CSE 331
Section 7 cheat sheet

OO design heuristics, MVC

Goals for modular software:

- ***Decomposable***: can be broken down into pieces
- ***Composable***: each piece is useful, and can be combined with other pieces in multiple ways
- ***Understandable***: each piece makes sense in isolation
- ***Safe***: an error affects few modules

Design heuristics help us reason about:

- What should be a class
- Responsibilities of a particular class
- Collaboration between classes

OO design heuristics

What should be a class?

Nouns from the project specification are good candidates.

- "A class should capture one and only one key abstraction."
- "Spin off non-related behavior into another class (i.e., non-communicating behavior)."
- "Be sure the abstractions that you model are classes and not simply the roles objects play."
- "Do not create god classes/objects in your system. Be very suspicious of a class whose name contains Driver, Manager, System, or Subsystem."
- "Model the real world whenever possible."
- "Eliminate irrelevant classes from your design."
- "Eliminate classes that are outside the system."
- "Do not turn an operation into a class. Be suspicious of any class whose name is a verb or is derived from a verb, especially those that have only one piece of meaningful behavior (don't count set, get, print)."
- "Agent classes are often placed in the analysis model of an application. During design time, many agents are found to be irrelevant and should be removed."
- "Beware of classes that have many accessor methods defined in their public interface."
- "Beware of classes that have too much noncommunicating behavior."

What responsibilities should a class have?

What should its public interface be?
Look at verbs used in the project specification.

- "Keep related data and behavior in one place."
- "All data should be hidden within its class."
- "Minimize the number of messages in the protocol of a class."
- "Implement a minimal public interface that all classes understand."
- "Do not put implementation details such as common-code private functions into the public interface of a class."
- "Do not clutter the public interface of a class with items that users of that class are not able to use or are not interested in using."
- "Classes should not contain more objects than a developer can fit in his or her short-term memory. A favorite value for this number is six."

How should classes collaborate? What should their relationship look like?

relationship	description	example
extension	"is a" – a superclass / subclass relationship	`ChocolateChipCookie` "is a" `Cookie`
composition	"is entirely made of" – a field that is the true essence of the state of the object containing it; stronger than aggregation	`Book` "is entirely made of" `Page`
aggregation	"is part of" – a field	`Engine` "is part of" `Car`
dependency	"uses temporarily" – often a parameter, rather than a field; an implementation detail	`LotteryTicket` "uses temporarily" `Random`

- "Users of a class must be dependent on its public interface, but a class should not be dependent on its users."
- "Classes should only exhibit nil or export coupling with other classes, that is, a class should only use operations in the public interface of another class or have nothing to do with that class."
- "Distribute system intelligence horizontally as uniformly as possible, that is, the top-level classes in a design should share the work uniformly."
- "In applications that consist of an object-oriented model interacting with a user interface, the model should never be dependent on the interface."

Model-View-Controller

Goal: separate classes that manage the user interaction (the *View*) from classes that contain the core data and behavior of the program (the *Model*).