— Java Exceptions —

An exception is a mechanism for handling errors during execution. A method indicates that an error occurs by throwing an exception. When this occurs the method immediately returns, and execution returns to the point where the method was called, where the exception is caught and the error dealt with.

Note that exceptions in Java are throwable objects. All types of exceptions are subclasses of the class java.lang.Exception.

To catch an exception use a try-catch block:

```
try {
    // Code which might throw an exception(s)
}
catch (exceptionClass e) {
    // Deal with problem
}
catch (otherExceptionClass e) {
    // Deal with other problem
}
```

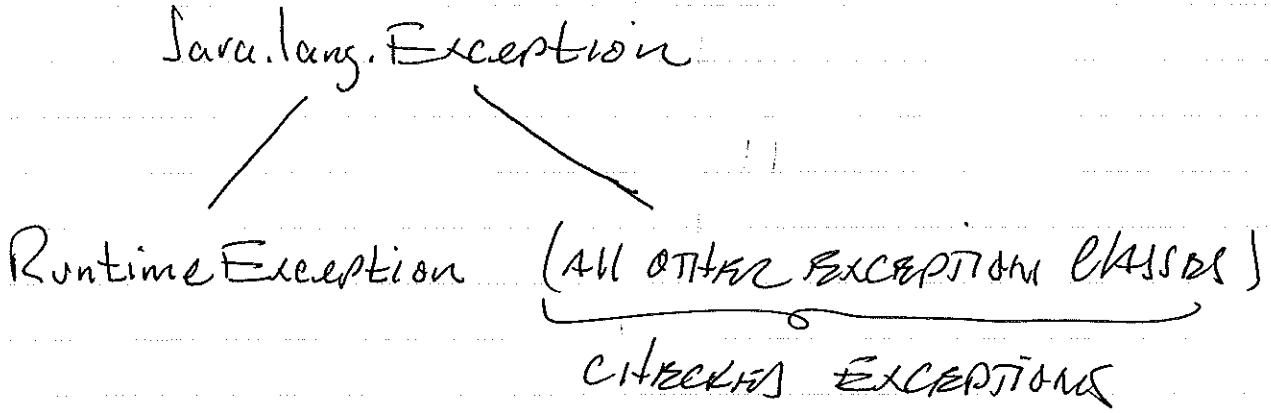
IF AN OPERATION IN try BLOCK THROWS AN EXCEPTION, REMAINDER OF THAT BLOCK WILL BE SKIPPED, AND CONTROL TRANSFERS TO THE FIRST CATCH BLOCK WITH A MATCHING EXCEPTION TYPE.

FOR THIS REASON, YOU CAN ASSUME WHEN WRITING try BLOCK THAT ALL OPERATIONS SUCCEED!

IF THE EXCEPTION DOES NOT MATCH THE TYPE IN ANY CATCH BLOCK, THE EXCEPTION WILL BE THROWN UP THE STACK OF FUNCTION CALLS TO THE CALLING FUNCTION & EITHER CAUGHT THERE OR THROWN FURTHER.

IF AN EXCEPTION IS THROWN ALL THE WAY UP TO THE main FUNCTION WHERE PROGRAM EXECUTION STARTS, THEN IS THROWN FURTHER, IT IS CAUGHT BY THE OS WHICH SIMPLY HALTS EXECUTION. (i.e. EXCEPTION 'PASS OUT OF THE PROGRAM.)

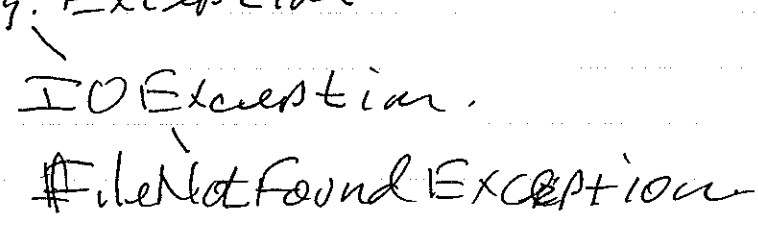
JAVA HAS TWO TYPES OF EXCEPTIONS : CHECKED EXCEPTIONS AND RUNTIME EXCEPTIONS

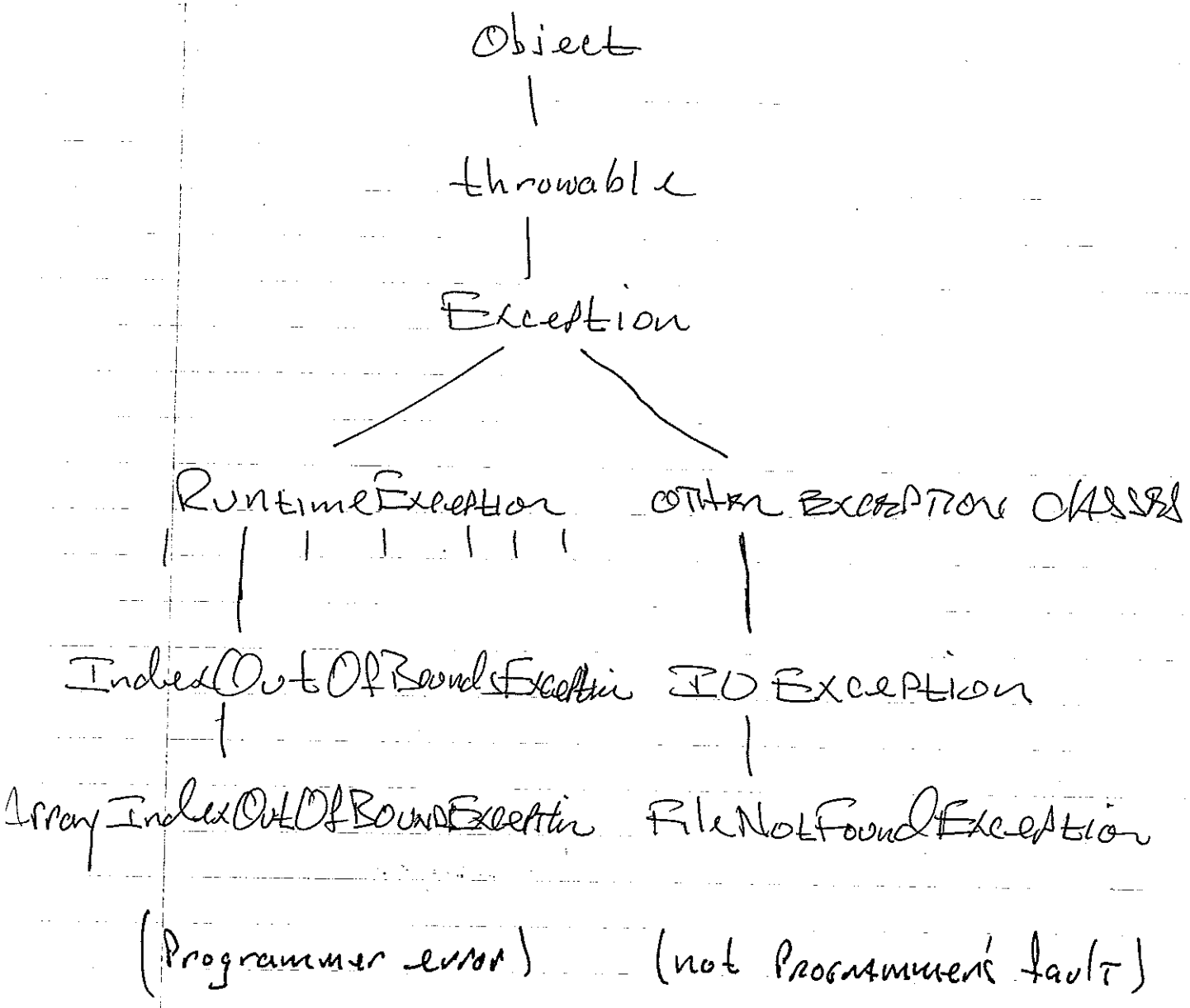


CHECKED EXCEPTIONS ARE USED IN SITUATIONS WHERE A METHOD HAS ENCOUNTERED A SERIOUS PROBLEM SUCH AS MAY REQUIRE PROGRAM TERMINATION.

IF A CHECKED EXCEPTION IS GENERATED IT MUST BE EITHER CAUGHT LOCALLY OR THROWN UP TO THE CALLING METHOD.

EG. THE CONSTRUCTOR OF File class CAN THROW FileNotFoundException. SO ANY method WHICH CALLS THIS method MUST EITHER PROVIDE A try-catch BLOCK OR DECLARE ITSELF AS THROWING SUCH AN EXCEPTION. (SEE LAB 2)

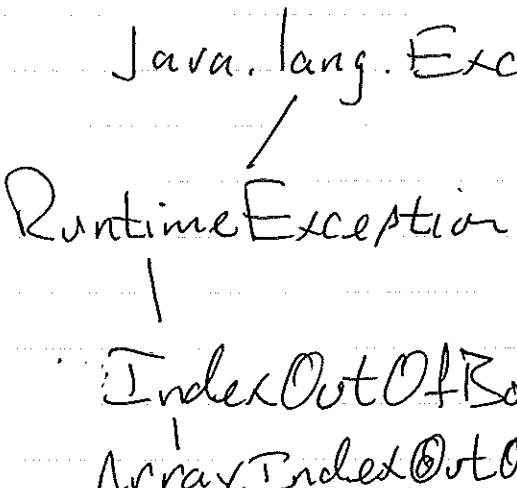



Object
Throwable
Exception
RuntimeException
OTHER EXCEPTION CLASSES
IndexOutOfBoundsException
IOException
ArrayIndexOutOfBoundException
FileNotFoundException
(Programmer error)
(not Programmer's fault)

RUNTIME EXCEPTIONS ARE USED WHERE A METHOD HAS ENCOUNTERED A PROBLEM WHICH IS NOT SO SERIOUS.

RUNTIME EXCEPTIONS NEED NOT BE CAUGHT locally or EXPLICITLY THROWN TO THE CALLING METHOD.

EX. Java.lang.Exception



TO THROW AN EXCEPTION DO:

```
throw new exceptionClass(stringMsg);
```

EX.
```
Public void someMethod() throws someException {
    // code which might cause problems
    throw new someException("msg");
}
```

IF someException IS A SUBCLASS OF RuntimeException THE throws CLAUSE IS NOT NECESSARY, OTHERWISE IT IS NECESSARY.

YOU MAY DEFINE YOUR OWN EXCEPTION CLASSES. USE EITHER Exception OR RuntimeException AS THE SUPERCLASS DEPENDING ON WHAT YOU BELIEVE IS THE SEVERITY OF THE PROBLEM.

Ex. Public class ListFullException
extends RuntimeException {

Public ListFullException(String s) {
super(s);
}
}

THIS CALLS THE CONSTRUCTOR OF THE SUPERCLASS RuntimeException. S IS PRINTED TO stderr WHEN EXCEPTION IS THROWN.

Ex. Public class ListIndexOutOfBoundsException
extends IndexOutOfBoundsException {

Public ListIndexOutOfBoundsException(String s){
super(s);
}
}

PUT THESE class DEFINITIONS IN SEPARATE FILES.

NOW REWRITE get, add, remove SO AS TO THROW EXCEPTIONS ← EXERCISE

```
// get
// Pre: 1 <= index <= size()
// Post: returns item AT index
Public int get(int index)
      throws ListIndexOutOfBoundsException {
    if (index < 1 || index > numItems) {
        throw new ListIndexOutOfBoundsException(
            "get: ~~ListIndexOutOfBoundsException~~");
    }
    return item[arrayIndex(index)];
}

//add
//Pre: size() < MaxLength, 1 <= index <= size()+1
//Post: !isEmpty()
Public void add(int index, int newItem)
    throws ListFullException,
           ListIndexOutOfBoundsException {
    if(numItems == MAX_LENGTH) {
        throw new ListFullException(
            "add: ~~ListFullException~~");
    }
    if(index < 1 || index > (numItems + 1)) {
        throw new ListIndexOutOfBoundsException(
            "add: ~~ListIndexOutOfBoundsException~~");
    }
    for(int i = numItems; i >= index; i--)
        :
}
```