Lecture 3

Wednesday, February 12, 2020 11:54 PM

- Good code
  - correctness
  - efficiency
  - clarity and readability
  - reusability and Maintainability
  - Extensibility
- Objected Oriented Programming
- Class
  - define a class
    - define/declare the class members (data and function)
    - implement member functions
  - protection
    - public: anyone can access a public member (data/function of a class)
    - private: only the members of the class can access a private member (data/function) of a class
    - protected

```
class Currency
{
private:
        string symbol_;
        double exchangeRate_;
public:
        Currency();
        Currency(string symbol, double rate);
        ~Currency();
        string GetSymbol();
        double GetExchangeRate();
        void SetExchangeRate(double rate);
};
```

  - Object
    - instance of class
  - constructor
    - name of the class
    - take parameter
    - also has a default without params
  - destructor
    - called when the obj is destroyed
    - free up resource
  - format
    - include guards
    - First letter of the class name is uppercase,
    - public member functions start with a upper case letter
    - private members use camelCase-->Member variable names end with (underscore), e.g. name_.
  - abstraction and encapsulation
    - Encapsulation refers to combining data and functions inside a class so that

data is only accessed through the functions of the class.
        - Data abstraction refers to the separation of interface (public functions of the class) and implementation)
    - copy constructor
        - Currency(const Currency& other); // head file
        - Currency::Currency(const Currency& other)
        - : symbol_(other.symbol_), exchangeRate_(other.exchangeRate_)
          { }
    - assignment operator
        - operator=(const Currency& other); // h file

```
Currency& Currency::operator=(const Currency& other)
{
        if (this!= &other)  // here we are using &other instead of other
        because
                                // we are detecting the address of the obj
        {
                symbol_ = other.symbol_;
                exchangeRate_ = other.exchangeRate_;
        }
        return *this;
}
```

        - this can be used for c2=c1 //both are currency obj
    - this keyword
        - this pointer is initialized with the object's own address.
    - other operator overloading
        - addition +
        - refer to Baruch cpp course for implementation
    - static member
        - We use static keyword to associate a member with the class.
        - A static data member cannot be accessed directly using a non-static member function
        - Static member variables must be initialized once before we use it (outside the class): int Counter::count_ = 0;
        - A static member (data/function) does not belog to an object -->We do not need an object of a class to use a static member.

```
class Counter
{
public:
        static int GetCount();
        static void Increment();
private:
        static int count_;
};
```

        - so if we have Counter c1,c2 and c1.Increment(), c2.GetCount() will show that the count_ variable has been increased by 1

- Struct
    - struct: Members have public protection level by default.

    - I class: Members have private protection level by default.
- include guards
    - #pragma once
    - a program can read an include file only once.
    - Currency.h
      #ifndef CURRENCY_H
      #define CURRENCY_H
      class Currency
      { … };
      #endif