An OO Design Exercise

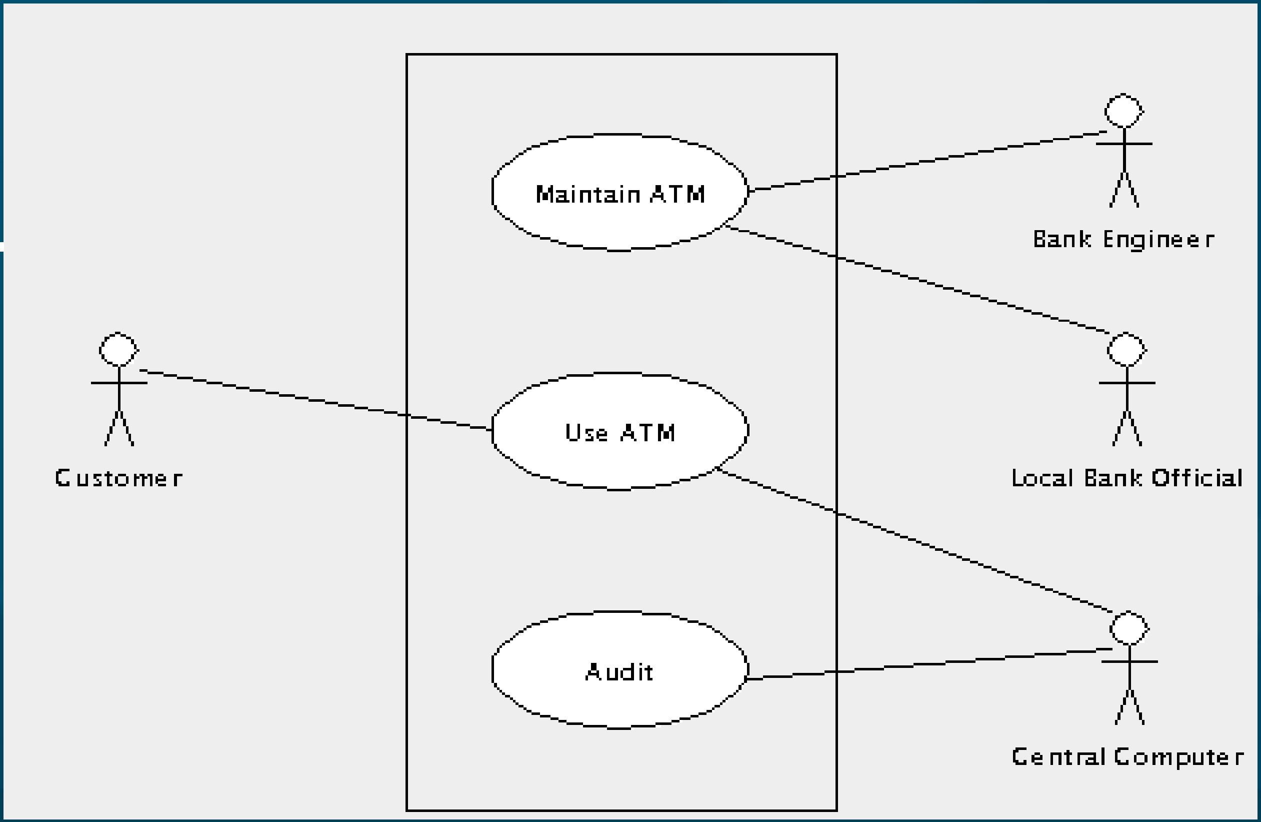
11 Mar 2009
CMPT166
Dr. Sean Ho
Trinity Western University

Steps to OO design: wADes

- (Prereq: understand client requirements)
- System behaviour
  - Use-case scenarios
  - User interface mockups
- Components
  - Self-contained blocks with narrow interactions
- From components to classes
  - Attributes, methods

(1) System behaviour: use-case

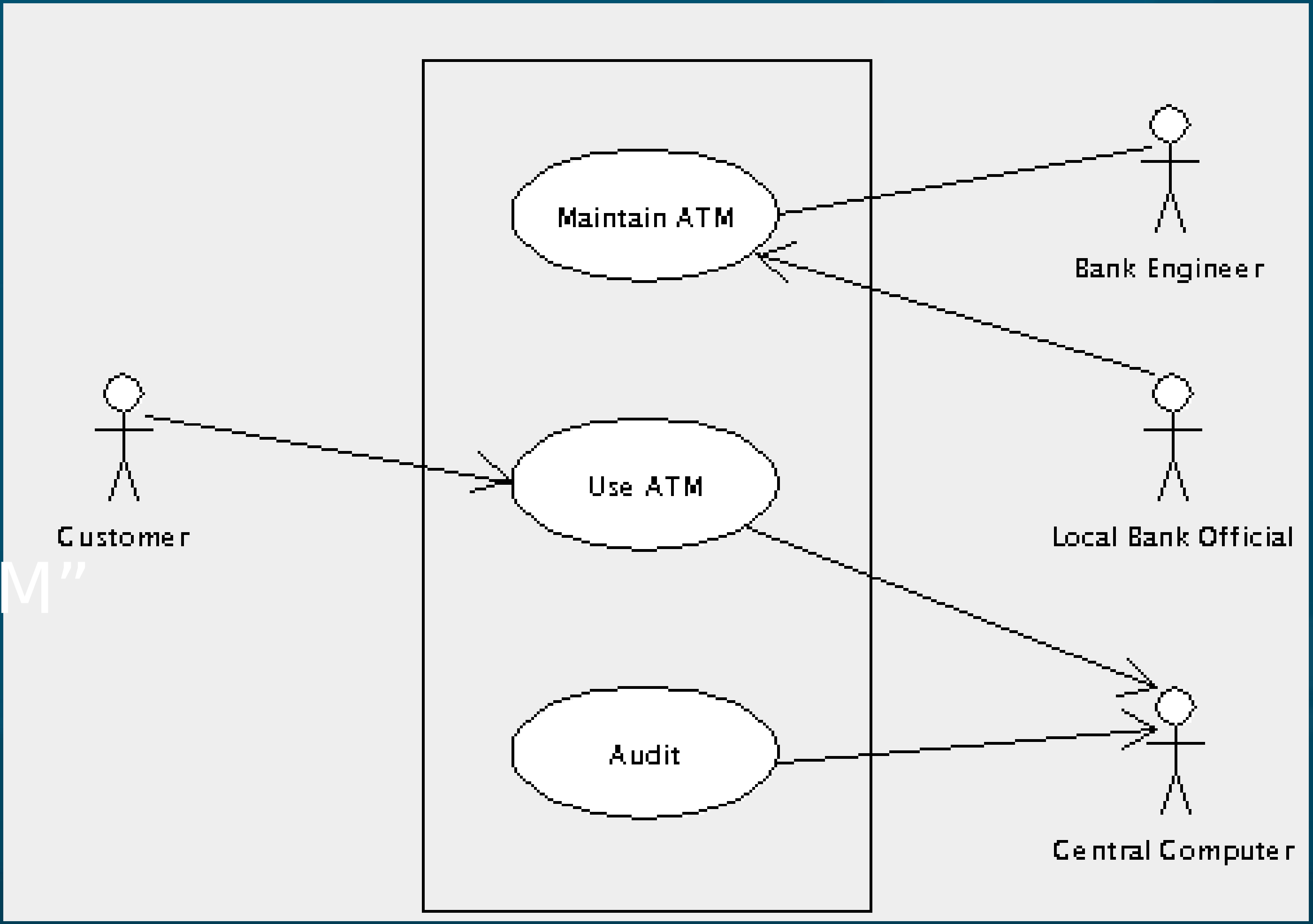
- UML use-case diagrams show:
  - The actors involved (may be nonhuman!)
  - Ways in which the actors interact: relationships, actions, use cases, etc.
- Example: ATM (thanks to ArgoUML)



Use case diagram: navigation

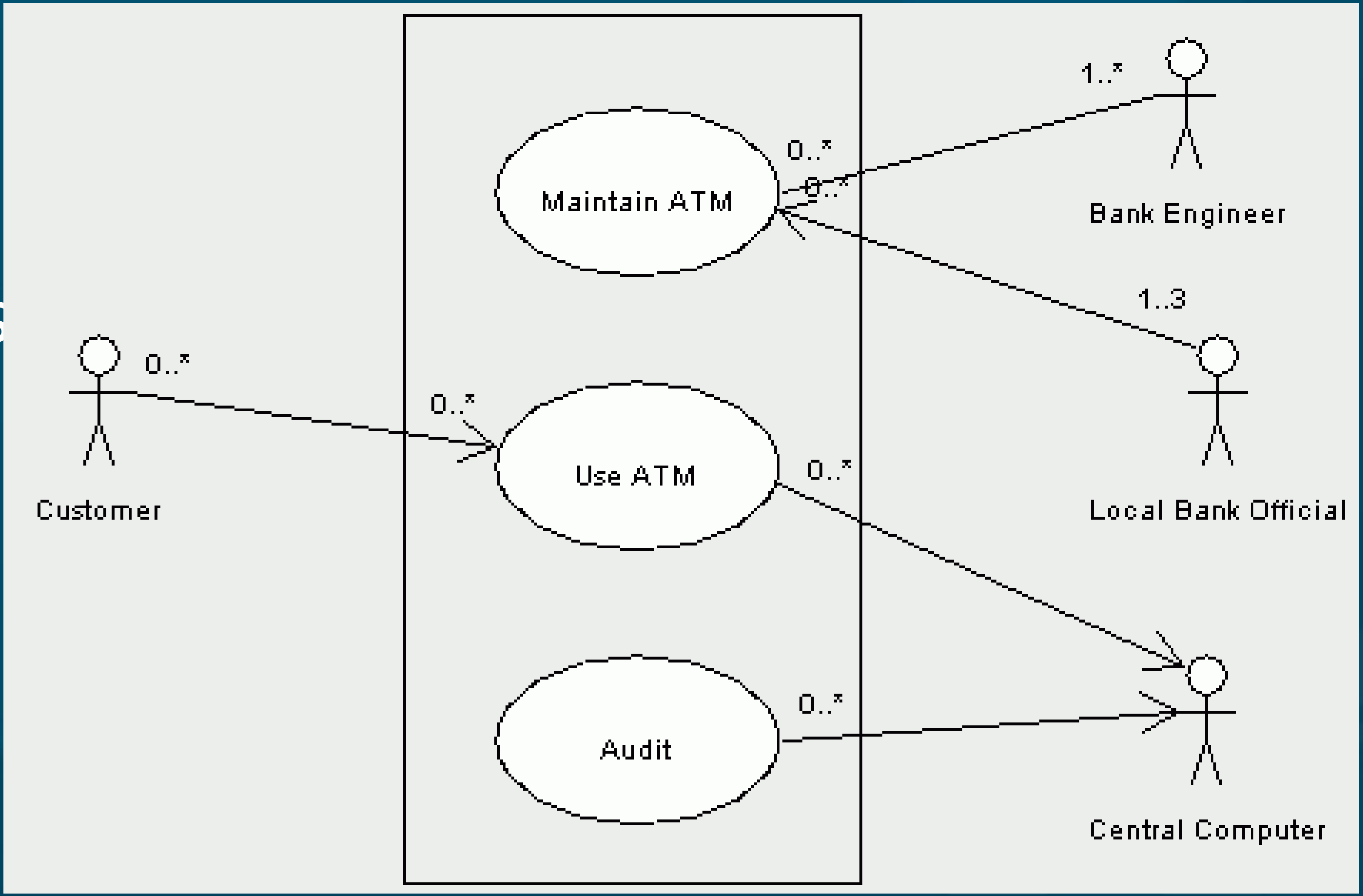
- Direction of arrows indicates which actor is passive and which is active:

- What direction should the arrows point between “Maintain ATM” and “Engineer”?



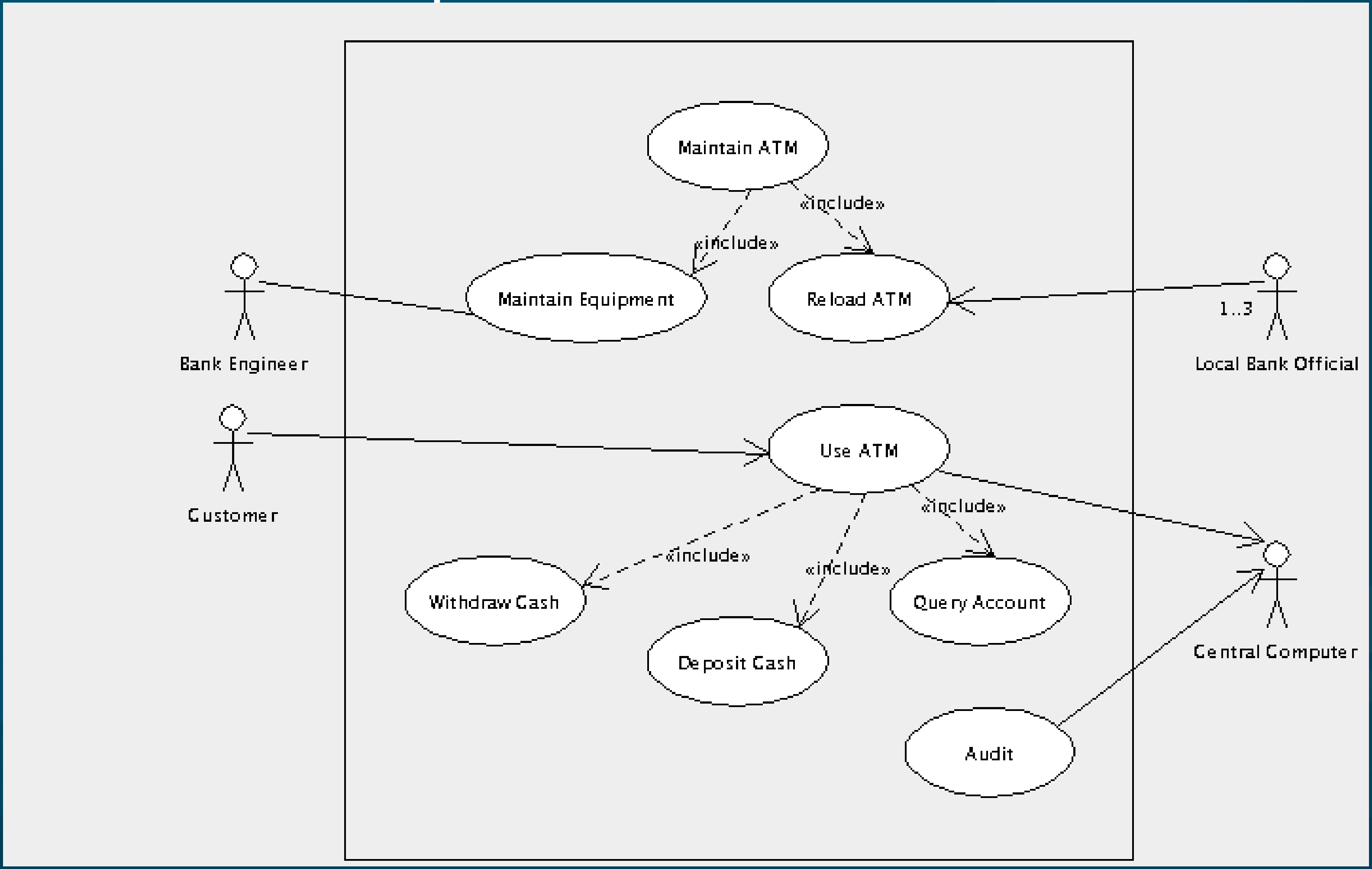
Use case diagram: multiplicity

- Numbers indicate how many instances of an actor can be doing how many instances of the use case
- e.g., only allow up to 3 Bank Officials



Use case diagram: includes

- We may need to break down each use case into smaller chunks to implement



Specifying a use case

- Each use case should have:
  - Short name
  - Goal: what does it achieve for its actors?
  - Names of actor(s) involved
  - Pre/post-conditions?
  - Basic flow: break down into steps (pseudocode!)
  - Alternate flows: what if user inputs something different from the usual?

Ex. use case: Withdraw Cash

- Name: Withdraw cash
- Goal: Customer gets cash; Computer ensures account has enough money and keeps a record
- Actors: Customer, Central Computer
- Basic flow:
  - Customer selects account to withdraw from
  - Customer inputs dollar amount of cash
  - ATM verifies with Computer enough money
  - ATM dispenses cash to Customer
  - ATM prints receipt

Ex. use case: alternate flows

- How might the basic flow not work? What might go wrong?
  - *input ($) too big or too small*
  - *can't give out coins (e.g., $4)*
  - *not enough $$$ in account*
  - *user cancels*
  - *no paper, or no cash, or ATM on fire*
  - *dropped connection to Computer*
- Each results in an alternate flow: how to handle that alternate situation

Example (midterm q. #10)

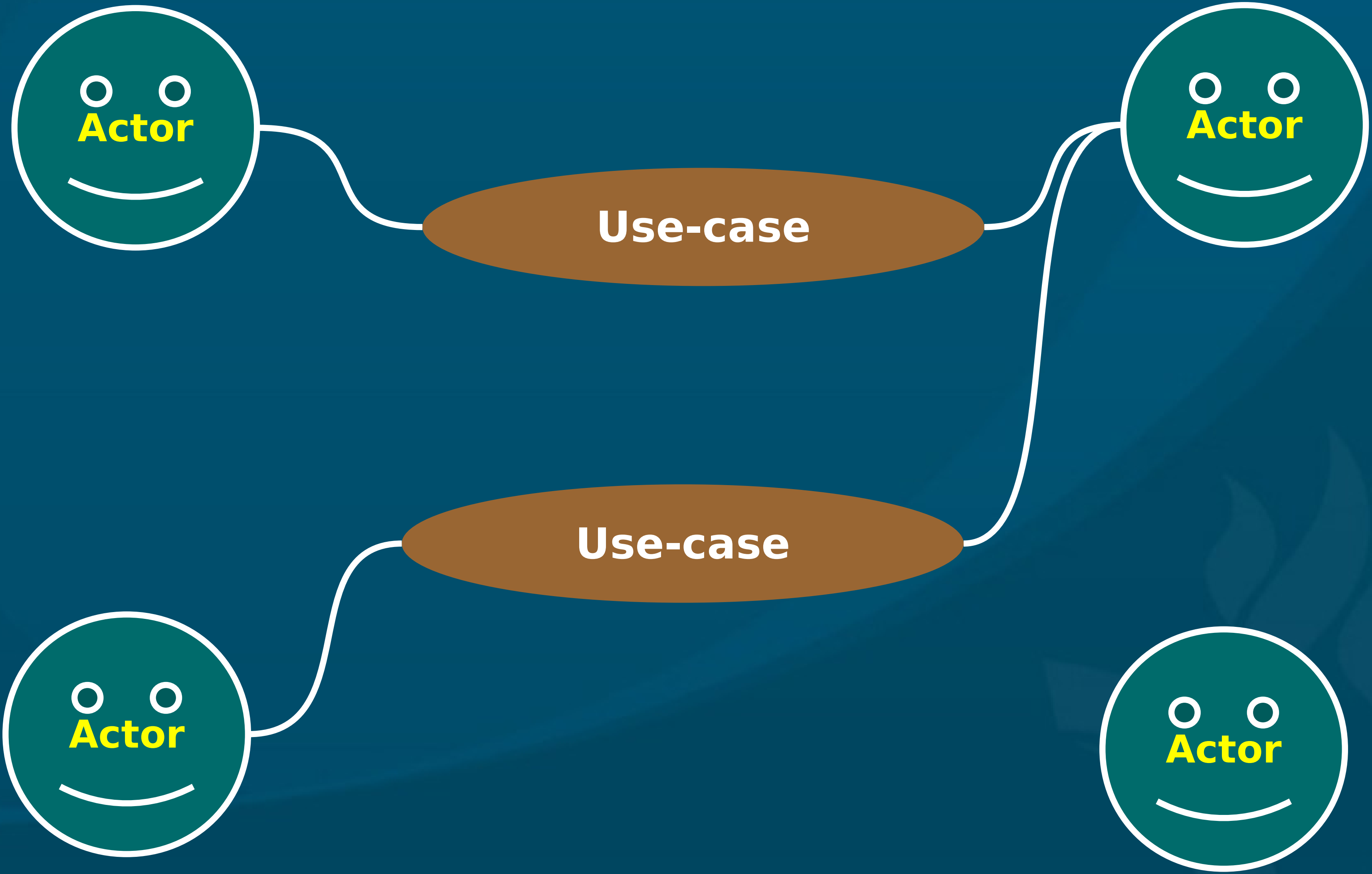
- Problem statement:

  Design a student enrolment database like we have at TWU

(1) Actors and actions

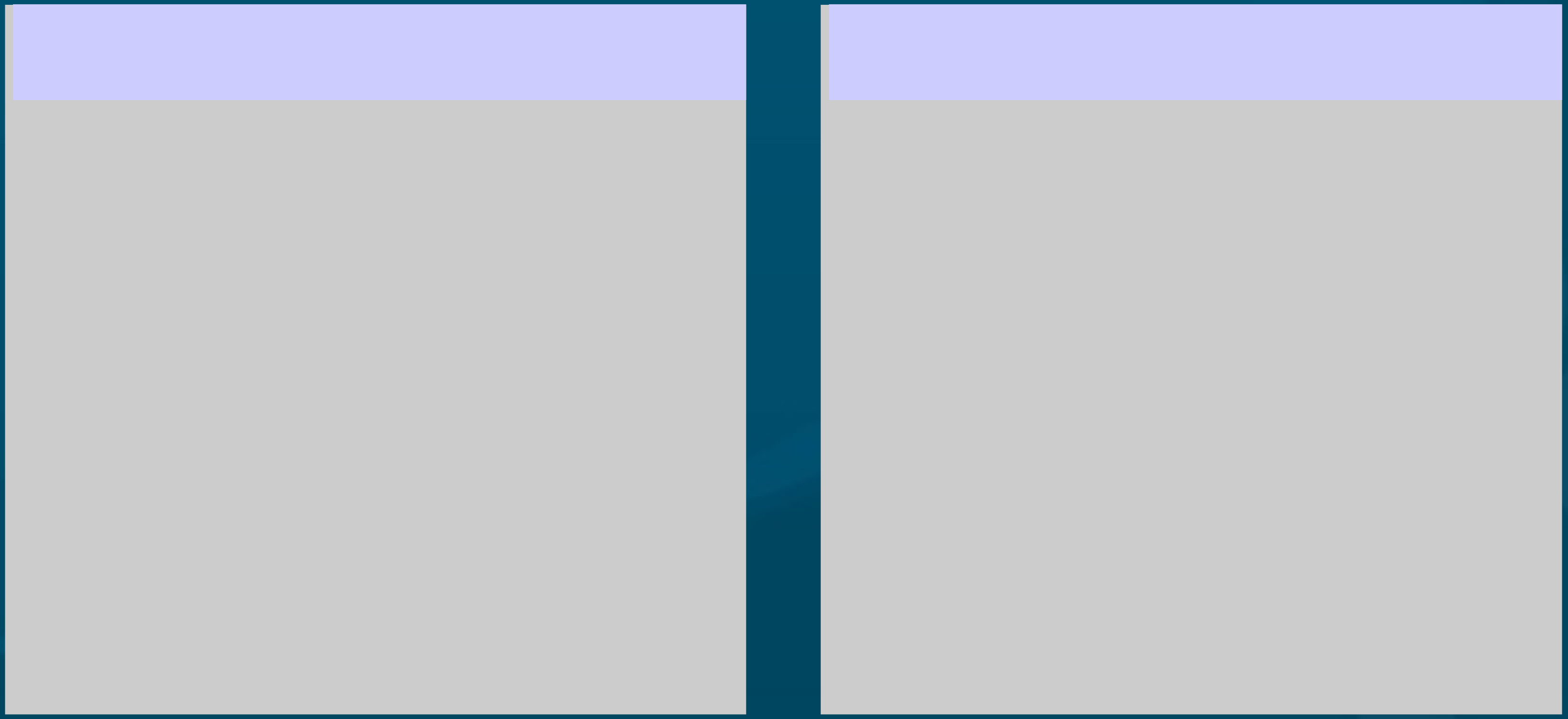
- Use-case scenarios: actors and actions
- Who are the actors? Who will interface with us?
  - *.Student, Alumni, Other students/public, Registrar, Database, Advisor, Instructor*
- What are the actions? Scenarios of use?
  - *ask for GPA, add class, drop class, change address, change advisor*

(1) Use-case diagrams



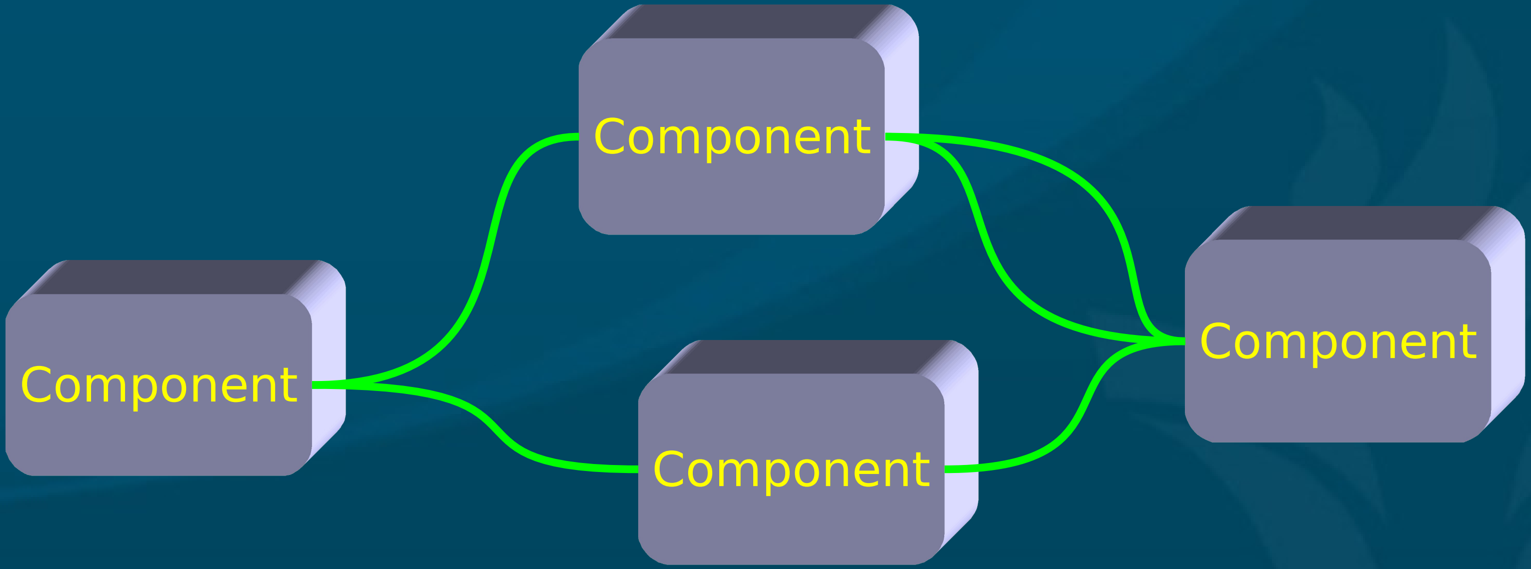
(1) UI mockup

- For each use-case (action), describe/mockup what the user interface will be like:
  - Text Q&A? Windows? Interactivity?



(2) Component design

- This is often the hardest part!
- Components need not be classes
- Thinly coupled: describe all interfaces between components



Component: ....

- Name: ...
- Description: ...
- Interface to (*component*):
    - ...
- Interface to (*component*):
    - ...

Component: ….

- Name: …
- Description: …
- Interface to (*component*):
    - …
- Interface to (*component*):
    - …

(3) From components to classes

- Each component may need several classes to implement it
- Component: …
    - Class: …
        - Attributes: …
        - Methods: …
    - Class: …
        - Attributes: …
        - Methods: …