# Lecture 34 — Abstract Classes

J. Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

September 6, 2016

Acknowledgments: W.D. Bishop

# The Shape of Things to Come

In an earlier example we considered the idea of a `Shape` class as a parent class.

It had several descendants, such as `circle`, `rectangle`, `triangle`...

And suppose that they should all have `double ComputeArea( )`.

But it makes no real sense to create an instance of a `Shape` object directly – it needs to be something.

The solution is an abstract class.

# Abstract Class

An abstract class can only be used as a base to derive other classes from.

You cannot create an instance of an abstract class, because it is "incomplete".

It is, however, a way to create a partial class definition that can be extended as is necessary.

To be abstract, the class must contain one pure virtual function.

# Pure Virtual Function

To declare a pure virtual function, we use the keyword `virtual` as well as an assignment to zero.

```
virtual double ComputeArea( ) = 0;
```

You may have multiple pure virtual functions in a class, and they may be of any type and take any parameters.

# Abstract Class

A class that extends an abstract class can go down one of two routes.

1. Either it implements all the pure virtual functions; or
2. It will itself also be an abstract class.

# Abstract Class Examples

Example: abstract class *A* has 5 pure virtual methods.

*B* extends *A* and implements all 5: *B* is not an abstract class.

*C* extends *A* and implements 4 of the 5: *C* is abstract.

*D* extends *C* and does not implement the "missing" abstract function: *D* is abstract.

*E* extends *D* and implements the last function: *E* is not abstract.

In theory, additional pure virtual functions could be introduced in, say, *D*, which *E* would also have to implement.

# Interfaces

Other programming languages like Java and C# have the concept of an interface.

An interface is like an abstract class that has no methods or fields implemented in it at all; all functions are virtual.

Thus, the same behaviour in C++ can be achieved with an abstract class with all pure virtual functions.