Exception Handling

Exception Handling in Java

The **exception handling in java** is one of the **powerful** ***mechanism to handle the runtime errors*** so that normal flow of the application can be maintained.

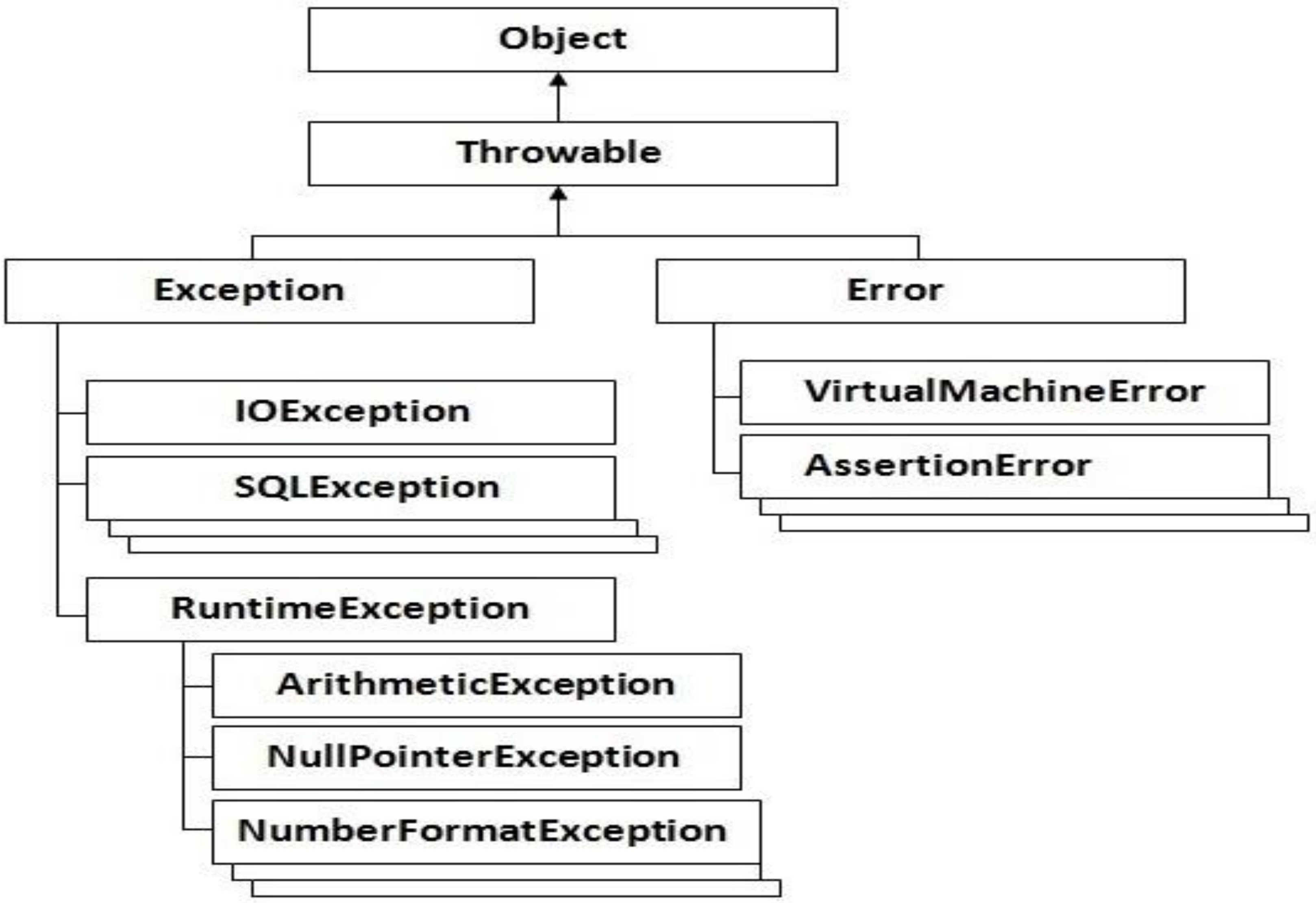
What is exception

Exception is an **abnormal condition**.
In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

What is exception handling

Exception Handling is a mechanism to handle runtime errors such as ClassNotFound, IO, SQL etc.

Hierarchy of Java Exception classes



Types of Exception

- There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception.
- The **sun microsystem** says there are three types of exceptions:

1. Checked Exception
2. Unchecked Exception
3. Error

1) Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException, SQLException etc. Checked exceptions are checked at **compile-time**.

2) Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at **runtime**.

3) Error

Error is **irrecoverable** e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

Java Exception Handling Keywords

There are 5 keywords used in java exception handling.

1. try
2. catch
3. finally

Java try block

- Java try block is used to **enclose the code that might throw an exception.**
- It must be **used within the method.**
- Java try block must be followed by either **catch or finally block.**

Syntax of java try-catch

```
try{
//code
}
catch(Exception_class_Name ref)
{
}
```

Syntax of try-finally block

```
try{
//code
}finally
{
}
```

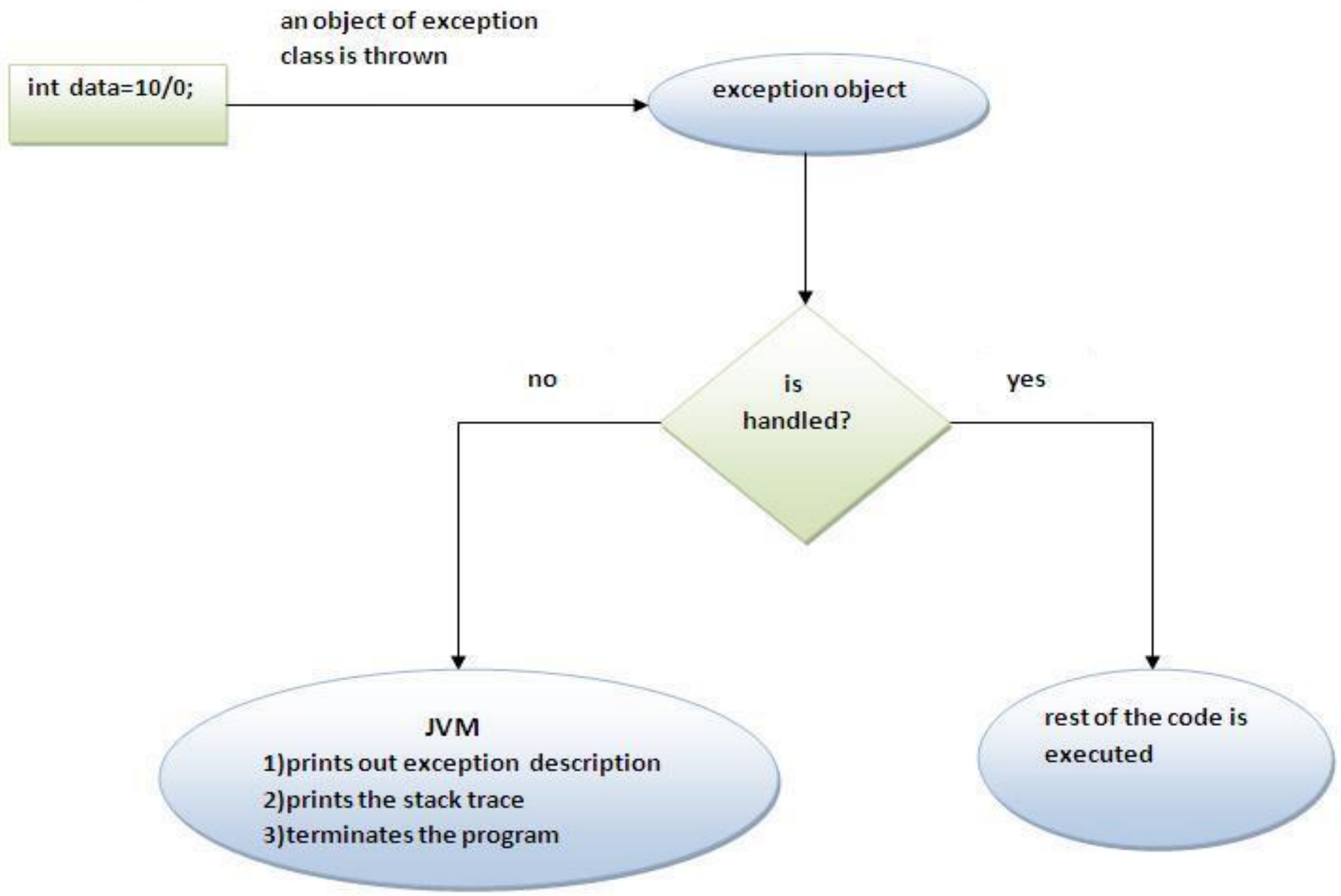
Java catch block

- Java catch block is used to **handle the Exception**. It must be used **after the try block only.**
- You can use **multiple catch block with a single try.**

Ex:-

```
public class Testtrycatch2
{
  public static void main(String args[]){
   try
{   int data=50/0;   }
catch(ArithmeticException e)
{System.out.println(e);}
   System.out.println("rest of the code...");  }
}
```

Internal working of java try-catch block


an object of exception class is thrown
int 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

Java Multi catch block

- If you have to perform different tasks at the occurrence of different Exceptions, use java multi catch block.
- Let's see a simple example of java multi-catch block.

```
class TestMultipleCatchBlock1{
  public static void main(String args[]){
   try{
    int a[]=new int[5];
    a[5]=30/0;
   }
   catch(Exception e)
{System.out.println("common task completed");}
   catch(ArithmeticException e)
{System.out.println("task1 is completed");}
   catch(ArrayIndexOutOfBoundsException e)
{System.out.println("task 2 completed");}
   System.out.println("rest of the code...");
 }
}
```

Java Nested try block

The try block within a try block is known as nested try block in java.

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.

Syntax:-

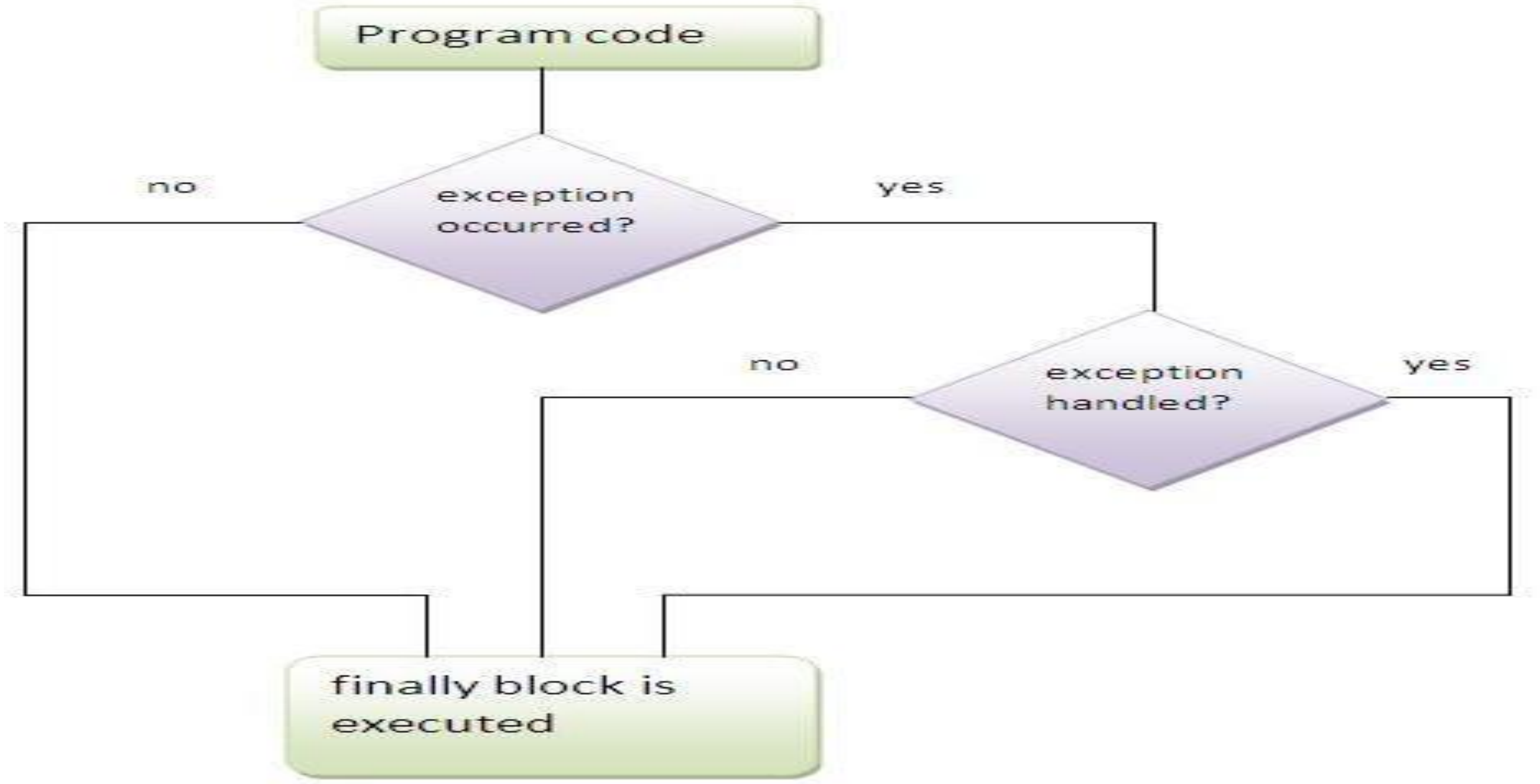
```
try
{
   statement 1;
   statement 2;
   try
   {
      statement 1;
      statement 2;
   }
   catch(Exception e)
   {
   }
}
catch(Exception e)
{
}
```

```
class Excep6
{
 public static void main(String args[]){
  try{
    try{
     System.out.println("going to divide");
     int b =39/0;
    }catch(ArithmeticException e){System.out.println(e);}
    B
    try{
    int a[]=new int[5];
    a[5]=4;
    }
catch(ArrayIndexOutOfBoundsException e)
{System.out.println(e);}
    System.out.println("other statement);
  }catch(Exception e){System.out.println("handeled");}

  System.out.println("normal flow..");
 }
}
```

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.


Program code
no
exception occurred?
yes
no
exception handled?
yes
finally block is executed