1. Lecture 6 – C++ Classes – Constructor(s) (Pt.1)
   1. Today’s Topic
      1. C++ Classes Cheatsheet
         1. ⮚ Declaration   
            ⮚ Members, Methods, Interface  
            ⮚ Implementation – Resolution Operator ( **::** )  
            ⮚ Instantiation – Objects  
            ⮚ Object Usage – Dot Operator ( **.** )  
            ⮚ Object Pointer Usage – Arrow Operator ( **->** )  
            ⮚ Classes as Function Parameters, Pass-by-Value, by-(**const**)-Reference, by-Address ⮚ Protection Mechanisms – **const** Method signature  
            ⮚ Classes – Code File Structure
      2. Constructor(s)
      3. Destructor
   2. Classes
      1. Class Cheatsheet
         1. Protection Mechanisms – Access Specifiers:
            1. **public**
         2. Anything that has access to a ***Car*** Object (scope-wise) also has access to all **public** Member Variables and Functions.
            1. ⮚ “Normally” used for Functions.
            2. ⮚ Need to have at least one **public** Member.
         3. **Private** Members (Variables and Functions) that can ONLY be accessed by Member Functions of the ***Car*** Class.
            1. ⮚ Cannot be accessed in **main()**, in other files, or by other functions.
            2. ⮚ If not specified, Members default to **private**.
            3. ⮚ Should specify anyway – good coding practices!
         4. **Protected** Members that can be accessed by:
            1. ⮚ Member Functions of the ***Car*** Class.
            2. ⮚ Member Functions of any *Derived* Class.
         5. Member Functions – Accessors (“Getters”) Name starts with **get**, ends with Member name.
         6. Allows retrieval of non-**public** Data Members.
         7. Member Functions – Mutators (“Setters”)
         8. Name starts with **set**, ends with Member name.
         9. *Controlled* changing of non-**public** Data Members.
   3. Constructor(s)
      1. Constructor – Description
         1. Special Class Methods that “build” an Object.
            1. ⮚  Object Initialization.
            2. ⮚  Supply *specific* default values (If necessary).
         2. Remember:  
            *Implicit* initialization (*Default* Constructor
            1. ⮚ Automatically called when an object is created.

Implicit: ***Date myDate*;**Explicit: ***Date myDate*(1,1,1917);**

* + 1. Description
       1. Called when a Class is *Instantiated*.
          1. ⮚ C++ won't automatically initialize Member Variables.
       2. Default Constructor:
          1. ⮚ Basic no-argument constructor, can have one or none in a Class.
          2. ⮚ If Class has no Constructors, the C++ Compiler will *make* a *Default*.
       3. Overloaded Constructors:
          1. ⮚  Constructors that take in arguments, can have none or many in a Class.
          2. ⮚  Appropriate version called based on number and type of arguments passed when an Object is created (*Instantiated*).
    2. Overloading
       1. Can have multiple versions of the Constructor: ⮚ *Overloading* the Constructor.
       2. Different constructors for when:
          1. ⮚ All Member values are known.
          2. ⮚ No Member values are known.
          3. ⮚ Some subset of Member values are known.
    3. Default Parameters
       1. Not really meaningful to have numerous Constructors just to set default Member values.
          1. ⮚ A lot of code duplication.
          2. ⮚ Can set Default Parameter values in Constructor.
    4. Deconstructor – Description
       1. Called when a Class goes out-of-scope or is freed from the heap (by **delete**).
       2. Destructor
          1. ⮚Hasthename**~ClassName()**, hasNO**return**type.
          2. ⮚ Can only specify a *single* one *user-provided* in a Class, or none and the compiler will provided an *automatically synthesized* one.
       3. Destructor will automatically (without writing any code in it) call Destructor of any
       4. Data Member *Objects*.
          1. ⮚ But not Pointed-to Objects by Member Pointers !
          2. ⮚ Define a Destructor if you need to return resources, deallocate pointer memory, etc.
  1. Classes (Pt 2)
     1. Ask yourself:
        1. ⮚  What properties must each Object have, what data types should each be?
        2. ⮚  Which should be **private**? Which should be **public**?
        3. ⮚  What operations must each Object have?
        4. ⮚  What Accessors, Mutators, Facilitators?
        5. ⮚  What parameters must each of these have?
        6. ⮚  **const**, by-Value, by-Reference, Default?
        7. ⮚  What **return** value should each of these have?   
           Beginner’s rules of thumb:
        8. ⮚  Data usually **private**, operations usually **public**.
        9. ⮚  Usually 1 Mutator & 1 Accessor per Data Member.