

#### So Far

4/9/07

4/9/07

- 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

CS250 Introduction to Computer Science II

- 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

CS250 Introduction to Computer Science II

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

CS250 Introduction to Computer Science II

### **Concrete Classes**

4/9/07

4/9/07

4/9/07

- 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

CS250 Introduction to Computer Science II

## Pure Virtual Functions

- A class is made abstract by having one or more pure virtual functions associated with the class as follows:
  - $\circ$  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?

CS250 Introduction to Computer Science II

# **Pure Virtual Functions**

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

## Example

4/9/07

• Let us create an abstract class called shape, and from this class inherit a point, circle, and cylinder class

CS250 Introduction to Computer Science II

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

CS250 Introduction to Computer Science II

## Example

4/9/07

4/9/07

- The abstract class will also contain two virtual functions:
  - o getArea: returns the area of the shape
  - o getVolume: returns the volume of the shape

CS250 Introduction to Computer Science II

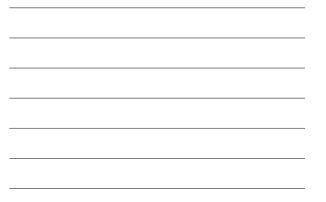
 Why would these be defined as virtual functions and not pure virtual functions?

### Shape Header

```
#ifndef SHAPE_H
#define SHAPE_H
#include <string>
using namespace std;
class Shape {
public:
    virtual double getArea() const;
    virtual double getVolume() const;
    virtual string getName() const = 0;
    virtual void print() const = 0;
};
#endif
4907 CS250 Introduction to Computer Science II
```

10

12



Shape Definition	
<pre>#include <iostream></iostream></pre>	
using namespace std;	
<pre>#include "shape.h"</pre>	
double Shape::getArea() const	
{	
return 0.0;	
}	
<pre>double Shape::getVolume() const</pre>	
{	
return 0.0;	
}	
4/9/07 CS250 Introduction	to Computer Science II 1

## Your Turn

4/9/07

- Using paired programming, I would like you to implement the shape, point, circle, cylinder hierarchy
- I have already implemented shape, and it is up to you to implement the other three classes
- Thoroughly test your classes in the main function
- Place the resulting program on Turing. Make sure you put both your names on it

CS250 Introduction to Computer Science II