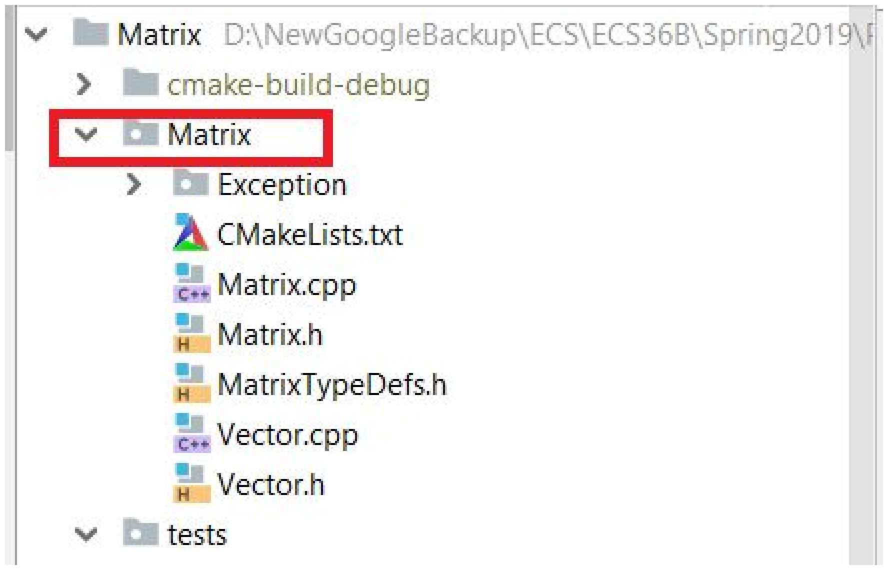
Adding Exceptions To Matrix

Matthew’s Stats

Time Taken: 1 hour
Files: 17
Lines of Code: 319

What to submit

- A zip file containing the Matrix **FOLDER**
    - Note I want a **FOLDER** this time and not just the files
    - This is the folder that contains your solution to the Matrix problem and the Exceptions, and not the one that contains the tests as well
    - Here’s a picture



```
Matrix  D:\NewGoogleBackup\ECS\ECS36B\Spring2019\
  cmake-build-debug
  Matrix
    Exception
    CMakeLists.txt
    Matrix.cpp
    Matrix.h
    MatrixTypeDefs.h
    Vector.cpp
    Vector.h
  tests
```

- Use the CMakeLists.txt that I’ve given you, updating them if you add or remove files from your solution but do **NOT** make your own CMakeLists as they won’t be compatible with the tester

Restrictions and Requirements

- You must implement all of the methods described in the .h files given in the assignment
    - You are free to **add** methods and members to the classes you are given or additional classes but you may not remove anything

- No global or static variables may be used

Description

In the previous version of our Matrix program, we assume that the parameters passed to our functions would always be good

- No one would try to go out of bounds on an element access
- No one would try to divide by 0
- No one would try to perform operations arithmetic operations on matrices/vectors that were of incompatible sizes.

Now we want to account for these possible errors and throw the appropriate exceptions if users of our classes make mistakes using them.

Exception Types

MatrixError

- Description: A generic error signifying that something went wrong in a Vector or Matrix operation
- Inherits From: std::exception
- Thrown by: Not directly thrown

DivideByZeroError

- Description: An error signifying that the user tried to divide a Vector or Matrix by a scalar with value 0
- Inherits From: MatrixError
- Thrown by: Thrown by Vector/scalar and Matrix/scalar when scalar is 0

OutOfBoundsError

- Description: A generic error signifying that the index used to access an element in a Vector or Matrix is out of bounds
- Inherits From: MatrixError
- Thrown by: Not directly thrown

VectorOutOfBoundsError

- Description: An error signifying that the index used to access a Vector is out of bounds

- Inherits From: OutOfBoundsError
- Thrown by: Vector.at or Vector.operator[] when the index is out of bounds

MatrixOutOfBoundsError

- Description: An error signifying that the index used to access a Matrix is out of bounds
- Inherits From: OutOfBoundsError
- Thrown by: Matrix.at or Matrix.operator[] when the index is out of bounds

DimensionMismatchError

- Description: A generic error signifying that the operation being performed on the Vectors or Matrices cannot be done as the Vectors/Matrices aren't the right size
- Inherits From: MatrixError
- Thrown by: Not directly thrown

VectorDimensionMismatchError

- Description: An error signifying that the operation being performed on 2 Vectors cannot be done because they are not the same size
- Inherits From: DimensionMismatchError
- Thrown by: Vector + Vector, Vector - Vector, and Vector * Vector when the Vectors are not the same size

MatrixDimensionMismatchError

- Description: An error signifying that the operation being performed on 2 Matrices cannot be done because they are not the same size
- Inherits From: DimensionMismatchError
- Thrown by: Matrix + Matrix and Matrix - Matrix when the Matrices are not the same size

MatrixInnerDimensionMismatchError

- Description: An error signifying that the operation being performed on 2 Matrices cannot be done because their inner dimensions do not match. The inner dimensions are the columns of A and the rows of B in the expression A OP B
- Inherits From: DimensionMismatchError
- Thrown by: Matrix * Matrix when the columns of A are not the same as the rows of B when doing A * B.

The what method

Since your exceptions inherit from std::exception your exceptions will need to implement the what method: `virtual const char* what() const noexcept;`

- This method will return a C string that reporting what the error is.
- The exact error message is not checked, just make it something that makes sense
- If you have a std::string, you can get a C string from it by using the std::string.c_str method
- You might find it easier to have one of the base classes implement the what method and then just have the children inherit it, as you probably will want to do the same thing in all of them

Testing

Testing will once again be through GoogleTest

Reimplementing the Matrix Class

You can copy and paste your solution to the previous Matrix problem into the Matrix folder so you don't have to retype everything.