UML Sequence Diagrams

Behavioral Modeling in UML

- Class Diagrams are used to model the static structure of a system
    - The entities in the system and the connections between them
- In addition to static structure, a system also has dynamic behavior
    - The system must DO something to be useful
    - How the objects in the system interact at runtime
        - When and by whom are the various objects created?
        - What is the message flow between the various objects?
    - What are the Algorithms?

Sequence Diagrams

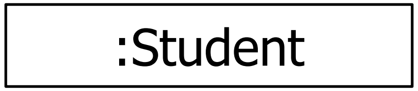
- Sequence Diagrams are used to show how messages flow between objects

- They provide one possible representation for algorithmic behavior
  - Pseudo code is another

- Sequence Diagrams contain:
  - Objects (instances)
  - Lifelines
  - Messages

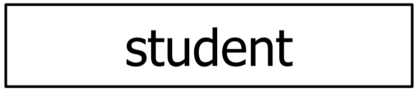
Representing an Object

- Objects are represented by a box and a lifeline
  - Object can be anonymous



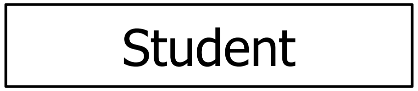
:Student

  - Can also omit the type

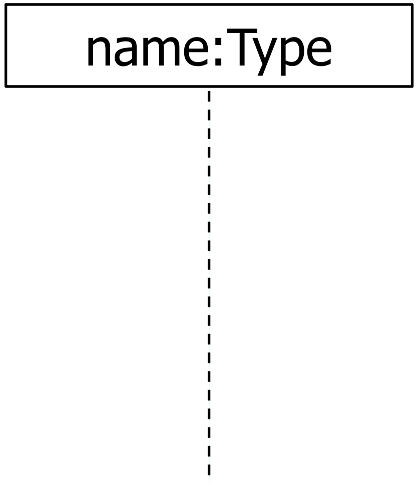


student

- In the case of invocation of static methods



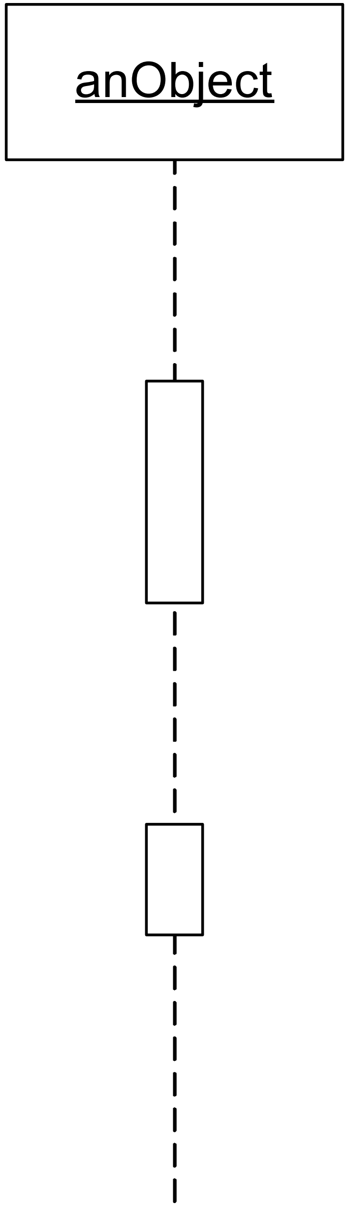
Student



name:Type

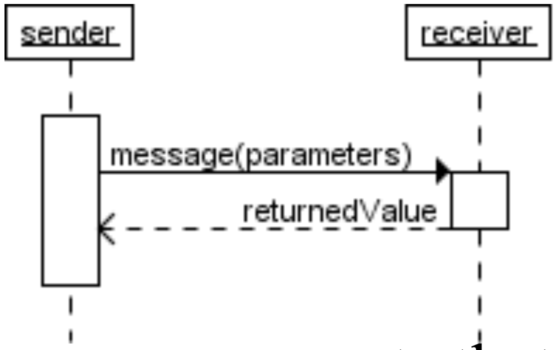
Object Lifelines

- Time proceeds from top to bottom
- Dashed line represents the lifetime of the object (the time during which the object exists)
- Activation boxes show when the object is active (i.e., executing an operation)



Messages

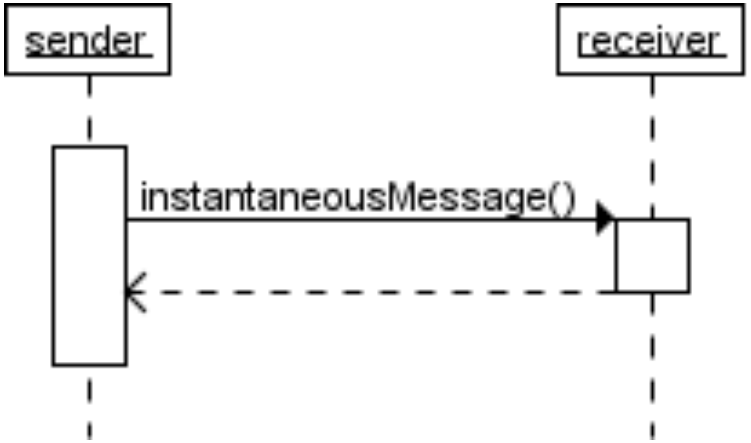
- Synchronous Message
  - Represented by a → between two objects



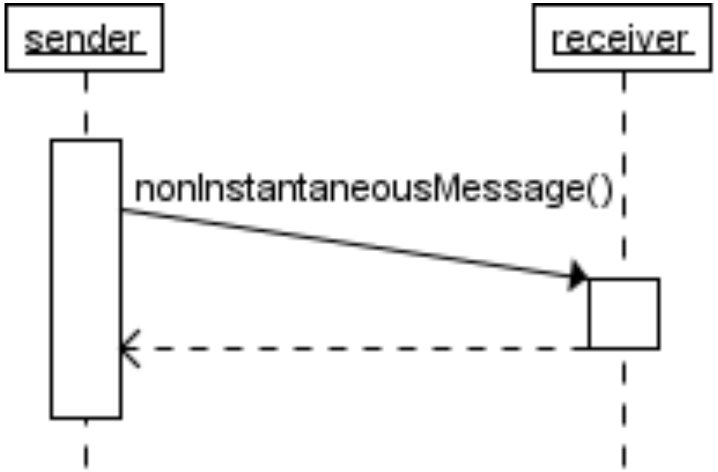
  - The dashed arrow represents that receiver has finished to process the message
  - This line is optional:
    - Should only use it when there is a return value
    - Otherwise, hide it to keep the diagram simple

Messages - II

- Instantaneous message

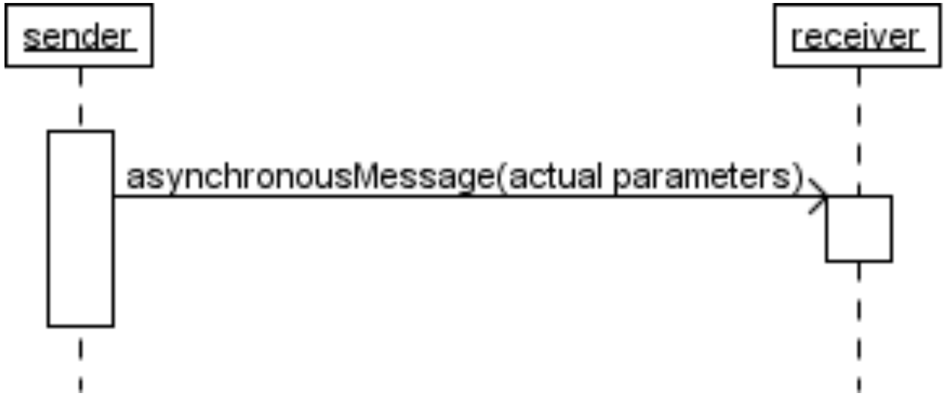


- Non-instantaneous message



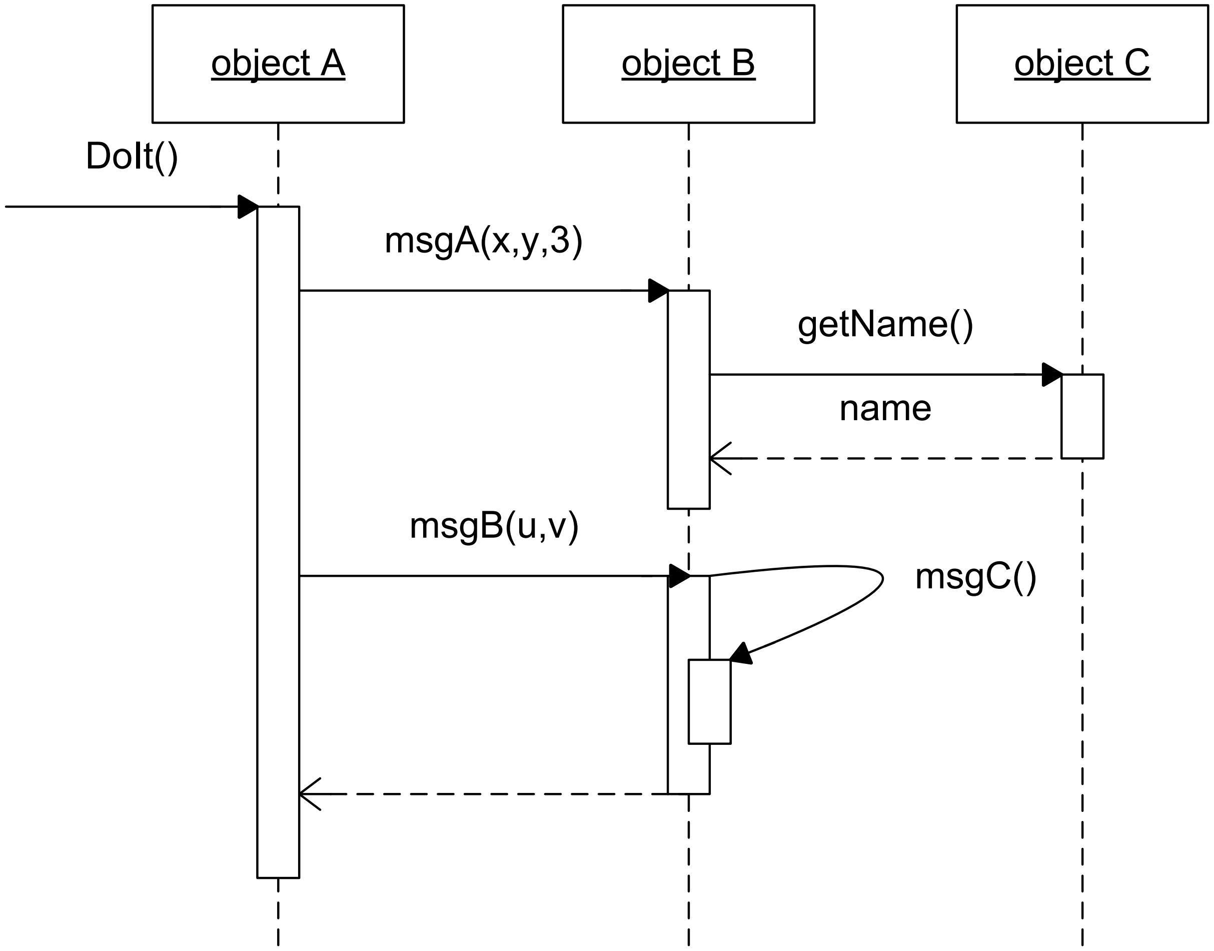
Messages - III

- Asynchronous message

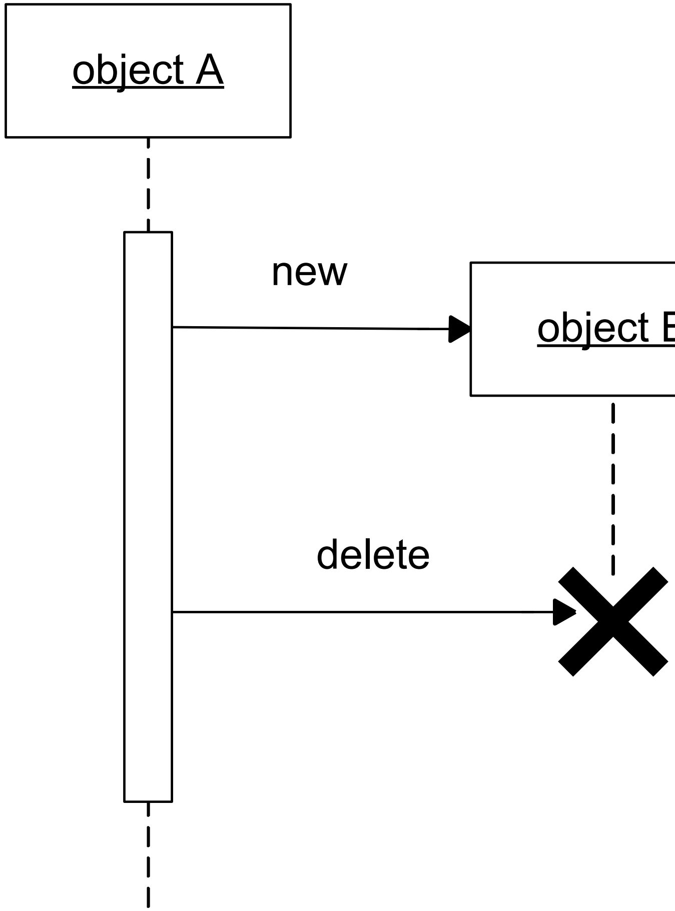


- And messages to itself?
- And callback messages?

Message Passing



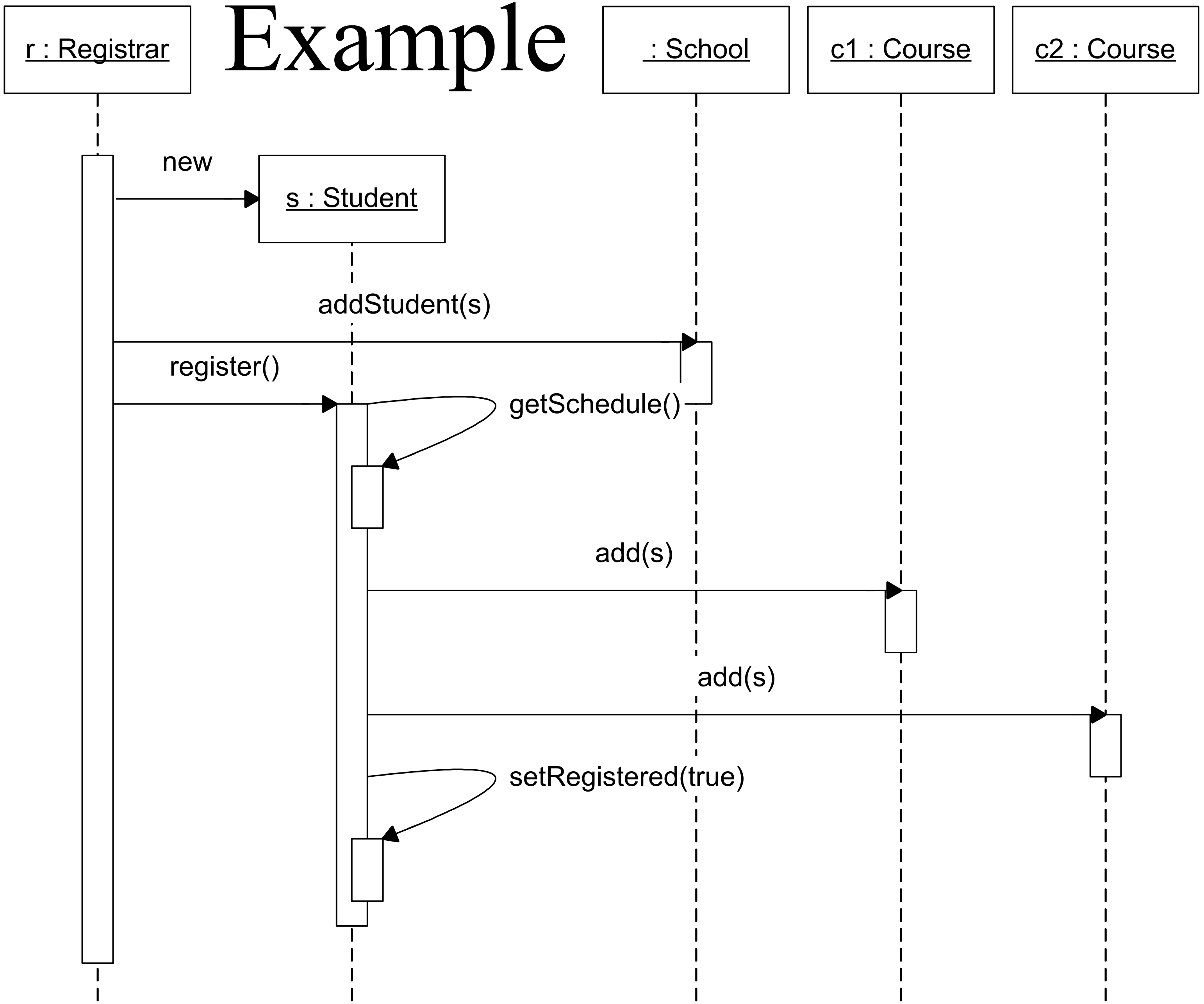
Creating/Deleting Objects



Pseudo Code

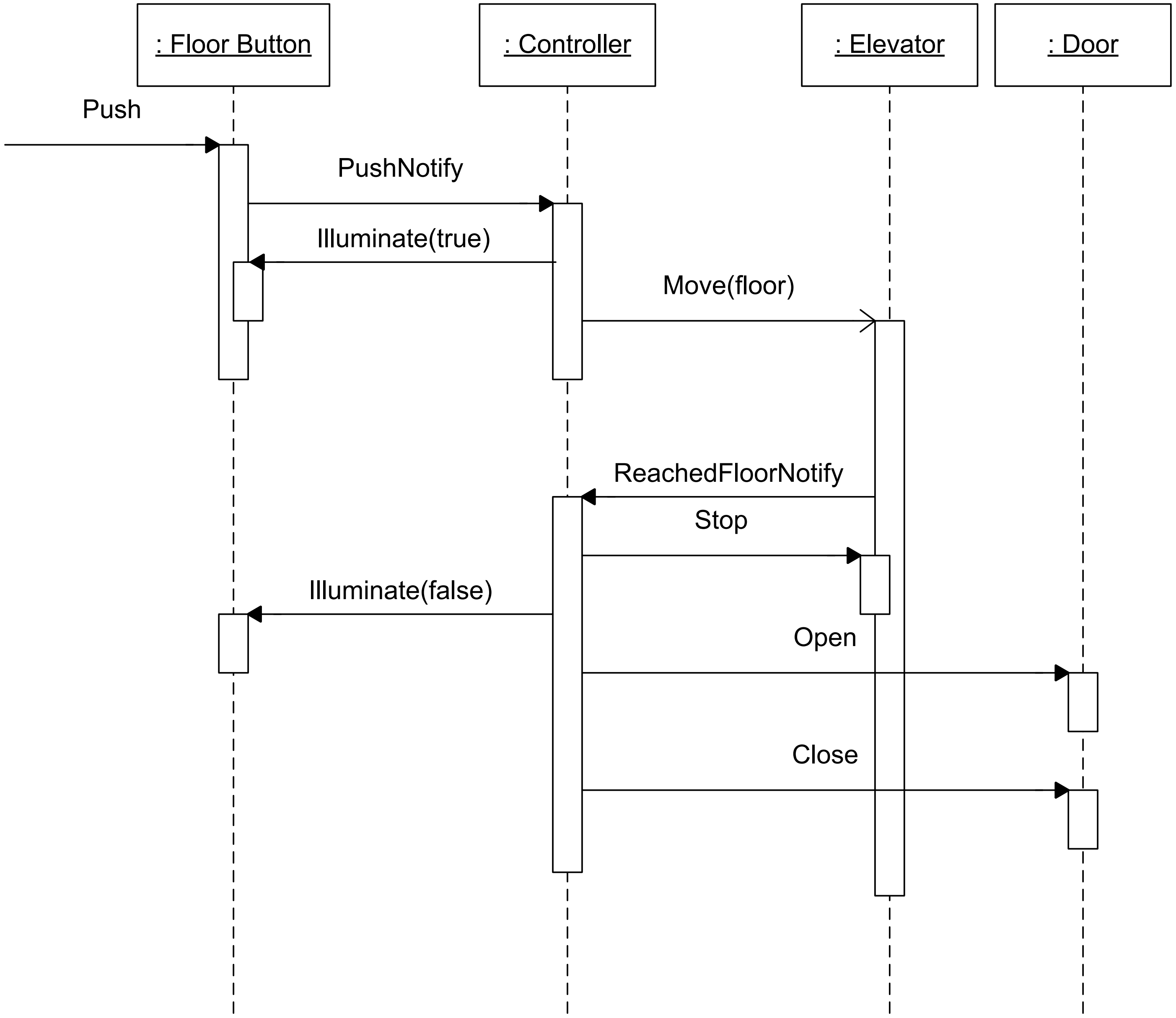
```
Create new student

Add new student to school

For each course in student's schedule
    Add student to course

Mark student as "registered"
```

Example



Source: Booch et. al., The UML User Guide

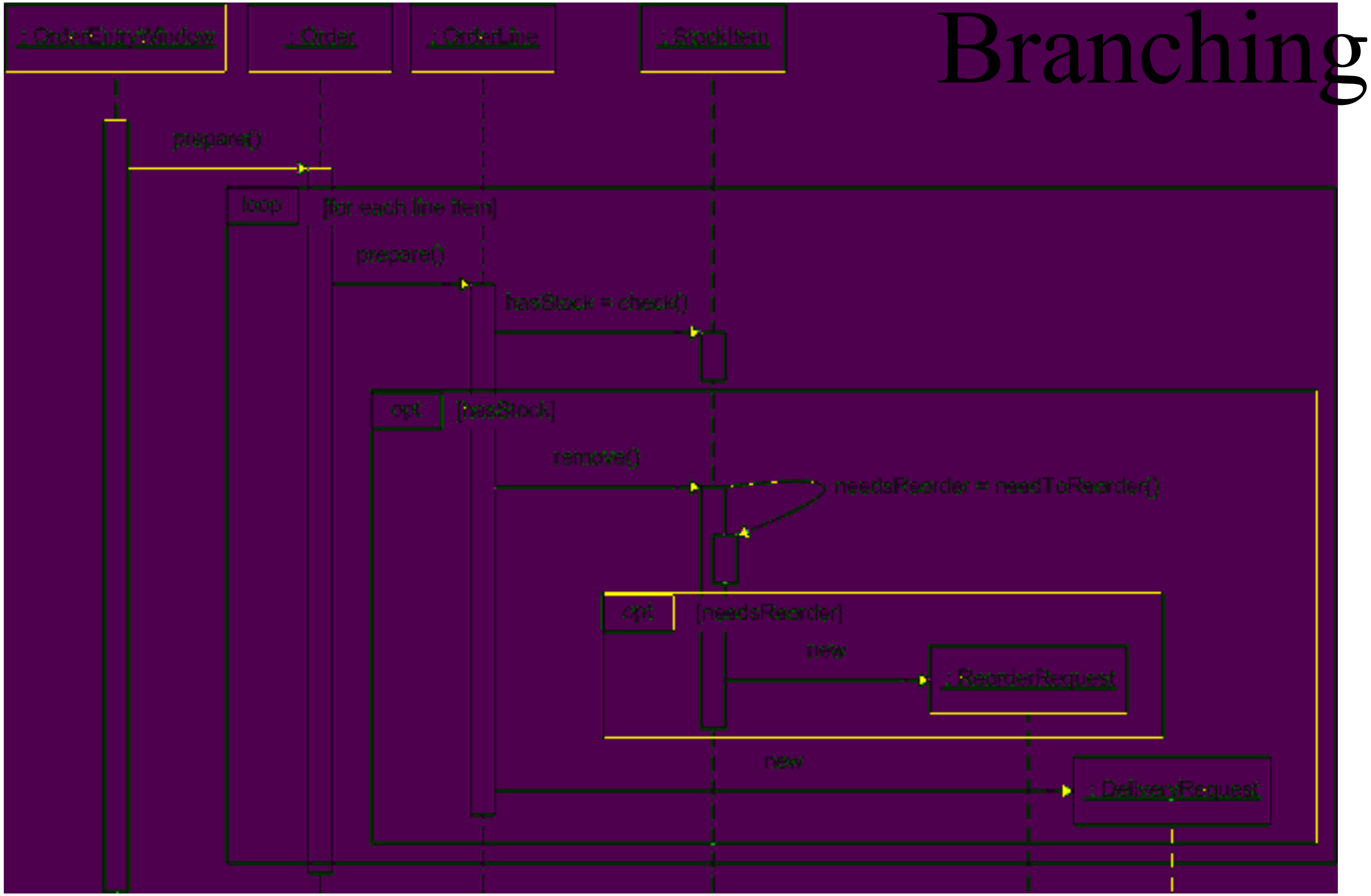
Example


: Floor Button
: Controller
: Elevator
: Door
Push
PushNotify
Illuminate(true)
Move(floor)
ReachedFloorNotify
Stop
Illuminate(false)
Open
Close

Pseudo Code

```
For each line item in the order

    If item is in stock

        Decrement quantity of item in stock

        If item needs to be reordered
            Create re-order request

        Create delivery request
```

Iteration and Branching



Source: Fowler, UML Distilled

Sequence Diagrams vs. Pseudo Code

- Sequence diagrams are good at showing how multiple objects work together to achieve a task
  - Algorithms are frequently distributed across multiple classes
  - Sequence diagrams excel at showing the message flow across participating objects
- Sequence diagrams are not good at showing complex logic
  - Complex logic = Lots of iteration and branching
  - If complex logic is needed, you can create a separate diagram for each major path, thus keeping each diagram as linear as possible
  - Pseudo code can be a better representation for complex algorithms