Questions:

* 7 different design patterns or just 7 total?
* Email Dr. Cerny about dropping offline feature

Design patterns

* Singleton – Service manager
* Visitor – For events
* Prototype – for copying / editing events
* Observer?
* Flyweight – Cache for DbDAO
* Adapter – Convert dayofweek to int and back
* We could use builder pattern to simplify long constructor for activity/deadline

Change view schedule screen to work with a schedule in general

Implement architecture – Kevin make skeleton to match design model

* Services
* Service Manager/configuration
* Persistence layer
* Change the event listener to use business layer / services instead of controller?

Event reference: isEditable bool or separate Readonly and Editable?

Features

* Make a popup box for two events in same cell, or right click menu - Kevin
* Assigning work times - Kevin
  + GUI
* Auto-Append - Kevin
  + Depends on assigned work times
  + Checkbox in event dialog
  + ScheduleManager#findFreeTime
* Generate To-do - Trent
  + Profile Screen – upcoming deadlines
  + Weekly schedule side panel
* Make monthly view – Josh?
* Groups - Samuel
  + Edit group permissions
    - Edit isEditable on references?
  + Share individual events to a group
    - Using reference
* Group Screen - Eric
  + One tab per group opened up
  + Access from profile screen
  + Show schedule
  + Show group members
* Profile Screen - Trent
  + Edit information
  + Groups
  + Categories
  + Upcoming deadlines
* View Time spent on specific task - Samuel
  + Int inputted by user when you edit an event
  + GUI
* View Time left on specific task - Josh
  + Calculated from end date - today
  + Right click menu or tooltip or something
* Calculate free time - Josh
  + How much time is free within working times

User input validation

Tests

User Guide

Live demo

Static code analysis

Git analysis (how many commits)

Time tracking

Point dist.

7 design pattern highlighted

Javadoc

Next meeting: Sunday 4/19 at 2:00