```cpp
//*****************************************************************************
// File name:  Model.h
// Author:     Chadd Williams
// Date:       5/24/2017
// Class:      CS485
// Assignment: Memory Game
// Purpose:    Practice with MVP
//*****************************************************************************

#pragma once
#include "Board.h"
#include "Player.h"
#include <vector>

class Model
{
public:

  Model (int seed = 1);
  ~Model ();

  bool isBoardDone () const;
  bool isWinner () const;
  std::string getCurrentPlayerName () const;
  int getCurrentPlayerScore () const;

  void setPlayerName (std::size_t index, std::string name);

  //bool makeMove (int x1, int y1, int x2, int y2);

  bool flip (int x, int y);

  bool legalMove (int x, int y) const;

  void resetGame ();

  void advanceTurn ();

  std::string getCard (int x, int y) const;


private:

  Board *mpcBoard;
  std::vector<Player> mcVecPlayers;
  int mCurrentTurn = 0;
  static const int NOT_FLIPPED = -1;
  int mLastX1 = NOT_FLIPPED, mLastY1 = NOT_FLIPPED;
  int mLastX2= NOT_FLIPPED, mLastY2 = NOT_FLIPPED;
};
```

```
1  //*************************************************************************
2  // File name:  Board.h
3  // Author:     Chadd Williams
4  // Date:       5/24/2017
5  // Class:      CS485
6  // Assignment: Memory Game
7  // Purpose:    Practice with MVP
8  //*************************************************************************
9
10 #pragma once
11 #include "Card.h"
12
13 class Board
14 {
15 public:
16   static const int BOARD_SIZE = 4;
17
18   Board (int seed=1, bool bEasy = false);
19
20   bool flip (int x, int y);
21   void putFaceDown (int x, int y);
22
23   bool match (int x1, int y1, int x2, int y2);
24
25   void replace (int x, int y);
26
27   bool isBoardDone () const;
28
29   std::string getCard (int x, int y) const;
30
31   void setFixed (int x, int y);
32   bool legalFlip (int x, int y) const;
33
34   void reset ();
35 private:
36
37   void setUpBoard (int seed=1, bool bEasy=false);
38
39
40   Card macCards[BOARD_SIZE][BOARD_SIZE];
41   bool maFixed[BOARD_SIZE][BOARD_SIZE];
42   static const int USED_CARD = 9;
43
44 };
```

```
1  //*************************************************************************
2  // File name:  Card.h
3  // Author:     Chadd Williams
4  // Date:       5/24/2017
5  // Class:      CS485
6  // Assignment: Memory Game
7  // Purpose:    Practice with MVP
8  //*************************************************************************
9
10 #pragma once
11 #include <string>
12
13 class Card
14 {
15
16 public:
17
18
19   Card (int value = 0);
20
21   int getValue () const;
22
23   std::string to_string () const;
24
25   void flip ();
26   void putFaceDown ();
27
28   bool isFaceUp () const;
29   bool operator==(const Card &rcCard) const;
30
31 private:
32   int mFaceValue;
33   bool mbFaceUp = false;
34 };
```

```
1  //*************************************************************************
2  // File name:  Player.h
3  // Author:     Chadd Williams
4  // Date:       5/24/2017
5  // Class:      CS485
6  // Assignment: Memory Game
7  // Purpose:    Practice with MVP
8  //*************************************************************************
9
10 #pragma once
11 #include <string>
12 #include "Observable.h"
13
14 class Player
15 {
16 public:
17
18   Player () = default;
19   Player (std::string cName);
20
21   void setName (std::string cName);
22   std::string getName () const;
23
24   int getScore () const;
25   void incrementScore ();
26   void resetScore ();
27
28 private:
29
30   std::string mcName;
31   int mScore = 0;
32 };
```

```
1  //************************************************************************
2  // File name:  IObserver.h
3  // Author:     Chadd Williams
4  // Date:       4/20/2017
5  // Class:      CS485
6  // Assignment: Observer Example
7  // Purpose:    Demonstrate how the observer pattern works
8  //************************************************************************
9
10 #pragma once
11 #include <string>
12
13 class IObserver
14 {
15 public:
16
17   virtual void notify (const std::string &rcData) = 0;
18
19 };
```

```cpp
//****************************************************************************
// File name:  Observable.h
// Author:     Chadd Williams
// Date:       4/20/2017
// Class:      CS485
// Assignment: Observer Example
// Purpose:    Demonstrate how the observer pattern works
//****************************************************************************

#pragma once
#include "IObserver.h"
#include <vector>
#include <string>

class Observable
{
public:

  Observable () = default;
  ~Observable () = default;

  virtual void addObserver (IObserver *pcObs);
  virtual void removeObserver (IObserver *pcObs);

  virtual void notifyAll (const std::string &rcData);


private:
  std::vector<IObserver *> mcObservers;
};
```