Classes Part 1: Encapsulation, How to Code a Class, UML

Homework 2 solution

- I saw a lot of people do a “C style” approach
- The “C style” approach is much harder than using C++ features
- Sample solution is actually in C++

Now we start Object Oriented Programming

However….

- We are still learning how to program in C++

- So now the class is not only about OO Concepts, but also about how to apply this in C++

- Java, C#, and other OO languages will have applications for almost everything we cover. Just slightly different syntax

Lets start getting in Object Oriented (OO)

- 1310 – Teaching you how to program
  - Get you in the mindset of a programmer
- 1320 – Learn more advanced programming
  - What goes on behind the scenes (memory, pointers)
- 1325 – Learn a different type of programming
  - Objects, Inheritance, Encapsulation
- 1310 and 1320 focused on Procedural Programming
- 1325 is focuses on OO Programming

3 main concepts to OO Programming

- Encapsulation
- Inheritance
- Polymorphism
- Think “PIE”

Basic Vocabulary

- Encapsulation
  - Dictionary definition – the action of enclosing something in or as if in a capsule
  - Book definition – protecting something meant to be private (such as implementation details) from unauthorized access
  - Other definition – Bundling functions, variables, data, and code into a restricted container (such as a class)
  - Allows us to make things in our program private to where other programs can not change them

Basic Vocabulary

- Class
  - Book Definition – A user defined type that may contain data members, function members, and member types
  - Other Definition – a template encapsulating data and code that manipulates it

Basic Vocabulary

- Method – a function that manipulates data in a class
- Instance – an encapsulated bundle of code
- Object – instance of a class containing a set of encapsulated data and associated methods
- Variable – block of memory associated with a symbolic name that contains an object or a primitive data value
- Operator – A symbol that modifies an object, or generates a new object from other objects

What is a Class (generally)

- Class consists of data and code (optional)
  - Data is information managed (variables)
  - Code (methods) manipulates the data
  - Methods can be written either inside the class or outside the class

- Basic types of Classes
  - Enumerations (Enum)
  - Structure (Struct)
  - Class

Structures

- Primarily used for data structures where members can take any value
- Basically a set of type and variable name declarations

Structures

- struct Date{
  int y;
  int m;
  int d;
  };
- Date today;
- today.y = 2017;
- today.m = 8;
- today.d = 28;

today.y = 28;

today.m = 2017;

today.d = 8;

Structures – Constructors and Member functions

- Can have constructors that check for proper input
- Constructor – an operation that initializes (“constructs”) the object
- Constructors have the same name as the object
- Member functions – functions declared as memebers of the class within the class body

Structures – Constructors and Member Functions

- struct Date{
  int x, y, z;
  Date (int y, int m, int d)
  {//Code Here}
  void add_day(int n)
  {//Code Here}
  };

- Date birthday; //error
- Date today {8, 28,2017}; //run time error
- Date tomorrow {2018, 1, 30}; //OK
- Date yesterday = {2018, 1, 28}; //OK
- Date today = Date{2018, 1, 29}; //OK

Structures – Constructors and Member Functions

- struct Date{
  int x, y, z;
  Date (int y, int m, int d)
  {//Code Here}
  void add_day(int n)
  {//Code Here}
  };

- today.add_day (1); //OK
- Add_day(7); //error

Why do we need Classes?

- Everything in a struct is publicly accessible.
- Variables can be accessed and changed at will
- Functions can be called at will
- Even by other programs sometimes

Issues with Sturcts

- Date today {2017, 8, 28};
- So far so good
- today.m = 26;
- Invalid month. Will still run

How to program a class

- Coding time
- How to create a class
- How to create a constructor – the method that creates our object
  - Object = instance of a class
- Other Methods
- Method Overloading
  - 2 methods – same name, different implantation
- Public vs Private
- Getter or Accessor Methods
  - Get our private variables
- Setting or Mutator Methods
  - Change our private variables (within reason)
- toString
- Operator Override

Public vs Private

- Public is the default
- Public means anyone can access and change our variable
- Private means outside people can’t change it
- But one problem, how do we change it then?

UML

- UML stands for Unified Modeling Languages
- Used to describe, specify, design, and document the structure and behavior of software systems, particularly in OO
- Can specify a system so that code can be generated
- Can specify a system to enhance team communication
- Can visually represent a system to enhance understanding

UML Class Diagrams

UML Class Diagram Elements

- + sign = public field
- - sign = private field
- Variables = variable name : type
- Methods = method name (variables) : return type
- How to use Umbrello