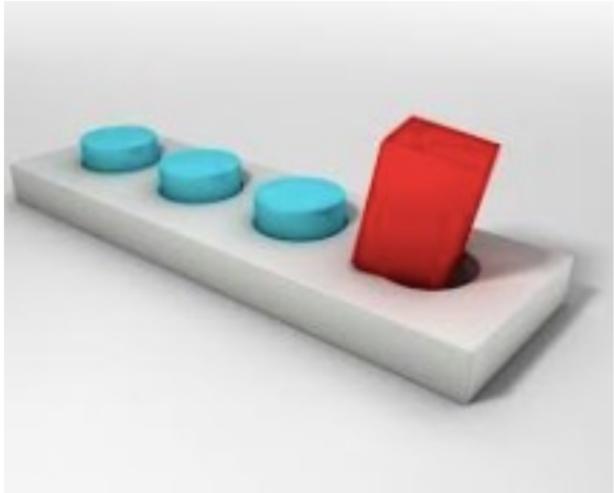
Input Validation

- Decide on a consistent model on how to handle bad input data
  - Pretend the method succeeded in a “vacuous” manner?
  - Have the method fail automatically?
  - Throw an exception?
  - Return an error code?

DALHOUSIE UNIVERSITY
Inspiring Minds

Input Validation – Check for unexpected data

- Objects (eg. String, Integer, ArrayList, …)
  - Watch for null objects
  - Watch for objects with no data in them
- Formatted data (eg. a date from a user in yyyy-mm-dd format)
  - Double-check the format of the data coming in



Input Validation – Check for unexpected data

- Data ranges or enumerated answers (eg. user response of “yes” or “no”; day number in a month)
  - If you’re expecting data to be in a range, check for that range

- Special characters
  - Scan strings for any characters that might have a special meaning to other libraries where you plan to pass the data
    - Eg. & character if you’re sending out HTML
      ; character in an SQL statement
      “ character in a string

Input Validation – Check for unexpected data

- Test the length of input data, if it has a potential of making a difference to your code
  - Strings and buffers are notorious here.

- Tables, arrays, or more complex data structures contain meaningful data on which to operate

DALHOUSIE UNIVERSITY
Inspiring Minds

Input Validation

- Generally a pile of “if” statements in your method where input data comes in
  - Acts as as preconditions to continue with the method

- Often exploit a common compiler optimization
  - In a big conjunction for an “if” statement, the conditions are evaluated left-to-right and stop as soon as one is false
    - Consequence: when you reach a condition then you assume that all the ones to the left of it in the expression are true
  - Sample use:
    - If ((node != null) && !node.word.equals(“”))
    - The “node.word.equals” would crash if node were null, but that case is cleared with the earlier part of the expression

Return Codes

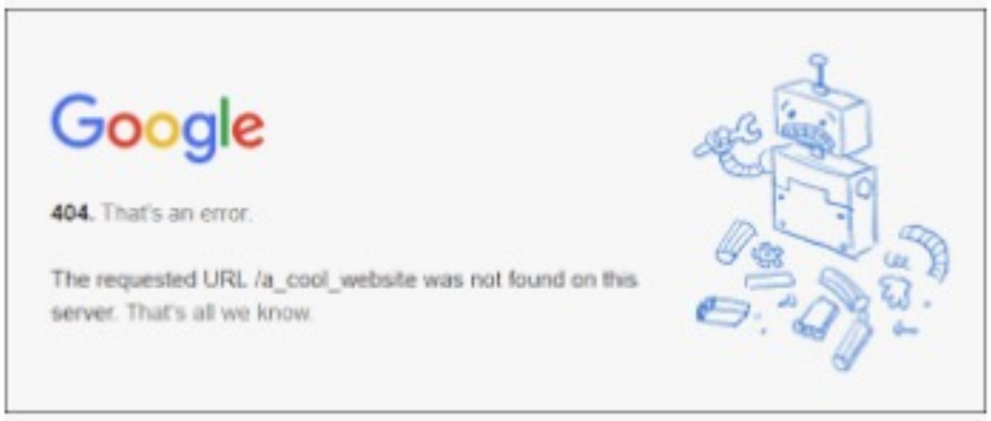
Return Codes

- Have functions return information about how the computation ended
  - Successfully
  - A category of error

- Come in addition to returned information

Return Codes

- Many return codes built structure or meaning into the codes
  - Eg. HTTP return codes
    - 100-199 – informational response
    - 200-299 – successful operation
    - 300-399 – redirection response
    - 400-499 – client-side data error
    - 500-599 – server-side error
  - Individual numbers gave more information about the nature of an error.



404. That's an error.

The requested URL /a_cool_website was not found on this server. That's all we know.

Common Structure in C

- Common to be the return value of the function while the function’s actual data returns as a pass-by-reference parameter

  Eg. int myFunction (int inParameter, char *outParameter);

  Caller then does

  Constant to be defined elsewhere as the success return code

```
if (myFunction( in, &out ) != OK) {
        /* Do error handling */
} else {
        /* Continue with good case code */
}
```

Return Codes

- Advantages
  - Portable concept across many languages
  - Easily recognized
  - Can structure the codes
- Disadvantages
  - Error-handling merged with regular control flow
  - Need to coordinate the meaning of the return codes
  - Relies on the calling function to check for and act on errors

Exceptions

Exceptions

- Report a not-uncommon problem to your calling method
- Use exceptions for error situations that you anticipate and whose origin may be out of your scope
  - Eg. bad input from a user, path a to a file that doesn’t exist

DALHOUSIE UNIVERSITY
Inspiring Minds

Exceptions

- Not all languages include exceptions
- An updated way to report an error condition to a calling method
  - Generally an upgrade to return codes

- Don’t use exceptions to just “pass the buck” to someone else to handle an error

Exceptions

- Exceptions are objects like any other in the system
  - They store information
  - The belong to a hierarchy and can inherit data and methods from their superclass
  - You need to create one to send it back

Exception Hierarchy

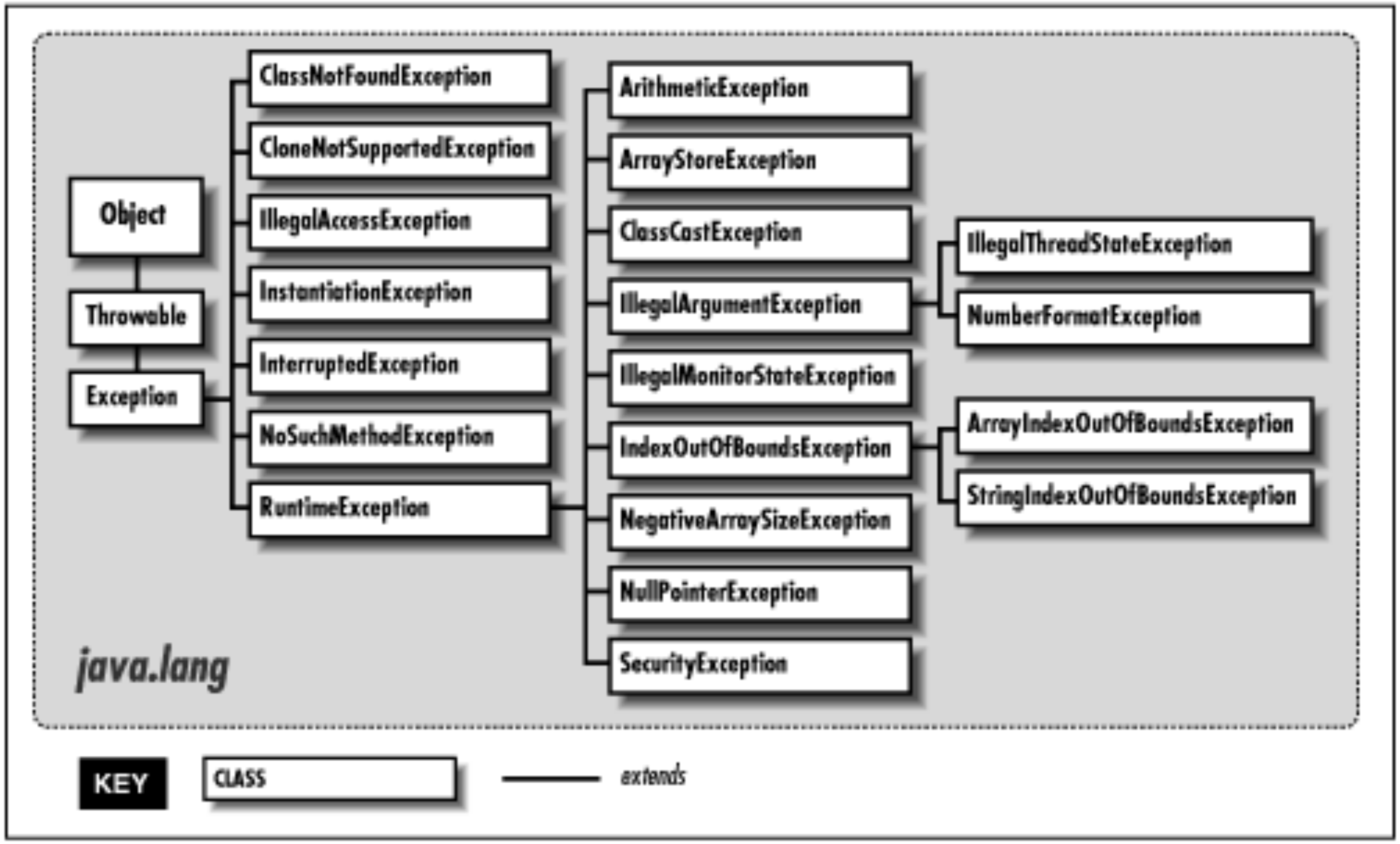


Image from https://docstore.mik.ua/orelly/java/langref/ch09_04.htm

2 Parts to Exceptions in Java

- Sending an exception out of one method
  - Declare that the method might send out an exception
  - Create an object of the exception type
  - Return the exception object with the “throw” keyword
- Receiving an exception in a calling method
  - Be prepared to receive an exception by placing the called code in a “try” block
  - List the exceptions that you will handle along with the code to handle it in a “catch” block
  - Provide clean-up code in a “finally” block

DALHOUSIE UNIVERSITY
Inspiring Minds

Throwing an Exception in Java

```
public void myMethod ( void ) throws IOException {
          …

          if ( exceptional file case detected ) {
                    throw new IOException( “Exception message” );
          }

          …
}
```

Catching an Exception in Java

```
public void someMethod( void ) {
        …
        try {
                myMethod( );
        } catch ( IOException e ) {
                // Do something with IOException and data in object “e”
        } catch (Exception f) {
                // Do something else for another exception type
        } finally {
                // Do code that runs no matter how we end
        }
        …
}
```

Java File Handling Example

```
//Java program to demonstrate FileNotFoundException
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;

 class File_notFound_Demo {

    public static void main(String args[])  {
         File file = null;
        try {

            // Following file does not exist
            file = new File("E://file.txt");

            FileReader fr = new FileReader(file);
        } catch (FileNotFoundException e) {
           System.out.println("File does not exist");
        }
    }
}
```

Doesn’t close the file!!

DALHOUSIE UNIVERSITY *Inspiring Minds*

Code modified from https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/

Java File Handling Example – Extra Care

```
//Java program to demonstrate FileNotFoundException
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;

 class File_notFound_Demo {

    public static void main(String args[])  {
         File file = null;
        try {

            // Following file does not exist
            file = new File("E://file.txt");

            FileReader fr = new FileReader(file);
        } catch (FileNotFoundException e) {
           System.out.println("File does not exist");
        } finally {
             file.close()
        }
    }
}
```

Code modified from https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/

Java File Handling Example – Try With Resource Example

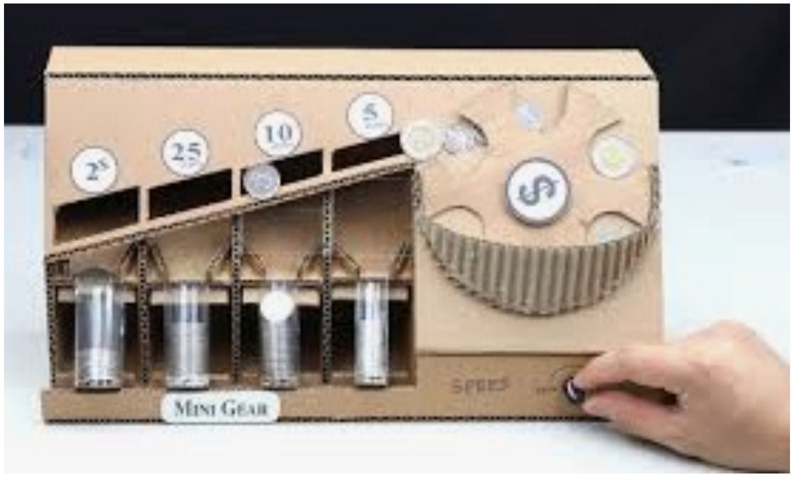
```
//Java program to demonstrate FileNotFoundException
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;

 class File_notFound_Demo {

    public static void main(String args[])  {
        try (new File file = new File("E://file.txt"); ) {

            FileReader fr = new FileReader(file);
        } catch (FileNotFoundException e) {
           System.out.println("File does not exist");
        }
    }
}
```

Will automatically invoke the close() method at the end

Code modified from https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/

Multiple Catch Statements

- The catch statements are checked in order
  - The first one to match gets the exception
  - Consequence: have the specific exceptions before the general exceptions



DALHOUSIE UNIVERSITY
Inspiring Minds

Exception Blocks Good Practices

- Do not leave a catch block empty
  - Basically ignores that an error has happened, which doesn’t fix the problem
- Include enough information in the exception to understand the error
  - You can create your own exceptions if existing ones don’t have enough information for you
- Know which exceptions are thrown to your code
- Standardize your project’s use of exceptions
- Catch specific exceptions when you can
  - Can include a more general catch-all exception after the specific ones