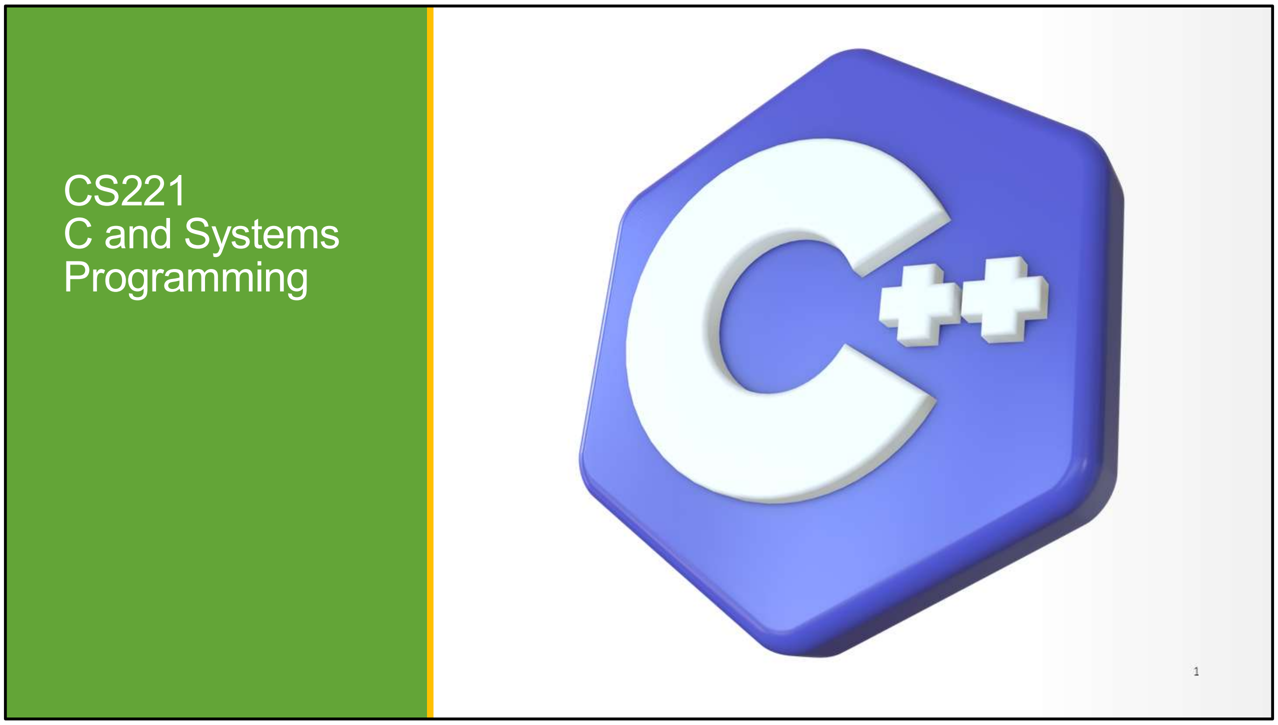
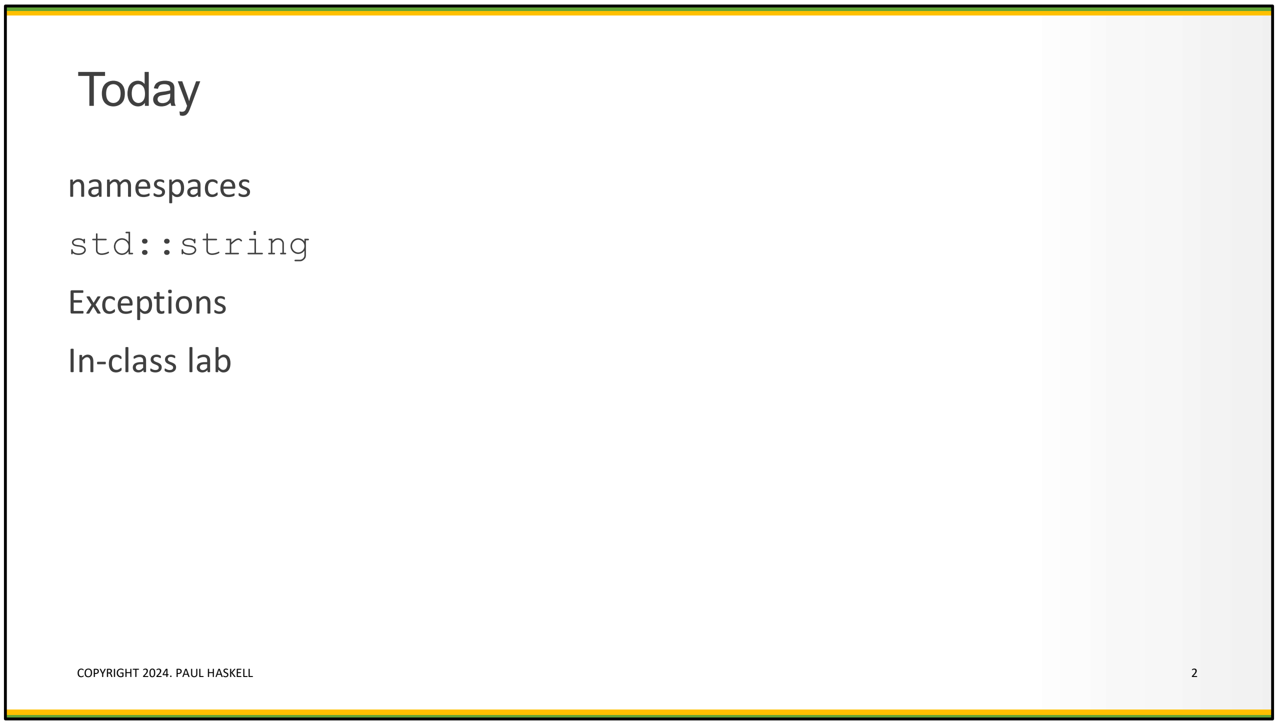


CS221
C and Systems Programming



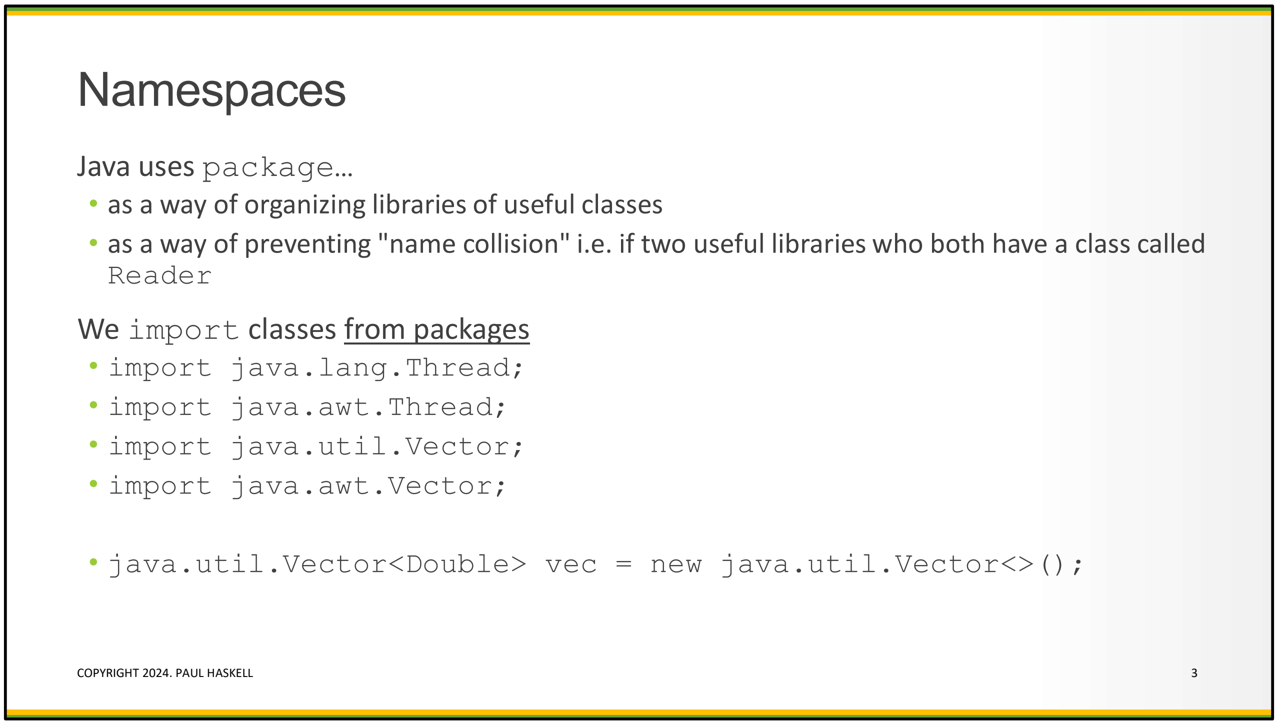
Today

namespaces

`std::string`

Exceptions

In-class lab



Namespaces

Java uses `package`…

- as a way of organizing libraries of useful classes
- as a way of preventing "name collision" i.e. if two useful libraries who both have a class called `Reader`

We `import` classes from packages

- `import java.lang.Thread;`
- `import java.awt.Thread;`
- `import java.util.Vector;`
- `import java.awt.Vector;`

- `java.util.Vector<Double> vec = new java.util.Vector<>();`

3

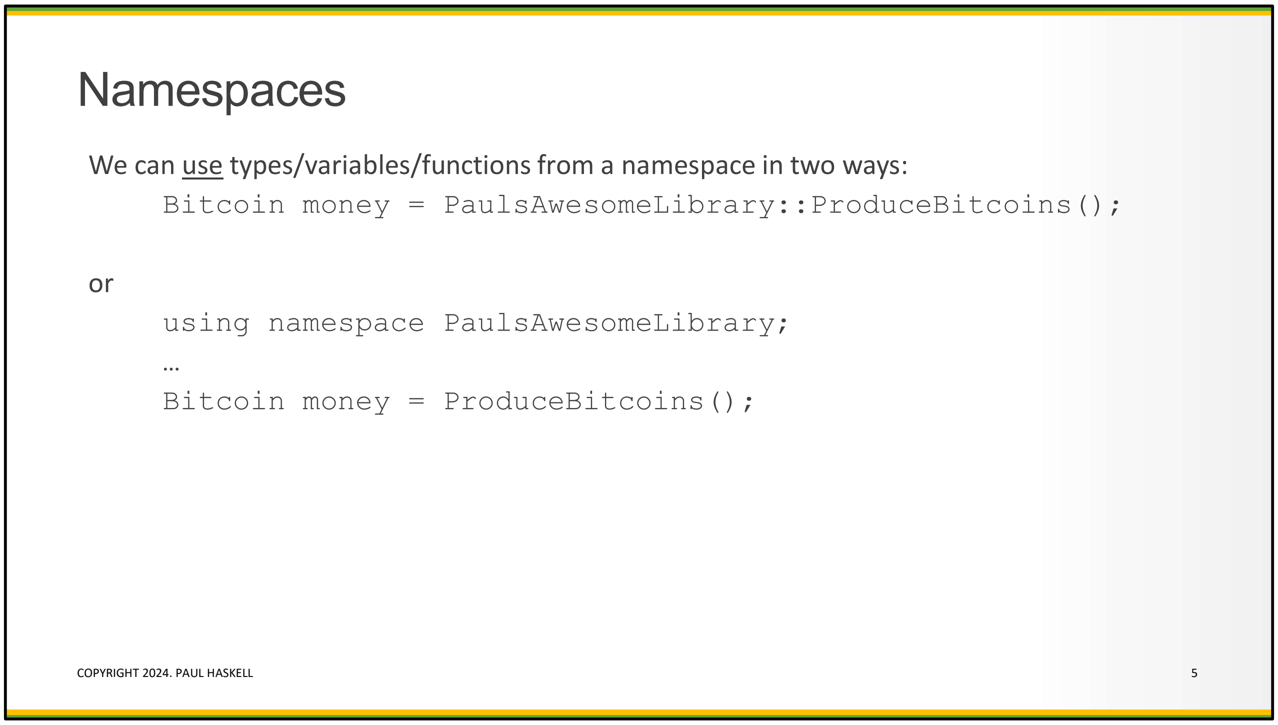
Uhh, look at that. Java lib does have multiple classes with the same names.

Namespaces

C++ uses `namespace` for the same purpose…

- By default, functions and data types are declared in the "global" namespace
- We can assign **types**, **variables**, and **functions** to a particular namespace like this:

```
namespace PaulsAwesomeLibrary {
  int myCoinCount;
  Bitcoin ProduceBitcoins() { … }
} // no semicolon!
```



Namespaces

We can use types/variables/functions from a namespace in two ways:

```
    Bitcoin money = PaulsAwesomeLibrary::ProduceBitcoins();
```

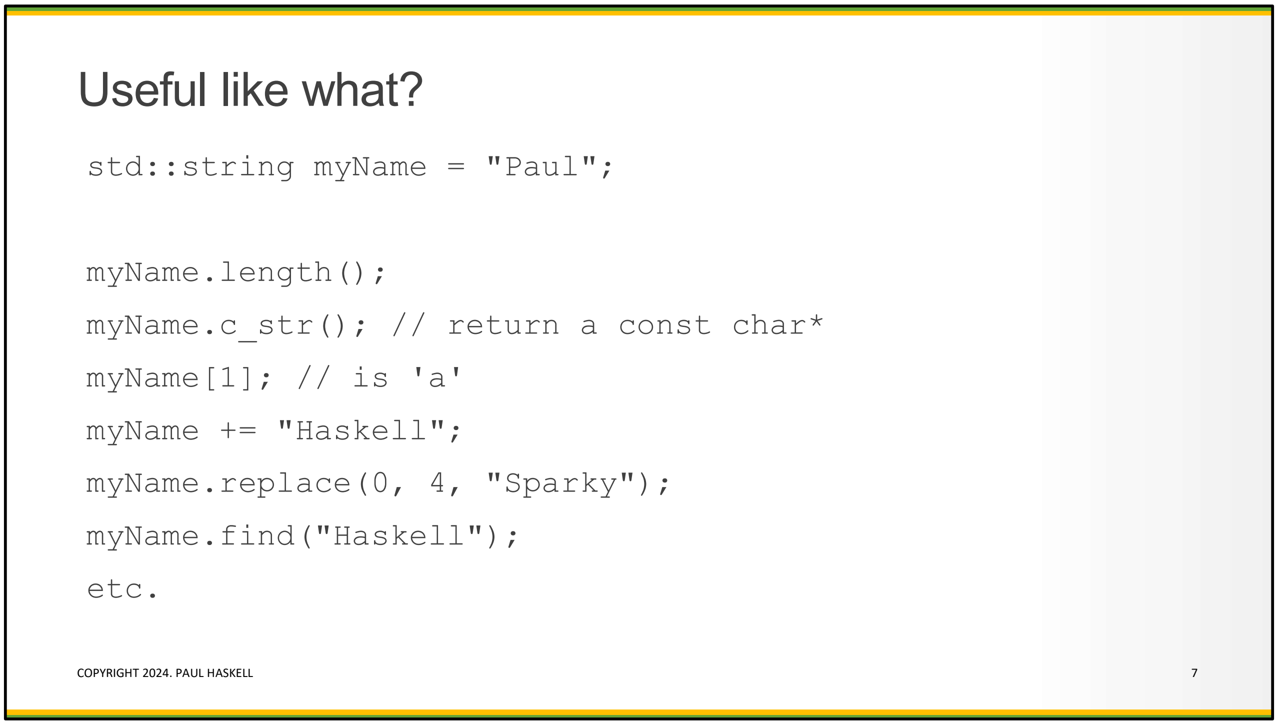
or

```
    using namespace PaulsAwesomeLibrary;
    …
    Bitcoin money = ProduceBitcoins();
```

Why are you telling us all this?

There is a namespace `std` with lots of useful types inside.

- `std` is not the same as the global namespace



Useful like what?

```
std::string myName = "Paul";

myName.length();
myName.c_str(); // return a const char*
myName[1]; // is 'a'
myName += "Haskell";
myName.replace(0, 4, "Sparky");
myName.find("Haskell");
etc.
```

Let's try a simple code-along: make a string with your first name. Append your middle and last names, if you have them. If not, make some up. Print out your name. Replace your middle name with "Sparky" and print out again.
SEE demo.cpp

What else is in `std`?

Complex numbers

Arrays

Queues, Lists, Hashmaps, Sets, Vectors

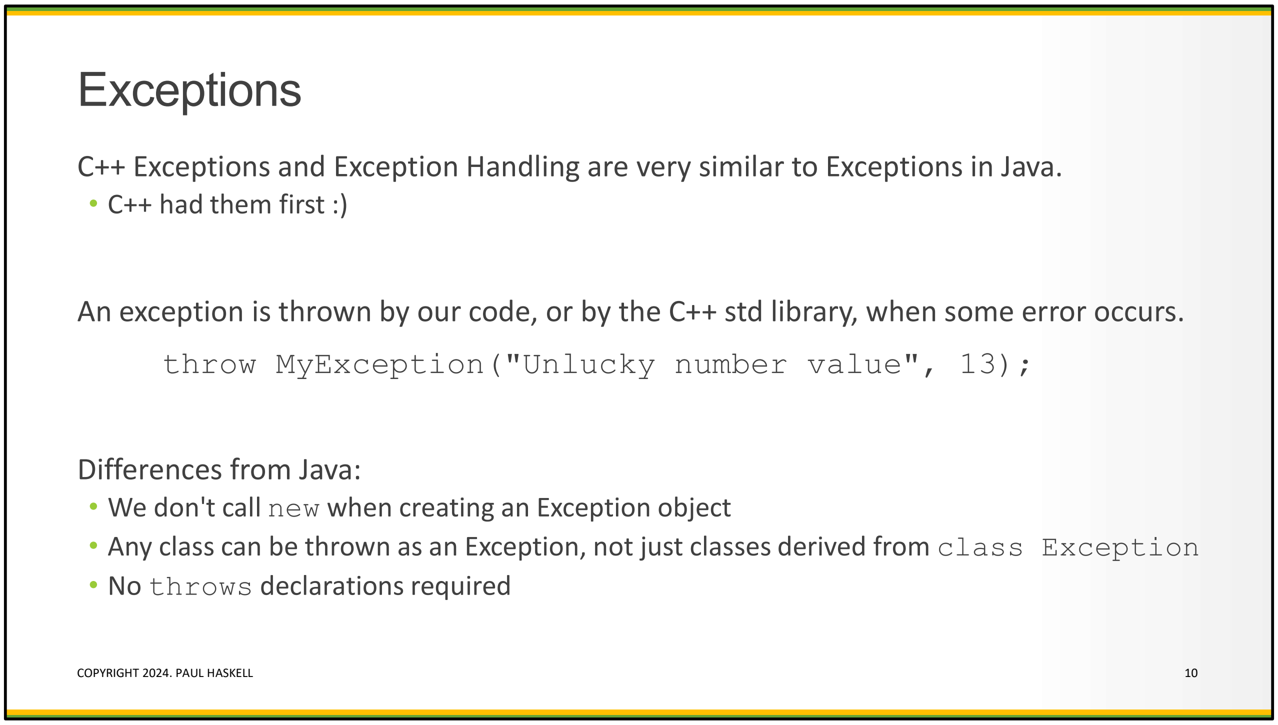
I/O streams

Threads, mutexes

Assertion checking

…and a lot more

One last C++ topic…



Exceptions

C++ Exceptions and Exception Handling are very similar to Exceptions in Java.

- C++ had them first :)

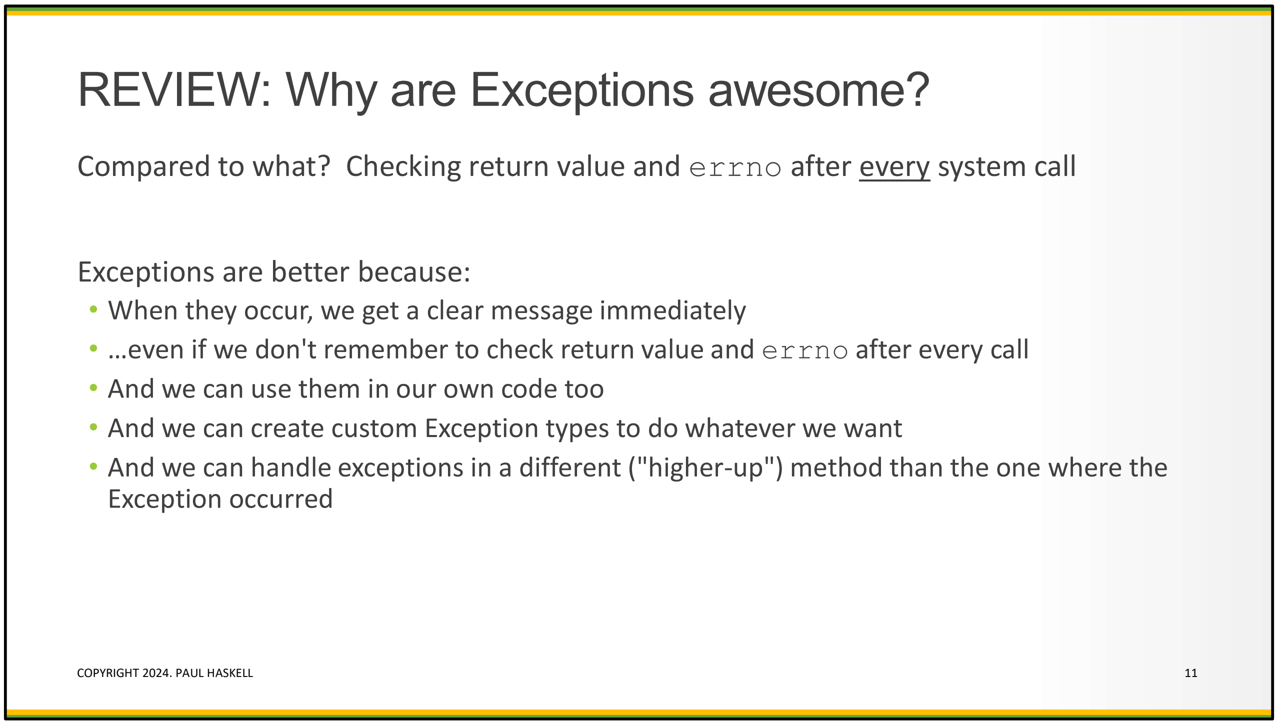
An exception is thrown by our code, or by the C++ std library, when some error occurs.

```
throw MyException("Unlucky number value", 13);
```

Differences from Java:

- We don't call `new` when creating an Exception object
- Any class can be thrown as an Exception, not just classes derived from `class Exception`
- No `throws` declarations required

Look at except.cpp

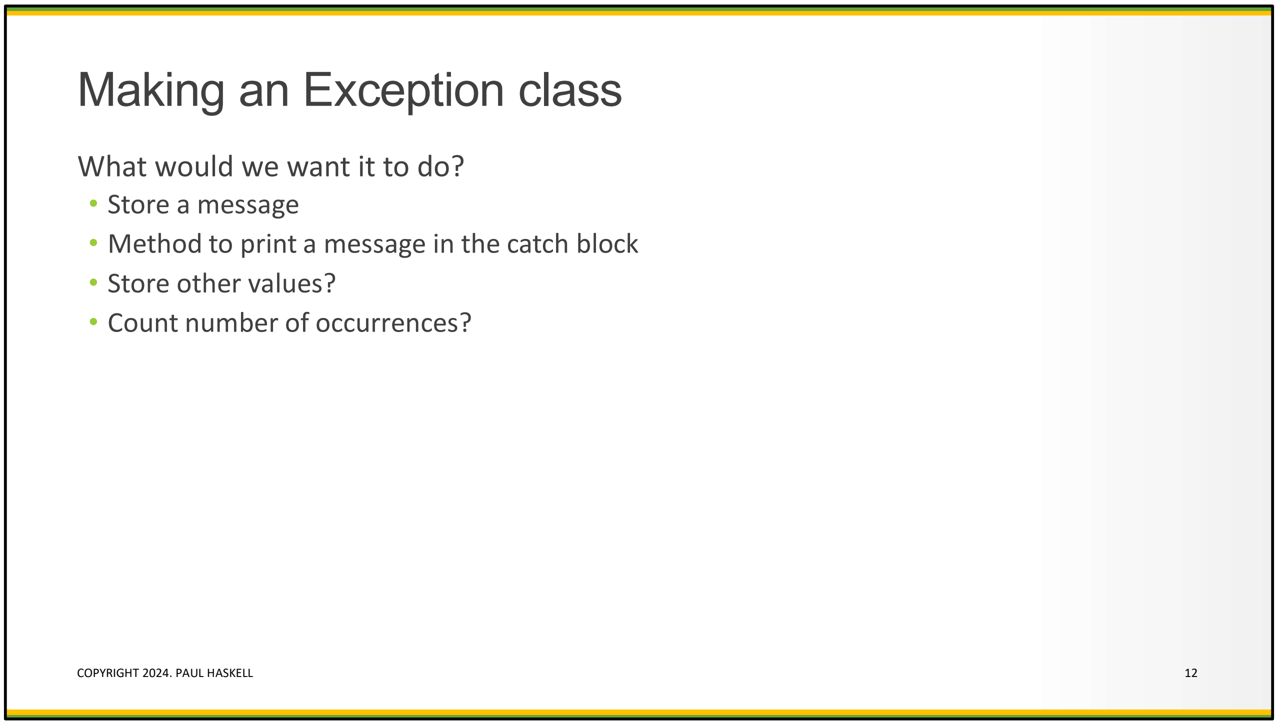

REVIEW: Why are Exceptions awesome?

Compared to what? Checking return value and `errno` after <u>every</u> system call

Exceptions are better because:

- When they occur, we get a clear message immediately
- …even if we don't remember to check return value and `errno` after every call
- And we can use them in our own code too
- And we can create custom Exception types to do whatever we want
- And we can handle exceptions in a different ("higher-up") method than the one where the Exception occurred

See except2.cpp



Making an Exception class

What would we want it to do?

- Store a message
- Method to print a message in the catch block
- Store other values?
- Count number of occurrences?

Code-along: Make MyException, that takes a string. Have a static variable that counts # of times exception was created. Make some code that throws the Exception and prints a msg that includes # of occurrences.

Exceptions vs Assertions

Assertions are used ONLY during development and testing

- If an assertion fails, source code must be changed to fix it.
- Assertions cause the program to exit

Exceptions are for handling errors that are reasonably expected to happen during run-time…in particular when the handling of the error is not in the same place as the detection of the error

- Missing file
- Incorrect user input
- Cannot connect socket
- etc

Final details

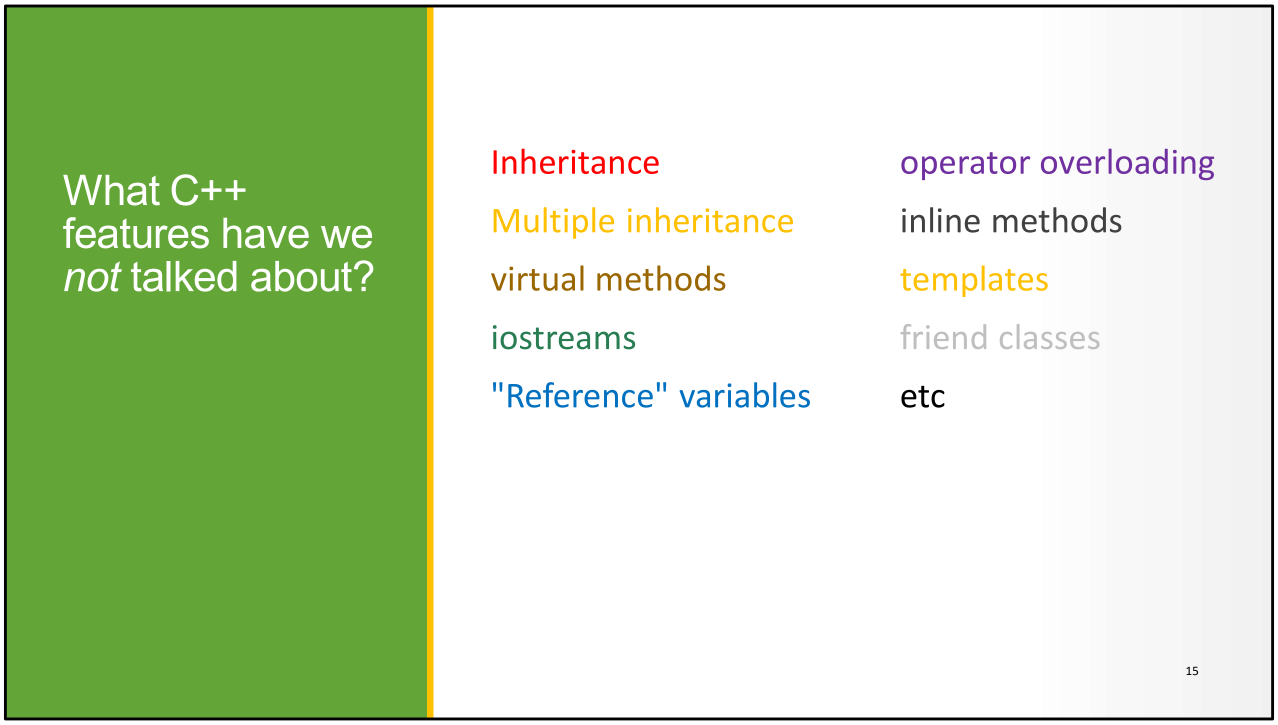
Does the C++ std library throw Exceptions?

- Yes!

Do the C system functions throw Exceptions?

- No :(

…and the `std` library defines many useful Exception classes



What C++ features have we *not* talked about?

- Inheritance
- Multiple inheritance
- virtual methods
- iostreams
- "Reference" variables
- operator overloading
- inline methods
- templates
- friend classes
- etc

15