Documenting classes

Documenting Classes

Recall
- Classes are defined in the header file
- Classes are defined by
  - Attributes (equivalent to variables)
  - Methods (equivalent to functions)
  - Interfaces
- The class definition provides an *interface* to the class
  - It describes…
    - What methods are available
    - What the methods do
    - What types of values they process and
    - What they produce

Components of a Class

Attributes

- They are equivalent as variables
- So, we will document attributes as variables using a Data Table
  - For each attribute we will:
    - State the use of the attribute
    - How its value is obtained/used

Methods

- They are equivalent as functions
- So, we will document methods similarly as functions

Documenting Class Definition

Attributes

- They will be document with a Data Table

Methods

- They will be grouped as:
  - Constructor and Destructor
  - Mutators
  - Accessors
- They will be documented as functions prototypes but listed right after the class definition

The Sheep Class (documented)

```
class Sheep
{
   public:
      /******************************
       ** CONSTRUCTOR & DESTRUCTOR **
       ******************************/

     Sheep ();                 // constructor
     ~Sheep ();                // destructor

     /***************
     /** MUTATORS  **
      ***************/

     void  SetAge (int age);
     void  SetName(string name);
     void  ChangePosition(int xCoord,
                          int yCoord);

     /***************
      ** ACCESSORS **
      ***************/

     int   GetAge       () const;
     void  PrintNeatly  () const;
     float DistanceFrom (int xCoord,
                         int yCoord) const;
     bool  GetPosition  (int xCoord,
                         int yCoord) const;
   private:
     string name;                               // IN/OUT - the sheep's name
     int    age;                                  // IN/OUT - the sheep's age in years
     int    x, y;                                 // IN/OUT - the sheep's position in the field

};
```

The Sheep Class (documented)

Below Class Definition (in the same .h file)

```
/*****************************************************************
 * Sheep Class
 *   This class represents a sheep object. It manages 4 attributes,
 *   name, age and position (x and y coordinate)
 *****************************************************************/


/******************************
 ** CONSTRUCTOR & DESTRUCTOR **
 ******************************/

/*****************************************************************
 * Sheep ();
 *   Constructor; Initialize class attributes
 *   Parameters: none
 *   Return: none
 *****************************************************************/

/*****************************************************************
 * ~Sheep ();
 *   Destructor; It does not perform any specific function
 *   Parameters: none
 *   Return: none
 *****************************************************************/
```

```
/*************
 ** MUTATORS **
 *************/

/*********************************************************************
 * void SetAge (int age);
 *
 *   Mutator; This method will update the age attribute to the
 *        parameter age value
 *-------------------------------------------------------------------
 *   Parameter: age (int) // IN – the age for the new attribute
 *-------------------------------------------------------------------
 *   Return: none
 *********************************************************************/

…
/**************
 ** ACCESSORS **
 **************/
/*****************************************************************
 * int GetAge () const;
 *
 *   Accessor; This method will return the age attribute
 * ---------------------------------------------------------------
 *   Parameters: none
 * ---------------------------------------------------------------
 *   Return: age (int)
 *****************************************************************/
```

Documenting Method Definition

Methods are defined in a .cpp file and should be documented similarly as a function above the definition

Documentation should include:

- Method name
- Class that method belongs to
- Description
- Pre-conditions including input parameters
- Post-conditions including return value and any other values changed (by reference)

Sample Method Definition Documentation
Methods should be presented in the same order as the interface

```
/**********************************************************
 *
 * Method ChangePosition: Class Sheep
 *_________________________________________________________
 * This method receives a x and y coordinate representing
 *   a new position for a sheep object and update the sheep's
 *   position – returns nothing.
 *_________________________________________________________
 * PRE-CONDITIONS
 *   The following need previously defined values:
 *      xCoord: New sheep's x coordinate
 *      yCoord: New sheep's y coordinate
 *
 * POST-CONDITIONS
 *     This function will update x and y coordinate attribute
 *     in the sheep object. There is no return value.
 *     <Post-conditions are how the program execution is
 *      affected by this method -  Did it output something
 *      is it modifying reference parameters – does it return
 *      something? – state that here  >
**********************************************************/
 void Sheep::ChangePosition(int xCoord, // IN – New sheep's x coordinate
                            int yCoord} // IN – New sheep's y coordinate
{
```