Object-Oriented Design

Lecturer: Raman Ramsin

Lecture 14:

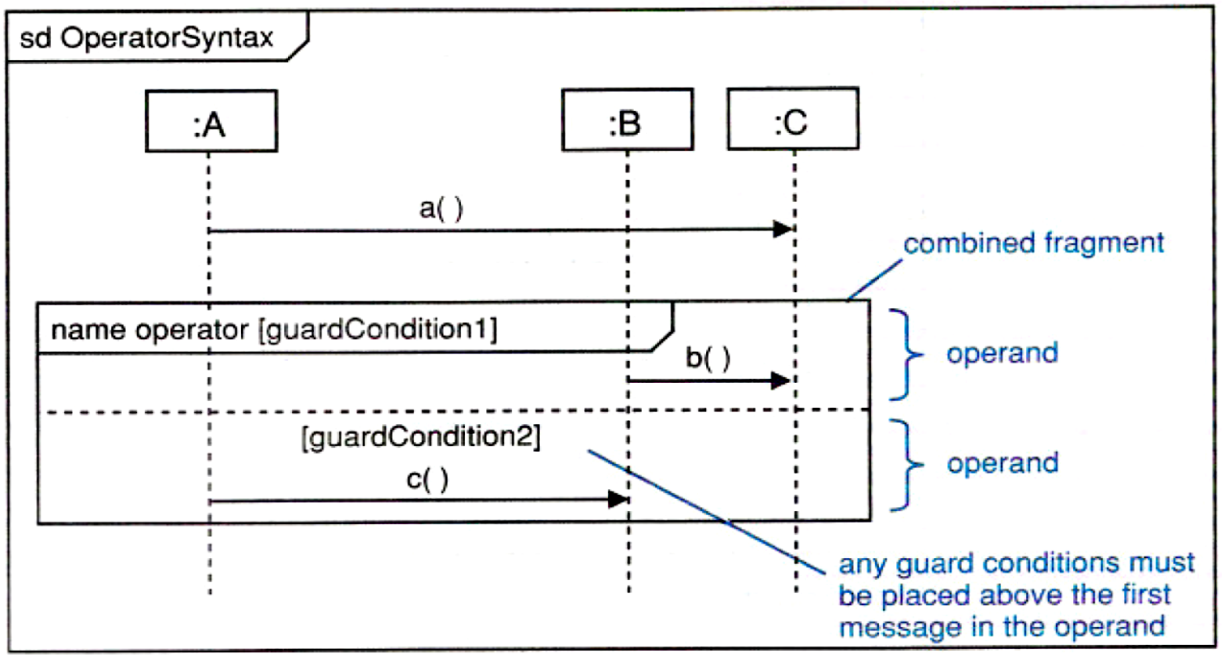
Use Case Realizations – Part 2

Analysis Workflow: *Analyze a Use Case*

- The *analysis workflow* consists of the following activities:
  - Architectural analysis
  - **Analyze a use case**
    - **Outputs:**
      - **analysis classes**
      - **use case realizations**
  - Analyze a class
  - Analyze a package

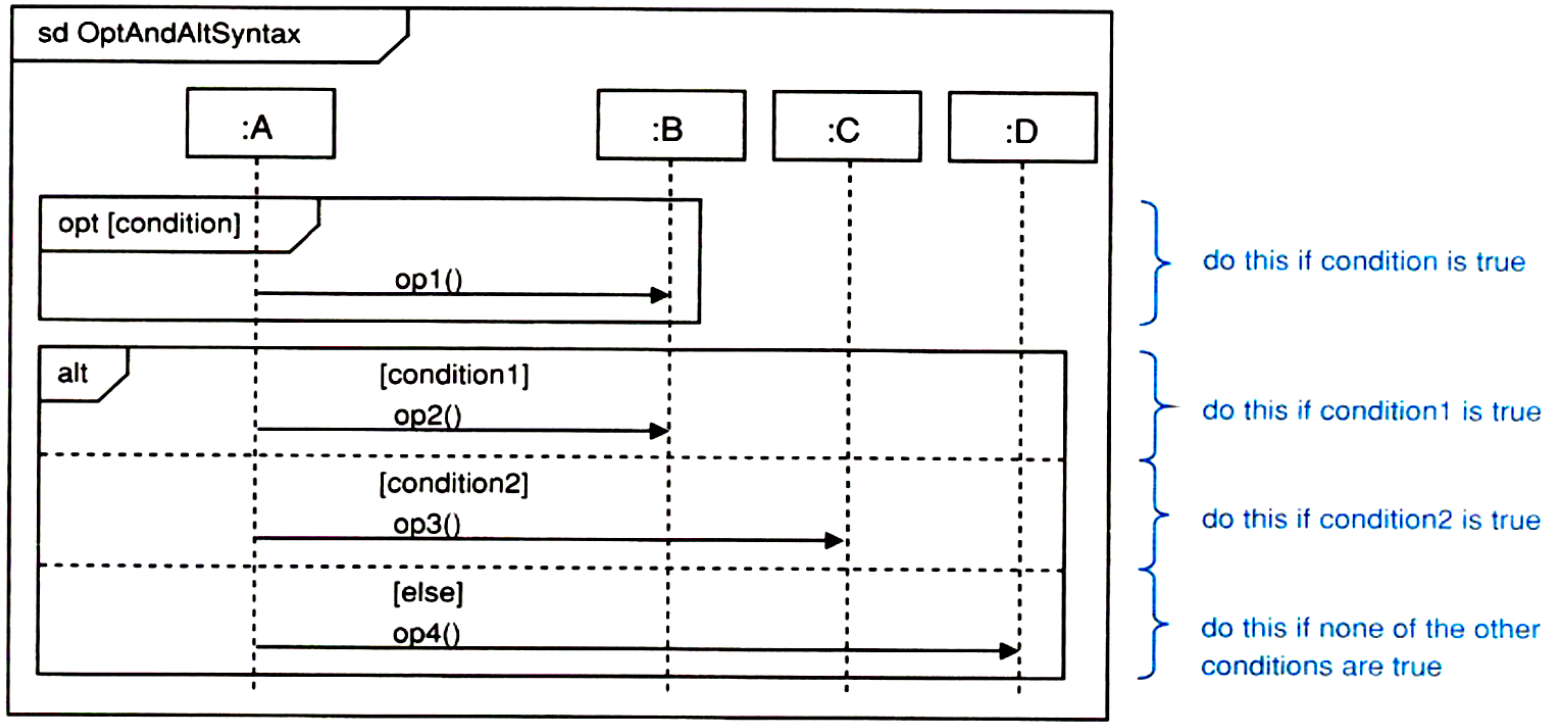
Combined Fragments

- Combined fragments - areas within a sequence diagram with different behavior.
  - The operator defines *how* its operands execute.
  - The guard condition defines *whether* its operand executes.
  - The operand contains the behavior.



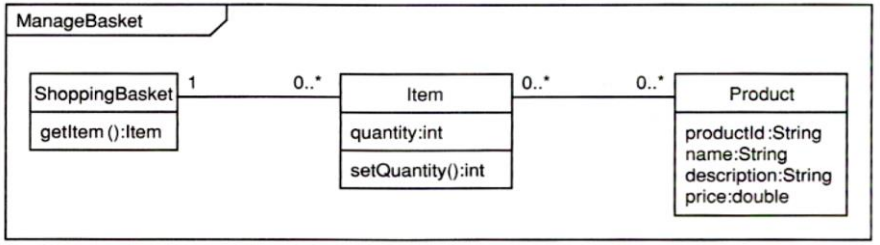
Combined Fragments: Operators – *opt* and *alt*

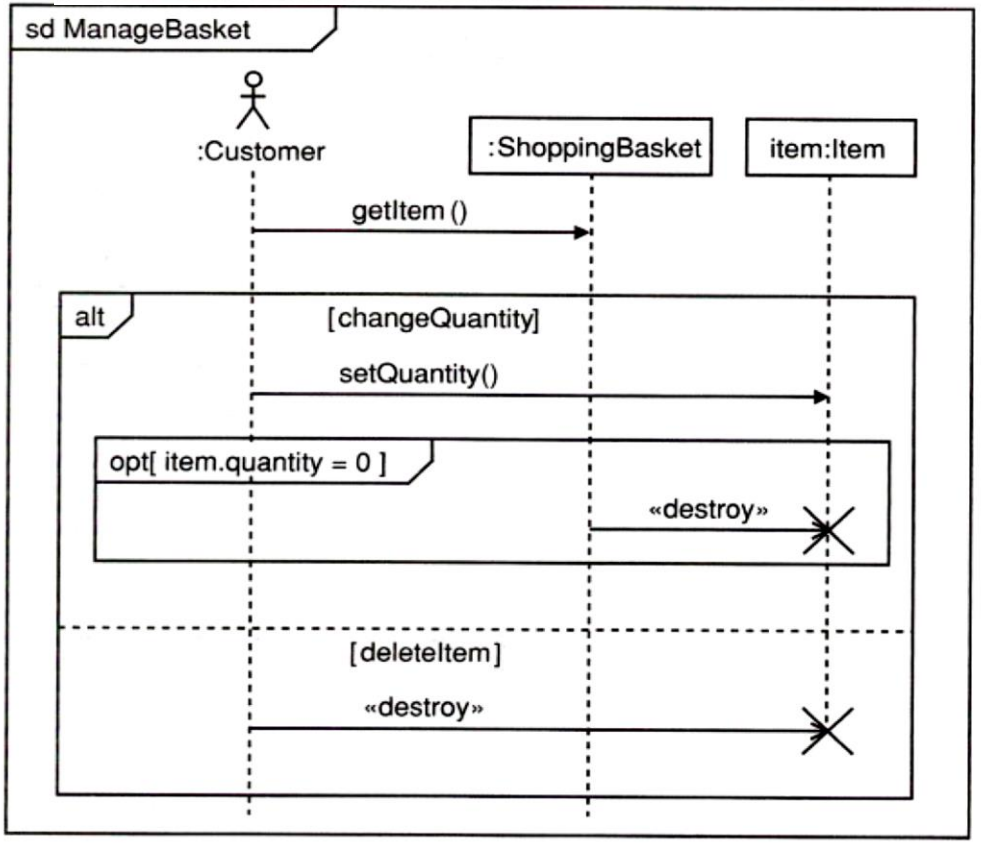
- **opt** - there is a single operand that executes if the condition is true (like if ... then).
- **alt** - the operand whose condition is true is executed.



Combined Fragments: Operators – *opt* and *alt*

Use case: ManageBasket
ID: 2
Brief description: The Customer changes the quantity of an item in the basket.
Primary actors: Customer
Secondary actors: None.
Preconditions: 1. The shopping basket contents are visible.
Main flow: 1. The use case starts when the Customer selects an item in the basket. 2. If the Customer selects "delete item" 2.1 The system removes the item from the basket. 3. If the Customer types in a new quantity 3.1 The system updates the quantity of the item in the basket.
Postconditions: None.
Alternative flows: None.



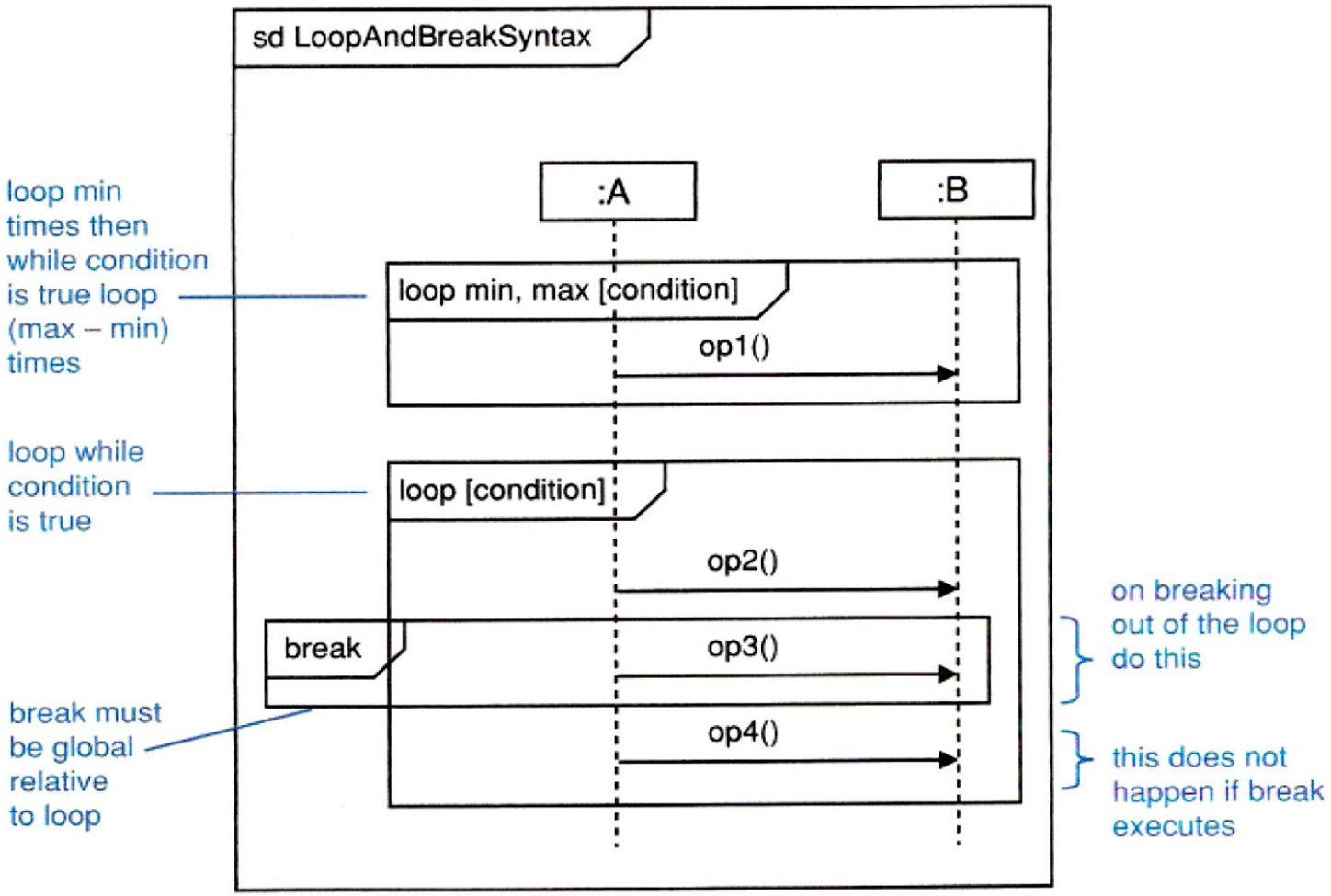


Combined Fragments: Operators – *loop* and *break*

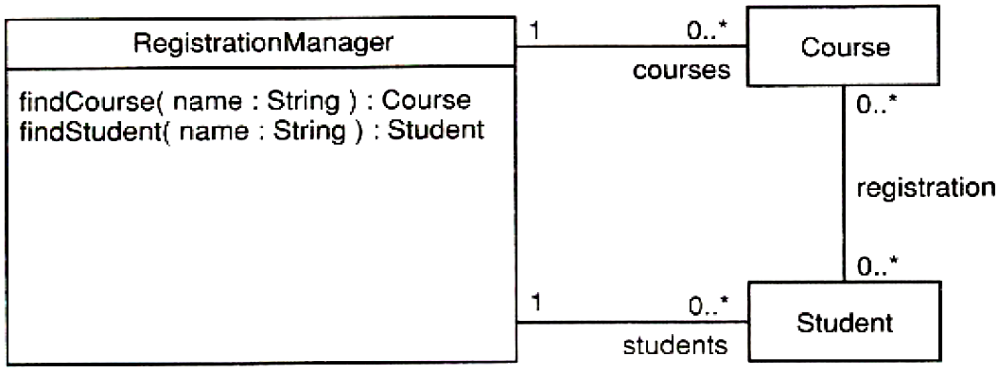
- **loop** - loop min, max [condition]
  - loop or loop * - loop forever;
  - loop n, m - loop (m – n + 1) times;
  - loop [booleanExpression] - loop while booleanExpression is true;
  - loop 1, * [booleanExpression] - loop once then loop while booleanExpression is true;
  - loop [for each object in collectionOfObjects] - execute the body of the loop once for each object in the collection;
  - loop [for each object in className] - execute the body of the loop once for each object of the class.

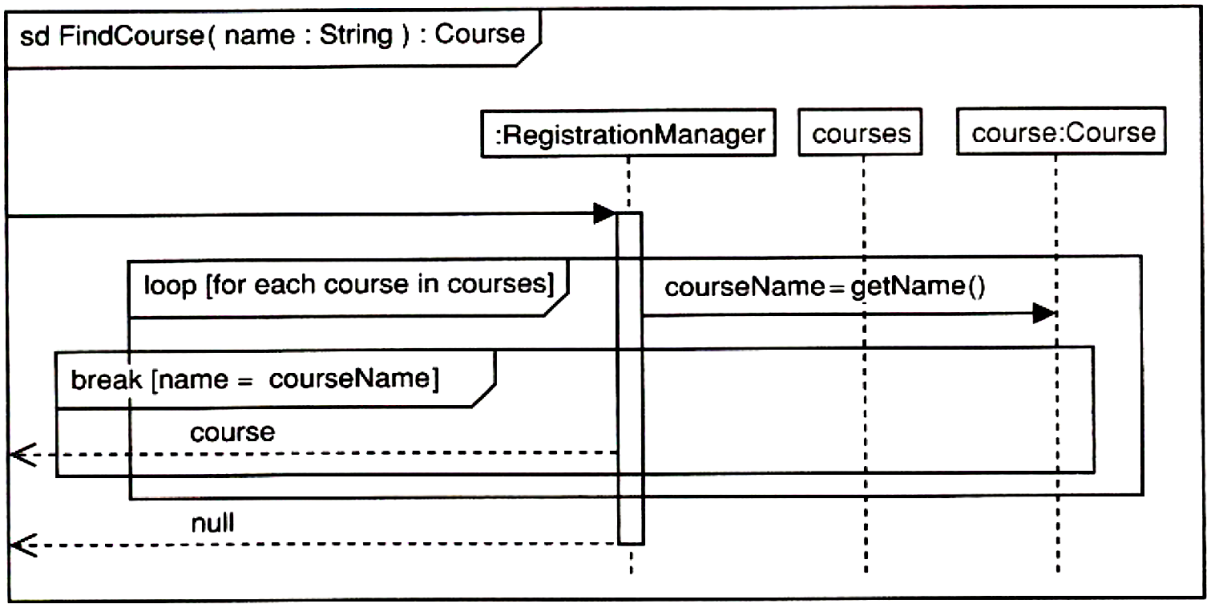
- **break** - if the guard condition is true, the operand is executed, not the rest of the enclosing interaction.

Combined Fragments: Operators – *loop* and *break* Syntax



Combined Fragments: Operators – *loop* and *break* Example



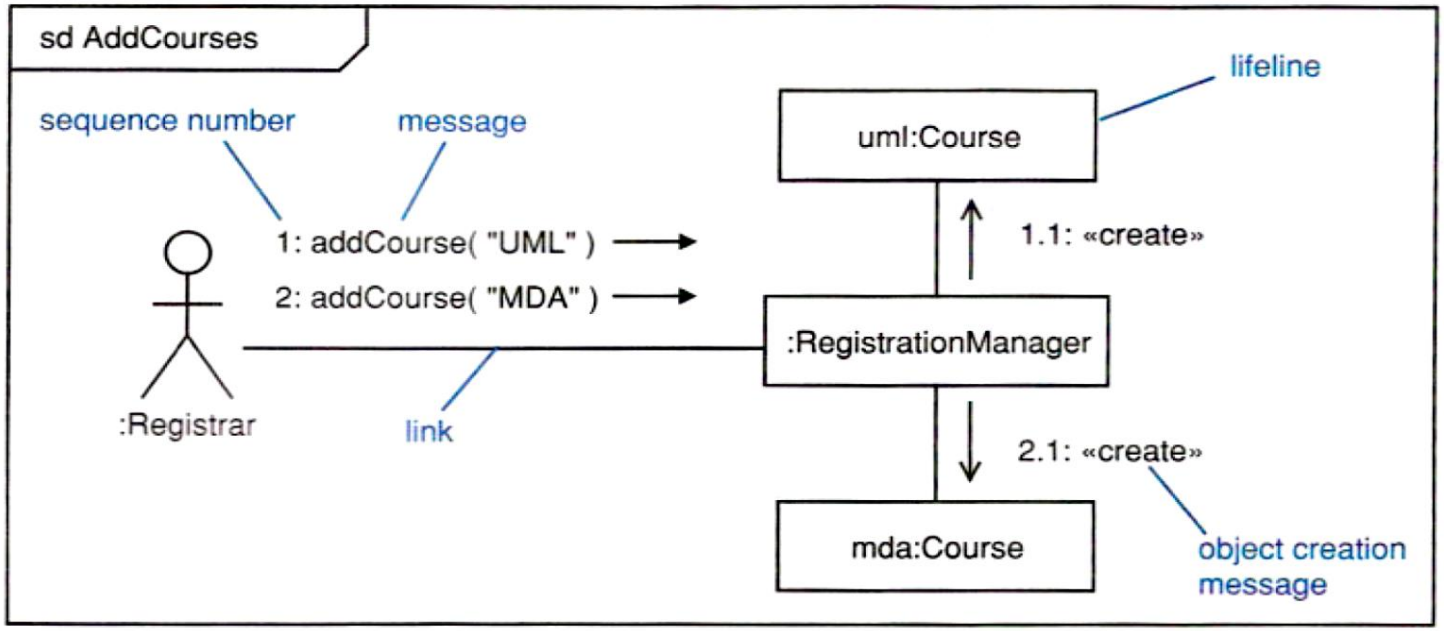


Combined Fragments: Operators – Other

- **ref** - the combined fragment refers to another interaction.
- **par** - all operands execute in parallel.
- **critical** - the operand executes atomically without interruption.
- **seq** - operands execute in parallel subject to the following constraint: events arriving on the same lifeline from different operands occur in the same sequence as the operands occur.
- **strict** - the operands execute in strict sequence.
- **neg** - the operand shows invalid interactions.
- **ignore** -lists messages that are intentionally omitted from the interaction.
- **consider** -lists messages that are intentionally included in the interaction.
- **assert** - the operand is the only valid behavior at that point in the interaction.

Communication Diagrams

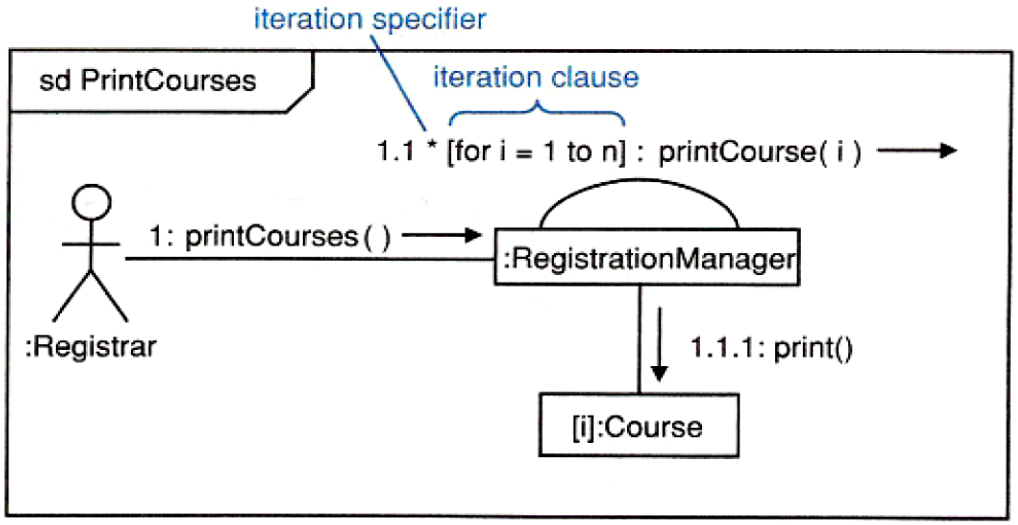
- Communication diagrams - emphasize the structural aspects of an interaction:
  - lifelines are connected by links;
  - messages have a sequence number - they are numbered hierarchically according to the nesting of the focus of control.

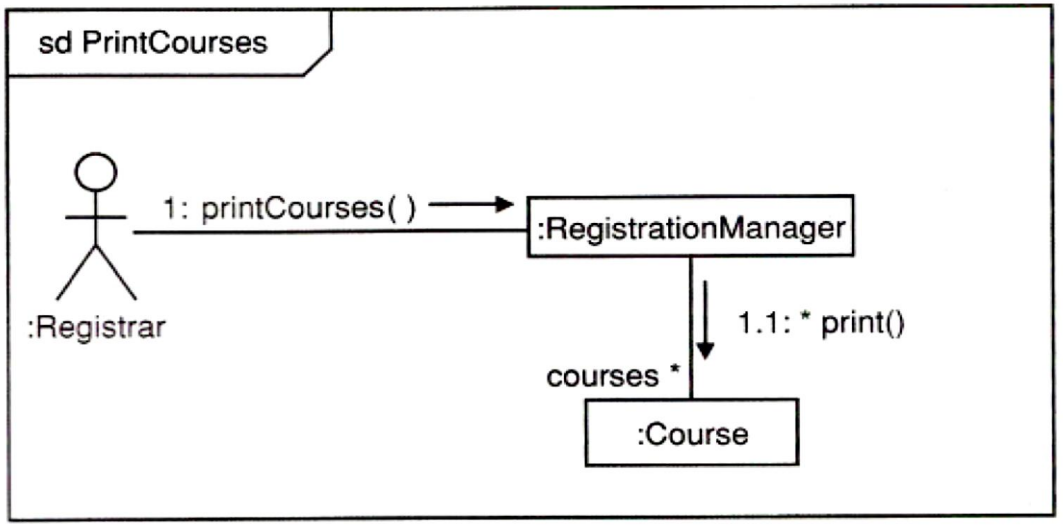


Iteration

- Iteration - use an iteration specifier (*) and an optional iteration clause on the message.
  - The iteration clause specifies the number of times to loop.
  - You can use natural language, pseudocode, source code, or sequence diagram loop notation for the iteration clause.
  - Iteration over a collection of objects:
    - Denoted by showing the role name and multiplicity (>1) on the target end of the link and prefixing the message with *.
    - The message is sent to each object in turn.
  - Use the parallel iteration specifier *// to indicate that messages are executed in parallel.

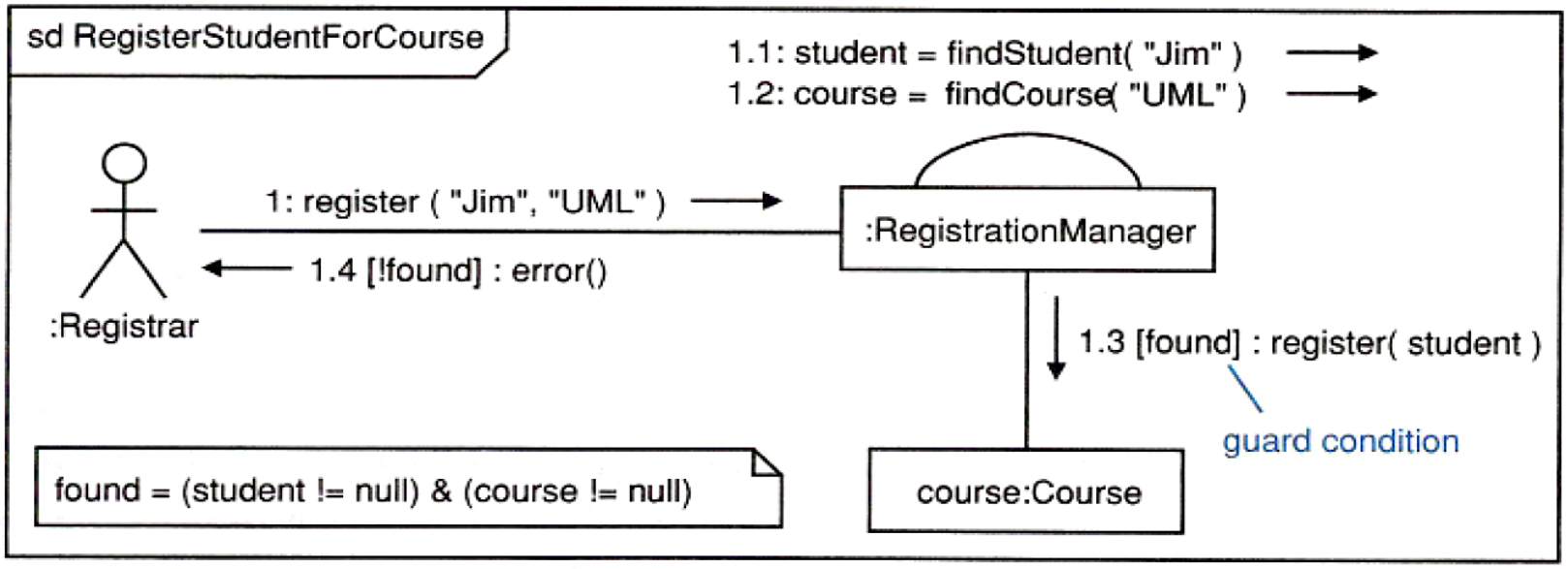
Iteration - Example





Branching

- Branching - prefix messages with guard conditions. The message executes if the guard condition is true.
- It can be hard to show branching clearly on a communication diagram - for complex branching, use sequence diagrams instead.



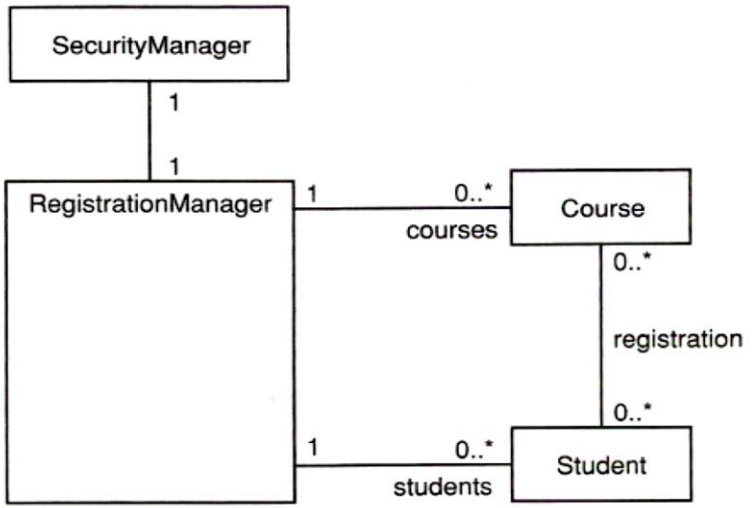
Interaction Occurrences

- Interaction occurrences: references to another interaction.
  - The flow of the referenced interaction is included in the flow of the referencing interaction.
  - Parameters - interaction occurrences may have parameters - use normal parameter notation.
  - Gates - inputs and outputs of interactions:
    - a point on the sequence diagram frame that connects a message outside the frame to a message with the same signature inside the frame.
  - Use parameters when you know the source and destination of all messages - use gates when you don't.

Interaction Occurrences – Example

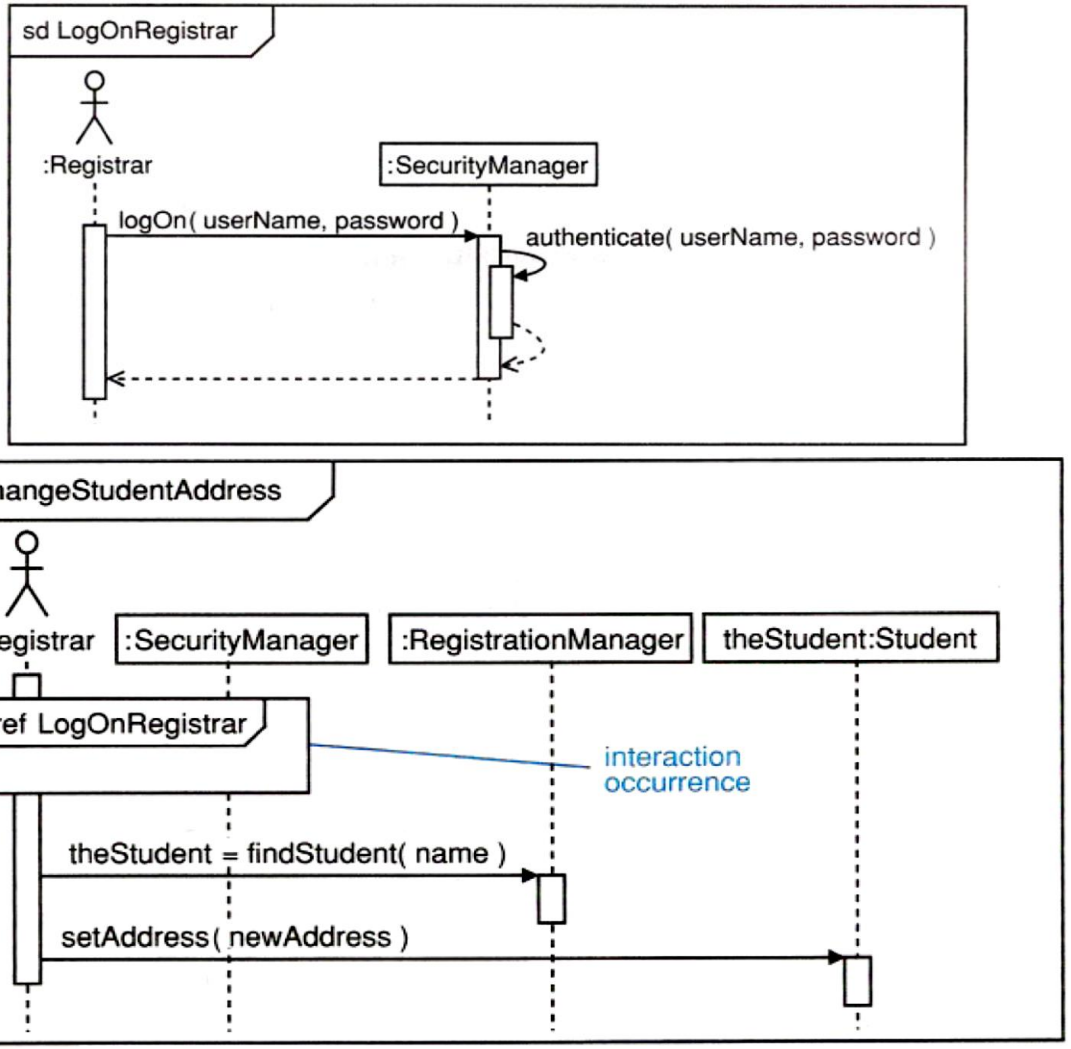
Use Case and Class Diagram

Use case: LogOnRegistrar
ID: 4
Brief description: The Registrar logs on to the system.
Primary actors: Registrar
Secondary actors: None.
Preconditions: 1. The Registrar is not logged on to the system.
Main flow: 1. The use case starts when the Registrar selects "log on". 2. The system asks the Registrar for a user name and password. 3. The Registrar enters a user name and password. 4. The system accepts the user name and password as valid.
Postconditions: 1. The Registrar is logged on to the system.
Alternative flows: InvalidUserNameAndPassword RegistrarAlreadyLoggedOn

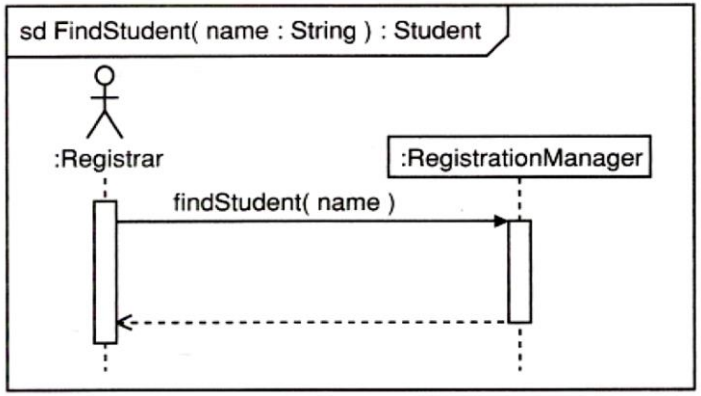


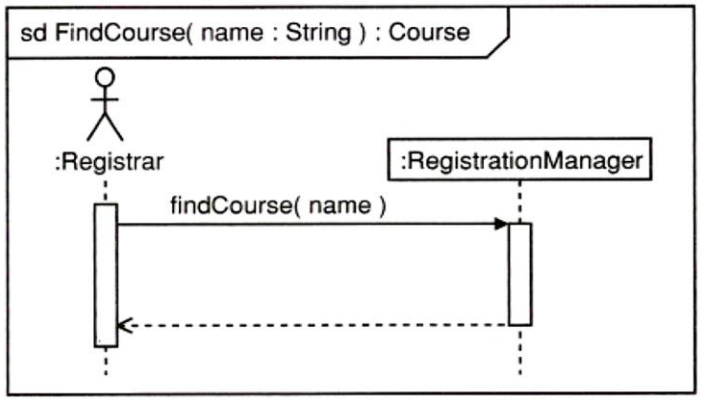
Interaction Occurrences – Example

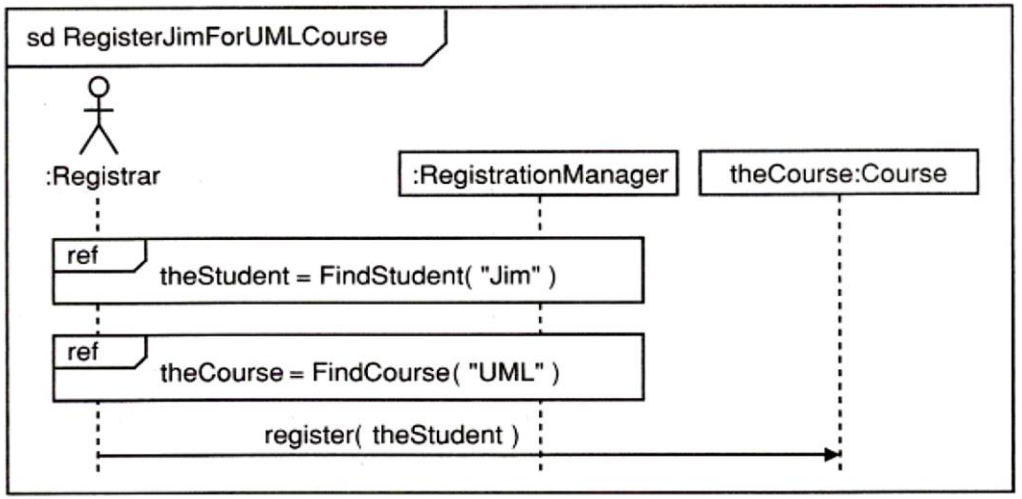
SDs



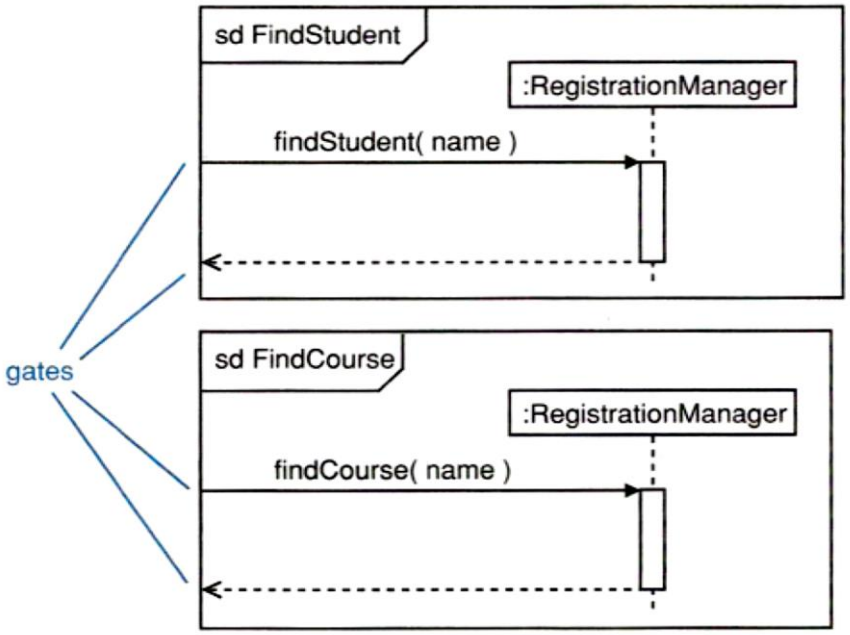
Interaction Occurrences – Parameters

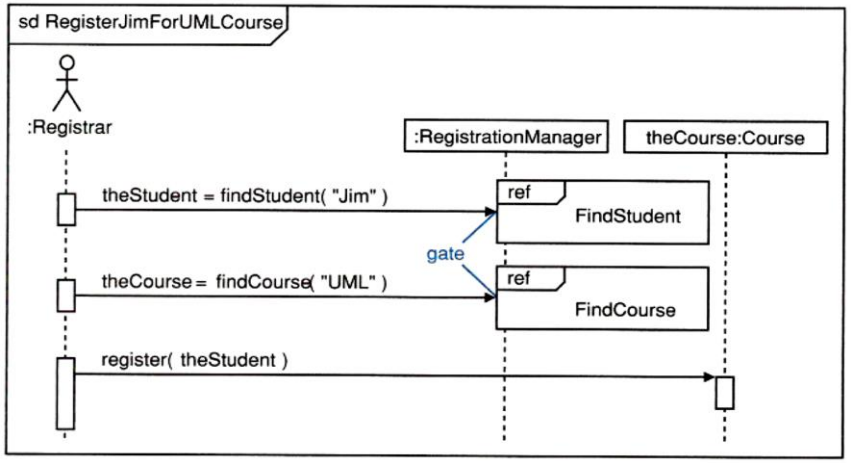


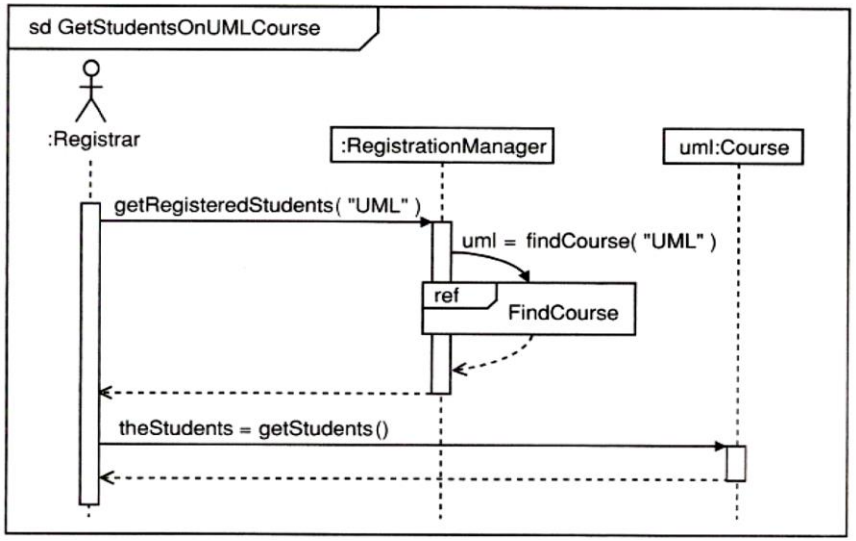




Interaction Occurrences – Gates







Reference

- Arlow, J., Neustadt, I., *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, 2nd Ed. Addison-Wesley, 2005.