Meeting Cost Tracker

Create a skeleton for the Meeting Cost Tracker with these user-facing features:

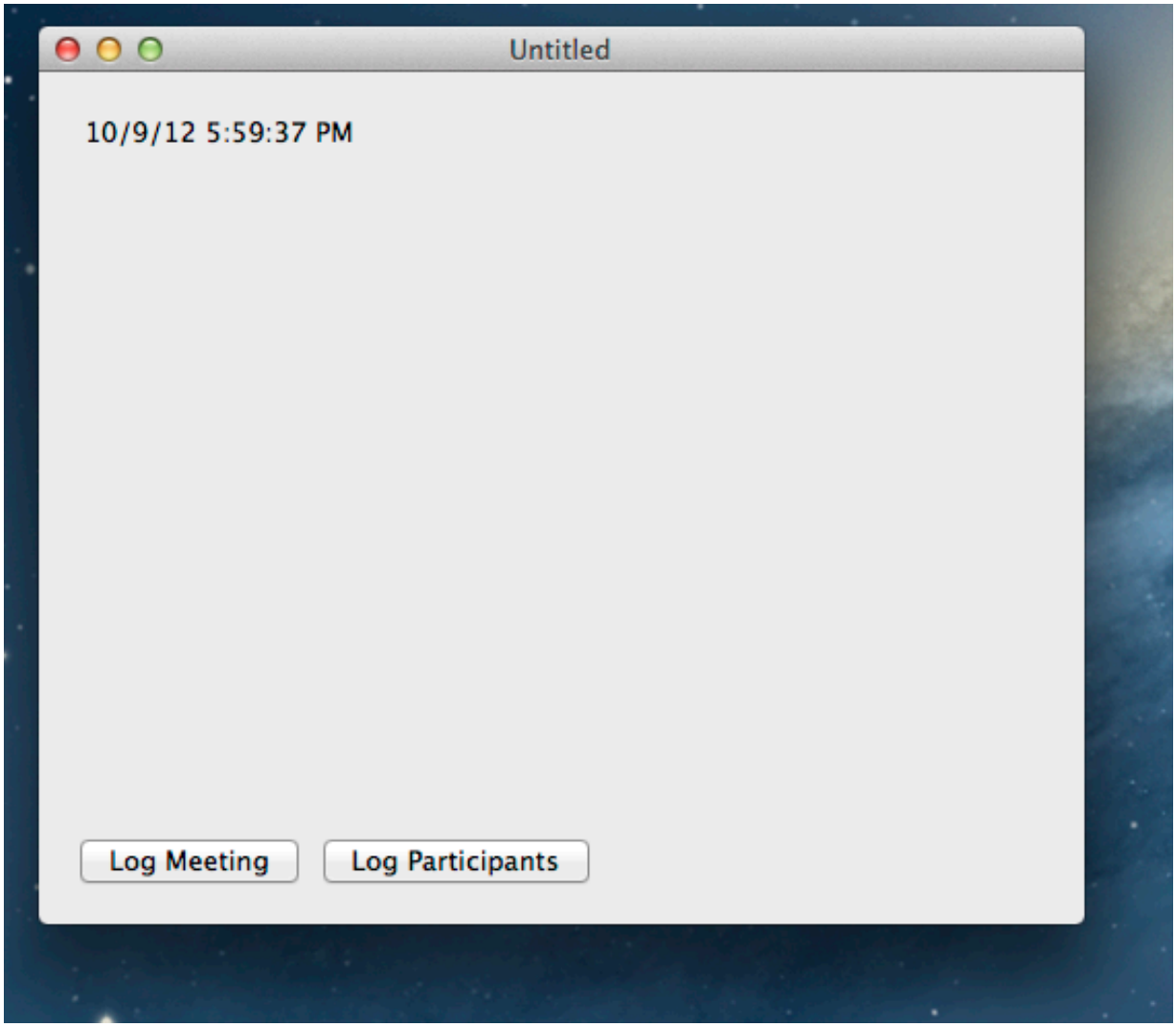
- an updating clock
- a button to print the meeting's -description to console
- a button to print each meeting participant's -description to console

This will be implemented in a Document-based application, with unit tests enabled. This week there is very little UI. The heavy lifting is in the creation of the Model classes.

Create a Document based Cocoa application. Within the Build Settings for your project, change "Use Automatic Reference Counting" to NO, and change "Analyze During 'Build'" to YES.

Download testPerson.m and testMeeting.m from Catalyst, and add them to your project. Use the checkboxes to add them to the Unit Tests target, but not to your primary application target. Do not modify these files. If you add more unit tests of your own, place them in a different file within your project.

Use the shortcut cmd-U to exercise the unit tests.



Required structure:

`Person` class implements these methods:

```
- (NSString *)description;
- (NSString *)name;
- (void)setName:(NSString *)aParticipantName;
- (NSNumber *)hourlyRate;
- (void)setHourlyRate:(NSNumber *)anHourlyRate;

+ (Person *)personWithName:(NSString *)name
                hourlyRate:(NSNumber *)rate;
- (id)initWithName:(NSString*)aParticipantName rate:
(double)aRate;
```

`Meeting` class implements these methods:

```
- (NSString *)description;
- (NSDate *)startingTime;
- (void)setStartingTime:(NSDate *)aStartingTime;
- (NSDate *)endingTime;
- (void)setEndingTime:(NSDate *)anEndingTime;
- (NSMutableArray *)personsPresent;
- (void)setPersonsPresent:(NSMutableArray
*)aPersonsPresent;
- (void)addToPersonsPresent:(id)personsPresentObject;
- (void)removeFromPersonsPresent:
(id)personsPresentObject;

- (void)removeObjectFromPersonsPresentAtIndex:
(NSUInteger)idx;
- (void)insertObject:(id)anObject
inPersonsPresentAtIndex:(NSUInteger)idx;

- (NSUInteger)countOfPersonsPresent;
- (NSUInteger)elapsedSeconds;
- (double)elapsedHours;
- (NSString *)elapsedTimeDisplayString;

- (NSNumber *)accruedCost;
- (NSNumber *)totalBillingRate;
```

Implement at least one of these methods, to create and return Meeting instances prepopulated with some or all of the Marx Brothers, Three Stooges, and Star Trek Captains.

```
+ (Meeting *)meetingWithStooges;
+ (Meeting *)meetingWithCaptains;
+ (Meeting *)meetingWithMarxBrothers;
```

Your NSDocument subclass implements these methods:

```
- (Meeting *)meeting;
- (void)setMeeting:(Meeting *)aMeeting;
- (NSTimer *)timer;
- (void)setTimer:(NSTimer *)aTimer;

- (IBAction)logMeeting:(id)sender;
- (IBAction)logParticipants:(id)sender;
```

```
- (void)updateGUI:(NSTimer *)theTimer;

- (void)windowWillClose:(NSNotification *)notification;
```

Do not use bindings or dot notation. Do not use `@property` or `@synthesize` within your Person and Meeting classes; you may use them for the NSDocument subclass.

Grading focus

- Model/view/controller separation.
- No GUI references in Person or Meeting classes.
- Instance variables begin with underscore. Proper accessors are written.
- There is no direct access to instance variables outside of the accessors.
- No memory leaks or other management errors.
- All unit tests pass.

Notes and hints

For the timer, refer to the definition of `stopGo`: from Hillegass and Preble, page 327. Substitute your own update method for `checkThem`:, and use accessors instead of making direct reference to the timer instance variable.

The subclass of NSDocument you'll create (perhaps called MeetingDocument) will be configured to save your data automatically. Since we haven't written code to do that yet, you'll see a distracting error message periodically. You can ignore it. But if you really want to get rid of that error message, find the -dataOfType:error: implementation and temporarily replace it with this version (getting it to work correctly will be later):

```
- (NSData *)dataOfType:(NSString *)typeName error:(NSError **)outError
{
    // FIXME: placeholder
    return [NSData data];
}
```

The NSDocument subclass needs an instance of Meeting, and an instance of NSTimer. I suggest creating the Meeting in NSDocument's -init, and

creating the timer in -windowControllerDidLoadNib:. Both of those methods will be stubbed out for you when you create your project. Make sure to populate your Meeting object, perhaps by using +meetingWithStooges or +meetingWithMarxBrothers to create your Meeting in the first place.

Try your project with multiple windows open at once. What happens when you close a window? What can you do with -windowWillClose: to fix that?

The total hourly rate of a Meeting is the sum of its participants' hourly rates. You might explore several ways to do that computation: a simple C for loop, a block, fast enumeration, or array key path operations. The externally exposed rates are NSNumbers; you'll have to convert between NSNumber and double for the arithmetic.

The -description method is already declared by NSObject. You're overriding that method, so you should not declare it in Meeting.h/Person.h. Simply implement it in Meeting.m and Person.m.

The -windowWillClose: method is already declared in protocol <NSWindowDelegate>. Your NSDocument subclass instance is the delegate of its window. Do not declare this method–simply implement it.

When you run my sample implementation, you'll need to have the Console application running in order to see the log messages. Console.app is found in the Utilities folder within your Applications folder.

The supplied unit test files are required. The supplied header files are optional; you can use them as a starting point or use them as they are.

It's okay if you can't remember the names of all of the Star Trek captains, Three Stooges, or Marx Brothers.