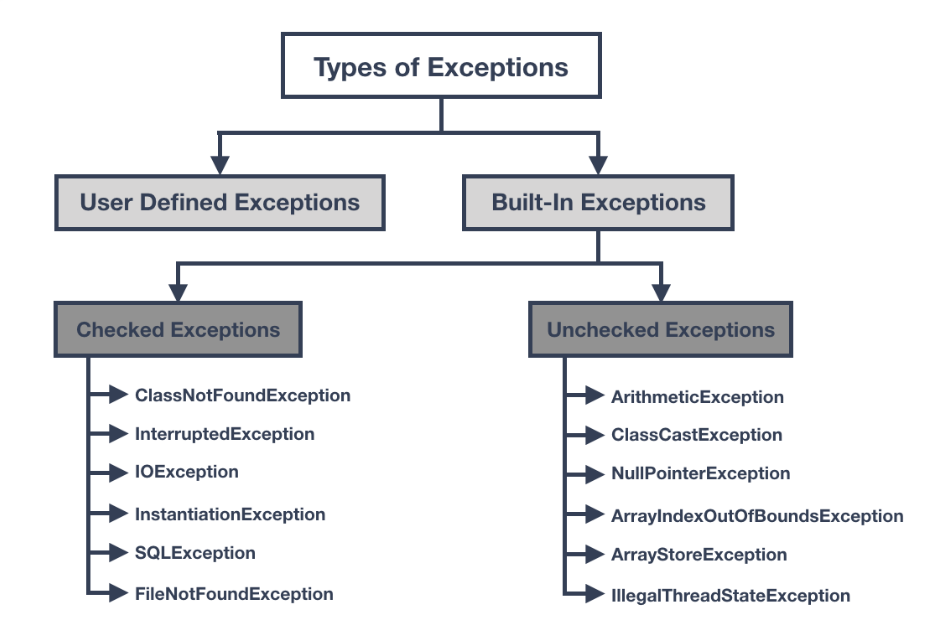
**Handling exceptions**

Exceptions are errors in a program.

Exception handling ensures that the flow of the program doesn't break when an error occurs.

Types of exceptions:

* Checked: Is something that has gone wrong in your code and is potentially recoverable. caught at compile time so if something throws a checked exception the compiler will enforce that you handle it.
* Unchecked: Is something that has gone wrong with the program and is unrecoverable. Not required to be handled.



**Types of classes that handle exceptions:**

* IllegalArgumentException: You passed an incorrect argument to a method.
* InputMismatchException: The console input doesn’t match the data type expected by a method of the Scanner class.
* ArithmeticException: You tried an illegal type of arithmetic operation, such as dividing an integer by zero.
* IOException: A method that performs I/O encountered an unrecoverable I/O error.
* ClassNotFoundException: A necessary class couldn’t be found.

**Throwing an exception**

* when an error occurs in a program, Java throws an exception back at the statement or block of code that created it.

**Try Block**

* Program statements that you think can raise exceptions are contained within a try block.

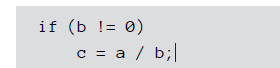
**Catching Block**

* Contains statement that allows you to define a block of code to be executed, if an error occurs in the try block.

**Handling Exceptions with a Pre-emptive Strike**

* Ensuring that you test your data before performing the operation that can lead to an exception and then skipping or bypassing the operation of the data that is problematic.

**e.g.:**

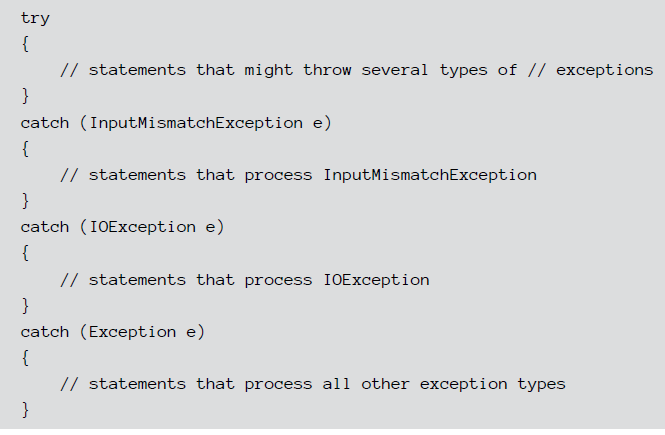


* **Methods to avoid exceptions:**
  + hasNextInt Method: Part of the Scanner class. Checks the next input value to make sure it’s a valid integer. Can be used as a conditional expression in a while loop to avoid an exception.

**Catching All Exceptions at Once**

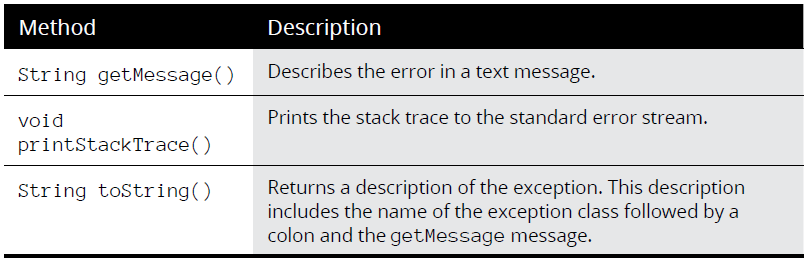
* Java has an all in one Exception class that accommodates all exceptions.
* This is for when you don’t want to be too specific and you have a list of exceptions.
* One try block with multiple exceptions

e.g.:

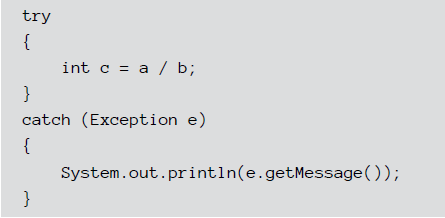


**Displaying the Exception Message**

* **Methods of the Exception Class (For Printing):**



Examples:



**Using a finally Block/Clause**

* Appears after all the catch blocks for a statement.
* always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

**Handling Checked Exceptions**

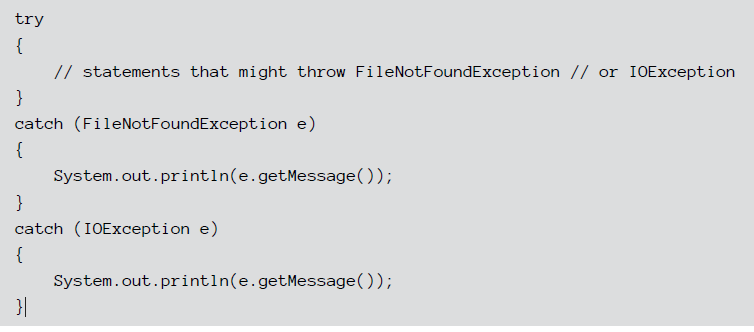
* **Viewing the catch-or-throw compiler error**
  + Gives an example of a compiler error message that occurs when a throw and a catch clause in not declared.
  + FileNotFoundException class is used as an example.
* **Catching FileNotFoundException**
  + One way to deal with the FileNotFoundException is to catch it by using an ordinary try statement.
* **Throwing the FileNotFoundException**
  + Avoiding an exception by passing the exception up to the method that calls the openFile method.
    - To do this you replace the try block with a throws block.
* **Throwing an exception from main**
  + Avoiding an exception by adding a throws clause in the main method.
  + Makes the program to terminate abruptly.
* **Swallowing exceptions**
  + If you don’t want to do anything with an exception, you can catch the exception in the catch block of a try statement but leaving the body of the catch block empty.

**Throwing Your Own Exceptions**

* writing methods that throw exceptions all on their own by using a throw statement.

**Catching Multiple Exceptions (Using a vertical bar in one catch block)**

**Before:**



**After:**

