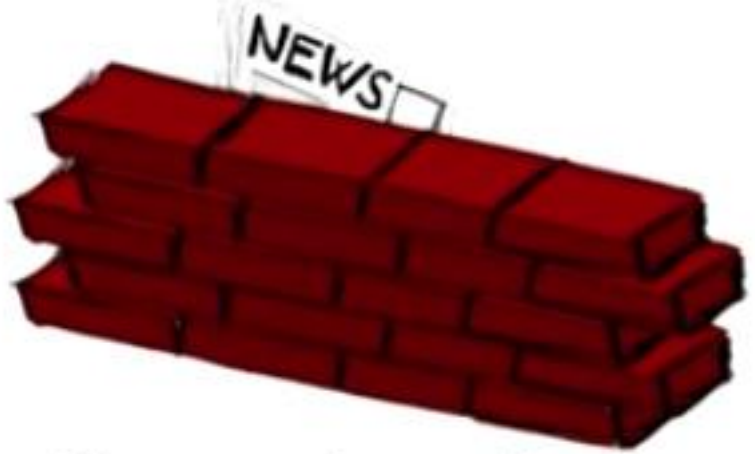
CSE3211-Software Engineering

Software Engineering and UML

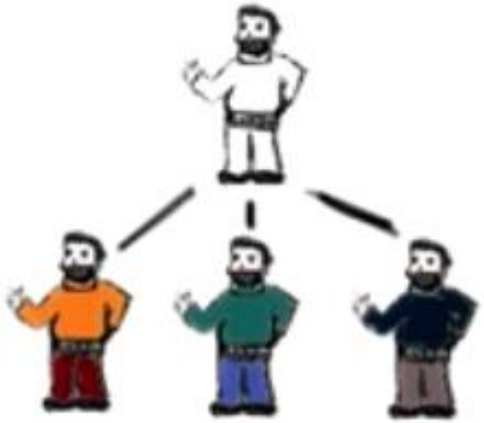
What is Object Orientation?

Data over function



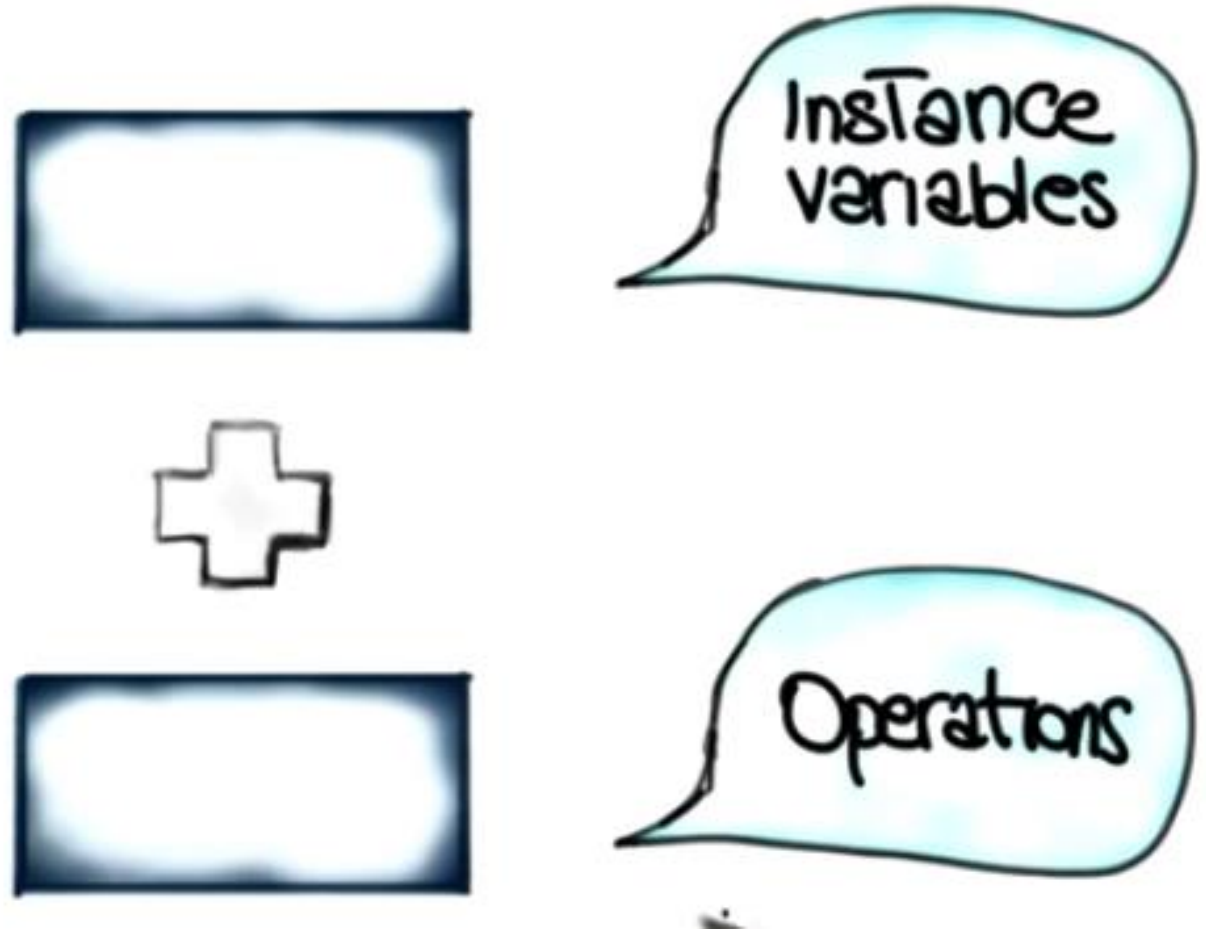
Information hiding

Encapsulation

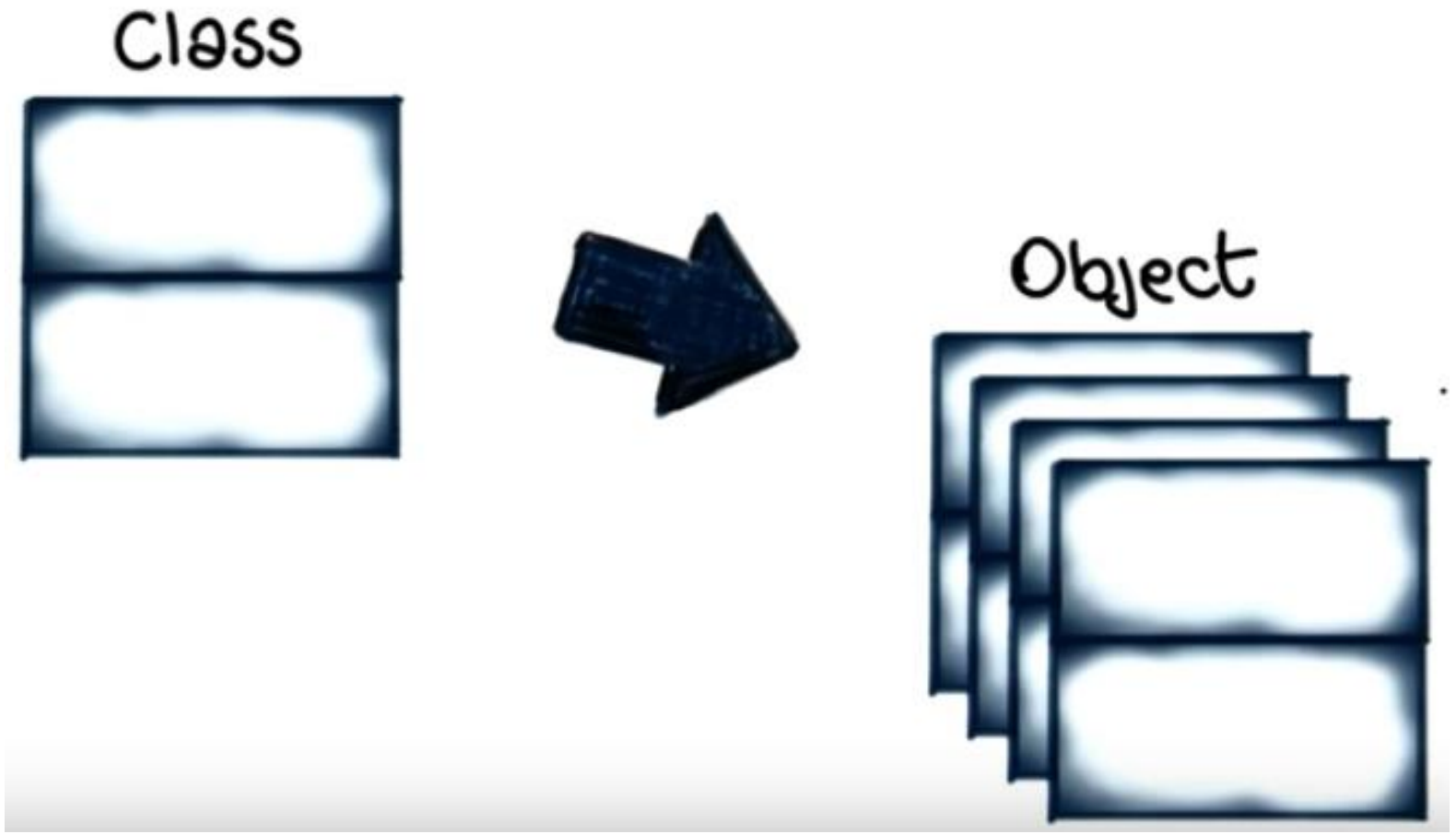


Inheritance

Objects and Classes



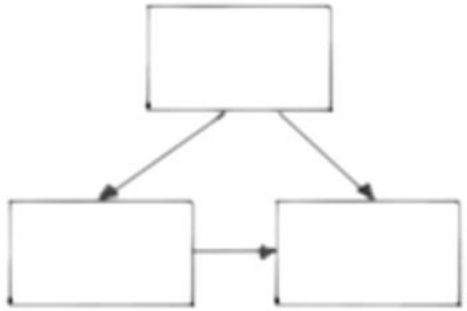
Objects and Classes



Why Use OO?

Reduce maintenance costs

Improve development process



Enforce good design

Acme Corporation decided to use an OO approach in its software development process. What benefits can they expect to receive from this decision?

[ ] Increased reuse because of the modular coding style

[ ] Increased maintainability because the system design can accommodate changes more easily

[ ] Increased speed because OO systems tend to run faster

[ ] Increased understandability because the design models real-world entities

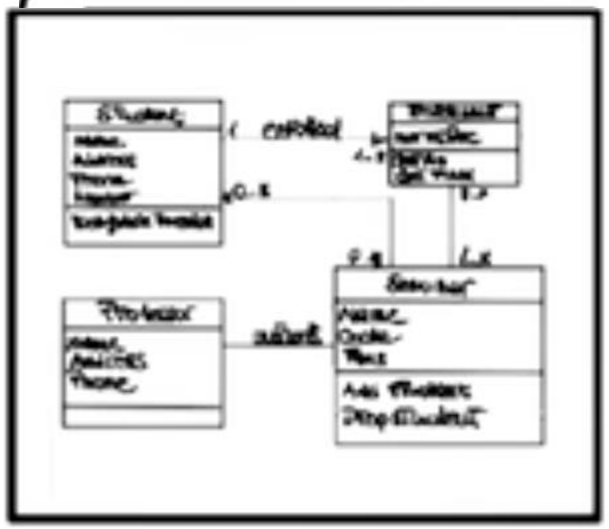
Acme Corporation decided to use an OO approach in its software development process. What benefits can they expect to receive from this decision?

- [x] Increased reuse because of the modular coding style
- [x] Increased maintainability because the system design can accommodate changes more easily
- [ ] Increased speed because OO systems tend to run faster
- [x] Increased understandability because the design models real-world entities

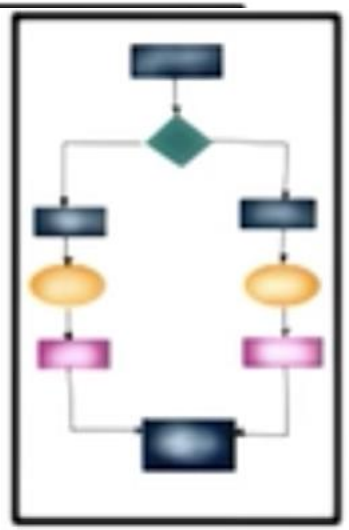
OOAD: OO Analysis and Design

Rumbaugh

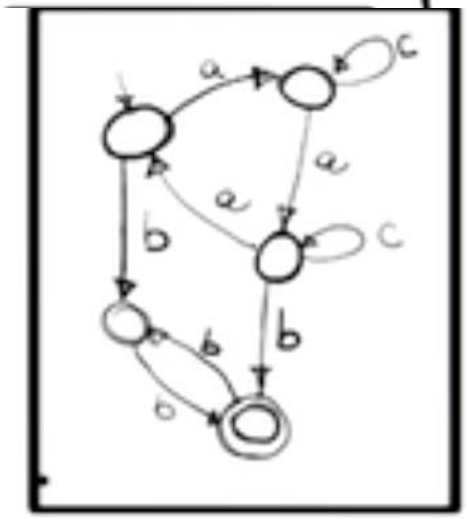
Object Modeling Technique (OMT)



Data



Functions

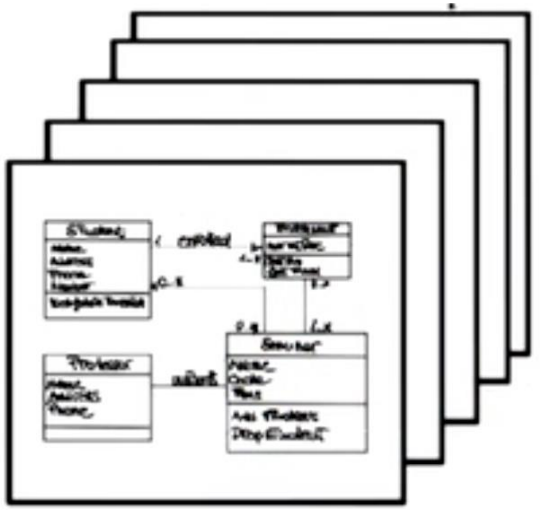


Control

Jacobson

Booch

Unified Modeling Language (UML)



OO Analysis

Functional oriented ⇒ data oriented

Real world objects ⇒ Requirements

obtain/prepare textual description of problem

underline nouns ⇒ classes

underline adjectives ⇒ attributes

underline active verbs ⇒ operations

Consider the following requirement for an online shopping website: "Users can add more than one item on sale at a time to a shopping cart"
Which of the following elements should be modeled as classes?

- [ ] Item
- [ ] Sale
- [ ] Shopping cart
- [ ] Time
- [ ] User

Consider the following requirement for an online shopping website: "Users can add more than one item on sale at a time to a shopping cart"
Which of the following elements should be modeled as classes?

- [x] Item
- [ ] Sale
- [x] Shopping cart
- [ ] Time
- [x] User

RUNNING EXAMPLE: COURSE MANAGEMENT SYSTEM

1 The Registration Manager sets up the curriculum for a semester using a scheduling algorithm

2 One course may have multiple course offerings

3 Each course offering has a number, location, and time

4 Students select 4 primary courses and 2 alternative courses by submitting a registration form

5 Students may use the system to add/drop courses for a period of time after registration

6 Professors use the system to receive their course offering rosters

7 Users of the registration system are assigned passwords which are used at logon validation

UNIFIED MODELING LANGUAGE

Structural Diagrams

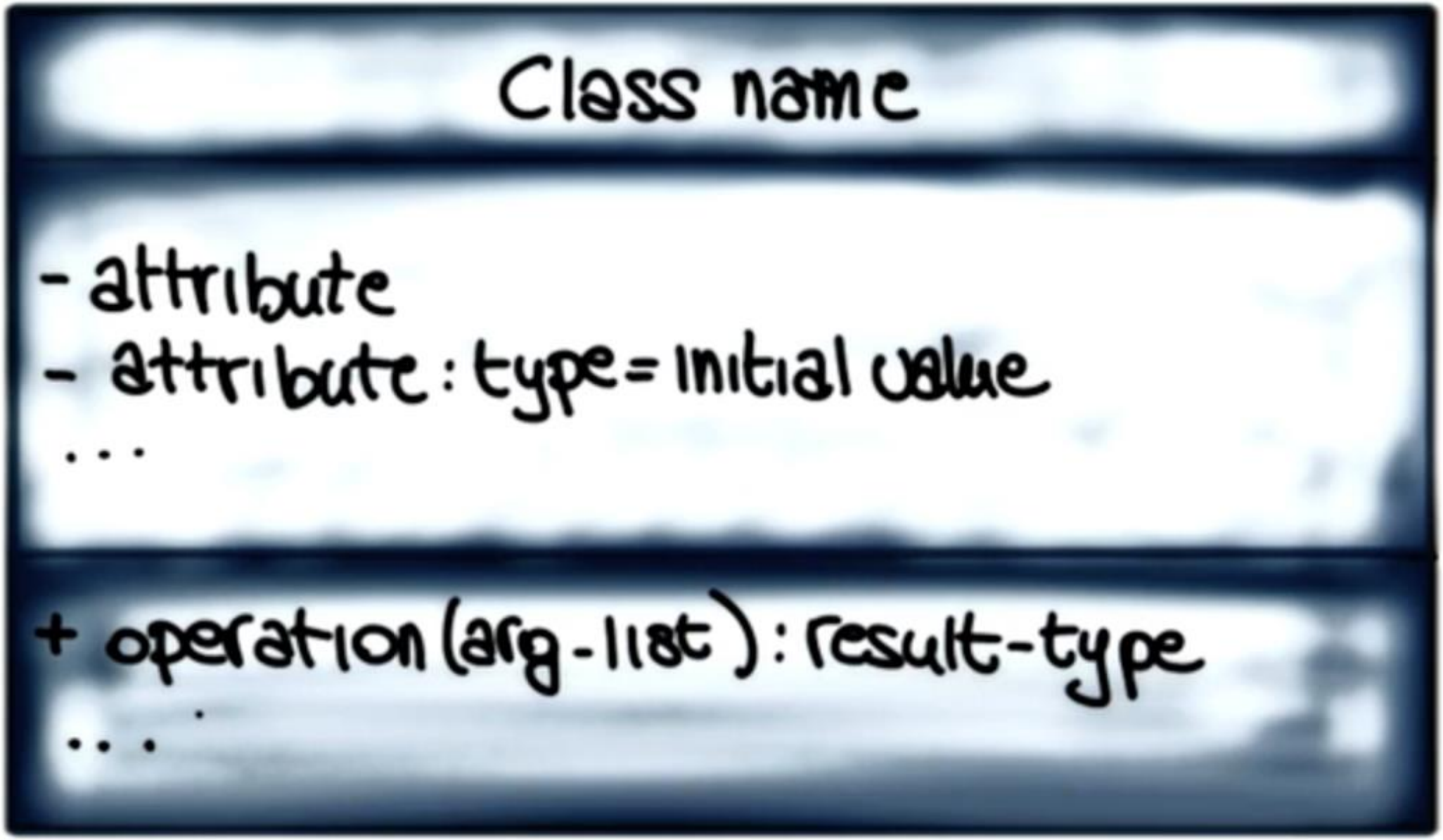
Class Diagram

Static, structural view of the system

Describes

- Classes and their structure
- Relationships among classes

Class Diagram: Class



Class Diagram for

COURSE MANAGEMENT SYSTEM

1 The Registration Manager sets up the curriculum for a semester using a scheduling algorithm

2 One course may have multiple course offerings

3 Each course offering has a number, location, and time

4 Students select 4 primary courses and 2 alternative courses by submitting a registration form

5 Students may use the system to add/drop courses for a period of time after registration

6 Professors use the system to receive their course offering rosters

7 Users of the registration system are assigned passwords which are used at logon validation

Class Diagram for our example

1. The Registration manager sets up the curriculum for a semester using scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offerings rosters
7. Users of the registration system are assigned password which are used at logon validation

Class Diagram for our example

- The **Registration manager** sets up the curriculum for a semester using **scheduling algorithm**
- One **course** may have multiple **course offerings**
- Each **course offering** has a number, location and time
- **Students** select 4 primary courses and 2 alternative courses by submitting a **registration form**
- Students may use the system to add/drop courses for a period of time after registration
- **Professors** use the system to receive their course offerings rosters
- Users of the registration system are assigned password which are used at logon validation

CLASS DIAGRAM FOR OUR EXAMPLE

Class Diagram: Attribute

Represent the structure of a class

May be found by

- examining class definitions
- studying requirements
- applying domain knowledge

Class Diagram for our example

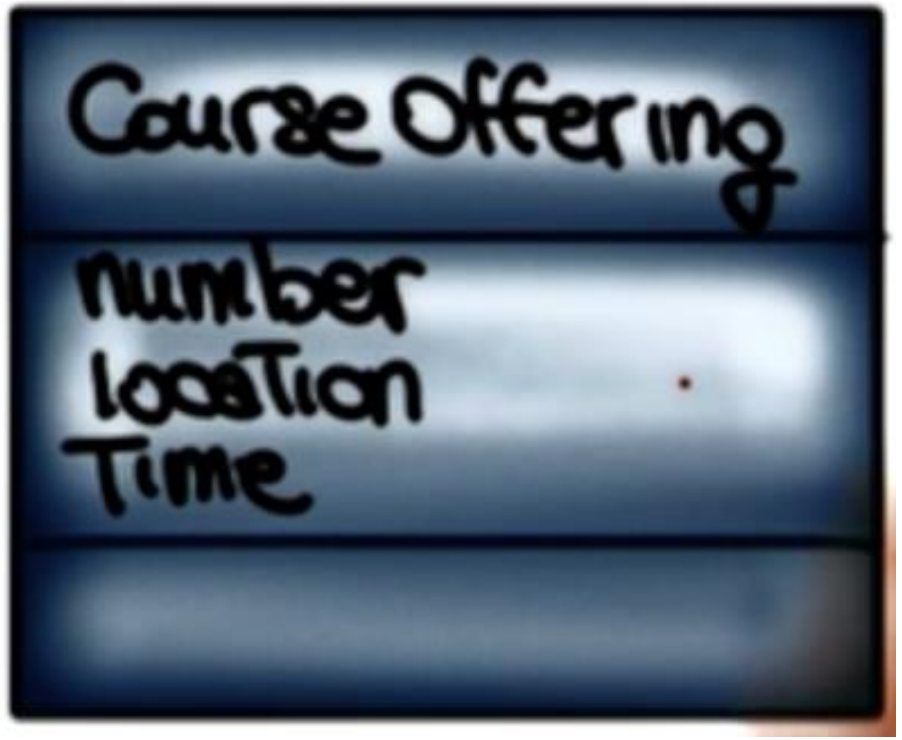
1. The Registration manager sets up the curriculum for a semester using scheduling algorithm
2. One course may have multiple course offerings
3. Each **course offering** has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offerings rosters
7. Users of the registration system are assigned password which are used at logon validation

Class Diagram: Attribute

Represent the structure of a class

May be found by

- examining class definitions
- studying requirements
- applying domain knowledge



Each **course offering** has
a number,
location and
time

Class Diagram: Operations

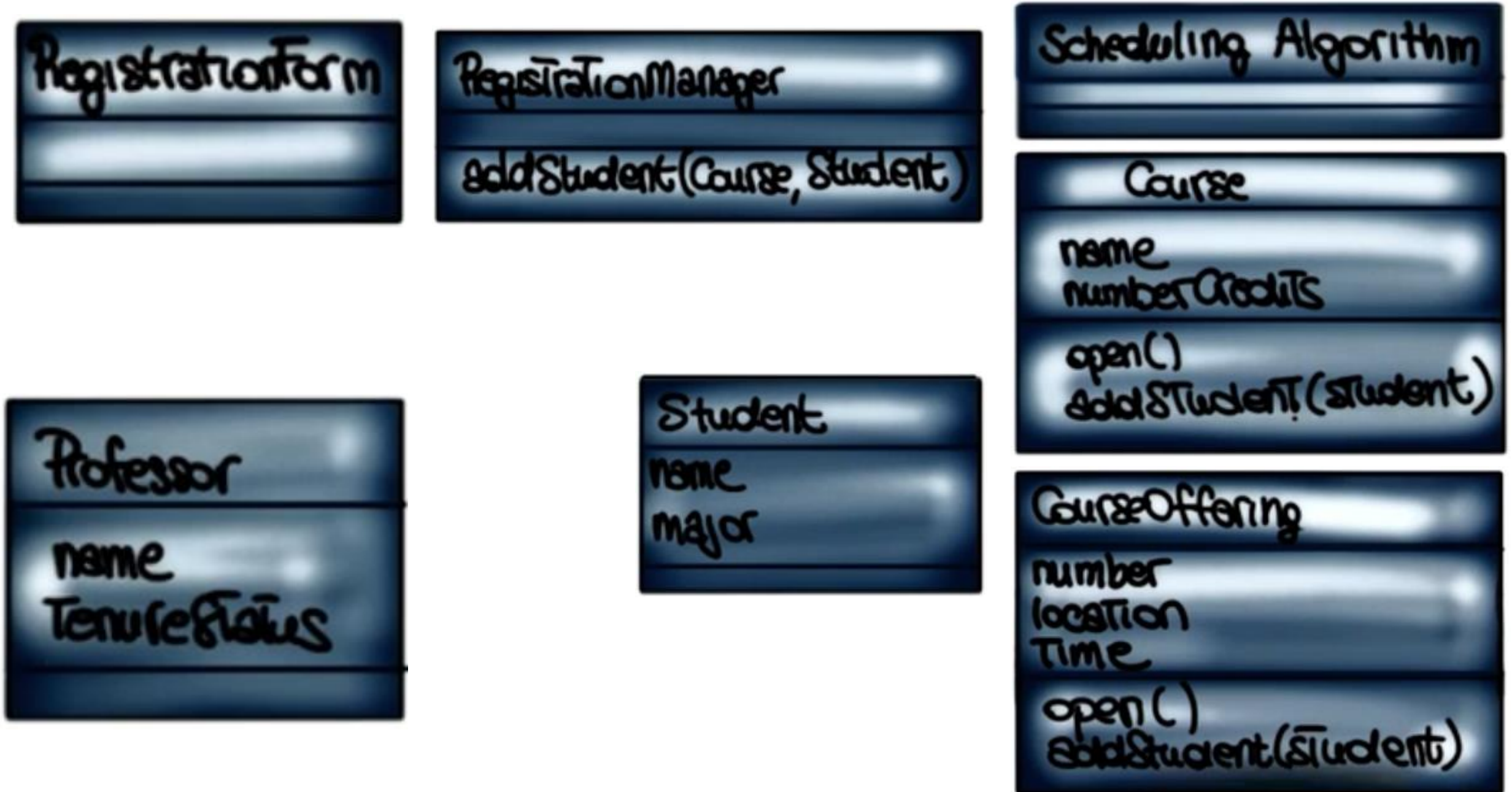
Represent the behavior of a class

May be found by examining interactions among entities

Class Diagram for our example

1. The Registration manager sets up the curriculum for a semester using scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. **Students may use the system to add/drop courses for a period of time after registration**
6. Professors use the system to receive their course offerings rosters
7. Users of the registration system are assigned password which are used at logon validation

Class Diagram for our example

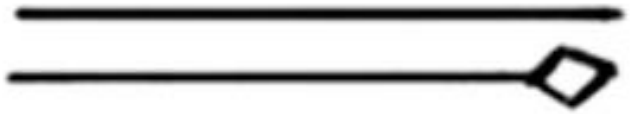


Class Diagram: Relationships

Describe interections between objects

Dependencies : X uses Y

Associations/ Aggregations : X has a Y



Generalization : X is a Y

Which of the following relationships is an actual relationship for the system we are modeling?

[ ] RegistrationManager uses SchedulingAlgorithm (dependency)

[ ] RegistrationManager uses Student (dependency)

[ ] Student uses RegistrationManager (dependency)

[ ] Student registers for CourseOffering (association)

[ ] Student consists of CourseOffering (aggregation)

[ ] Course consists of CourseOffering (aggregation)

[ ] CourseOffering is a Course (generalization)

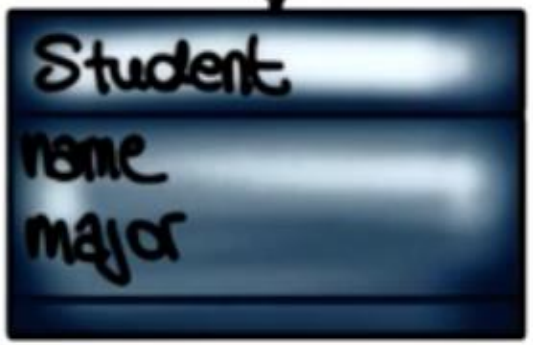
[ ] Student is a RegistrationUser (generalization)

[ ] Professor is a Registration user (generalization)

Which of the following relationships is an actual relationship for the system we are modeling?

- [x] RegistrationManager uses SchedulingAlgorithm (dependency)
- [x] RegistrationManager uses Student (dependency)
- [ ] Student uses RegistrationManager (dependency)
- [x] Student registers for CourseOffering (association)
- [ ] Student consists of CourseOffering (aggregation)
- [x] Course consists of CourseOffering (aggregation)
- [ ] CourseOffering is a Course (generalization)
- [x] Student is a RegistrationUser (generalization)
- [x] Professor is a Registration user (generalization)

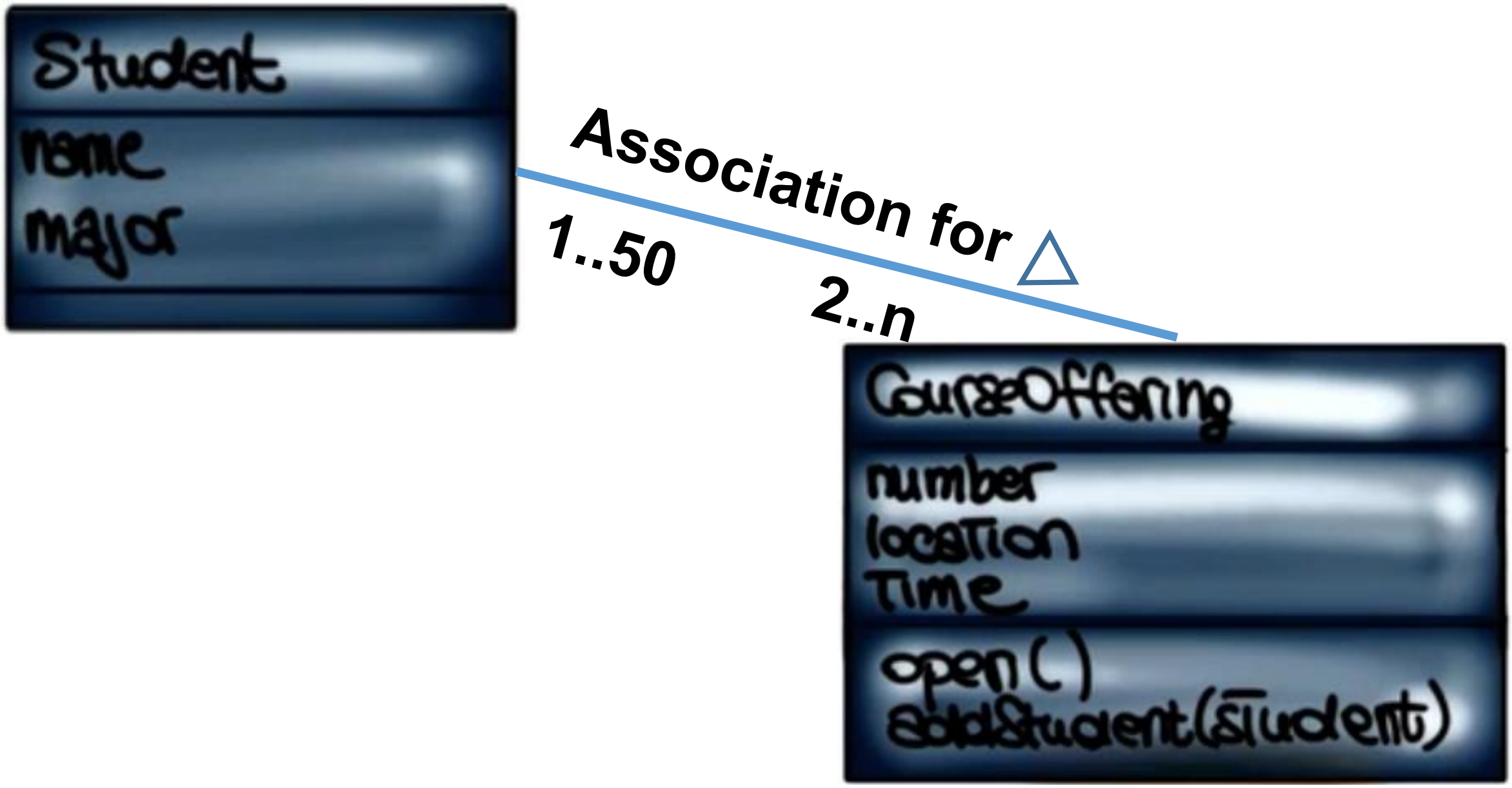
Dependency Example



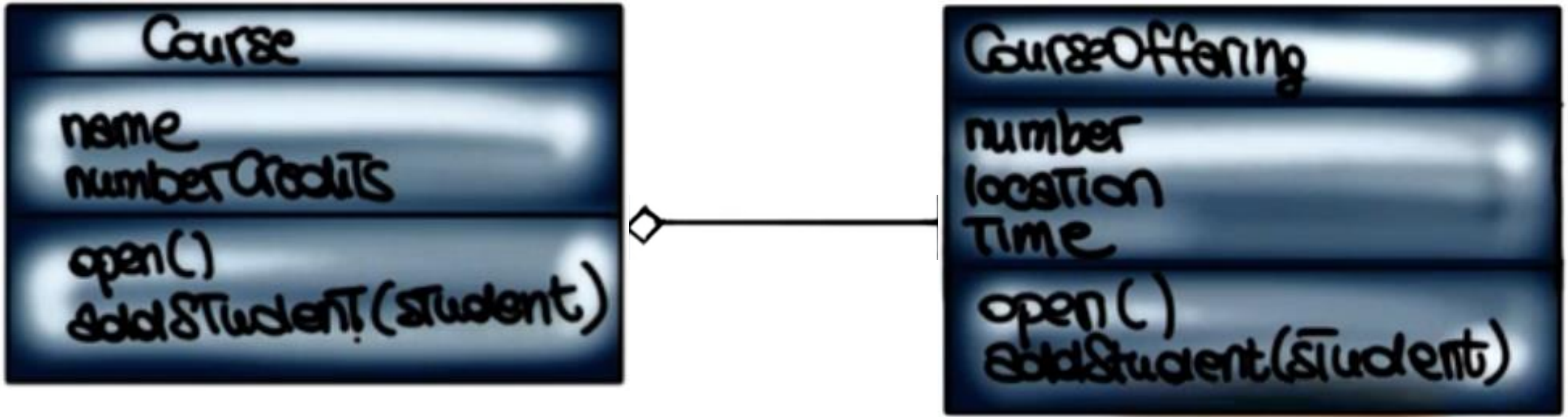
RegistrationManager gets a Student object as a parameter

RegistrationManager uses SchedulingAlgorithm in its implementation

Association Example

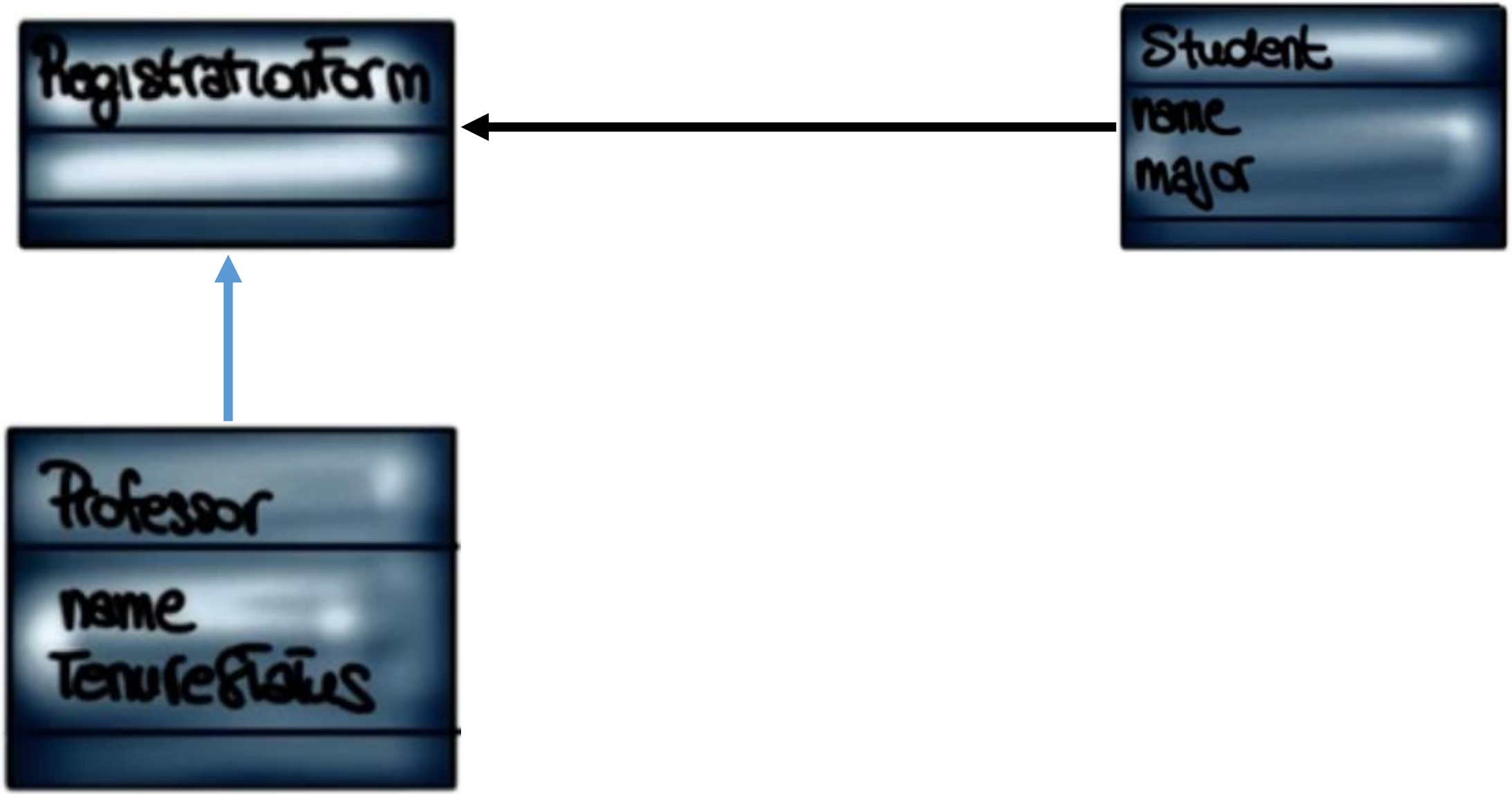


Aggregation Example



A Course consists of multiple CourseOfferings.

Generalization Example



Class Diagrams: Creation Tips

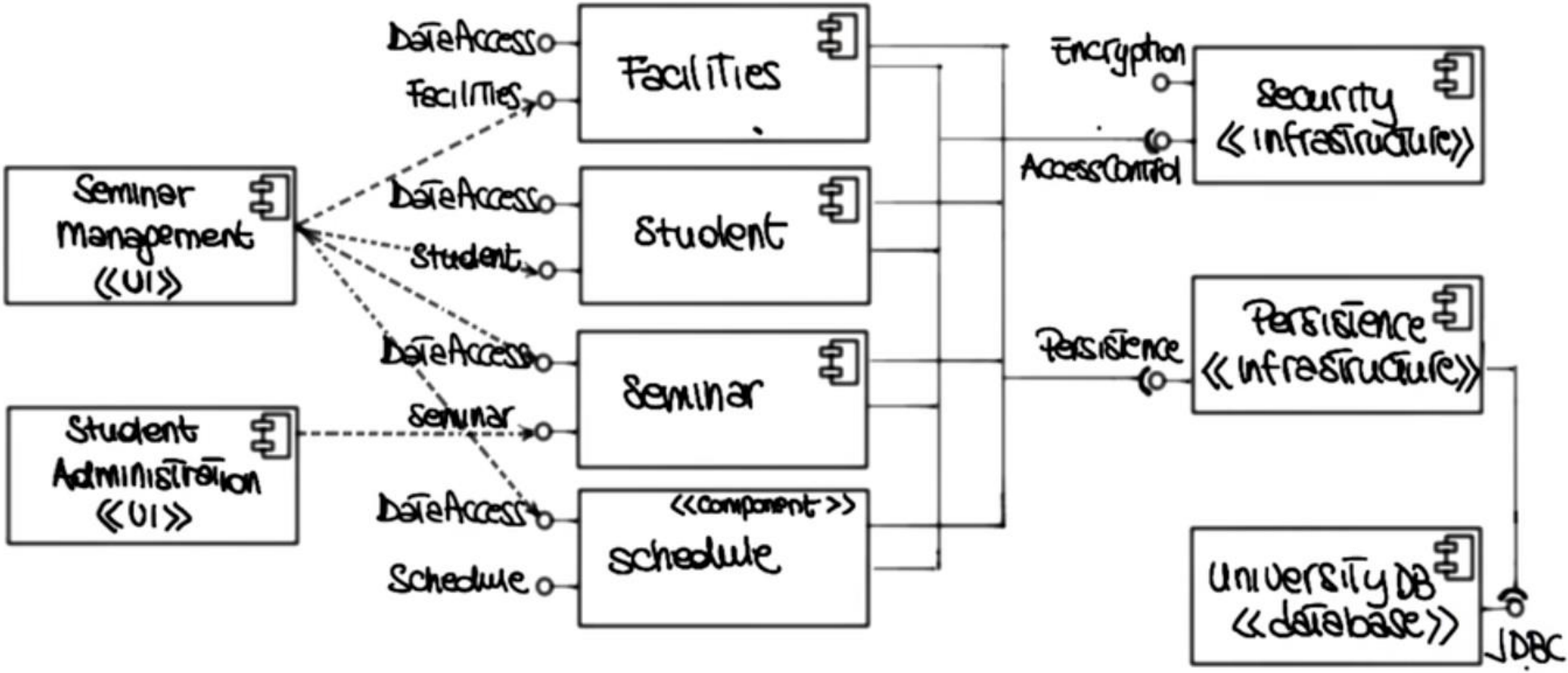
- Understand the Problem
- Choose good class names
- Concentrate on the WHAT
- Start with a simple diagram
- Refine until you feel it is complete

Component Diagrams

- Static view of components and their relationships
- Node = Component Set of classes with a well-defined interface
- Edge = Relationship

  ”Uses services of”
- Can be used to represent and architecture

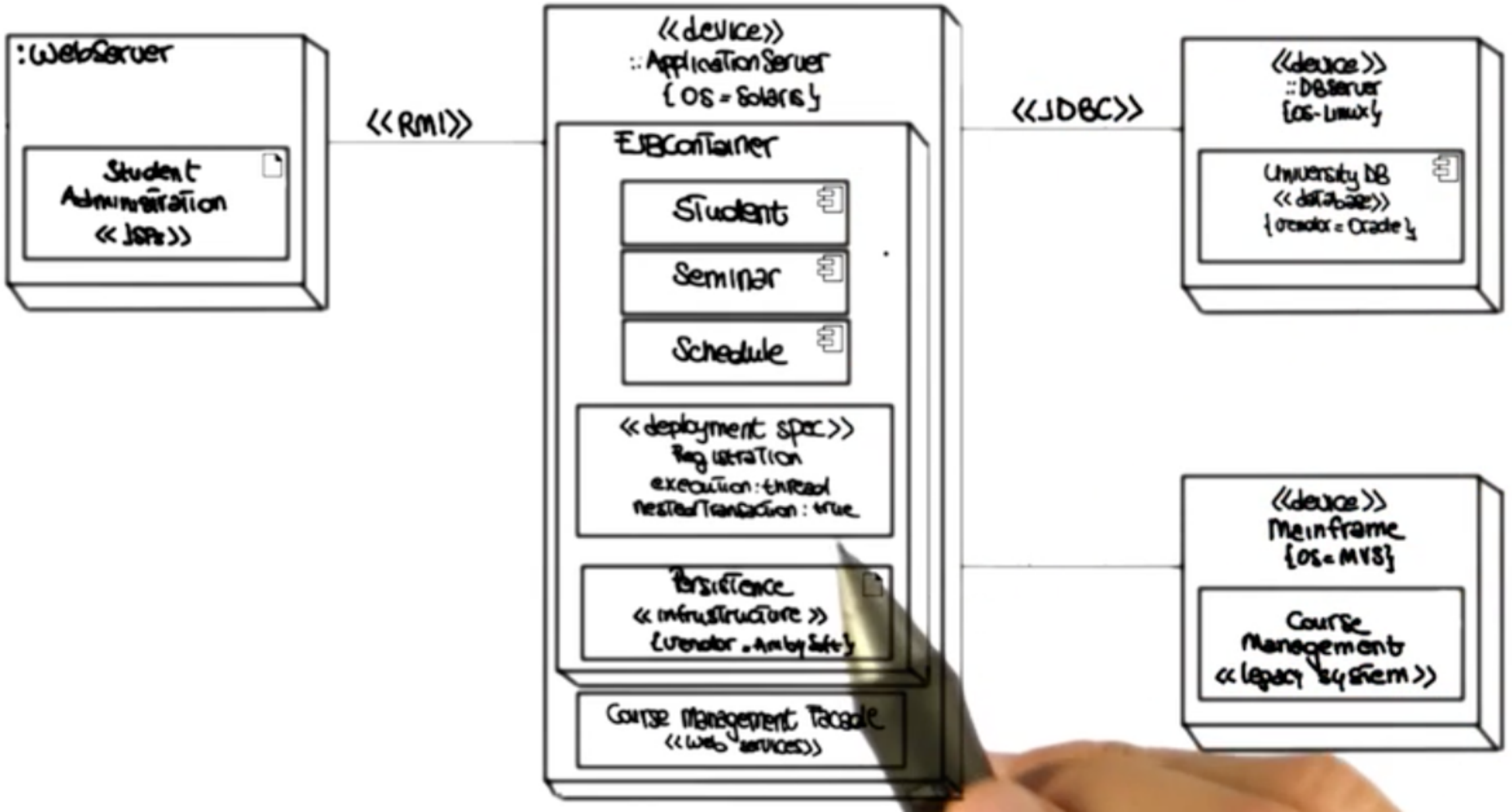
Component Diagrams Example



Deployment Diagram

- Static deployment view of a system
- Physical allocation of components to computational units
- Node = Computational unit
- Edge = Communication

Deployment Diagram Example



UNIFIED MODELING LANGUAGE

Behavioral Diagrams

Use Case

Describes the outside view of the system

- Sequence of interactions of outside entities (actors) with the system
- System actions that yields an observable result of value to the actors
- AKA Scenarios, scripts or user stories

Use Case: BASIC NOTATION

Use case

Actor

Is the actor of

Use Case: ACTOR

Entity: human or device

Plays a role

- An entity can play more than one role
- More than one entity can play the same role

May appear in more than one use case

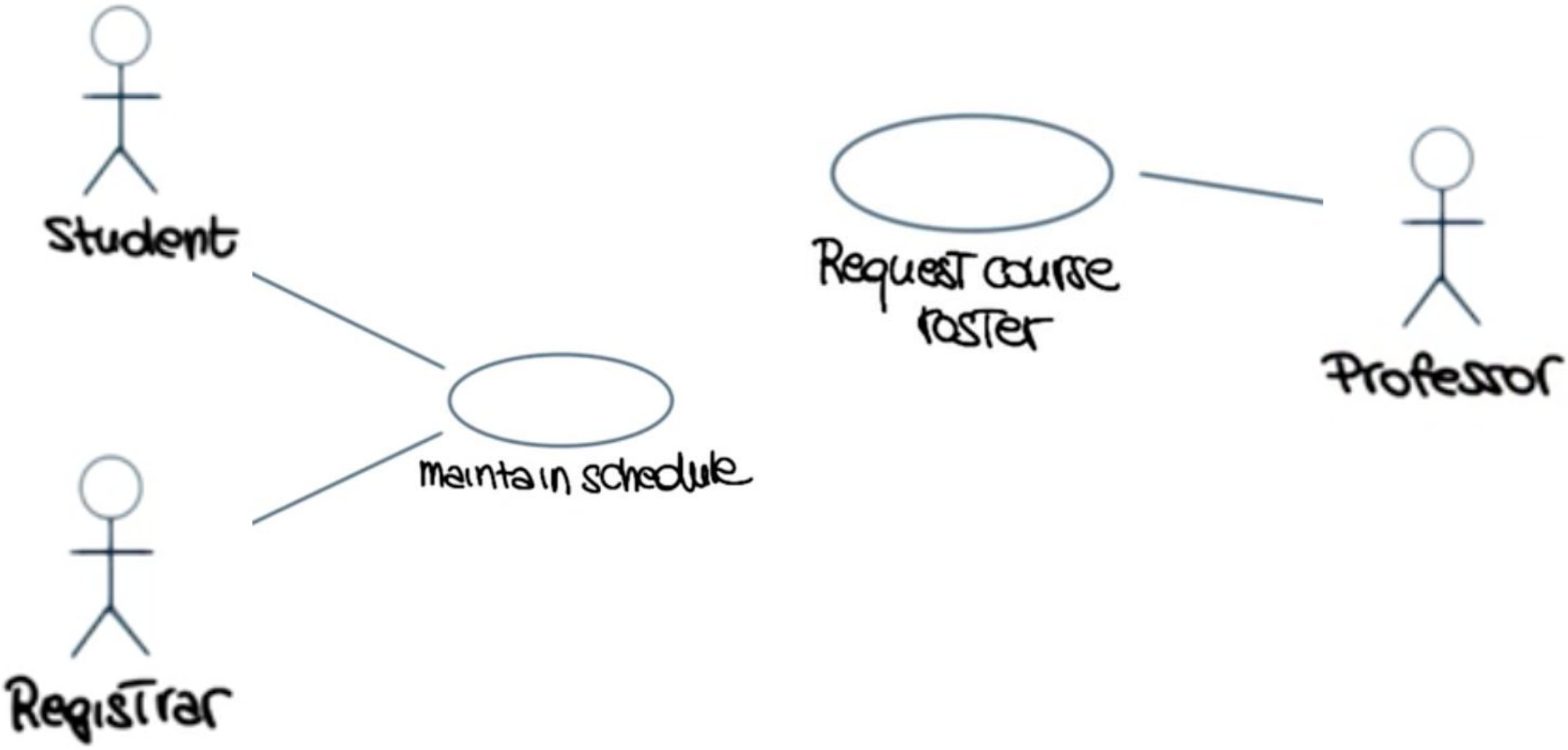
ACTORS for our example

1. The Registration manager sets up the curriculum for a semester using scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offerings rosters
7. Users of the registration system are assigned password which are used at logon validation

ACTORS for our example

1. The Registration manager sets up the curriculum for a semester using scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offerings rosters
7. Users of the registration system are assigned password which are used at logon validation

Use Case Diagram for our example



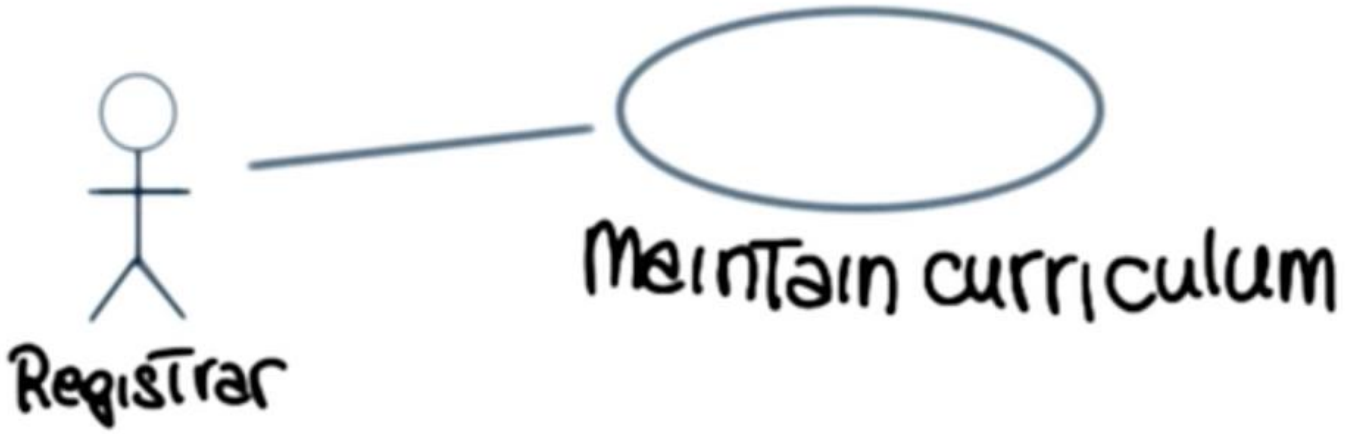
Documenting USE CASES

The behavior of a use case can be specified by describing its flow of events (formal or informal)

- How the use cases starts and ends
- Normal flow of events
- Alternative flow of events
- Exceptional flow of events

Use Case Diagram for our example

Let’s define the flow of events for



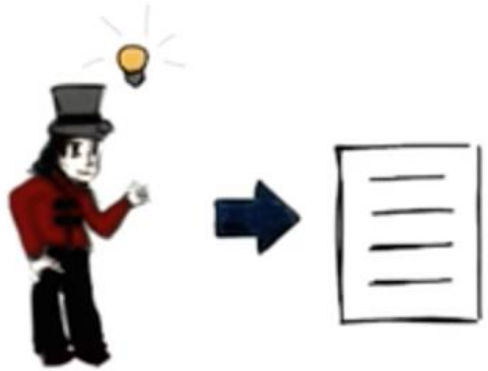
Maintain Curriculum USE CASE: Informal Paragraph

1. The Registration manager sets up the curriculum for a semester using scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offerings rosters
7. Users of the registration system are assigned password which are used at logon validation

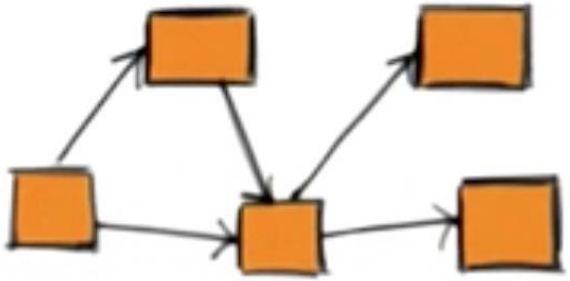
Maintain Curriculum USE CASE: Informal Paragraph

1. Registrar logs onto the system and enters password
2. If password is valid, system asks to specify a semester
3. Registrar enters the described semester
4. System prompts the Registrar to select the desired activity: ADD, DELETE, REVIEW, or QUIT
5. If Registrar selects ADD, System allows Registrar to add a course to Course list for selected semester
6. If Registrar selects DELETE, System allows Registrar to delete a course from the Course List for selected semester
7. If Registrar selects Review, System display course information in the course List for selected semester
8. If Registrar selects QUIT, System exits (Use case ends)

Role of USE CASES



Requirements elicitation



Architectural analysis

User prioritization

Planning

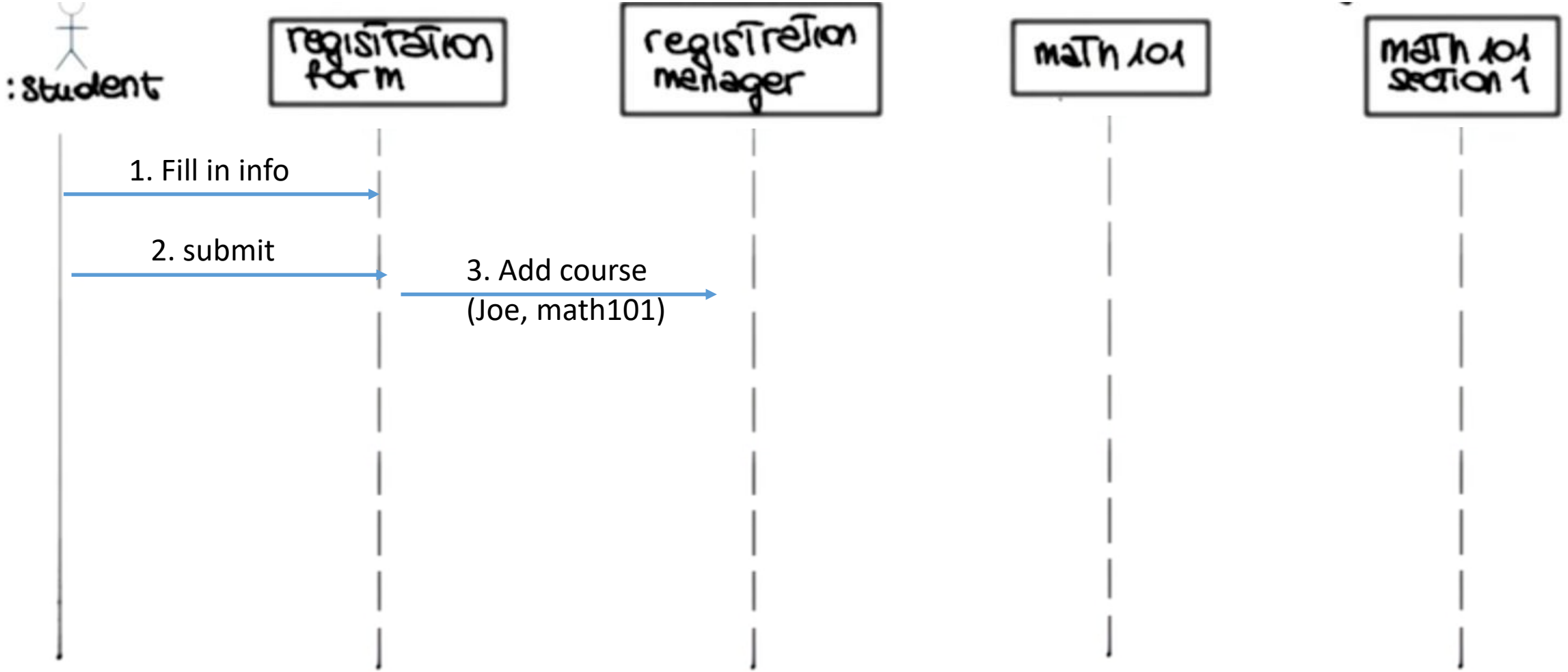
Testing

USE CASE Diagram: Creation Tips

- Use name that communicates purpose
- Define one atomic behavior per use case
- Define flow of events clearly
- Provide only essential details
- Factor common behaviors
- Factor variants

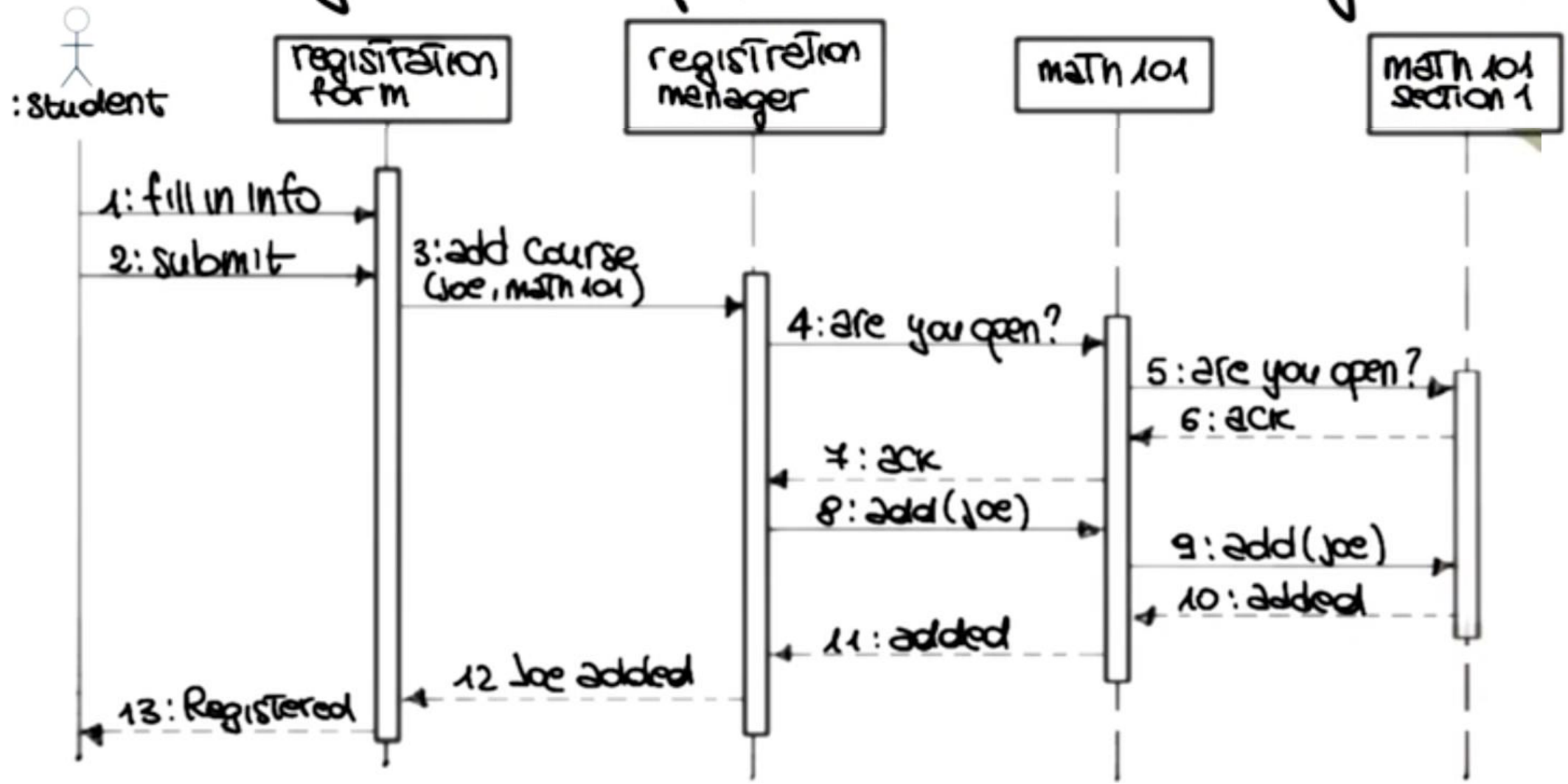
Sequence Diagram

Interaction diagram that emphasizes the time ordering of messages



SEQUENCE DIAGRAM

Interaction diagram that emphasizes the time ordering of messages

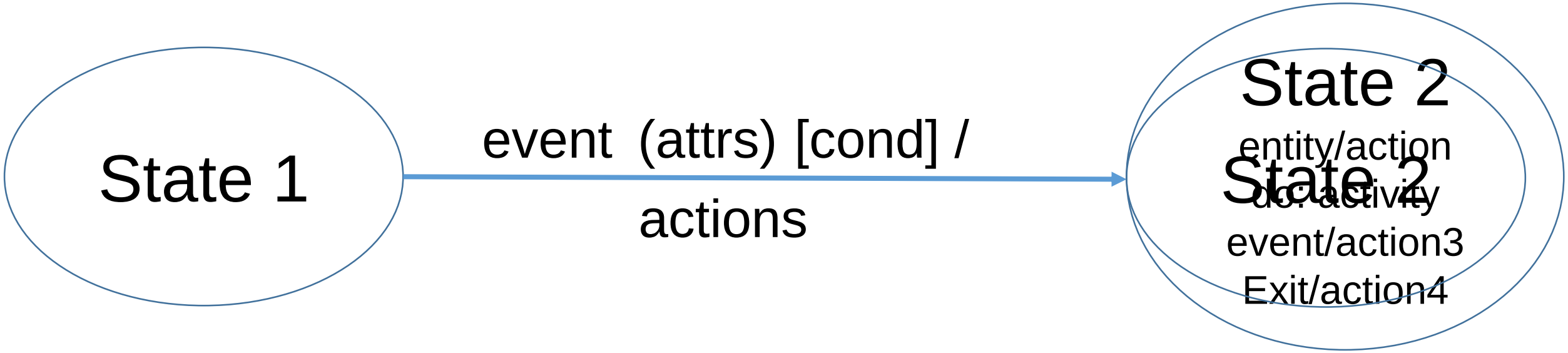


State Transition Diagram

For each relevant class

- Possible states of the class
- Events that cause a transition from one state to another
- Action that result from a state change

State Transition Diagram: Notation



STATE TRANSITION DIAGRAM FOR (PART OF) OUR EXAMPLE

An UML state transition diagram specifies :

[ ] A set of objects that work together to perform some action

[ ] The events that cause an object to move from one state to another

[ ] The set of components in a system

[ ] The effects of a state change

An UML state transition diagram specifies :

[ ] A set of objects that work together to perform some action

[X] The events that cause an object to move from one state to another

[ ] The set of components in a system

[X] The effects of a state change

Which of the following diagrams are UML Structural Diagrams?

[ ] Use case diagram

[ ] Class diagram

[ ] Deployment diagram

[ ] Sequence diagram

Which of the following diagrams are UML Structural Diagrams?

[ ] Use case diagram

[x] Class diagram

[x] Deployment diagram

[ ] Sequence diagram

Q/A?