4156

10/13/16

reminders - team requirements due tonight
1st 360-degree feedback
due Tuesday

next Thursday - team architecture due
show assignment

also coming up first exam
read the book!
review in class next Thursday

4156

10/13/16

design patterns

someone somewhere has already solved your programming problem or a very similar problem

sometimes this results in a library or framework where you can directly reuse open source code

sometimes this results in a pattern or template showing a way to organize code & code interactions, enabling reuse of "experience" w/o specific code

sometimes many developers already found that a particular approach was a REALLY BAD IDEA "anti-patterns"

many books & websites list design patterns with code samples in various languages

4156

10/13/16

many discussions of design patterns mention "GoF" or "Gang of Four" refer to the authors of a 1995 book titled "Design Patterns"

show sample lmhs

we are not going to cover design patterns in this class, except for one - MVC

MVC = Model View Controller is composed of several design pattens, so really an architecture

what is an architecture?

client - server
peer - to - peer
3 - tier

draw

MVC is employed for almost all web & mobile apps and is built into nearly every web/mobile development framework

4156

10/13/16

model = application data
classes & methods that
manipulate application state
business logic

view = rendered displays data for
user to see
updated when model changes
may be multiple views of
same data
(e.g., web & mobile versions
of website)

controller = translates user actions
(menu selections, mouse
actions, data entry,
gestures, etc.)
into operations on model
may be multiple controllers
for same model

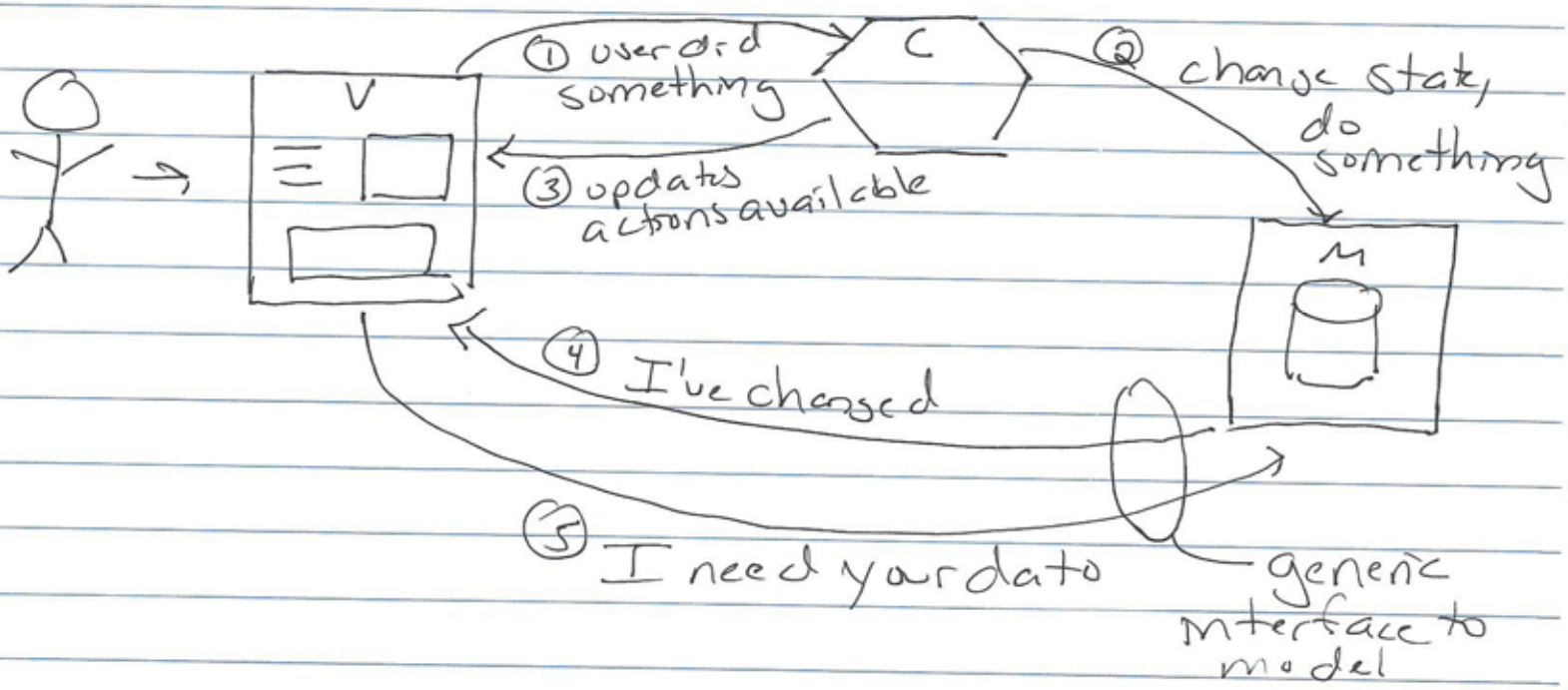
4156 10/13/16

example - itunes or similar music player

view - playlist, current song, graphics, sound, video, etc.
some elements accept user selection, action, data entry, etc.

controller - accepts user actions from view
tells model what to do

model - all data & application state, business logic
stores content, knows how to play songs, shuffle, rip, etc.



4156

10/13/16

M, V, & C loosely coupled
model & view know nothing
about each other, use
generic interface
controller has to know enough
about both to translate
user requests from options
provided by view to what
model can really do

coupling = interdependencies among
program units, "need to know"

cohesion = degree to which a program
unit exhibits a single purpose
(SRP - single responsibility principle)

MVC separates UI from application logic
different skill sets to implement
nice UI vs. good data structures
& algorithms, enables
independent change

Some variants of MVC treat model
purely as data & put logic in
controller, or force all interactions
between M & V to go through C

4156 10/13/16

notice I drew a little database w/in the model

V+C may have some ~~persistent~~ state, such as saved layout + preferences, but domain data belongs to M

how exactly you implement persistence depends on your choice of framework + database

databases + agile processes
- you do not need to design entire schema up front
- you do need to convert any existing stores when schema changes
- developers often test first with a mock DB or in-memory database, before testing with persistent store

do not store sensitive data in cleartext — encrypt