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 Controller

- GUI architecture pattern
- made up of many Design Patterns
- Goals
  - separate underlying model from UI
  - reuse of model for different UIs
  - provide easily tested layers
- 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).

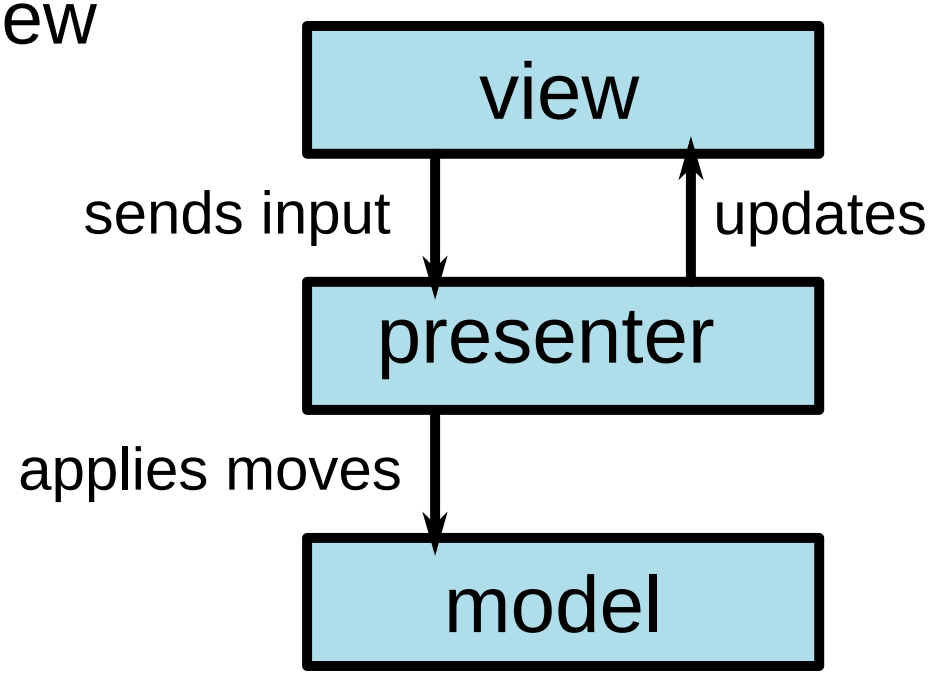
Benefits & Concerns

- 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 input
  - 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

Option 2:
Model-View-Presenter



https://frozenfractal.com/blog/2016/1/15/models-views-controllers-presenters-oh-my/

MVP

- Each layer provides an *interface* to code to
  - easy to swap out the View
- Clear separation between layers
  - Loosely Coupled

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

Model View Presenter

- Model
  - business logic
  - stateful
- View
  - Passive (dumb) display
  - stateful
- Presenter
  - process user action
  - retrieve data from model
  - stateless

Model View Presenter

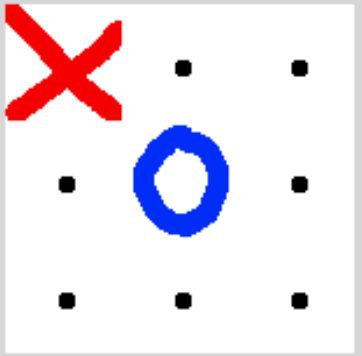
- Model
  - Player, Money, SavingsAccount
- View
  - Strings - all UI data are Strings
- Presenter
  - Money → String
  - String → Money
  - Translate button press to SavingsAccount::Deposit()
  - Update View with new balance

Example Code

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

SDL Tic Tac Toe

```
Player1: Chadd  Player2: Computer
X               O


        X # #
        # O #
        # # #
```



TextUI Tic Tac Toe

MVP workflow

Note..

Tic Tac Toe
Model-View-Presenter

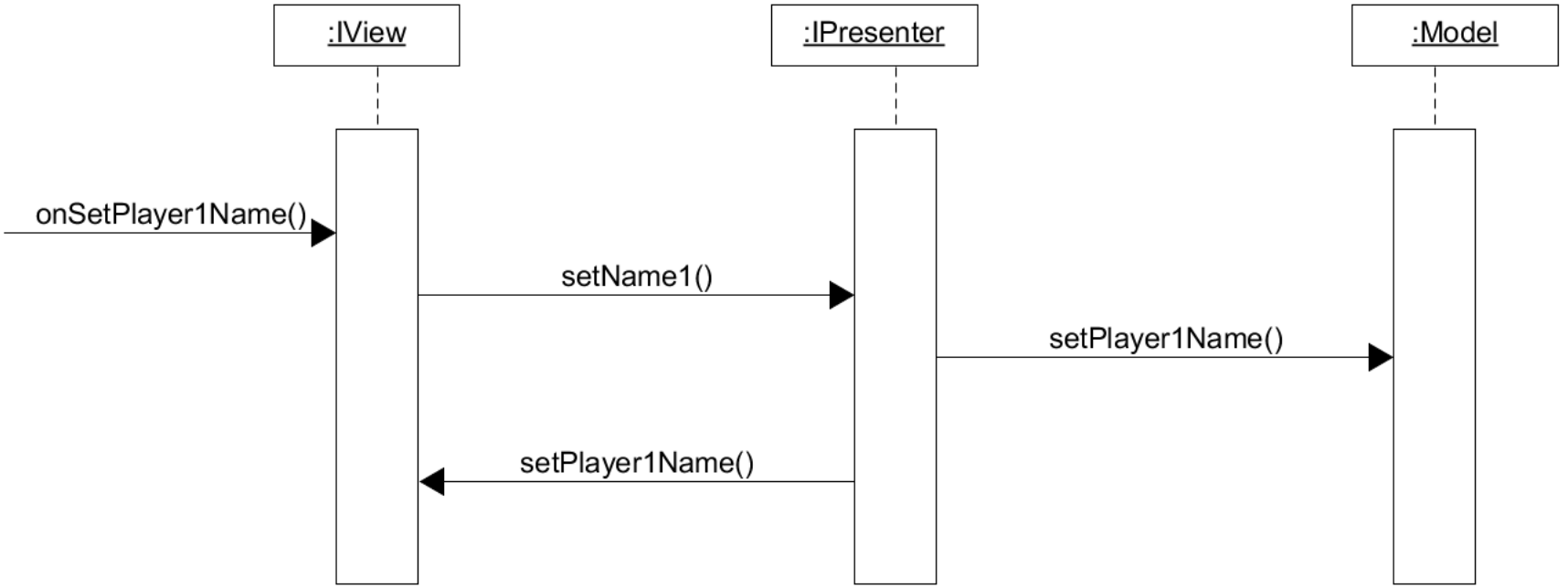
IView
+onMakeMove() = 0 *+setMove(int, int, string)=0*

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

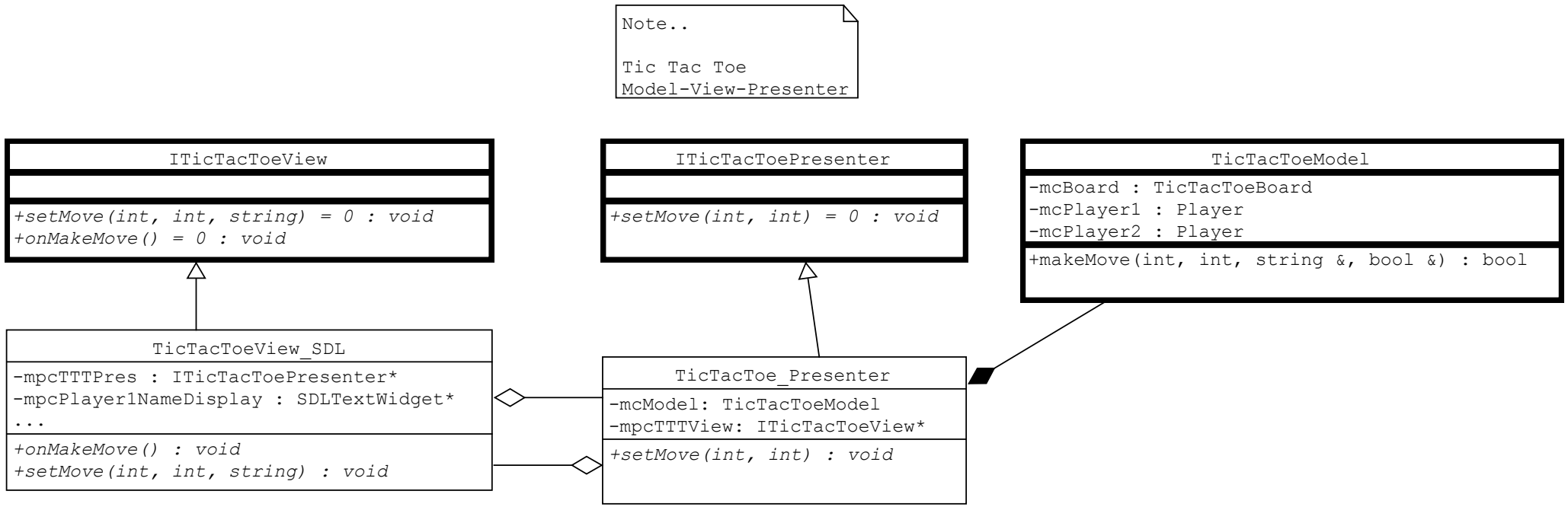
IPresenter
+setMove(int, int)=0

Sequence Diagram

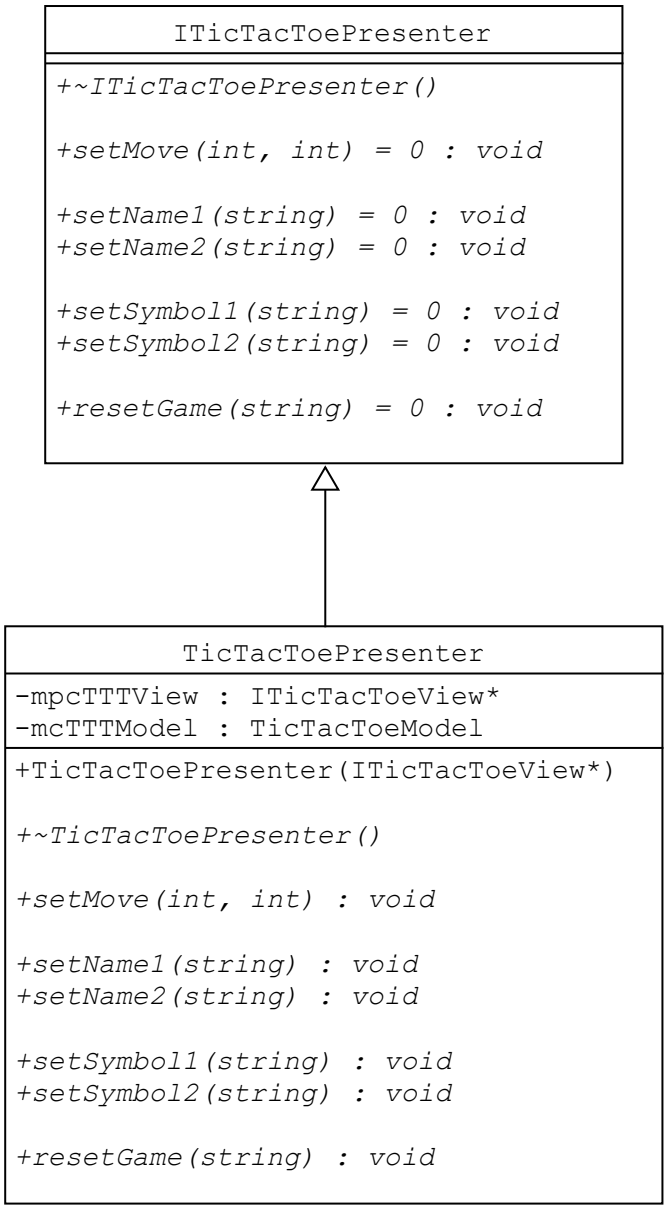
- What order to the messages flow between objects
  - Shalloway, page 34, 44, 167



UML - SDL Tic Tac Toe



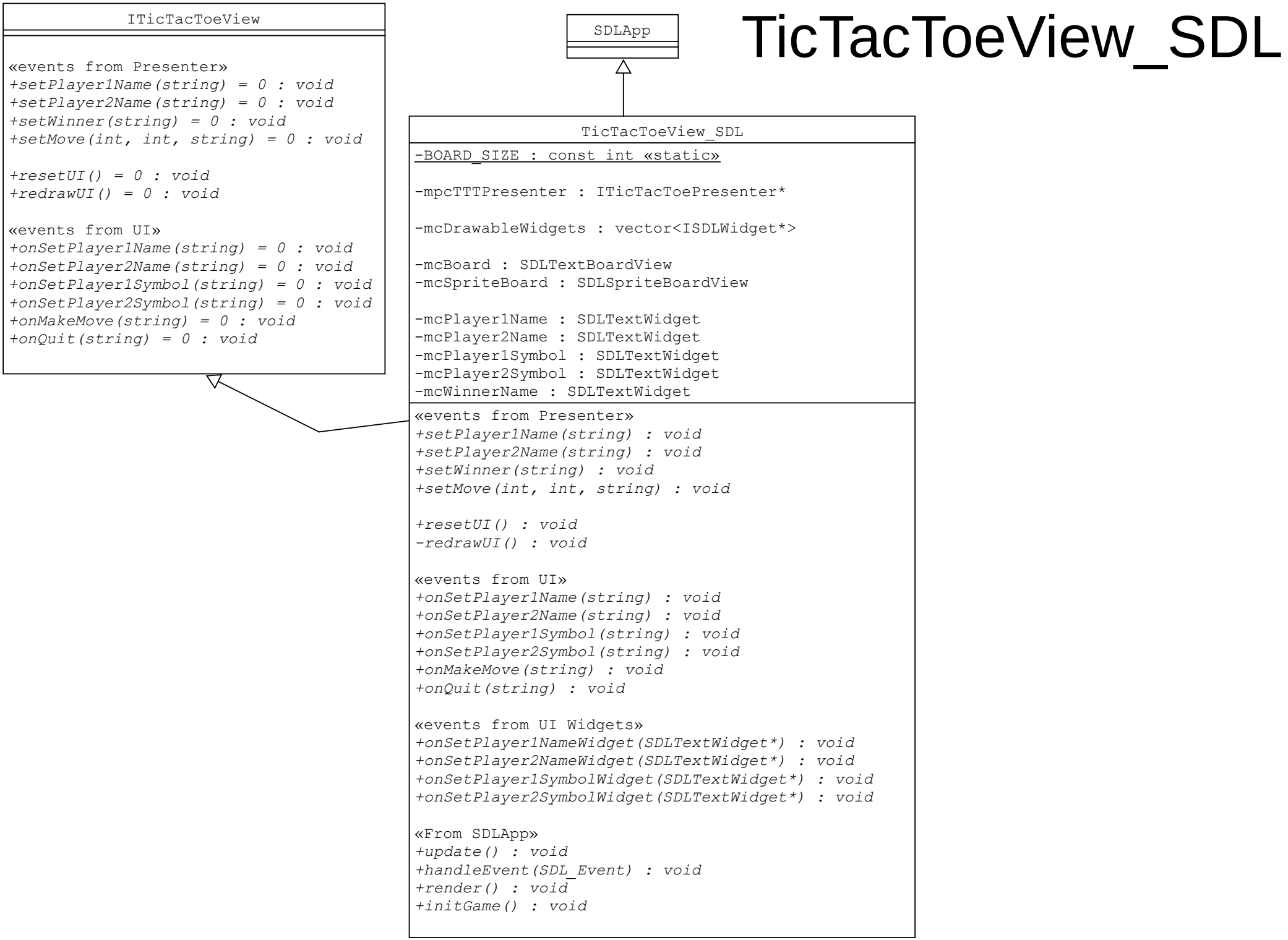
Presenter


ITicTacToePresenter
+~ITicTacToePresenter()
+setMove(int, int) = 0 : void
+setName1(string) = 0 : void
+setName2(string) = 0 : void
+setSymbol1(string) = 0 : void
+setSymbol2(string) = 0 : void
+resetGame(string) = 0 : void
TicTacToePresenter
-mpcTTTView : ITicTacToeView*
-mcTTTModel : TicTacToeModel
+TicTacToePresenter(ITicTacToeView*)
+~TicTacToePresenter()
+setMove(int, int) : void
+setName1(string) : void
+setName2(string) : void
+setSymbol1(string) : void
+setSymbol2(string) : void
+resetGame(string) : void

```
void TicTacToePresenter::setMove (int x, int y)
{
  std::string symbol;
  bool bRetVal;
  bool bWin=false;

  bRetVal =  mcTTTModel.makeMove (x, y, symbol, bWin);

  if (bRetVal)
  {
    mpcTTTView->setMove (x, y, symbol);
  }

  if (bWin)
  {
    mpcTTTView->setWinner (mcTTTModel.getCurrentPlayerName ());
  }
}
```

```
void TicTacToePresenter::setName2 (std::string name)
{
  removeSpaces (name);
  mcTTTModel.setPlayer2Name (name);
  mpcTTTView->setPlayer2Name (name);
}
```

TicTacToeView_SDL



TextUI Tic Tac Toe

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;


};
```