CS 271 / 462
Programming Assignment # 11

Reading: Chapter 19

Purpose: Understand how to implement inheritance in C++
Understand the object-oriented terms inheritance (is-a), parent, child, subclass, superclass, base class, derived class, composition (has-a)

Grading: Documentation & Style 10 points

In addition to what is given in the Documentation and Style Guidelines document:

- All variables must be declared at the top of the function, before any executable statements.
- No global variables.
- All closing curly braces must have an "end" comment

LongDate Class (LongDate.h and LongDate.cpp)	30
PA 11.cpp (contains the main function)	5
makefile	5
Total possible	50 points

The following C language features are prohibited:

- printf
- scanf
- C library names
- \t in output statements
- \n in output statements

*** If you submit a program that doesn't compile, you will receive a grade of zero for this assignment. ***

Object Oriented Concepts

Inheritance : creating a new class that absorbs (inherits) an existing class's capabilities, then customizes or enhances them.

Base Class (also called parent class or superclass) **:** the existing class

Derived Class (also called the child class or subclass) : the new class

"is - a" relationship: phrase that indicates inheritance

Accessibility Modifiers

Private: Private members of a class are only accessible in the member functions of the class where they are defined. Private members of a base class are inherited, but not directly accessible in a derived class.

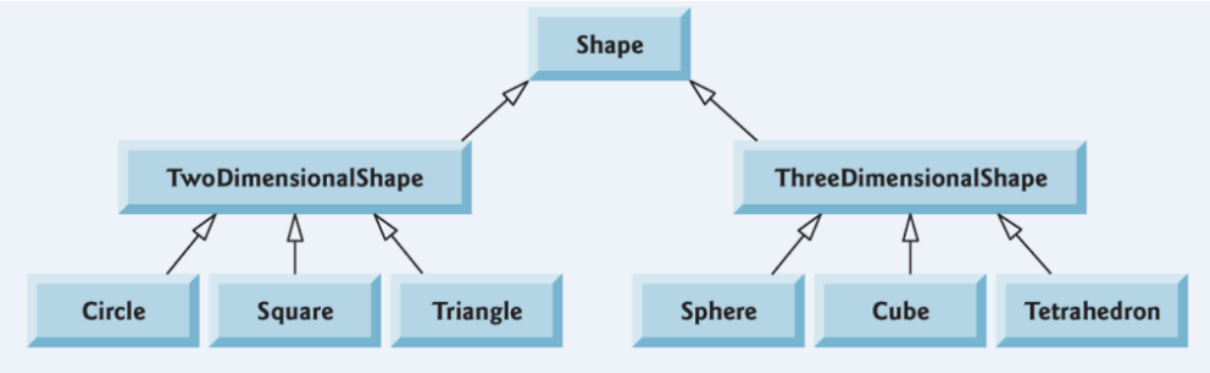
Public: Public members of a class are accessible in any function in any program.

Protected: Protected members of a base class can be access within member functions of the base class, by friends of the class, and by member functions and friends of any classes derived from that base class.

Friend: A class or a function can be declared as a friend of a class. Friends have access to the private and protected members of a class.

Friend privileges are given to a class or function. They cannot be taken.

Class Hierarchy or Class Relationship Diagram: A UML diagram showing the inheritance relationships among classes.



Types of Inheritance

C++ has three types of inheritance: public, protected, and private.

Public inheritance

To specify that a new class is to be publicly-derived from a base class:

```
class DerivedClassName : public BaseClassName {

   // derived class member definitions

   ...
};
```

Public inheritance is the only type of inheritance we will cover this semester.

Explicitly Calling a Base Class Constructor from a Derived Class Constructor

To specify that a constructor of a derived class is supposed to call the constructor of the base class:

```
    // in the constructor definition (not the prototype)

DerivedClassName::DerivedClassName ( parameter list )

  : BaseClassName( parameter names to be passed to the base class constructor )

{
     // body of derived class constructor function

}
```

Order of Constructor and Destructor Calls

When an object is created, the base class constructor is executed before the body of the derived class constructor executes.

When an object is destroyed, the derived class destructor is executed before the destructor of the base class is called.

Lab Instructions

A. Download Date.h and Date.cpp

- You will find that these files are very similar to those you wrote for PA 10.
- Compile Date.cpp to ensure that there are no syntax errors.

B. Create LongDate.h and LongDate.cpp

- Create a new class called LongDate that is publicly derived from class Date.
- Class LongDate should have one new private data member:

```
string dayOfTheWeek;
```

- You may want to add an array of strings to store the days of the week. If so, you must add it as a static const data member of the LongDate class. (See the "days" array in the Date.cpp file for an example of how a static const array is declared and initialized.)
- You may also want to add an array of strings for the names of the days.
- It should have a mutator for dayOfTheWeek.

```
void setDayOfTheWeek( );  // yes, there is no parameter
```

The mutator is not your typical mutator. It will set the dayOfTheWeek string according to the values that are stored in month, day, and year.

- It should have an accessor for dayOfTheWeek.

```
string getDayOfTheWeek( ) const;
```

- It should have one new constructor:

```
LongDate ( m : int, d : int, y : int );
```

The constructor should do the following:

1) Explicitly call the base class constructor and pass it the 3 parameters.
2) Determine and set (using the mutator) the string that is appropriate for the dayOfTheWeek. You can find algorithms for determining the day of the week for any given date on the Internet.

- It should override these "helper" functions (write new versions that replace the inherited ones) :

void helpIncrement( );
void setDate(int, int, int);

C. Create PA11.cpp

- Write a test program named PA11.cpp that will test the functions of the LongDate class.

D. Create a makefile

- Write a makefile to compile all programs and create an executable called PA11.

E. Zip all files and submit the .zip file

- Zip together the 6 files: Date.h, Date.cpp, LongDate.h, LongDate.cpp, PA11.cpp, and makefile. Submit the zip file.