CS 485
Advanced Object Oriented Design

Observer Pattern (Ch 18)

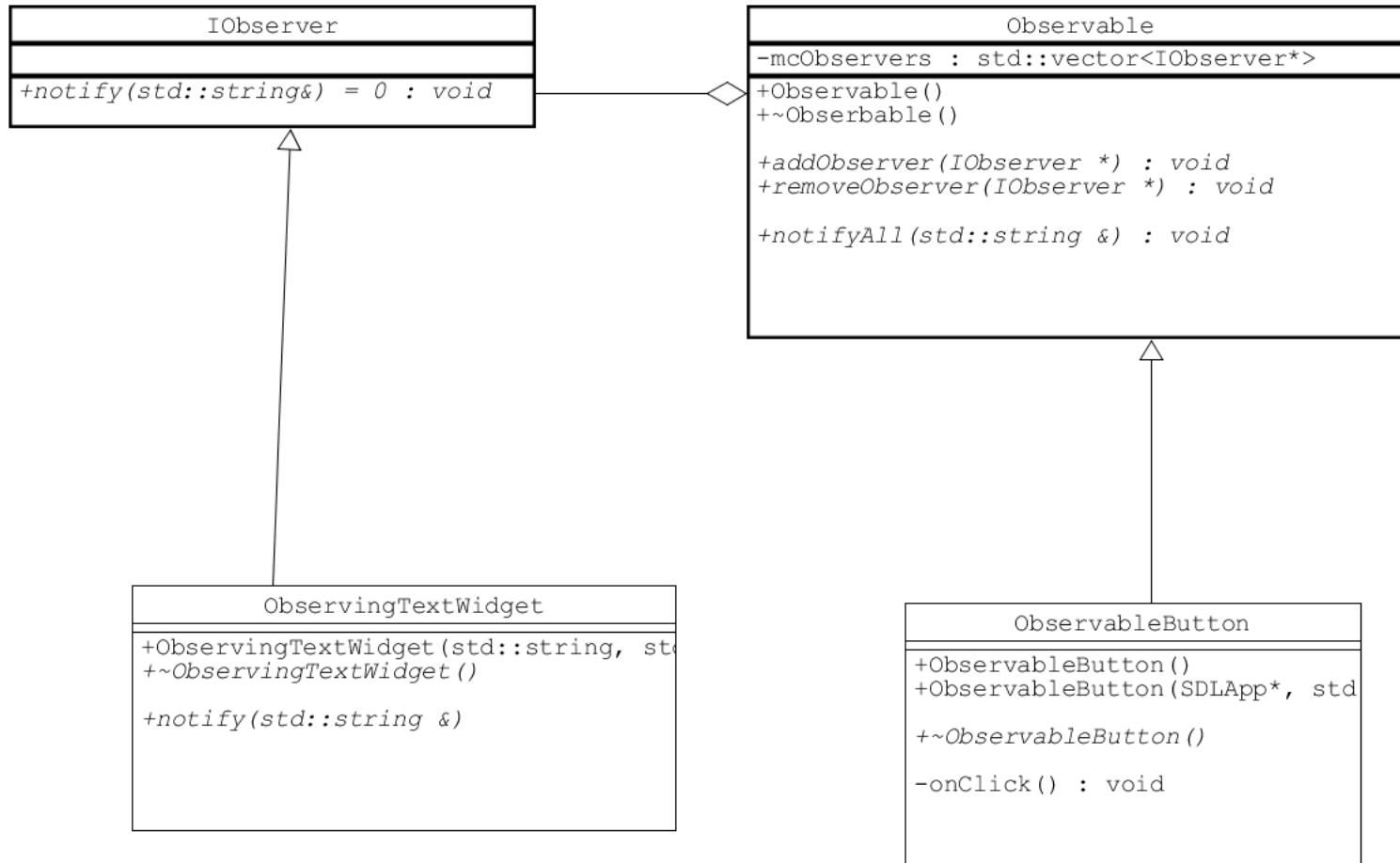# Spring 2019

http://www.netobjectives.com/PatternRepository/index.php?title=PatternsByAlphabet

http://www.netobjectives.com/files/books/dpe/design-patterns-matrix.pdf

Observer

- Behavioral Pattern

- One object advertises itself as *observable*

- Other objects may register as *observers*

- Observers are notified when the observable object changes state.

UML

```
            IObserver                                      Observable
─────────────────────────────         ──────────────────────────────────────────
                                       -mcObservers : std::vector<IObserver*>
+notify(std::string&) = 0 : void  ◇─── ──────────────────────────────────────────
                                       +Observable()
                                       +~Obserbable()

                                       +addObserver(IObserver *) : void
                                       +removeObserver(IObserver *) : void

                                       +notifyAll(std::string &) : void

```

```
        ObservingTextWidget                            ObservableButton
─────────────────────────────────         ──────────────────────────────────
+ObservingTextWidget(std::string, st      +ObservableButton()
+~ObservingTextWidget()                    +ObservableButton(SDLApp*, std

+notify(std::string &)                     +~ObservableButton()

                                           -onClick() : void
```

Goals

- Allow one object to be notified of a state change in another object

- Allow objects to *request* to be notified

- Notified objects may receive data in notification or request data after a notification
  - push vs pull

http://www.oodesign.com/observer-pattern.html

Often used for....

- View or Presenter may observe the Model
  - wait for state change in Model
  - update display based on notification

- UI Event Management
  - many widgets can register to be notified on a UI event

- News reader
  - register to get notified when an online news site publishes a new article

Example

```cpp
class IObserver
{
public:

  virtual void notify (const std::string &rcData) = 0;

};


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;
};
```

TicTacToe-MVP Repository