Model-View-Presenter

https://realm.io/news/eric-maxwell-mvc-mvp-and-mvvm-on-android/

https://martinfowler.com/eaaDev/uiArchs.html

https://github.com/ericmaxwell2003/ticTacToe

http://aspiringcraftsman.com/2007/08/25/interactive-application-architecture/

https://github.com/googlesamples/android-architecture/tree/todo-mvp

http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

Original UI Applications

- Forms and Controls
  - button, text box, ….
- Each control has and onClick()/onChange()

- Business logic and state is in the main UI and scattered across the various onClick() methods
- Hard to reuse code
- Hard to move UIs
- Hard to automate testing

Model View Control

- GUI architecture pattern
- made up of many Design Patterns
- Goals
  - separate underlying model from UI
  - reuse of model for different UIs
  - provide an easily tested interface
- Many slightly different definitions!

MVC

- Model
  - Data, State, Business logic
  - can interact directly with View when a state change occurs
    - Observer Pattern
- View
  - Visual representation of Model (UI)
  - can interact directly with the View to retrieve data
  - no smarts at all
- Controller
  - “defines the way the UI reacts to user input” - Gang of Four
    - Strategy Pattern
  - often contains the main control loop

Often, MVC is done at the individual control level (text box, etc).

Benefits & Concerns

- Model and View are well separated
  - loosely coupled
    - multiple views on the same model
  - well define Observer interface required
- Controller
  - easy to change how the system responds to inpu
- Controller
  - tightly tied to View

Model View Presenter

- Model
  - same
  - might directly update View via Observer or not
- View
  - UI Loop here
  - might update itself
- Presenter
  - Only tied to View Interface

Model View Presenter

- Model
  - Player, Money, SavingsAccount
- View
  - Strings - all UI data are Strings
- Presenter
  - Money → String
  - String → Money

MVP workflow

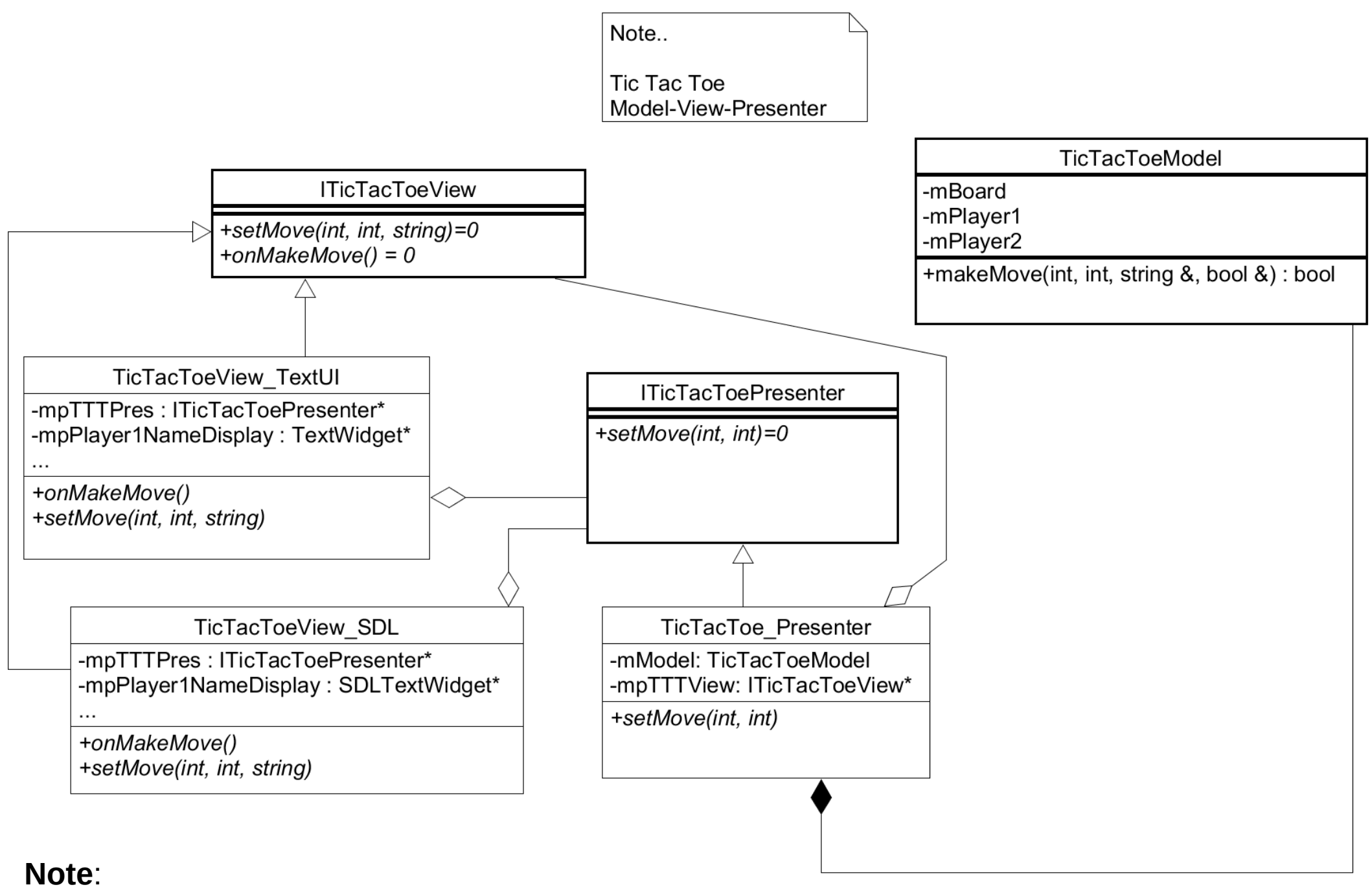
Note..

Tic Tac Toe
Model-View-Presenter

IView
+onMakeMove() = 0 *+setMove(int, int, string)=0*

Model
-mBoard -mPlayer1 -mPlayer2
+makeMove(int, int, string &, bool &) : bool

IPresenter
+setMove(int, int)=0

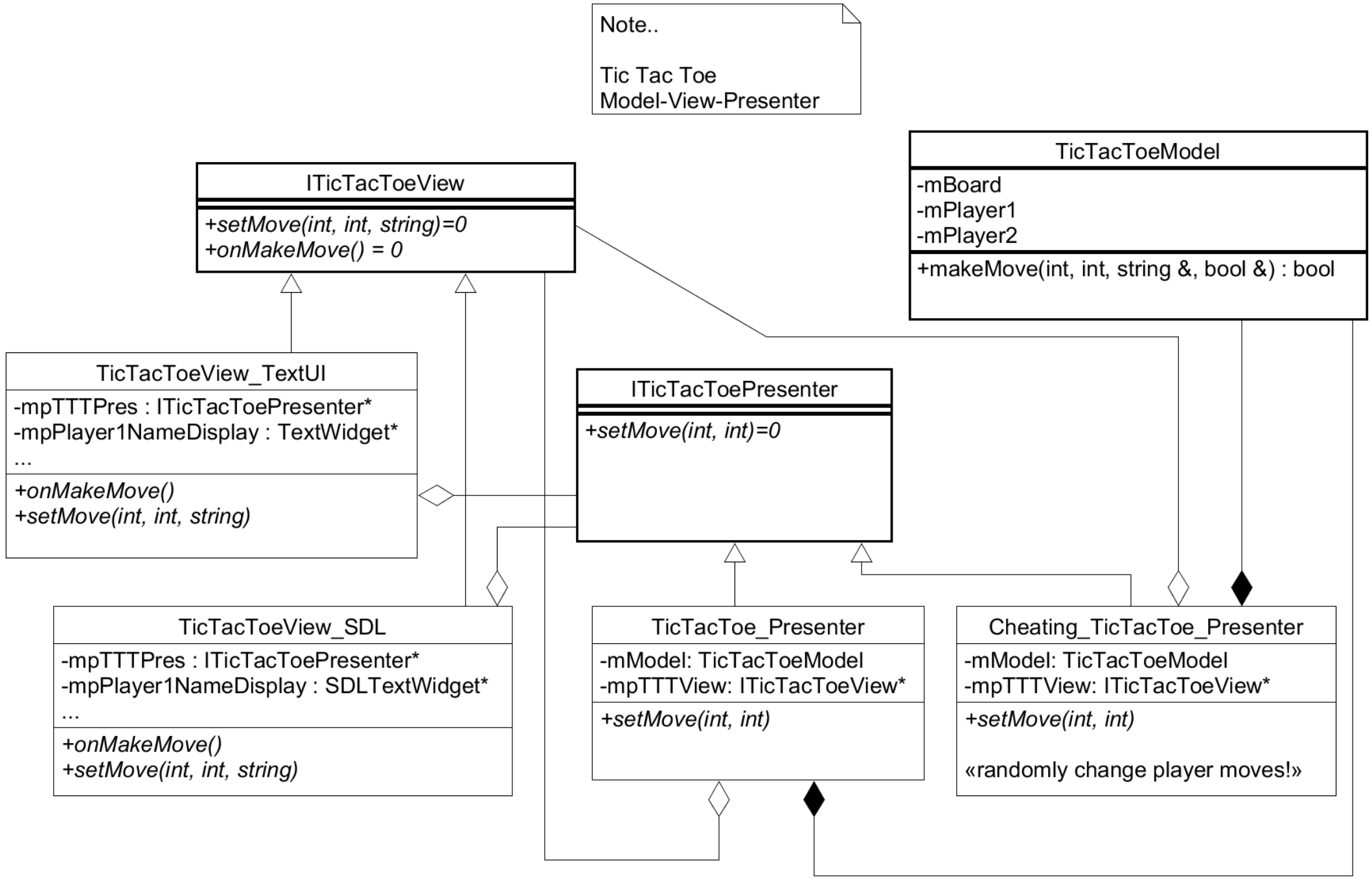


Note:

TicTacToeSDL_View is a subclass of SDLApp and ITicTacToeView

TicTacToeView_TextUI is a subclass of TextUI and ITicTacToeView

These really should be Composition, not Inheritance relations! (version 2.0)


Note..
Tic Tac Toe
Model-View-Presenter
TicTacToeModel
-mBoard
-mPlayer1
-mPlayer2
+makeMove(int, int, string &, bool &) : bool
ITicTacToeView
+setMove(int, int, string)=0
+onMakeMove() = 0
TicTacToeView_TextUI
-mpTTTPres : ITicTacToePresenter*
-mpPlayer1NameDisplay : TextWidget*
...
+onMakeMove()
+setMove(int, int, string)
ITicTacToePresenter
+setMove(int, int)=0
TicTacToeView_SDL
-mpTTTPres : ITicTacToePresenter*
-mpPlayer1NameDisplay : SDLTextWidget*
...
+onMakeMove()
+setMove(int, int, string)
TicTacToe_Presenter
-mModel: TicTacToeModel
-mpTTTView: ITicTacToeView*
+setMove(int, int)
Cheating_TicTacToe_Presenter
-mModel: TicTacToeModel
-mpTTTView: ITicTacToeView*
+setMove(int, int)
«randomly change player moves!»

Example Code

- Tic Tac Toe
  - Text Based
  - SDL Based
- Model
  - TicTacToeModel
  - TicTacToeBoard
  - TicTacToePlayer

Example Code

- Presenter
  - What interface does the View need?
  - How do we need to respond to changes in the Model?

```
class TicTacToePresenter : public ITicTacToePresenter
{
public:

  TicTacToePresenter (ITicTacToeView *pcView) ;

  virtual ~TicTacToePresenter () = default;

  // from View
  virtual void setMove (int x, int y);
  virtual void setName1 (std::string name);
  virtual void setName2 (std::string name);

  virtual void setSymbol1 (std::string);
  virtual void setSymbol2 (std::string);

  virtual void resetGame (std::string);

private:
  ITicTacToeView *mpcTTTView;

  TicTacToeModel mTTTModel;
};
```

Example Code

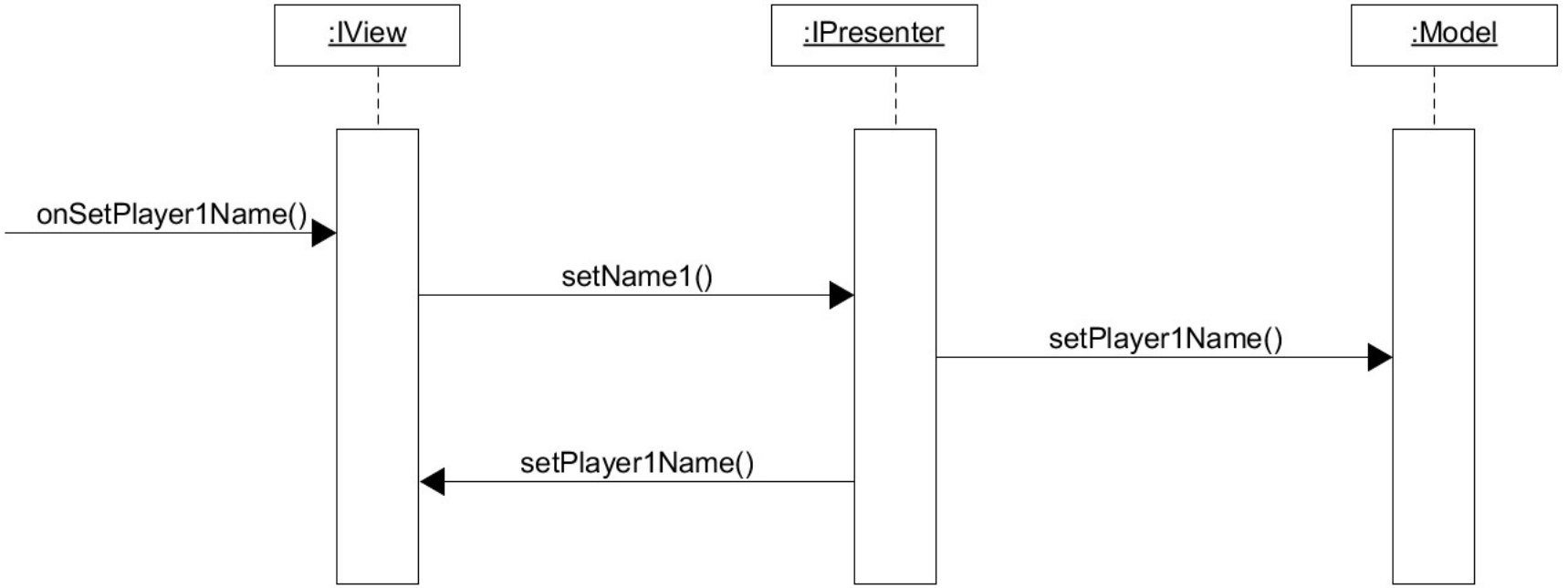
- View
  - What events can happen?
  - How should the presenter notify us of changes?

```
class ITicTacToeView
{
public:

  // events from Presenter
  virtual void setPlayer1Name (std::string name) = 0;
  virtual void setPlayer2Name (std::string name) = 0;
  virtual void setWinner (std::string name) = 0;
  virtual void setMove (int x, int y, std::string symbol) = 0;

  virtual void resetUI () = 0;
  virtual void redrawUI () = 0;

  // events from UI
  virtual void onSetPlayer1Name (std::string name) = 0;
  virtual void onSetPlayer2Name (std::string name) = 0;
  virtual void onSetPlayer1Symbol (std::string name) = 0;
  virtual void onSetPlayer2Symbol (std::string name) = 0;
  virtual void onMakeMove (std::string move) = 0;
  virtual void onQuit (std::string msg) = 0;
};
```

```
class TicTacToeView_TextUI : public ITicTacToeView, public TextUI
{
public:

  TicTacToeView_TextUI ();

  virtual ~TicTacToeView_TextUI ();

  // events from Presenter
  virtual void setPlayer1Name (std::string name);
  virtual void setPlayer2Name (std::string name);
  virtual void setWinner (std::string name);
  virtual void setMove (int x, int y, std::string symbol);
  virtual void resetUI ();


  // events from UI
  virtual void onSetPlayer1Name (std::string name);
  virtual void onSetPlayer2Name (std::string name);
  virtual void onSetPlayer1Symbol (std::string name);
  virtual void onSetPlayer2Symbol (std::string name);
  virtual void onMakeMove (std::string move);
  virtual void onQuit (std::string msg);

private:
  virtual void redrawUI ();

  static const int BOARD_SIZE = 3;

  ITicTacToePresenter* mpcTTTPresenter;

  TextBoardView mBoard;

  TextUITextWidget *mpPlayer1Name;
  TextUITextWidget *mpPlayer2Name;
  TextUITextWidget *mpWinnerName;

};
```

Sequence Diagram

- What order to the messages flow between objects
  - Shalloway, page 34, 44, 167



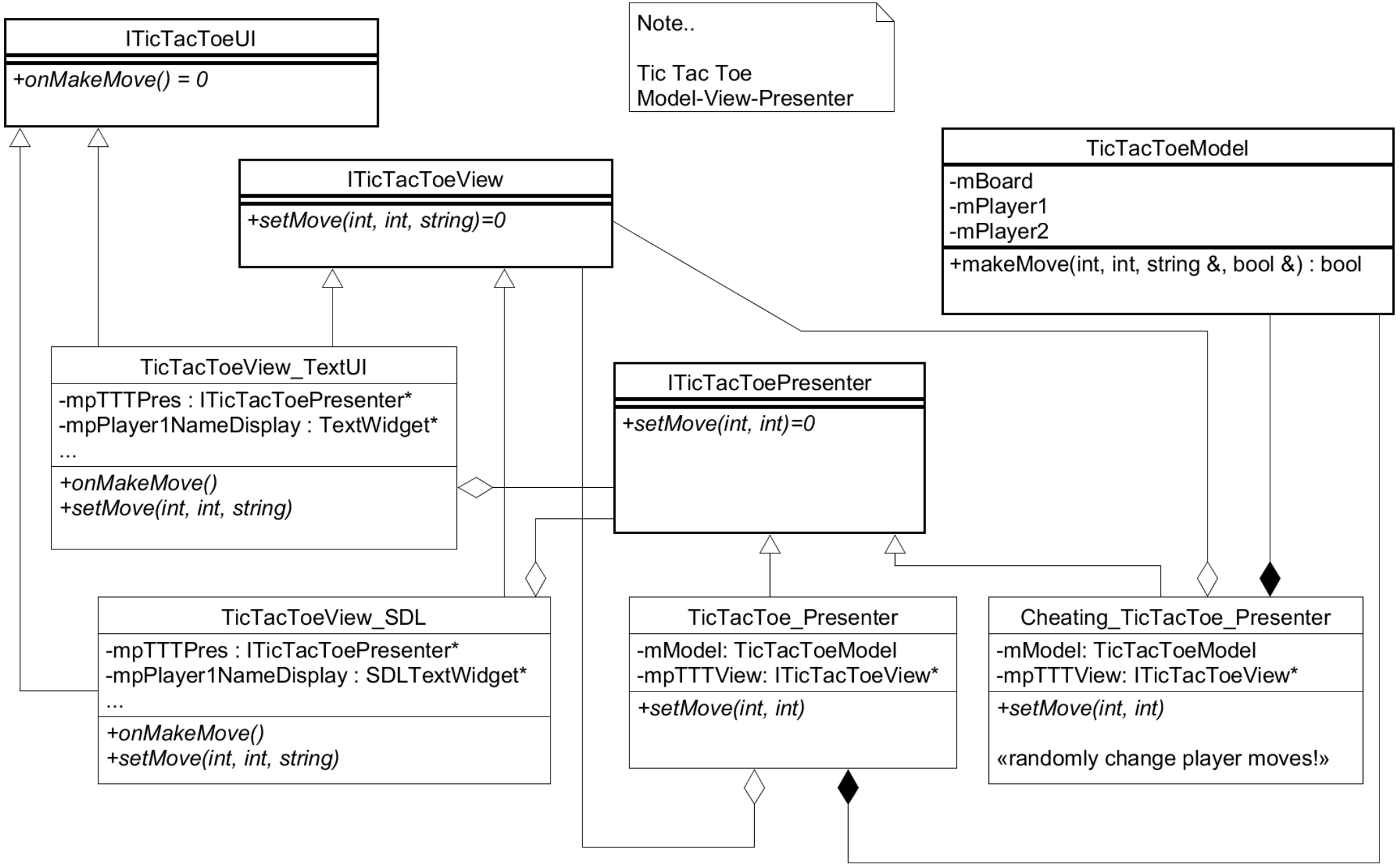
Interface Segregation

- Clients only need to know about methods that interest them

```
class ITicTacToeUI
{

public:

  // events from UI
  virtual void onSetPlayer1Name (std::string name) = 0;
  virtual void onSetPlayer2Name (std::string name) = 0;
  virtual void onSetPlayer1Symbol (std::string name) = 0;
  virtual void onSetPlayer2Symbol (std::string name) = 0;
  virtual void onMakeMove (std::string move) = 0;
  virtual void onQuit (std::string msg) = 0;
};
```

```
class ITicTacToeView
{
public:

  // events from Presenter
  virtual void setPlayer1Name (std::string name) = 0;
  virtual void setPlayer2Name (std::string name) = 0;
  virtual void setWinner (std::string name) = 0;
  virtual void setMove (int x, int y, std::string symbol) = 0;

  virtual void resetUI () = 0;
  virtual void redrawUI () = 0;


};
```

```
class TicTacToeView_TextUI : public ITicTacToeView,
  public ITicTacToeUI, public TextUI
{
public:

  TicTacToeView_TextUI ();

  virtual ~TicTacToeView_TextUI ();

  // events from Presenter
  virtual void setPlayer1Name (std::string name);
  virtual void setPlayer2Name (std::string name);
  virtual void setWinner (std::string name);
  virtual void setMove (int x, int y, std::string symbol);
  virtual void resetUI ();


  // events from UI
  virtual void onSetPlayer1Name (std::string name);
  virtual void onSetPlayer2Name (std::string name);
  virtual void onSetPlayer1Symbol (std::string name);
  virtual void onSetPlayer2Symbol (std::string name);
  virtual void onMakeMove (std::string move);
  virtual void onQuit (std::string msg);
```


ITicTacToeUI
+onMakeMove() = 0
Note..
Tic Tac Toe
Model-View-Presenter
ITicTacToeView
+setMove(int, int, string)=0
TicTacToeModel
-mBoard
-mPlayer1
-mPlayer2
+makeMove(int, int, string &, bool &) : bool
TicTacToeView_TextUI
-mpTTTPres : ITicTacToePresenter*
-mpPlayer1NameDisplay : TextWidget*
...
+onMakeMove()
+setMove(int, int, string)
ITicTacToePresenter
+setMove(int, int)=0
TicTacToeView_SDL
-mpTTTPres : ITicTacToePresenter*
-mpPlayer1NameDisplay : SDLTextWidget*
...
+onMakeMove()
+setMove(int, int, string)
TicTacToe_Presenter
-mModel: TicTacToeModel
-mpTTTView: ITicTacToeView*
+setMove(int, int)
Cheating_TicTacToe_Presenter
-mModel: TicTacToeModel
-mpTTTView: ITicTacToeView*
+setMove(int, int)
«randomly change player moves!»