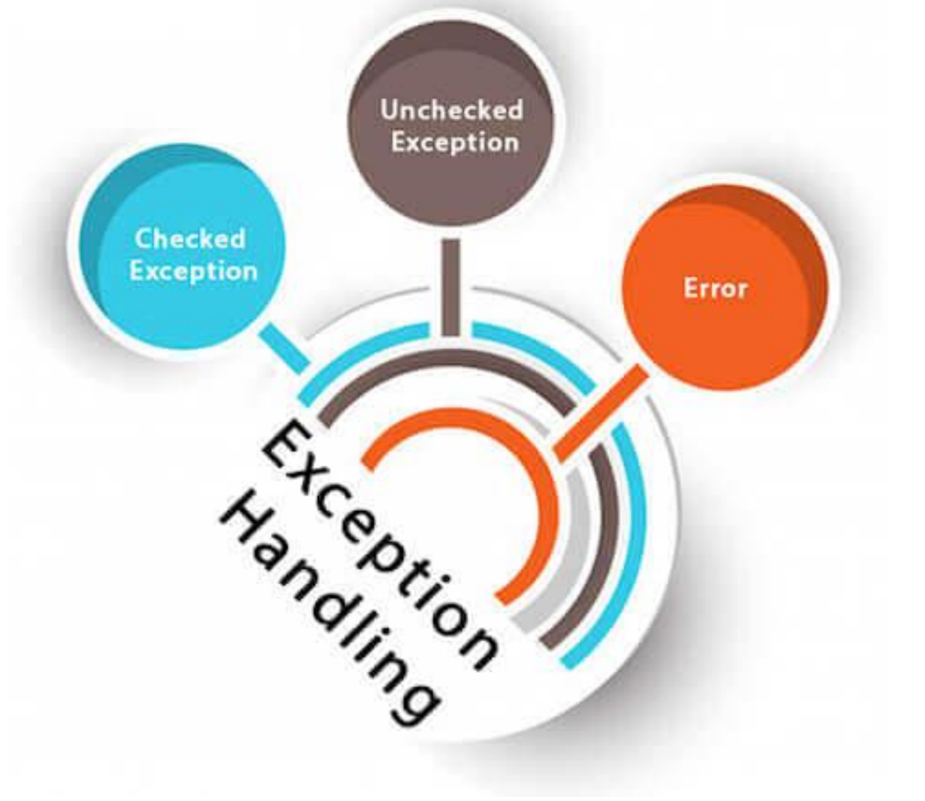
Unchecked Exception
Checked Exception
Error
Exception Handling

CATEGORIES OF JAVA EXCEPTIONS

Errors – These are not exceptions at all, but problems that arise beyond the control of the user or the programmer.

Errors are typically ignored in your code because you can rarely do anything about an error.

For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.



JAVA EXCEPTION KEYWORDS

Keyword	Description
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.

lóhika

Exceptions in Java

```
try{
```

```
}catch( Exception ){
    //Do nothing
}
```

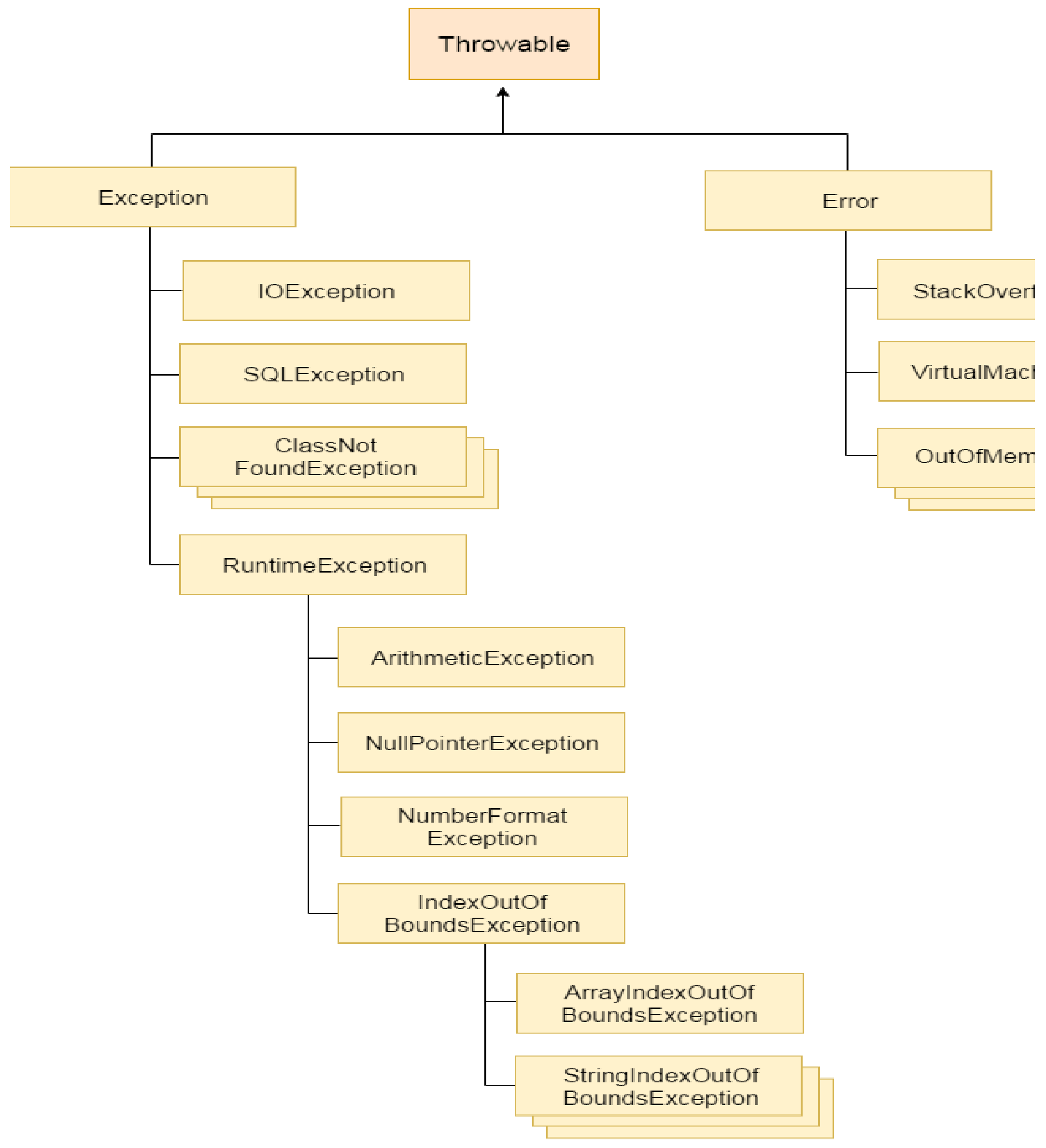
Exceptions...

Gotta catch 'em all!

Author: Vadim Lotar
Lohika (2011)

EXCEPTION HIERARCHY

The java.lang.Throwable class is the root class of Java Exception hierarchy which is inherited by two subclasses: Exception and Error.



EXAMPLE OF EXCEPTION

```
public class JavaExceptionExample{
  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...");
  }
}
```

COMMON SCENARIOS OF JAVA EXCEPTIONS

1) A scenario where ArithmeticException occurs

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

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

1.A scenario where NullPointerException occurs

If we have a null value in any variable, performing any operation on the variable throws a NullPointerException.

```
String s=null;

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

COMMON SCENARIOS OF JAVA EXCEPTIONS

1. **A scenario where NumberFormatException occurs**

The wrong formatting of any value may occur NumberFormatException.

Suppose I have a string variable that has characters, converting this variable into digit will occur NumberFormatException

```
String s="abc";
int i=Integer.parseInt(s);//NumberFormatException
```

1. A scenario where ArrayIndexOutOfBoundsException occurs

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

```
int a[]=new int[5];

a[10]=50; //ArrayIndexOutOfBoundsException
```

ERROR VS EXCEPTION

Error: An Error indicates serious problem that a reasonable application should not try to catch.

Exception: Exception indicates conditions that a reasonable application might try to catch.

HOW JVM HANDLE AN EXCEPTION? (AGAIN REVISIT)

Default Exception Handling : Whenever inside a method, if an exception has occurred, the method creates an Object known as Exception Object and hands it off to the run-time system(JVM).

The exception object contains name and description of the exception, and current state of the program where exception has occurred.

Creating the Exception Object and handling it to the run-time system is called throwing an Exception.

There might be the list of the methods that had been called to get to the method where exception was occurred. This ordered list of the methods is called **Call Stack**.

JAVA TRY BLOCK

Java **try** block is used to enclose the code that might throw an exception.

It must be used within the method.

If an exception occurs at the particular statement of try block, the rest of the block code will not execute.

So, it is recommended not to keeping the code in try block that will not throw an exception.

Java try block must be followed by either catch or finally block.

Syntax of Java try-catch

```
try{
//code that may throw an exception
}
catch(Exception_class_Name ref){}
```

Syntax of try-finally block

```
try{
//code that may throw an exception
}finally{}
```

JAVA CATCH BLOCK

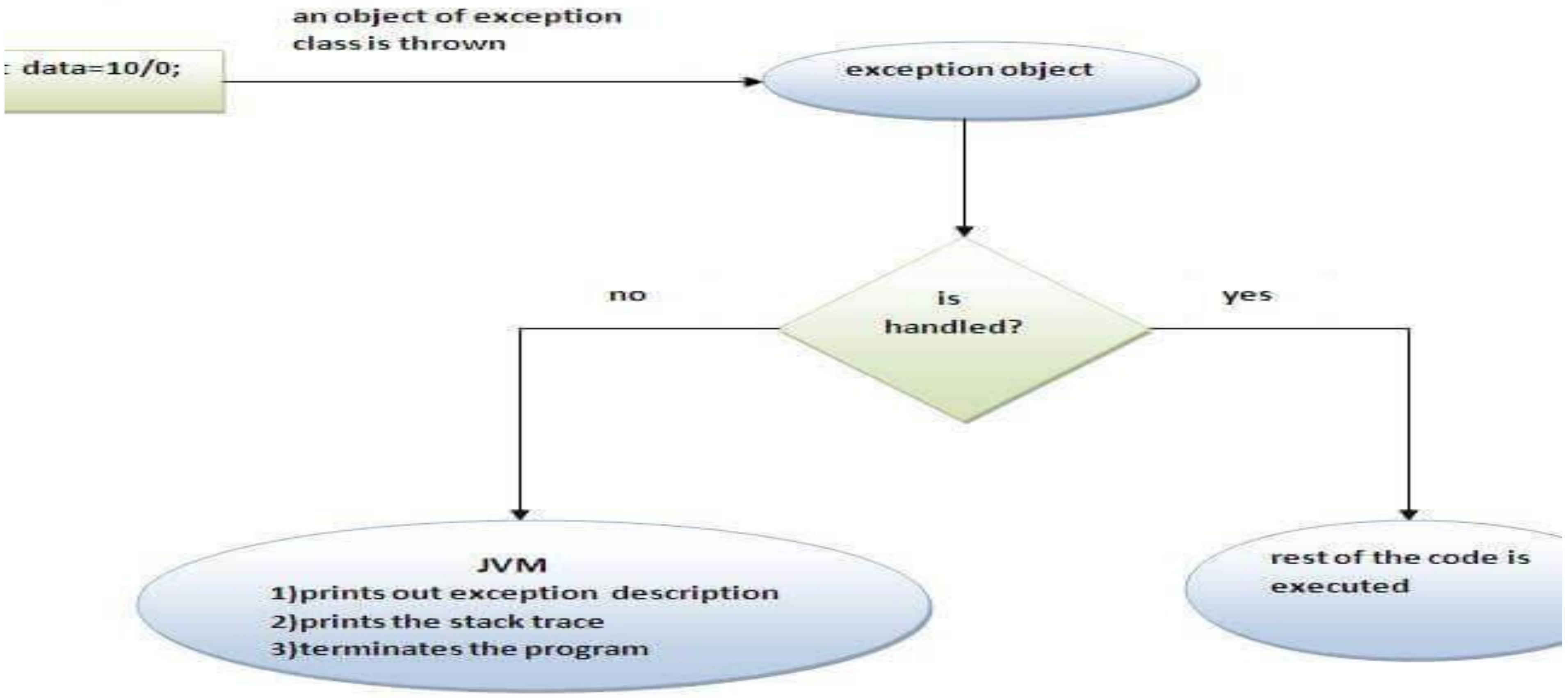
Java catch block is used to handle the Exception by declaring the type of exception within the parameter.

The declared exception must be the parent class exception (i.e., Exception) or the generated exception type.

However, the good approach is to declare the generated type of exception.

The catch block must be used after the try block only. You can use multiple catch block with a single try block.

INTERNAL WORKING OF JAVA TRY-CATCH BLOCK

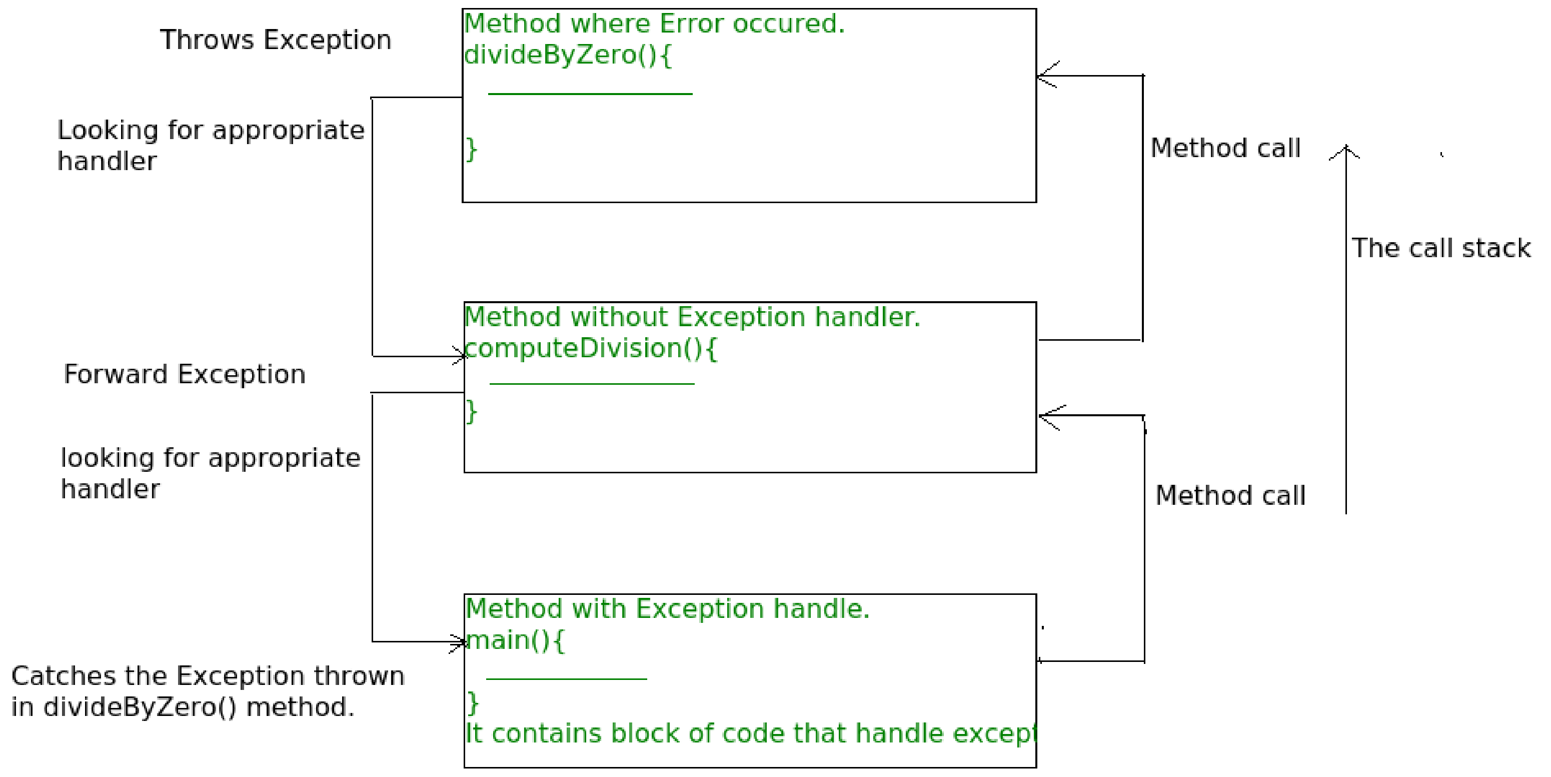

an object of exception class is thrown
: data=10/0;
exception object
is handled?
no
yes
JVM
1)prints out exception description
2)prints the stack trace
3)terminates the program
rest of the code is executed

The JVM firstly checks whether the exception is handled or not.

If exception is not handled, JVM provides a default exception handler that performs the following tasks:

- Prints out exception description.
- Prints the stack trace (Hierarchy of methods where the exception occurred).
- Causes the program to terminate.

But if exception is handled by the application programmer, normal flow of the application is maintained i.e. rest of the code is executed.


Throws Exception
Method where Error occured.
divideByZero(){
}
Looking for appropriate handler
Method call
The call stack
Method without Exception handler.
computeDivision(){
}
Forward Exception
looking for appropriate handler
Method call
Method with Exception handle.
main(){
}
It contains block of code that handle except
Catches the Exception thrown in divideByZero() method.

The call stack and searching the call stack for exception handler.

JAVA MULTI-CATCH BLOCK

A try block can be followed by one or more catch blocks.

Each catch block must contain a different exception handler.

So, if you have to perform different tasks at the occurrence of different exceptions, use java multi-catch block.

•At a time only one exception occurs and at a time only one catch block is executed.

•All catch blocks must be ordered from most specific to most general, i.e. catch for ArithmeticException must come before catch for Exception.

JAVA NESTED TRY BLOCK

Why use nested try block

Sometimes a situation may arise where a part of a block may cause one error and the entire block itself may cause another error.

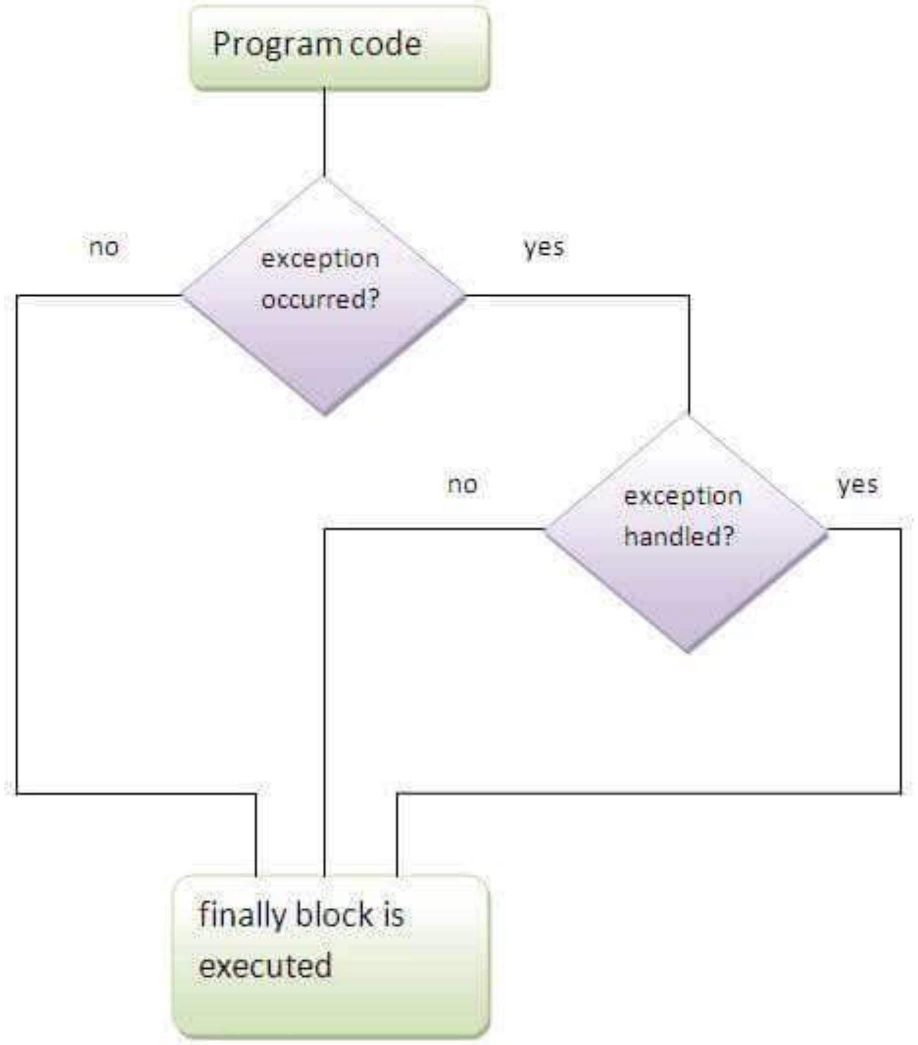
In such cases, exception handlers have to be nested.

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.

Java finally block follows try or catch block.



JAVA THROW EXCEPTION

The Java throw keyword is used to explicitly throw an exception.

We can throw either checked or uncheked exception in java by throw keyword.

The throw keyword is mainly used to throw custom exception.

JAVA THROWS KEYWORD

The **Java throws keyword** is used to declare an exception.

It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

Exception Handling is mainly used to handle the checked exceptions.

If there occurs any unchecked exception such as NullPointerException, it is programmers fault that he is not performing check up before the code being used.

Rule: If you are calling a method that declares an exception, you must either caught or declare the exception.

There are two cases:

Case1: You caught the exception i.e. handle the exception using try/catch.

Case2: You declare the exception i.e. specifying throws with the method.

A)In case you declare the exception, if exception does not occur, the code will be executed fine.

B)In case you declare the exception if exception occures, an exception will be thrown at runtime because throws does not handle the exception.

DIFFERENCE BETWEEN THROW AND THROWS IN JAVA

Sr. No.	throw	throws
1)	Java throw keyword is used to explicitly throw an exception.	Java throws keyword is used to declare an exception.
2)	Checked exception cannot be propagated using throw only.	Checked exception can be propagated with throws.
3)	Throw is followed by an instance.	Throws is followed by class.
4)	Throw is used within the method.	Throws is used with the method signature.
5)	You cannot throw multiple exceptions.	You can declare multiple exceptions e.g. public void method()throws IOException,SQLException.

IMPORTANT POINTS TO REMEMBER ABOUT THROWS KEYWORD

•throws keyword is required only for checked exception and usage of throws keyword for unchecked exception is meaningless.

•throws keyword is required only to convince compiler and usage of throws keyword does not prevent abnormal termination of program.

•By the help of throws keyword we can provide information to the caller of the method about the exception.

JAVA CUSTOM EXCEPTION

If you are creating your own Exception that is known as custom exception or user-defined exception.

Java custom exceptions are used to customize the exception according to user need.

By the help of custom exception, you can have your own exception and message.