



CS250 Intro to CS II

Spring 2014

Topics

- Virtual Functions
- Pure Virtual Functions
- Abstract Classes
- Concrete Classes
- Binding Time, Static Binding, Dynamic Binding
- Overriding vs Redefining

- Reading: pp. 907-928
- Problems: pp. 925-928 15.9-15.15 (all very good)

Abstract Class

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

Abstract Class

- 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

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 () = 0;`
- Each derived class must provide its own draw function that overrides the draw function of the abstract class

Abstract Class Example

```
class Shape
{
    public:
        Shape (int = 0, int = 0);
        void setX (int);
        void setY (int);
        int getX () const;
        int getY () const;
        virtual void draw () = 0;
        virtual double area () = 0;
    private:
        int mX;
        int mY;
};
```

Concrete Class

-
- A concrete class is any class that can be instantiated
 - An object of a concrete class can be created

Of Sprite, Avatar, Monster, and Castle, which are abstract and which are concrete? Why?

Concrete Class Example

```
class Circle : public Shape
{
    public:
        Circle (int = 0, int = 0, double = 0);
        void setRadius (double);
        double getRadius () const;
        virtual void draw ();
        virtual double area ();
    private:
        double mRadius;
};
```


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

Binding Time

- Binding time - the time at which something becomes known
- Static Binding - binding time that happens during compilation (e.g. a variable's type)
- Dynamic Binding - binding time that happens during runtime (e.g. the heap address of a dynamically allocated memory)

Redefining vs Overriding

- A derived class can “redefine” a base class member (static binding)
- A derived class that redefines a virtual function of a base class is said to “override” the base class function (dynamic binding)

Problem

1. Grab the solution PolymorphismExample from Turing.
2. In driver.cpp
 - a) Create an array of Shape pointers
 - b) Create a random number of Circle objects where each circle has a random (x,y) and a random radius between 25 and 50 inclusive.
 - c) Output the Circle objects on the screen
 - d) Free the dynamically allocated space
 - e) Implement Box.h/.cpp
 - f) Create a random number of Circle and Box objects placed randomly on the screen
 - g) Draw all objects
 - h) Free the dynamically allocated space

