

Before we start...

If you are not on Slack, you should probably be on Slack.

- Don't leave me hanging.

If your team has 5 members, contact me immediately, you're going to be dissolved.

If your team has 6 members, good news, message me.

Did you read my e-mail from yesterday? I hope so.

Before we start...

We will start reviewing your proposals tomorrow.

- If it's good-to-go, we'll be in touch shortly
- If "*it's complicated*", you will be assigned your project next week

Don't ever hesitate to contact me or your TA.

- I'll lurk your Slacks during weekdays
- I like Slack
 - I like it when you mention me on Slack.
 - Did I mention that I like Slack?
- After your initial meeting, though, use your TAs as much as possible - they will be monitoring your progress more closely

Before we start...

Next ~3 weeks is design stage.

This means we need to cover:

- Design Patterns (MVC and related)
- UML
- Getting Started with Grails (including a tutorial)

After that:

- Git and GitHub (definitely a tutorial)
- User Interfaces (including, probably, a couple of tutorials)
- Other curriculum material we need to do
 - Development methodologies, project management
 - i.e. the ultimate big business buzzword extravaganza

Stage 1

Software Project Design

Due Friday February 10th at Midnight

Stage 1 is the culmination of your project's design.

Components

- Team Website
- UML Class Diagram
- User Stories and Tasks

Team Website

- Home Page
- Team Roster Page
- Software Design Page
- UI Design Page
- Project Plan Page

Home Page

- Team name and number
- Brief description of your project

Team Roster Page

- List of team members
- Each member's skills/background/hobbies/philosophy/whatever
- Primary role on team
 - e.g. Code Champion, Repo Hero, Database Ace, Team Overseer, UX Guru, etc.

Software Design Page

- UML Class Diagram
- More details to be added later.

UI Design Page

- Look and feel / design language
 - e.g. flat design, minimal, Material Design, etc.
 - You can cite existing software as inspiration
- High-level description of intended user interactions
 - How does the user navigate between screens?
 - Summarize the most typical use case / workflow
- UI Mockups
 - Graphic representing each user interface view
 - Use of software recommended
 - Make sure all user stories are covered

Project Plan Page

Your returned project proposals will likely include

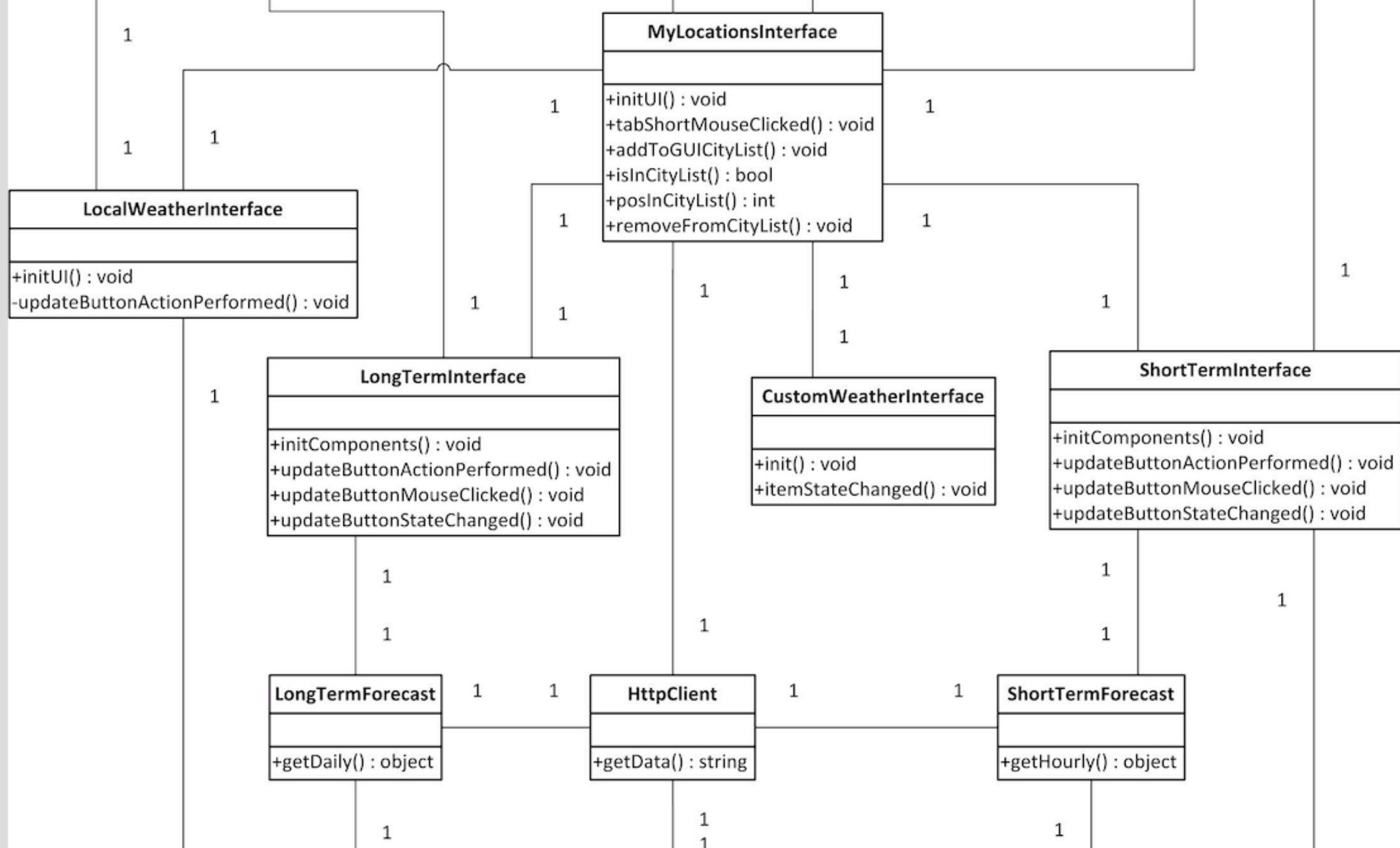
- Most of the features you originally proposed
- Requested, under specified features you need to elaborate

For all features (original and requested) you must include

- Title
- Description (brief)
- Feasibility statement and implementation plan
- Statement of dependency
 - i.e. whether/how other features depend on this one

UML Class Diagram

- Don't worry, we'll review this.
- Regardless of your project's framework (Grails, Vanilla Java, Android, Angular, React, etc...)
 - State the high level design pattern your project is following
 - Don't worry, we'll introduce this.
- Class names, relationships, instance variables, methods
- You can generate UML class diagrams using software
- **DO NOT INCLUDE USER INTERFACE CLASSES**
 - But do include any classes that *respond* to your UI
 - These are the 'C' in MVC



User Stories and Tasks

Basically, your Project Plan Page implemented using a project management tool.

- You can use GitHub for this!


Every user feature should be reflected by a user story including

- The story itself
- Any acceptance criteria
- Assigned team member(s)
- Associated Milestone with Due Date (if applicable)

Stock Price Charting #2

Edit

New Issue

 **Open** ethamajin opened this issue 6 minutes ago · 0 comments



ethamajin commented 6 minutes ago



User can see charts of tracked stock prices since tracking began.

Acceptance criteria:

- ☐ Charts working for all tracked stocks
- ☐ Passes unavailable resources tests



ethamajin added the **enhancement** label 6 minutes ago



ethamajin added this to the **Milestone 1** milestone 6 minutes ago



ethamajin self-assigned this 6 minutes ago

Assignees



ethamajin

Labels



enhancement

Projects

None yet

Milestone



Milestone 1

Notifications

Unsubscribe

You're receiving notifications because you were assigned.

1 participant

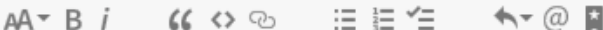


Lock conversation



Write

Preview



Leave a comment

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

 Styling with Markdown is supported

Close issue

Comment

For more details, find the Stage 1 description on
the course website!

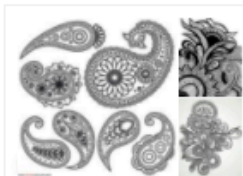
Primer on MVC and Grails

Today: Motivation

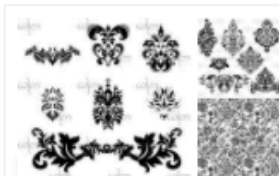
Friday: More on MVC, Grails*, UML, etc.



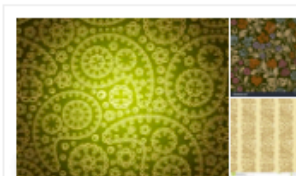
Graphic Design Geometric



Drawing



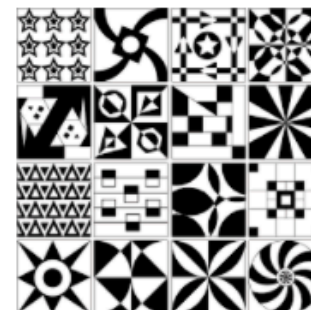
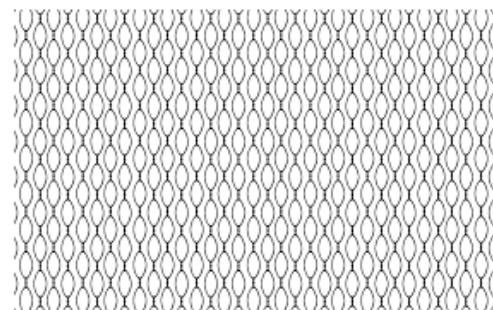
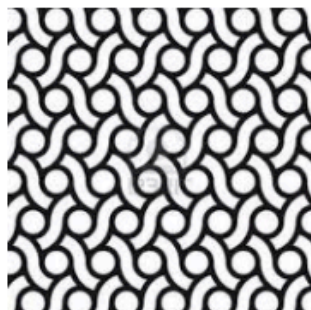
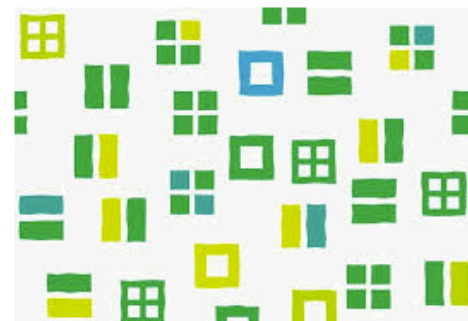
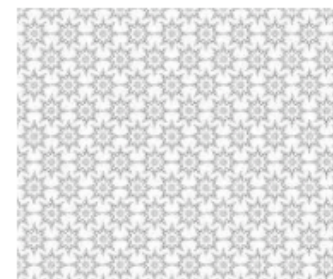
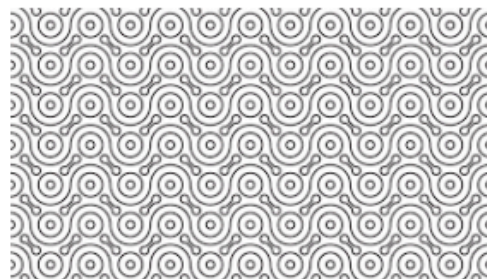
Art



Vintage Floral



Graphic



What is a design pattern?

...not that.

Design patterns are abstract code organization solutions.

They help with:

- Maintainability
- Code Reuse
- Separation of Concerns
- Security
- Resource Management
- Communication

Why are we learning this?

Design patterns are ubiquitous in OO-programming.

- MUCH more on this next year

In this course, we will build on the concepts of class hierarchies and modularity to write thoughtfully organized code.

At the highest level, we will consider the MVC design pattern.

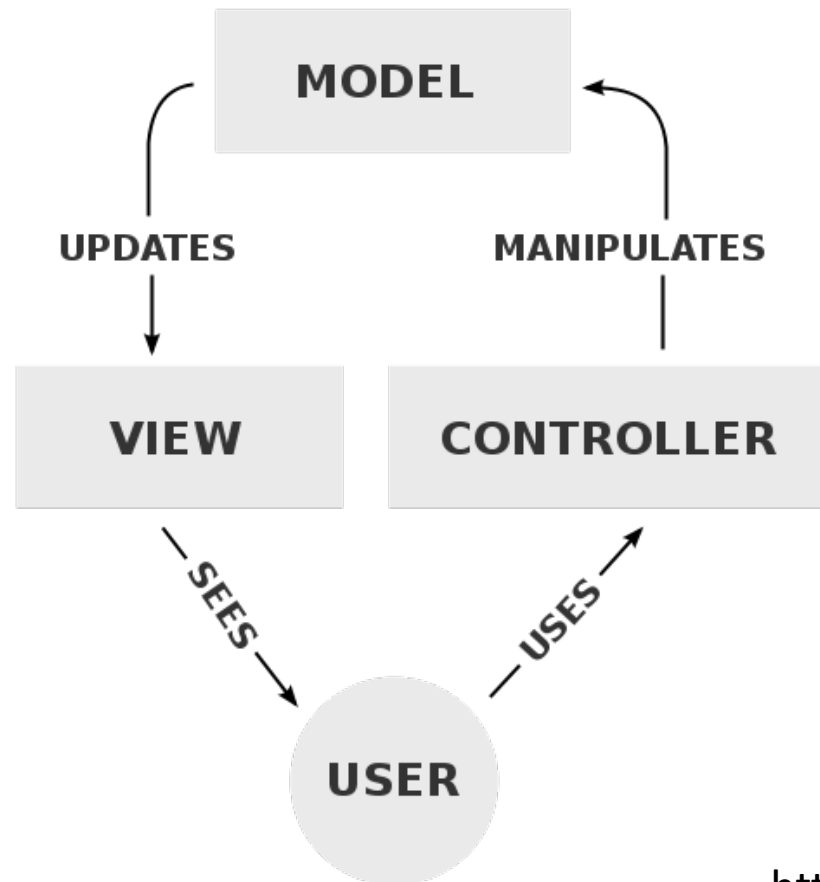
- MVC framework skills are in (very) high demand

Even if your project is in plain Java, you will benefit from a basic knowledge of MVC.

What is MVC?

MVC = Model View Controller

Definitions and explanations of MVC are inconsistent.



Model

Represents your data, logic, and rules.

- Domain Classes
- Interpretation/Presentation of Database

The model consists of a set of classes to represent all persistent entities in your program.

View

A visual interpretation of information at the model level.

- Multiple views can be defined for one data item.
- Changes to a data item in the model level should be reflected by any view that is bound to it.

In your project, views will correspond to user interface screens/components and may be written in Java, HTML, Javascript*, or *other* depending on your framework.

Controller

Responds to user interactions or other sources of input.

- Actions that change the state of the program or data should not be coded into views.
 - Remember, a view should be an *interpretation* of your data
- Instead, controller classes respond to input (e.g. filling out a form, clicking a button, writing a comment, navigating, etc.)
 - Updates the model
 - Issues commands to views

MVC and You

Regardless of your project

- Design your project classes with MVC in mind.
- Use the design pattern to help you decide what functionality belongs in what class.
- Be prepared to state how your design relates to the 3 components of MVC

MVC and You

If you decide to use Grails or other managed framework (mobile, etc.)

- In many cases MVC adherence is enforced
- Trying to fight against MVC will result in broken or hard-to-read code
- Always ask yourselves:
 - Is our user interface doing too much?
 - Have we designed our program so that it will be easy to implement a new user interface component without rewriting backend code?
 - Are data access and manipulation managed by specific classes?