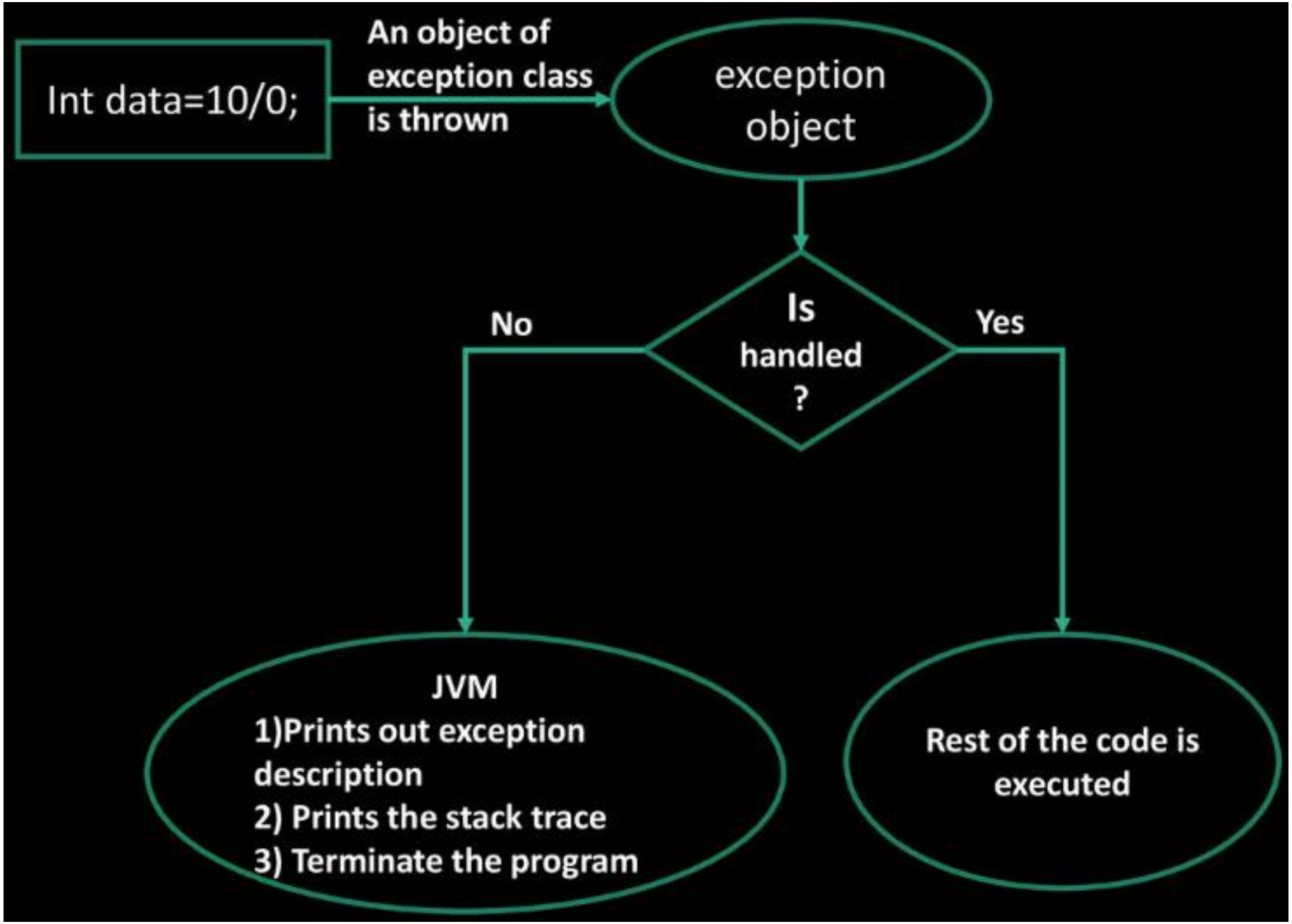
Object Oriented Approach
Exception

Exception

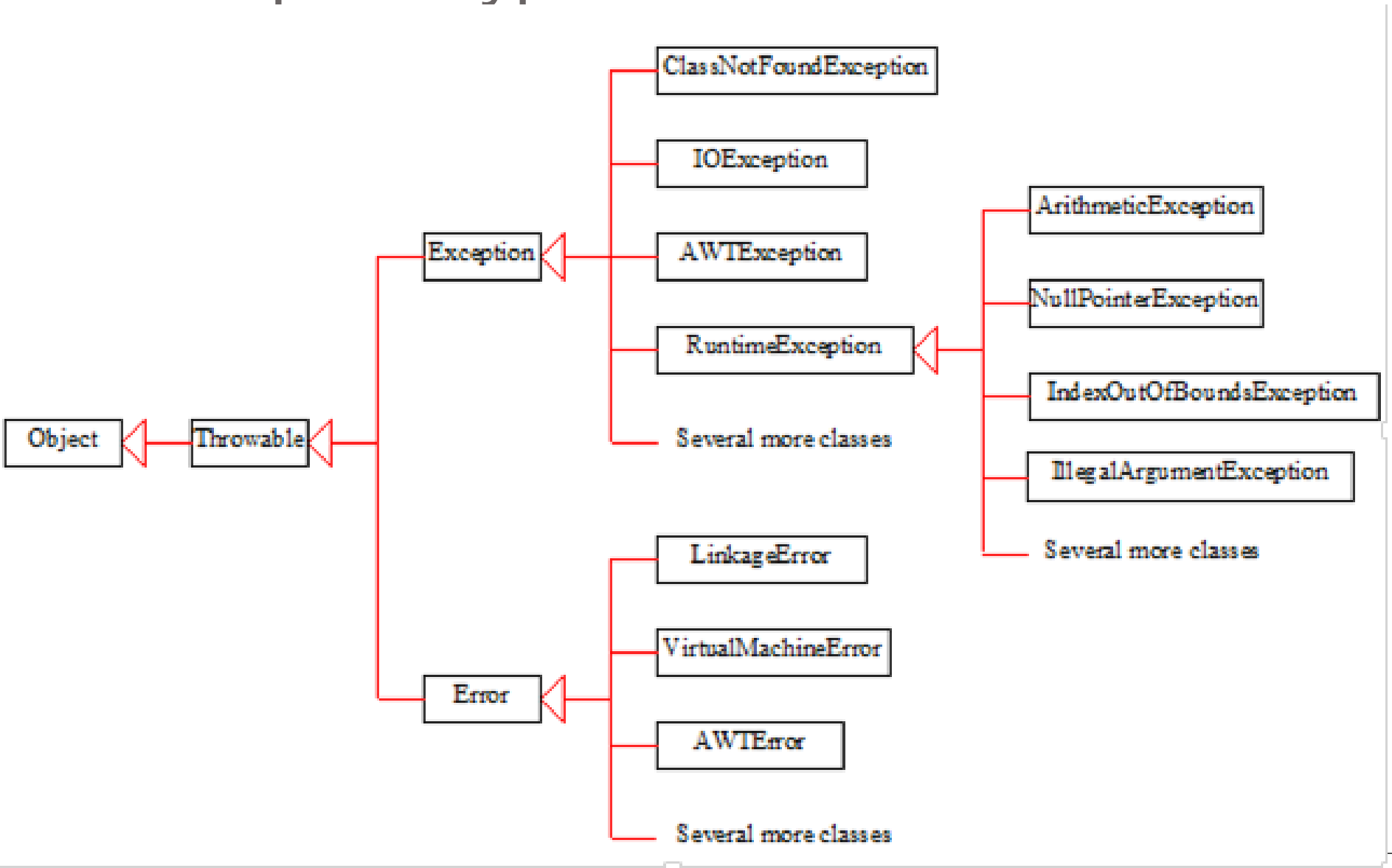
- Exception is a run-time error which arises during the execution of Java program.
- is an event that occurs during the execution of a program that interrupts the normal flow of instructions.
- Some abnormal and typically an event or conditions that occur during the execution which may interrupt the normal flow of program
- An exception can occur for many different reasons:
  - User entered an invalid data
  - A file that needs to be opened cannot be found
  - …

Exception

- An exception is an object that describes an exceptional error that has occurred in a piece of code at run-time.


Int data=10/0;
An object of exception class is thrown
exception object
Is handled ?
No
Yes
JVM
1)Prints out exception description
2) Prints the stack trace
3) Terminate the program
Rest of the code is executed

Exception types



Exception types

- *System errors* are thrown by JVM and represented in the Error class. The Error class describes internal system errors. Such errors rarely occur. If one does, there is little you can do beyond notifying the user and trying to terminate the program gracefully.

- Exception describes errors caused by your program and external situations. These errors can be caught and handled by your program
- RuntimeException is caused by programming errors, such as bad casting, accessing an out-of-bounds array, and numeric errors
-

Class Throwable

public class java.lang.Throwable extends java.lang.Object

Constructeurs

Throwable()
Constructs a new *Throwable* with *null* as its error message string.

Throwable(*String message*);
Constructs a new *Throwable* with the specified error message.

Quelques méthodes

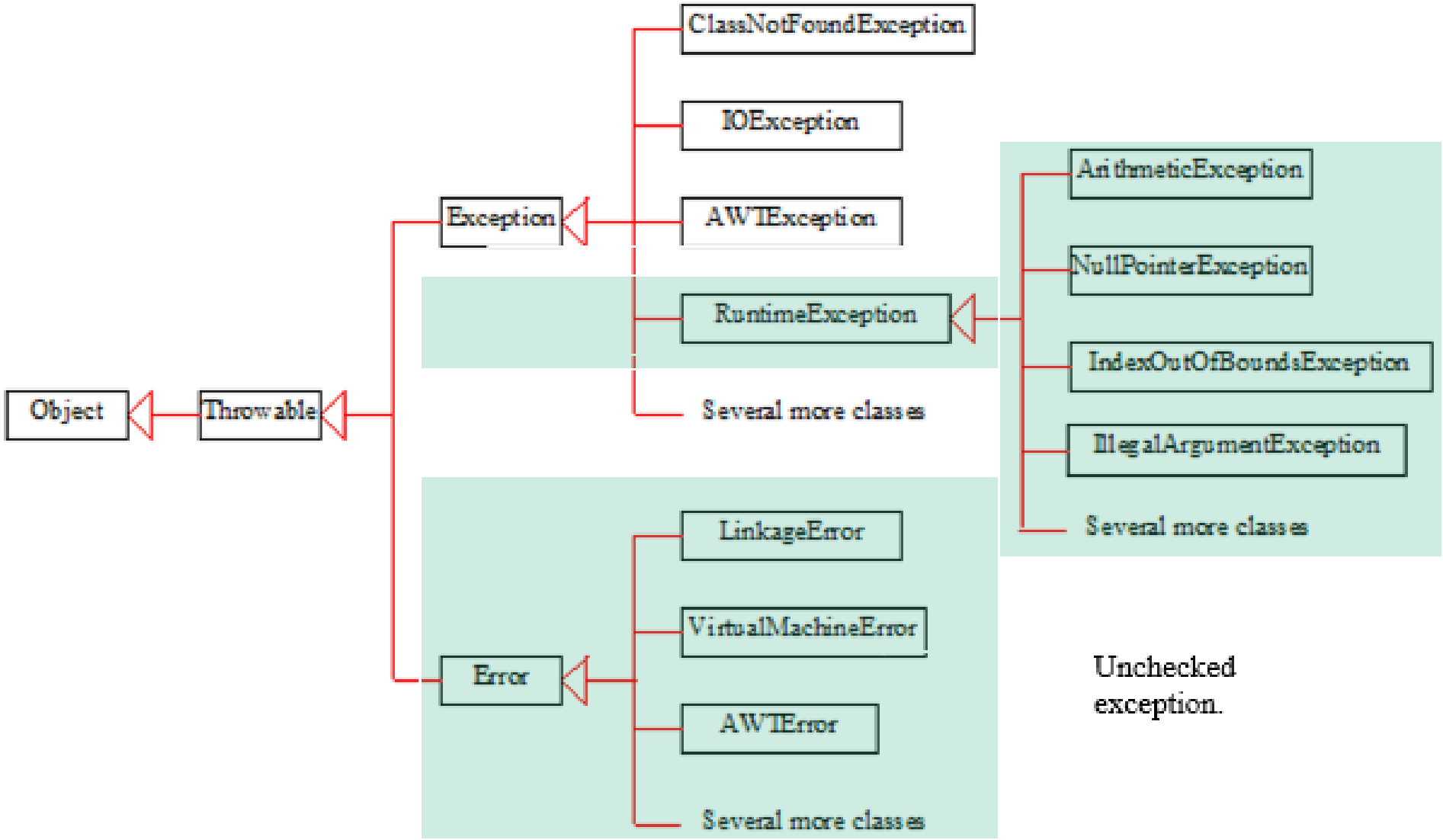
String **getMessage**();
Returns the error message *String* of this *Throwable* object.

void **printStackTrace**();
Prints this *Throwable* and its backtrace to the standard error stream.

String **toString**();
Returns a short description of this *Throwable Object*.

Exception

- RuntimeException, Error and their subclasses are known as *unchecked exceptions*. All other exceptions are known as *checked exceptions*, meaning that the compiler forces the programmer to check and deal with the exceptions



Unchecked exception.

Declaring, throwing, catching exception

```
try {
  statements;  // Statements that may throw exceptions
}
catch (Exception1 exVar1) {
  handler for exception1;
}
catch (Exception2 exVar2) {
  handler for exception2;
}
…
catch (ExceptionN exVar3) {
  handler for exceptionN;
}
```

Example

```
import java.util.Scanner;
public class EXCEPTION1 {
   static Scanner k=new Scanner(System.in);
 public static void P( ) {
       int x =k.nextInt( ) ;
        if ( x >0)  throw new Stop("Positive") ;}
public static void main (String [ ] args) {
  System.out.println ( "Message 1 " ) ; / / 1
   try {
        P ( ) ;
        System.out.println ( "Message 2 " ) ; / / 2
       } catch(Stop e) { System.out.println (e.getMessage());}
 System.out.println( "Message 4 " ) ; / / 4 }
}
}
class Stop extends RuntimeException {
public Stop(){}
public Stop(String s){super(s);}
}
```

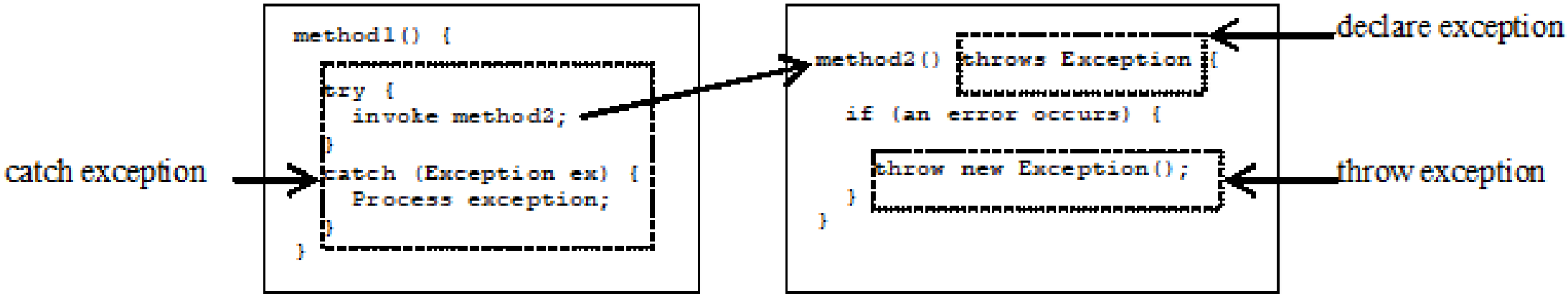
Exception thrown by a method

```
public int add(int a, String str) throws NumberFormatException
{
   int b = Integer.parseInt(str);
   a = a + b;
   return a;
}
```

Exception

```
public int add(int a, String str) {
   try {
         int b = Integer.parseInt(str);
         a = a + b;
   } catch (NumberFormatException e) {
         System.out.println(e.getMessage());
   }
   return a;
}
```

Example


method1() {
try {
invoke method2;
}
catch (Exception ex) {
Process exception;
}
}
method2() throws Exception {
if (an error occurs) {
throw new Exception();
}
}
catch exception
declare exception
throw exception

Exercise

- Define a class exception AccountException
- Throw exception =>> message to be displayed sum>balance..