Abstract Classes

So Far

- We have covered polymorphism
  - What is it?
- And virtual functions
  - What are those?
- Today we will learn about
  - Abstract class
  - Pure virtual functions

Abstract Classes

- Consider a base class called GameObject that contains a draw function
- Avatar, Monster, and Castle are classes that are derived from GameObject, and each one has a unique draw function
- If some kind of array of GameObjects is maintained, a simple draw command can be sent to each object invoking the specific draw method for each object type
- This is where we are heading

Abstract Classes

- An abstract class is a class where the programmer never intends to instantiate an object of the abstract class type
- These classes are typically base classes and are used in an inheritance hierarchy to build more generic derived classes
- Parts of the abstract class are not implemented in the base class; therefore, this logic must be implemented in the derived class

Concrete Classes

- A concrete class is any class that can be instantiated
  - An object of that class can be created
- Consider an abstract class called Shape2D with concrete classes Circle, Square, and Triangle derived from Shape2D
- Shape2D has a draw method that is not implemented while Circle, Square, and Triangle must have implemented draw methods

Pure Virtual Functions

- A class is made abstract by having one or more pure virtual functions associated with the class as follows:
  - `virtual void functionName () const = 0;`
- Each derived class must provide its own draw function that overrides the draw function of the abstract class
- How is this different from virtual functions?

Pure Virtual Functions

- A virtual function
  - Allows the derived class the ability to override the function and
  - Must have an implementation
- A pure virtual function
  - Requires the derived class to override the function
  - Cannot have an implementation

Example

- Let us add an abstract class to the point, circle, cylinder hierarchy
- The abstract class will contain two pure virtual functions
  - print: to print the data for the shape
  - getName: returns a string containing the name of the shape (I.e. point, circle, or cylinder)

Example

- The abstract class will also contain two virtual functions:
  - getArea: returns the area of the shape
  - getVolume: returns the volume of the shape
- Why would these be defined as virtual functions and not pure virtual functions?

Shape Header

```
#ifndef SHAPE_H
#define SHAPE_H
#include <string>
using std::string;
class Shape {
public:
   virtual double getArea() const;
   virtual double getVolume() const;
   virtual string getName() const = 0;
   virtual void print() const = 0;
};
#endif
```

Shape Definition

```
#include <iostream>
using std::cout;
#include "shape.h"
double Shape::getArea() const
{
   return 0.0;
}
double Shape::getVolume() const
{
   return 0.0;
}
```

Summary

- We covered virtual functions
- We covered:
  - Pages 672 - 679