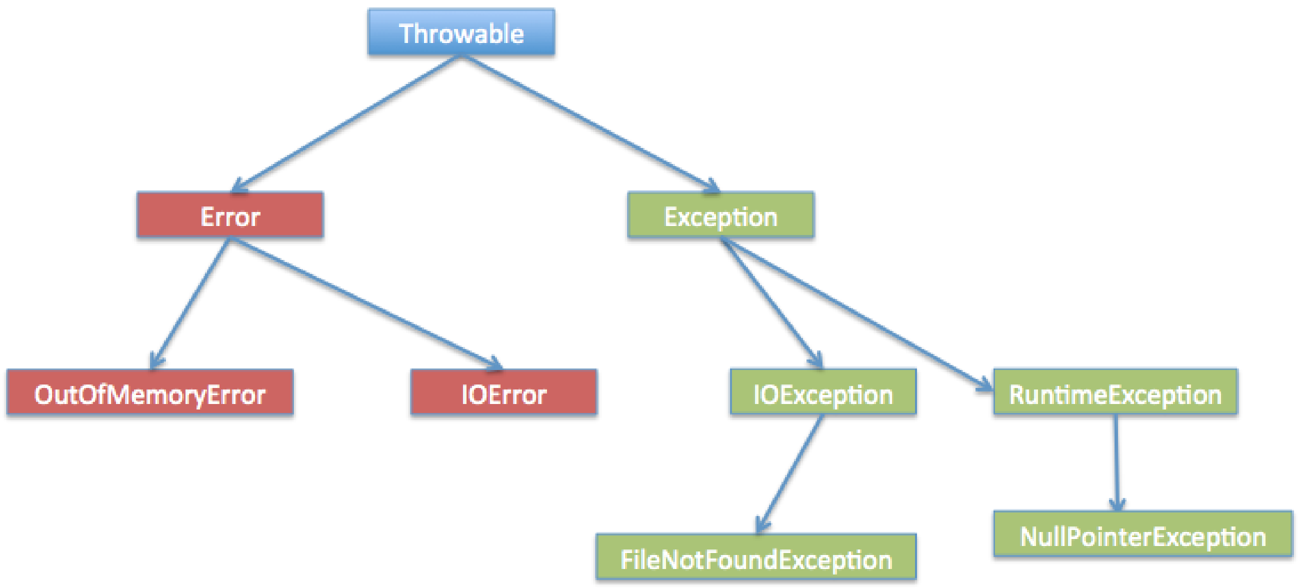
2. Exception handling

Exception Handling

> All exceptions are `class` that are derived from the `Throwable class`

Exception hierarchy



```
Exception
│
├─ RuntimeException
│   │
│   ├─ NullPointerException
│   │
│   ├─ IndexOutOfBoundsException
│   │
│   ├─ SecurityException
│   │
│   └─ IllegalArgumentException
│       │
│       └─ NumberFormatException
│
├─ IOException
│   │
│   ├─ UnsupportedCharsetException
│   │
│   ├─ FileNotFoundException
│   │
│   └─ SocketException
│
├─ ParseException
│
├─ GeneralSecurityException
│
├─ SQLException
│
└─ TimeoutException

Source:
https://www.liaoxuefeng.com/wiki/1252599548343744/1264737765214592
```

`Throwable` is a derived class of `object`. `Throwable` has two subcategories: `Error` and `Exception`

`Error`: severe problems occurred that cannot be solved

`Exception`: problems that occur when the program is running. It can be captured by `try...catch`

- `RuntimeException` as well as its subclasses
- Other Exceptions / Non-RuntimeException Exceptions

> - Exceptions other than RuntimeException **must be** `try...catch`; else the compiler would report and halt
> - `Errors` and `RuntimeException` as well as its derived classes are not forced to be `try...catch`

Capture the exception: `try...catch`

Single catch

```
try{
    //Program
}catch(ExceptionName e){
    //What to perform when Exception is detected in the try program
}
```

Multiple catch

It is possible to have multiple catch, so that different program could be run when different exceptions are encountered.

Be aware that only one catch will be executed:

- When the `try` program is run, it halts and jumps to the `catch` section when an exception occur
- The compiler reads the catch one by another in sequence. If the exception coincides, the corresponding program is executed.
- And then it ends

> Hence, it is important to note that Exception1 should be the subclass of Exception2, and Exception2 should be the subclass of Exception3. Else it is impossible to execute the second and third catch

```
try{
   //Program
}catch(ExceptionName1 e1){
  //Program
}catch(ExceptionName2 e2){
  //Program
}catch(ExceptionName3 e3){
  //Program
}
```

Finally

> The statement that will be executed anyway no matter whether the Exception occur in the try processes

```
try{
    process1();
    process2();
}catch(IOException e){
    System.out.println("IO excpetion detected");
}finally{
    System.out.println("END");
}

//This is equivalent to:
try{
    process1();
    process2();
    System.out.println("END");
}catch(IOException e){
    System.out.println("IO excpetion detected");
    System.out.println("END");
}
```