

GLS UNIVERSITY

SEM – III
0301301 - CORE JAVA

Dr. Disha Shah
Prof. Vidhi Thakkar



Index

Exception Handling

- Introduction
- ExceptionTypes
- Handling Techniques
 - try..catch
 - throw keyword
 - throws keyword
 - finally keyword
 - Multi-catch
 - User-defined exception

Multi-threading

- Introduction
- Thread Life cycle
- java.lang.Thread
- Main thread
- Creation of New Thread
 - By inheriting Thread class
 - By implementing Runnable interface
 - Thread Priorities



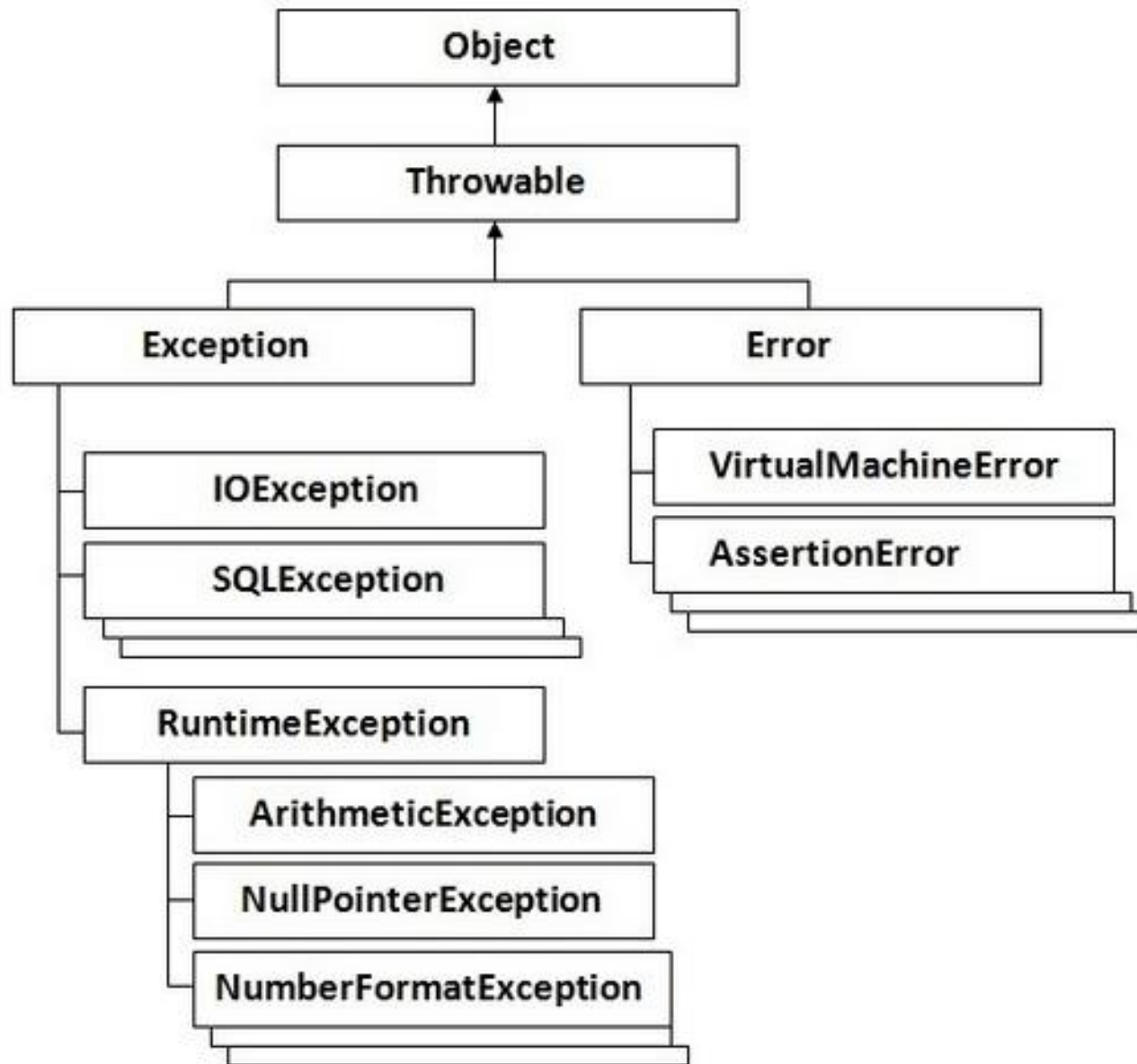
Unit – 4

Exception Handling in Java

Exception Handling

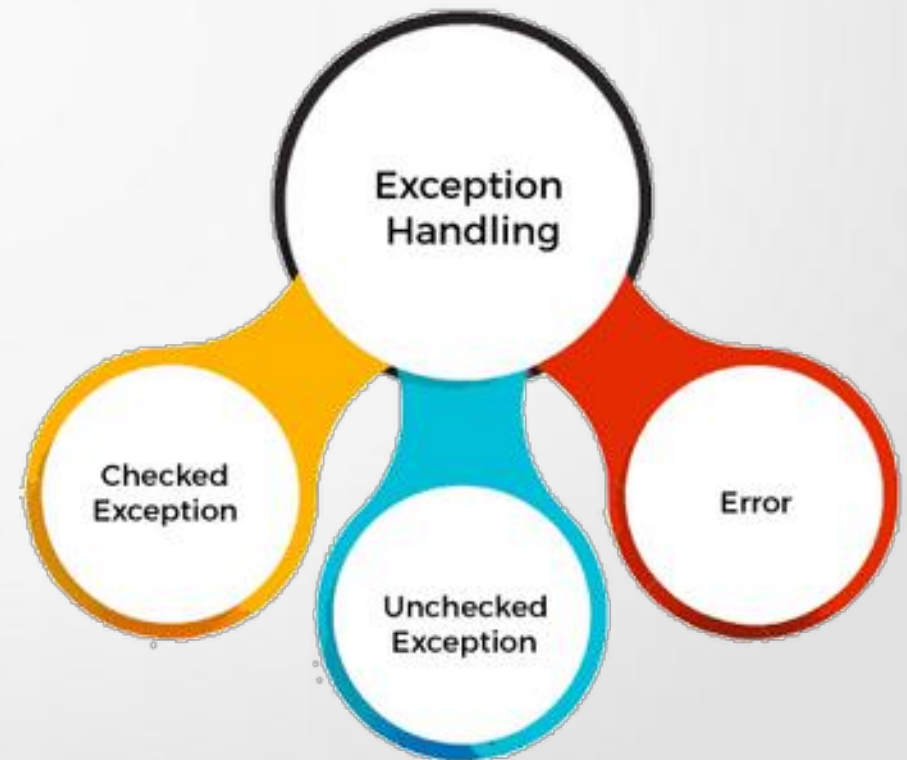
- The exception handling in java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.
- **Definition:**
Exception is an abnormal condition. In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.
- **What is exception handling?**
Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IO, SQL, Remote etc.
- The core advantage of exception handling is to maintain the normal flow of the application.

Exception Handling



Types of Exception

- The sun microsystem says there are three types of exceptions:
 - Checked Exception
 - Unchecked Exception
 - Error



Types of Exception

- **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.

- **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.

- **Error**

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

Some scenarios where exceptions may occur

1) Scenario where `ArithmeticException` occurs

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

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

2) Scenario where `NullPointerException` occurs

- If we have null value in any variable, performing any operation by the variable occurs an `NullPointerException`.

```
String s=null;
```

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


Some scenarios where exceptions may occur

3) Scenario where `NumberFormatException` occurs

- The wrong formatting of any value, may occur `NumberFormatException`. Suppose I have a string variable that have characters, converting this variable into digit will occur `NumberFormatException`.

```
String s="abc";
```

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

4) 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
```



There are 5 keywords used in java exception handling.

- try
- catch
- finally
- throw
- throws

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**

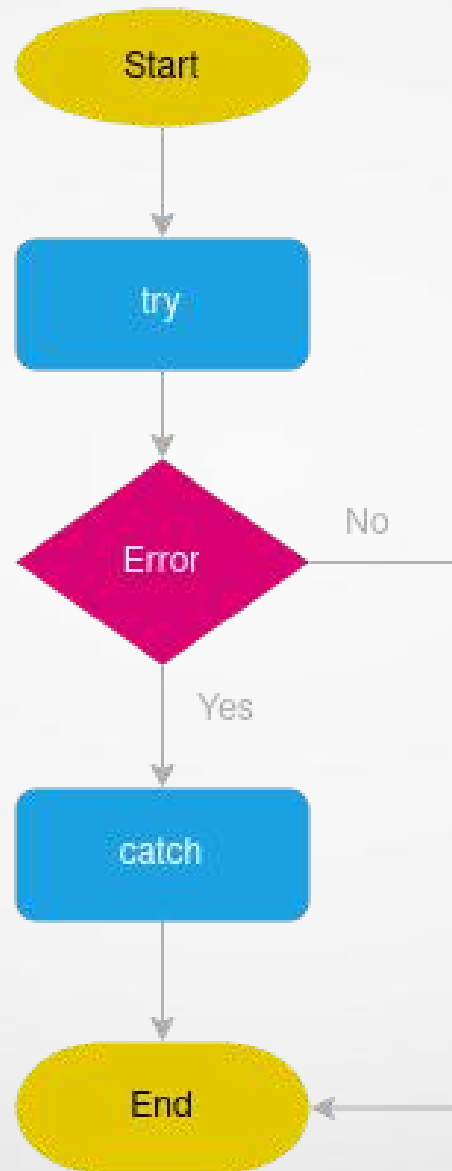
```
try{  
    //code that may throw exception  
}  
catch(Exception_class_Name ref)  
{ }
```

- **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.
- At a time only one Exception is occurred 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 .



Try..catch

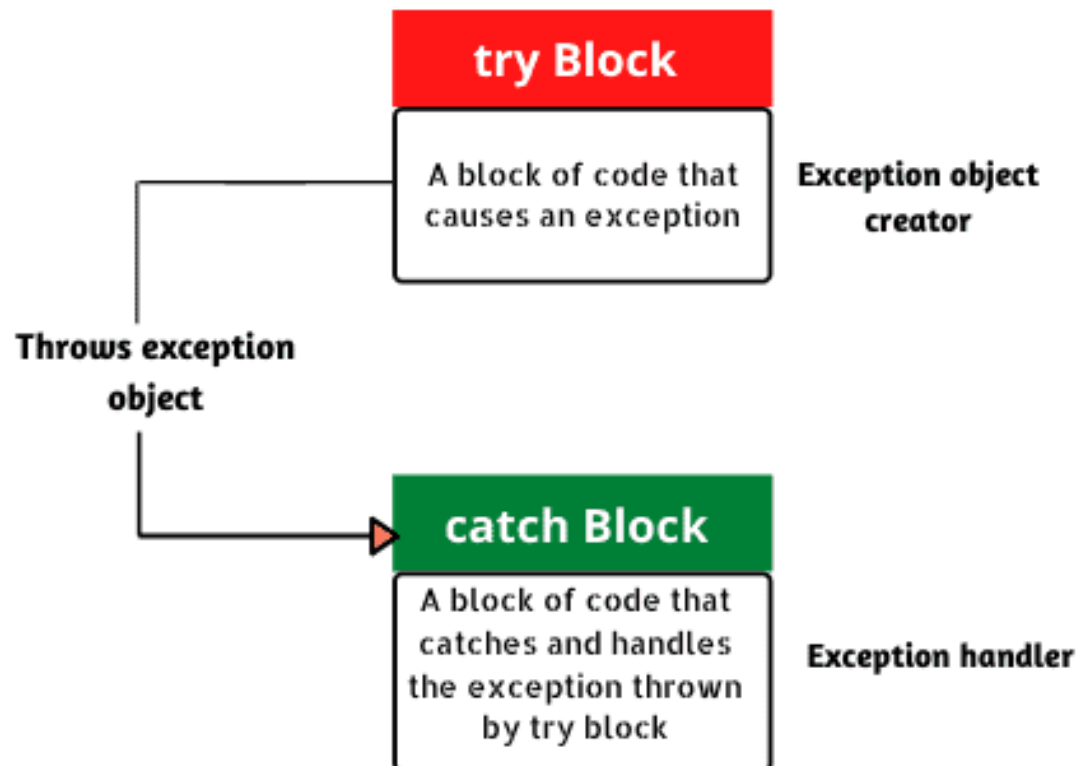


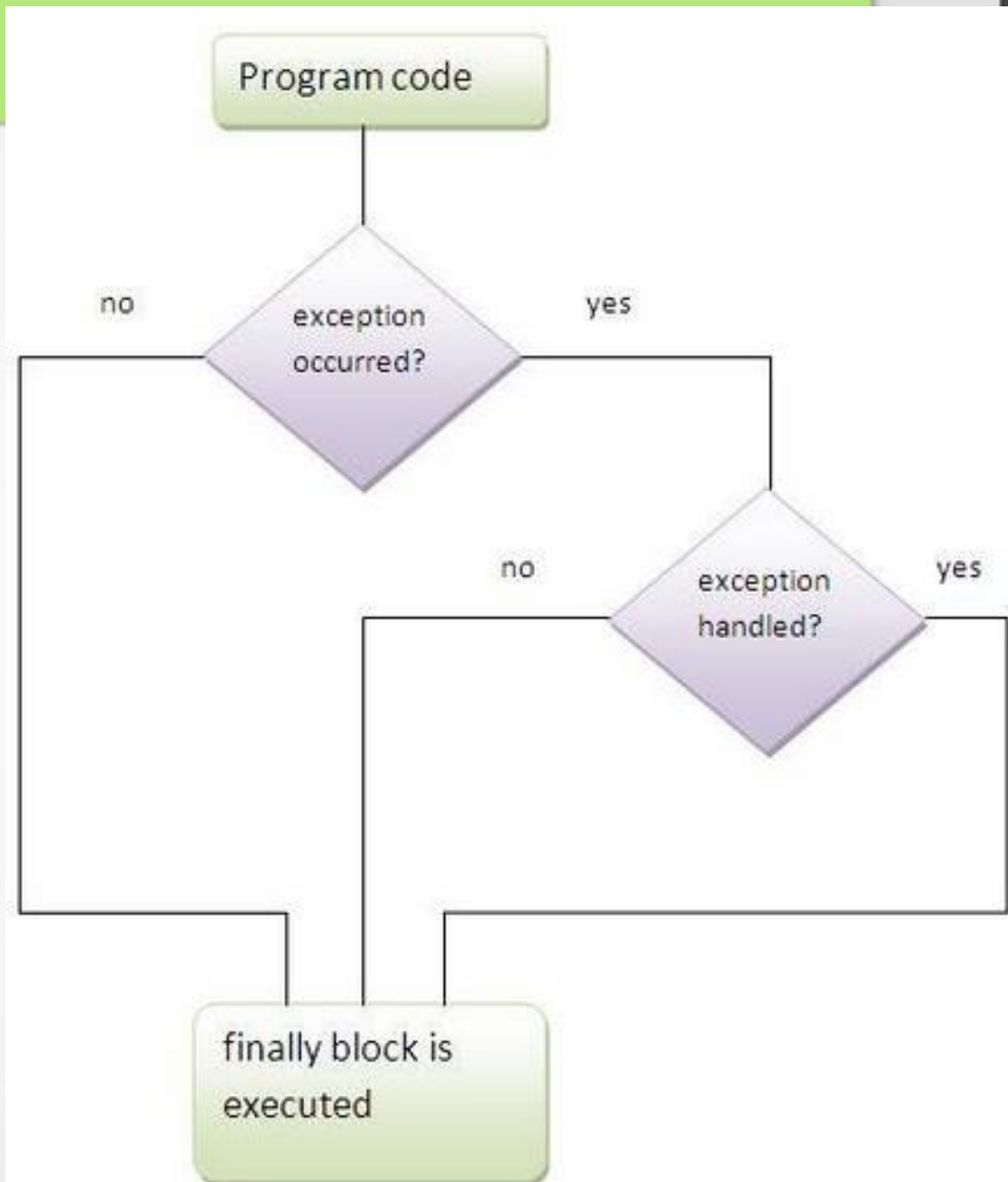
Fig: Exception handling mechanism

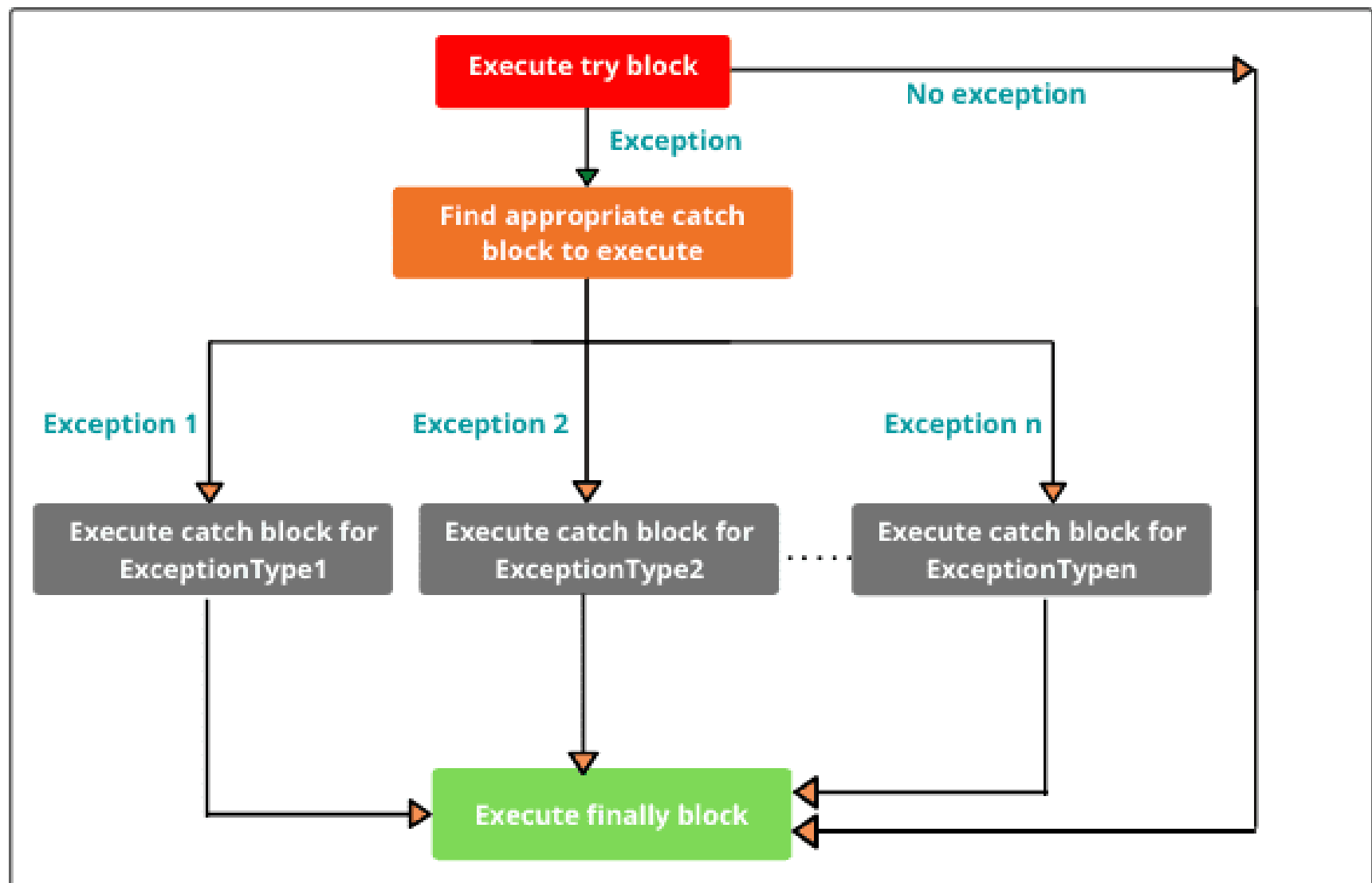
Java finally block

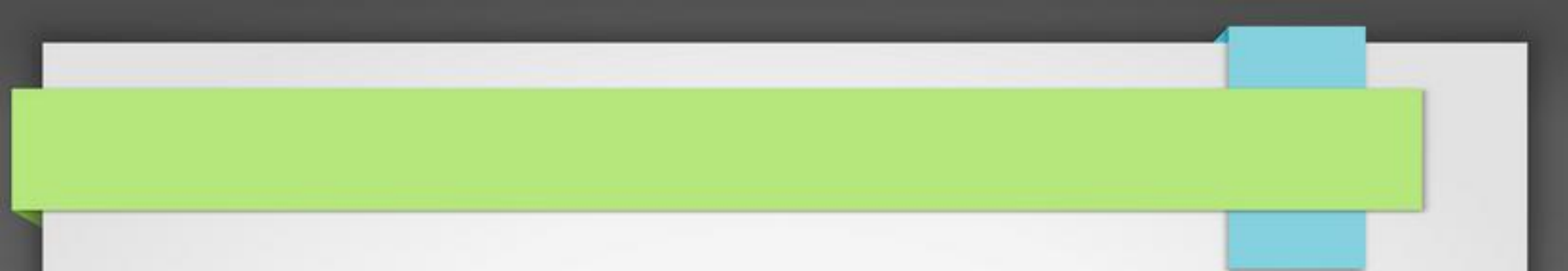
- Java finally block is a block that is used to execute important code such as closing connection, stream etc.
- Java finally block is always executed whether exception is handled or not.
- Java finally block follows try or catch block.
- If you don't handle exception, before terminating the program, JVM executes finally block(if any).
- Finally block in java can be used to put "cleanup" code such as closing a file, closing connection etc.

Java finally block

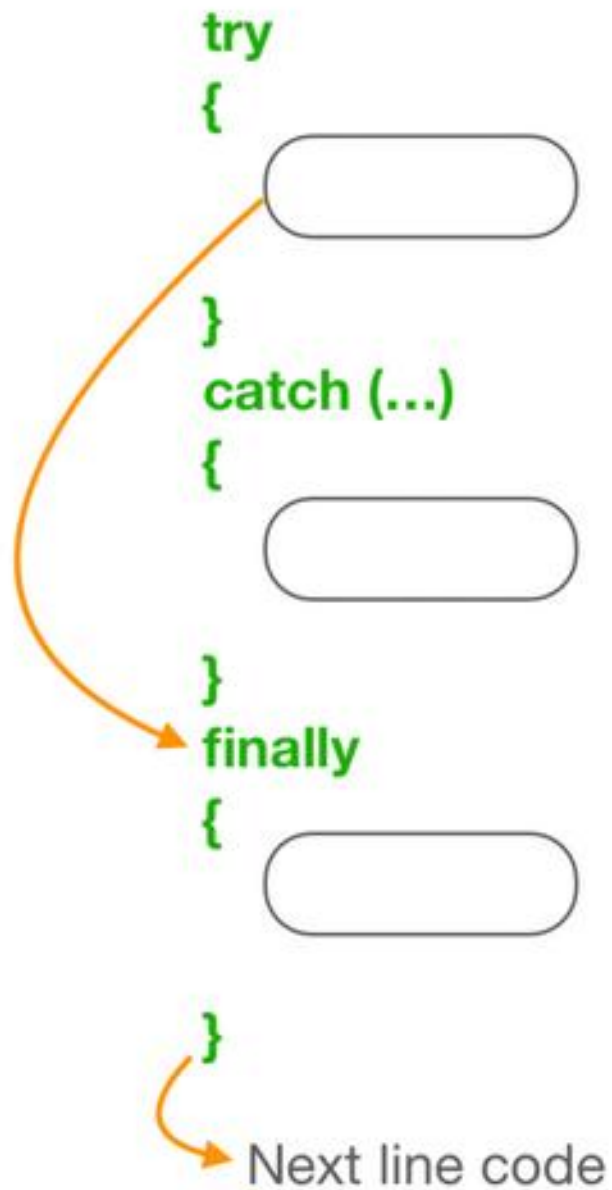
```
try
{
}
catch (ExceptionType name)
{
}
catch (ExceptionType name)
{
}
finally
{
}
```



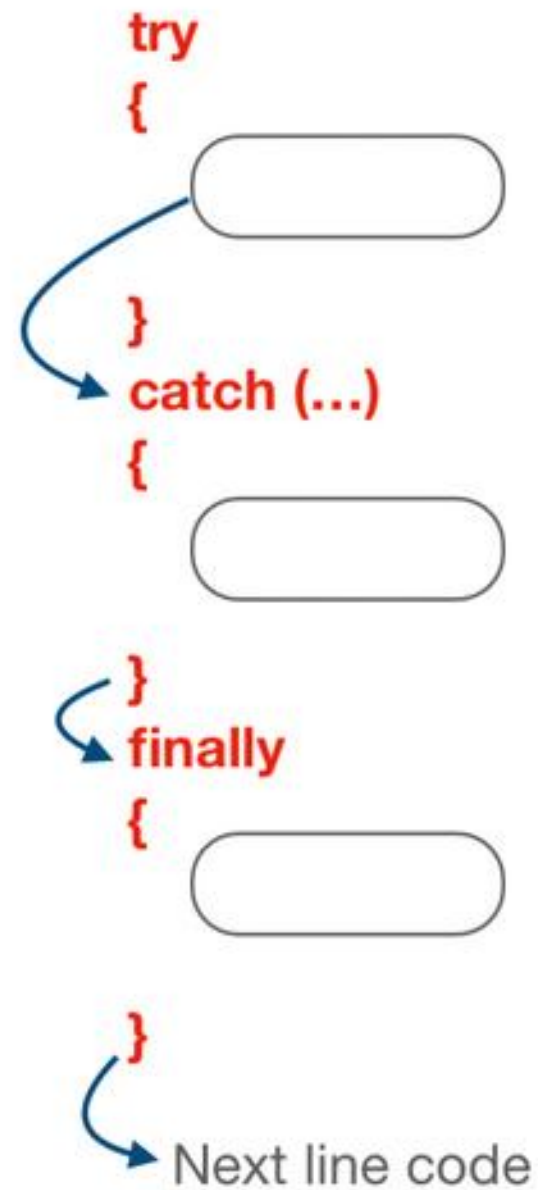


- 
- For each try block there can be zero or more catch blocks, but only one finally block.
 - The finally block will not be executed if program exits. (either by calling `System.exit()` or by causing a fatal error that causes the process to abort).

Without Exception



With Exception



Java throw keyword

- The Java throw keyword is used to explicitly throw an exception.
- We can throw either checked or unchecked exception in java by throw keyword.
- The throw keyword is mainly used to throw custom exception.
- The syntax of java throw keyword is given below.

`throw exception;`

Java throws keyword

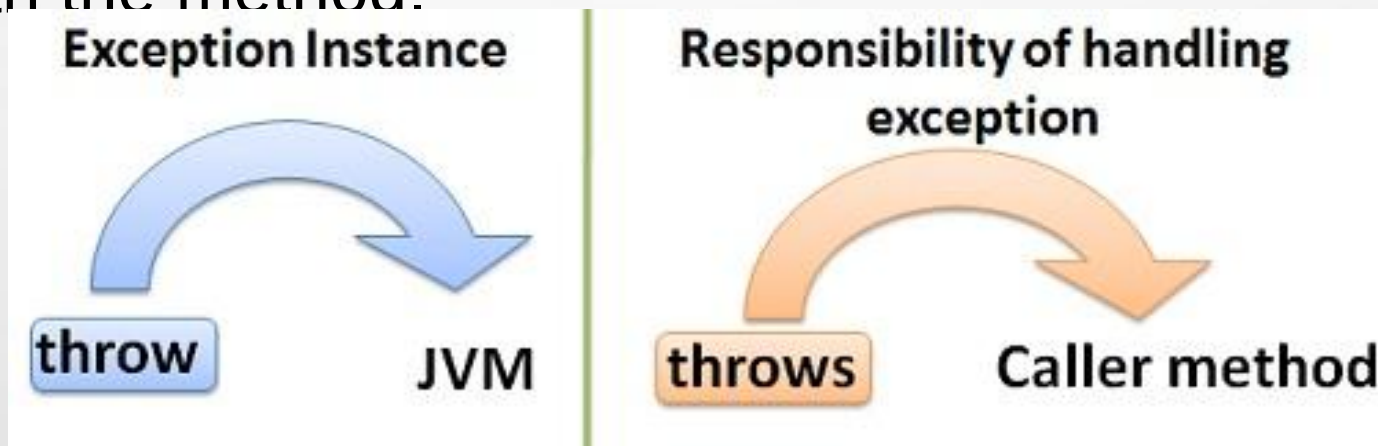
- The Java throws keyword is used to declare an exception.
- It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.
- Syntax of java throws

```
return_type method_name() throws exception_class_name  
{  
    //method code  
}
```

Java throws keyword

- Which exception should be declared
- Ans) checked exception only, because:
 - unchecked Exception: correct the code.
 - error: beyond the control
- **Advantage of Java throws keyword**
 - Now Checked Exception can be propagated (forwarded in call stack).
 - It provides information to the caller of the method about the exception.

- If you are calling a method that declares an exception, you must either catch 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.

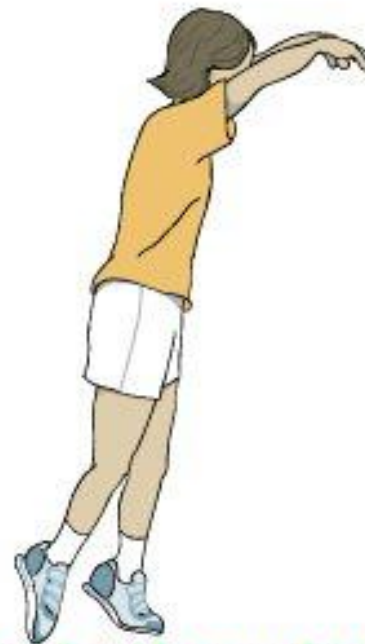


Difference between throw and throws

throws He will throw



throw He is throwing
right now



Difference between throw and throws

throw	throws
1. Java throw keyword is used to explicitly throw an exception	1. Java throws keyword is used to declare an exception.
2. <pre>void m(){ throw new ArithmeticException("sorry"); }</pre>	2. <pre>void m()throws ArithmeticException{ //method code }</pre>
3. Checked exception cannot be propagated using throw only.	3. Checked exception can be propagated with throws.
4. Throw is followed by an instance.	4. Throw is followed by a class.
5. Throw is used within the method.	5. Throws is used with the method signature.
6. You cannot throw multiple exceptions.	6. You can declare multiple exceptions e.g. <pre>public void method()throws IOException,SQLException.</pre>

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.