Class Design – Draft Version  
  
Card  
------  
  
The card class is responsible for drawing the card, updating it, and positioning it.  
It will use a bool member variable to decide whether or not it is alive, (e.g. no longer alive if the matching card has been found so no need to perform any more actions on it).  
  
Member Variables  
--------------------------  
  
private bool \_MAlive // if true the card is alive, else false  
private Vector<int> \_MPosition // Vector that holds the x,y,z of the card  
private string \_MType // the type of card, used for finding a correct pair, e.g. “UltraViolet”  
private texture \_MTexture // Relevant texture info for the card  
  
Methods  
-------------  
  
public Card(bool inAlive, string inType, Vector<int> inPosition) // Constructor  
private bool IsAlive() // Returns \_MAlive  
private string SetAliveStatus(bool inAlive) // Sets the alive status, if successful returns an empty string, else returns an error message  
private void Draw() // Draws the card  
private void Update() // Updates the cards, N.B. no deltaTime is used, the Engine class will handle that  
private bool IsTypeValid() // If the type, e.g. “UltraViolet”, is valid true is returned, else false  
private Vector GetPosition() // Gets the position of the card – useful for collision (e.g. mouse-click)  
  
Player  
---------  
  
The player class will handle the aspects of the player, such as their name and current score.  
  
Member Variables  
--------------------------  
  
private string \_MName // The name of the player  
private int \_MScore // The player’s current score  
  
Methods  
------------  
  
public Player(string inName) // Constructor, set score to 0  
private string ValidateName() // Validates the player’s name, if it’s valid return an empty string, else return an error message  
private string GetName() // Returns the name of the player  
private string SetName(string inName) // Sets the player’s name, calls GetName(), returns an empty string if successful, else returns an error message  
  
ScoringSystem  
--------------------  
  
The ScoringSystem class is in charge of loading the scores, writing the scores, and saving the scores.  
  
Member Variables  
-------------------------  
  
private Hashtable \_mScores // A hashtable of scores and the corresponding player’s name  
  
Methods  
--------------  
  
public ScoringSystem() // constructor  
  
private string LoadScoresFromFile() // Loads the scores from a file into a hashtable that contains the player’s name and the corresponding score, returns an empty string if successful, else returns an error message  
  
private string WriteScoresToFile() // Writes the scores to a file, returns an empty string if successful, else returns an error message  
  
private void SortScores() // Sorts the scores into descending order  
  
Timer  
--------  
Used from previous projects. Main purposed - to calculate deltaTime (for updating the scene per frame) and to calculate how much time the player has left.  
  
UserInterface  
--------------------  
Responsible for drawing the playing area, and displaying scores, player name, highscores table, and the game menu  
  
Member Variables  
------------------------  
  
private Vector<int> \_MTableDimensions // the height, width, and depth of the table  
private Vector<int> \_MTablePosition // the x,y,z of the table  
  
Methods  
------------  
  
public UserInterface() // Default constructor, provide reasonable member variable values  
  
public UserInterface(Vector<int> inTableDimensions, Vector<int> inTablePosition) // Constructor, supplies values for the member variables  
  
private void Draw() // Draws the correct ui for that action, e.g. table for playing, menu for when menu is selected.  
  
Engine  
---------  
This is the class that provides the rules and workings for the game  
  
Member Variables  
-------------------------  
  
private enum Difficulty // enumeration of difficulty  
private Vector<Card> \_MCard // A vector of instances of the Card class  
private Player \_MPlayer // instance of the Player class  
private ScoringSystem \_MScoring // instance of the ScoringSystem class  
private Timer \_MTimer // instance of the Timer class  
private UserInterface \_MUserInterface // instance of the UserInterface class  
  
Methods  
------------  
private void Draw() // Draws the game scene  
private void Update(float deltaTime) // Updates the game scene, calls CalculateScores  
private void CalculateScores(Difficulty difficulty) // Calculates the score based on the Difficulty enum  
  
Main  
-------  
  
Main instantiates the Engine class