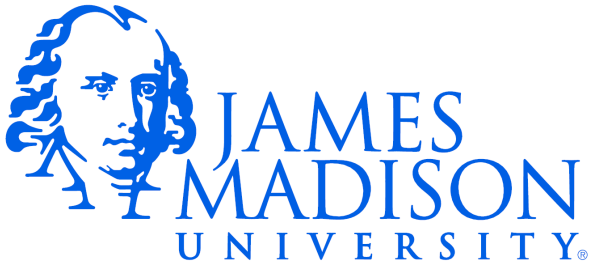
CS159



Review Class Design "Algorithm"

- Good for tackling "designer's block"

Some Criteria for Evaluating Designs

- Coupling:
  - The degree to which two classes (or methods) are "tied together." How likely is it that changing the internals of one will require modification of the other.
  - We aim for LOW coupling.
- Cohesion:
  - The degree to which the components of a class "hang together".
  - We aim for HIGH cohesion

Some Design Proposals

- Let's combine the GUI code and the SeatingChart class. Each call to step or solve can automatically refresh the display!

Some Design Proposals

- Let's combine the GUI code and the SeatingChart class. Each call to step or solve can automatically refresh the display!
  - Reduces Cohesion
  - Makes it harder to test and reuse the SeatingChart logic. Makes it harder to modify the GUI.

Some Design Proposals

- Let's give Student objects a reference to the SeatingChart they are in, and give them responsibility for moving themselves:

  bob.moveTo(3, 5)

  logic is similar to placeStudent

  no longer need setRow + setColumn methods?

Some Design Proposals

- Let's give Student objects a reference to the SeatingChart they are in, and give them responsibility for moving themselves:

  bob.moveTo(3, 5)

  logic is similar to placeStudent

  no longer need setRow + setColumn methods?

  - Not actually a *bad* idea, but it does increase coupling. In this design the Student class needs to "know about" the SeatingChart class.

Some Design Proposals

- Let's give Student objects a reference to the SeatingChart they are in, and give them responsibility for moving themselves:

  bob.moveTo(3, 5)

  - Not actually a *bad* idea, but it does increase coupling. In this design the Student class needs to "know about" the SeatingChart class.

Some Design Proposals

- Let's add code to the Student class for abbreviating names:

  getName() $\rightarrow$ “Nathan Sprague”

  getAbbreviatedName $\rightarrow$ “N. Sprague”

    Splits the string, pulls out the first letter etc.

- (Users have requested this as an option)

Some Design Proposals

- Let's add code to the Student class for abbreviating names:

  getName() $\rightarrow$ “Nathan Sprague”

  getAbbreviatedName $\rightarrow$ “N. Sprague”

    Splits the string, pulls out the first letter etc.

- (Users have requested this as an option)
  - Lowers cohesion. Perhaps create a Name class, or a Utility class that can perform these conversions.
  - Otherwise, introducing a Teacher class will require us to copy-paste this functionality.