RMIT UNIVERSITY

Unified Modeling Language

Activity Diagram and Sequence Diagram

Mojtaba Shahin

Week #7: Lecture - Part #2

Topics covered

- **Part 1**
  - Activity Diagram
- Part 2
  - Sequence Diagram

Notes and Acknowledgements

- Slides/images come from the following main sources:
  - Arlow, J., Neustadt, I., UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd Ed. Addison-Wesley, 2005.
  - UML Distilled, Martin Fowler
  - Object-oriented Design course by Raman Ramsin, Sharif University of Technology, Iran, http://sharif.edu/~ramsin/index_files/undergradcourse_OOD.htm
  - Sebastian Rodriguez, Software Engineering Fundamentals for IT (2110), RMIT University, Course Materials on RMIT Canvas
  - Schaum's Outlines UML (2nd edition)
  - Melina Vidoni, Software Engineering Fundamentals (2050), RMIT University, Course Materials on RMIT Canvas
  - Halil Ali , Software Engineering Fundamentals (Semester 1, 2020), RMIT University, Course Materials on RMIT Canvas
  - Ian Sommerville, Software Engineering, 10th Edition, 2015.

Classification of UML 2.2 Diagrams*

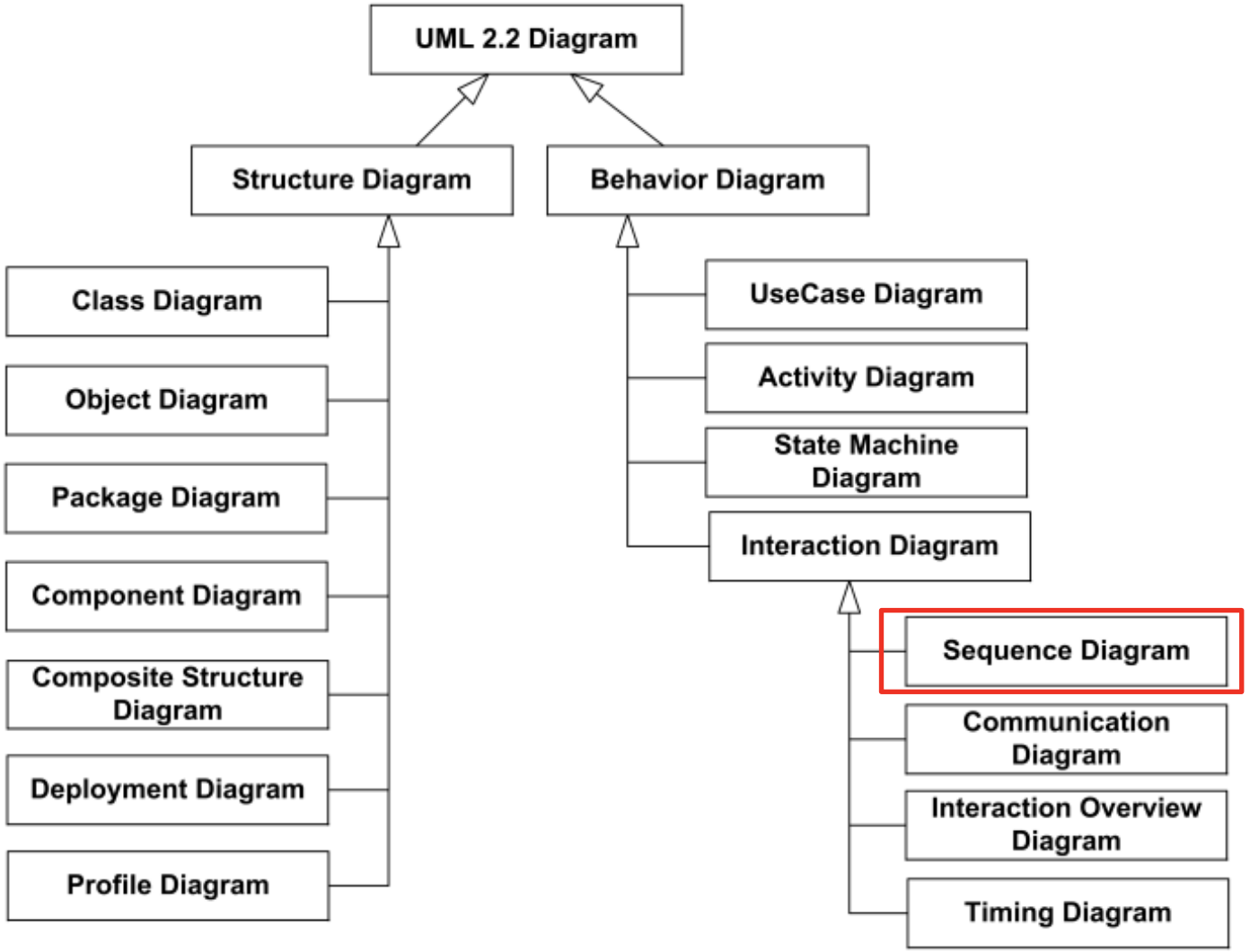


Image Source: https://www.uml-diagrams.org/uml-22-diagrams.html

Interactions and Interaction Diagrams

- **Interactions** in Interaction Diagrams are simply units of **behavior of a classifier** (e.g., objects, classes, instances of an object).
  - This classifier provides the context for the interaction.

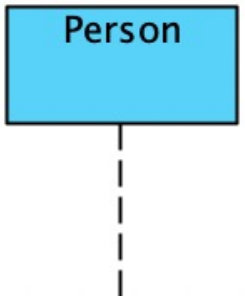
- **Interaction Diagrams** show how classifier instances interact to realize system behavior.
  - Generally, an Interaction Diagram describes one Use Case
  - An Interaction Diagram shows **how a use case is realized**

Four Types of Interaction Diagram

- Four types of interaction diagram provide different perspectives on object interaction.
  - **Sequence diagrams** – show interactions between lifelines as a time-oriented sequence of events.
  - **Communication diagrams** - emphasize structural relationships between objects.
  - **Interaction overview diagrams** – show how complex behaviour is realized by a set of simpler interactions.
  - **Timing diagrams** - emphasize real-time aspects of interactions.

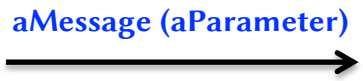
Lifelines and Messages

- The key elements in Interaction Diagrams are:
  - Lifelines
  - Messages
- A **lifeline** represents a **participant** in an interaction
  - It represents how an instance of a classifier participates in the interaction.
  - Lifeline syntax.



Messages

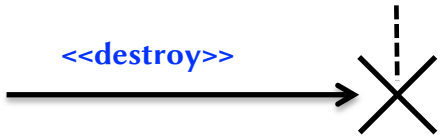
- To complete the interaction, we need to specify **messages** that are sent between the lifelines
- A message represents a specific kind of communication between two lifelines in an interaction.
- **Synchronous message** 

  - The sender waits for the receiver to finish executing the requested operation.
- **Asynchronous message** 

  - The sender does not wait for the receiver but continue to the next step.

Messages

- **Return message** 
  - The receiver of an earlier message returns focus of control the sender of that message.
- **Object creation** 

  - The sender creates an instance of the classifier specified by the receiver.
- **Object destruction** 

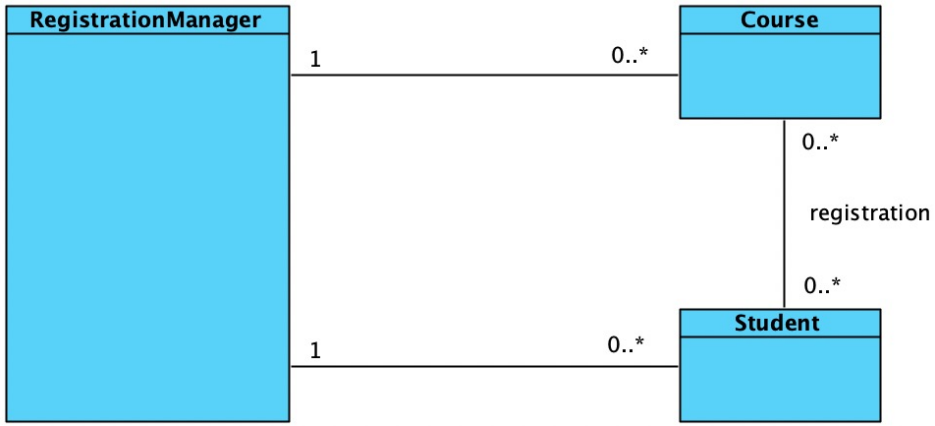
  - The sender destroys the receiver.
  - If its lifeline has a tail, this is terminated with **X**.

Sequence Diagrams: General Notation

- **Time** runs top to bottom.
- **Lifelines** run left to right:
  - lifelines have dashed vertical tails that indicate the duration of the lifeline;
  - lifelines may have activations to indicate when the lifeline has focus of control;
  - organize lifelines to minimize the number of crossing lines.
- **Constraints** - place constraints in {} on or near lifelines.

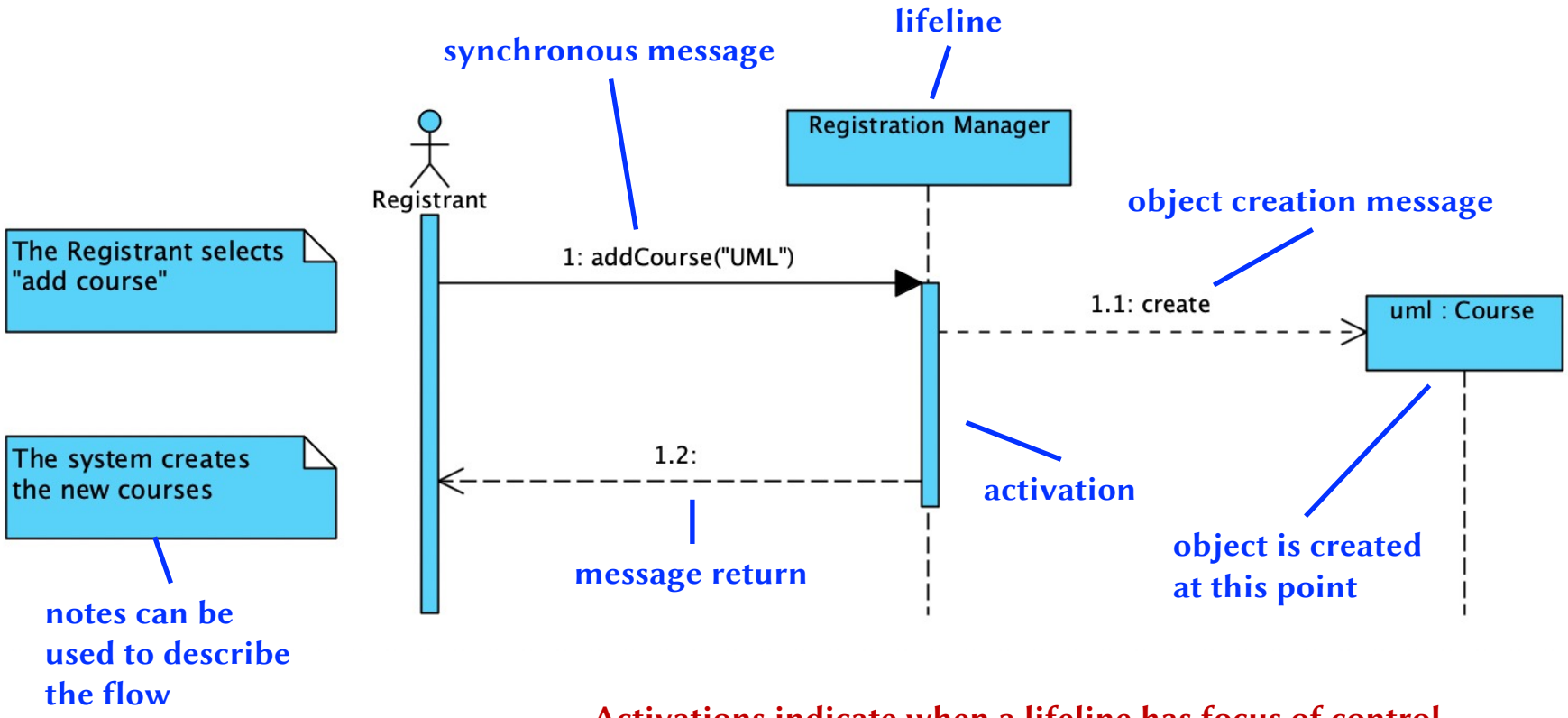
Sequence Diagrams: Example 1

Use Case: AddCourse
ID: 8
Brief description The Registrar adds details of a new course to the system.
Primary actors Registrar
Secondary actors None
Preconditions The Registrar has logged on to the system.
Main flow 1. The Registrar selects “add course”. 2. The Registrar enters the name of the new course. 3. The system creates the new course.
Postconditions A new course has been added to the system
Alternative flows CourseAlreadyExists.



Sequence Diagrams: Example 1 -

A sequence diagram that realizes the behavior of the **AddCourse** use case

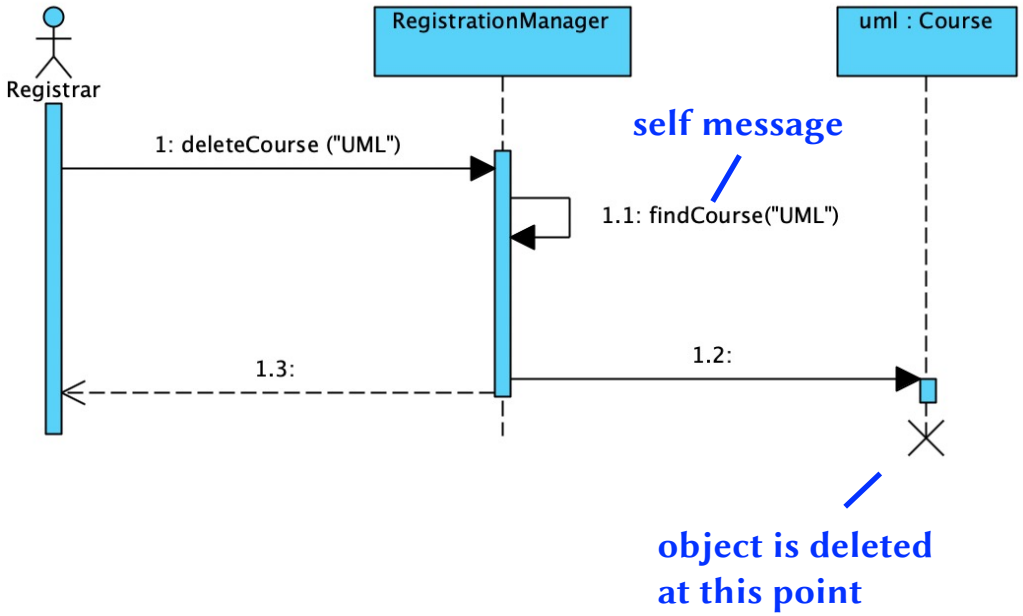


Activations indicate when a lifeline has focus of control

Sequence Diagrams: Example 2

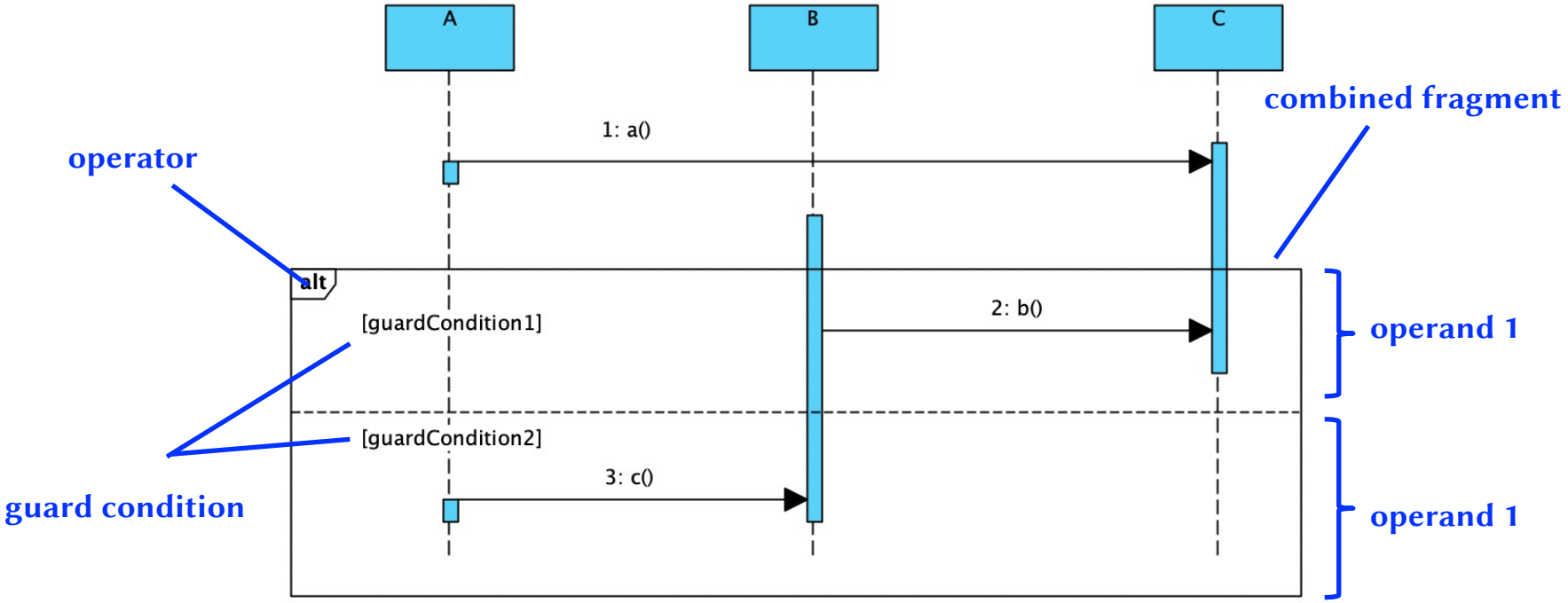
Use Case: DeleteCourse
ID: 10
Brief description The Registrar removes a course from the system.
Primary actors Registrar
Secondary actors None
Preconditions The Registrar has logged on to the system.
Main flow 1. The Registrar selects “delete course”. 2. The Registrar enters the name of the course. 3. The system deletes the course.
Postconditions A course has been removed from the system.
Alternative flows CourseDoesNotExist

A sequence diagram that realizes the behavior of the **DeleteCourse** use case



Fragments

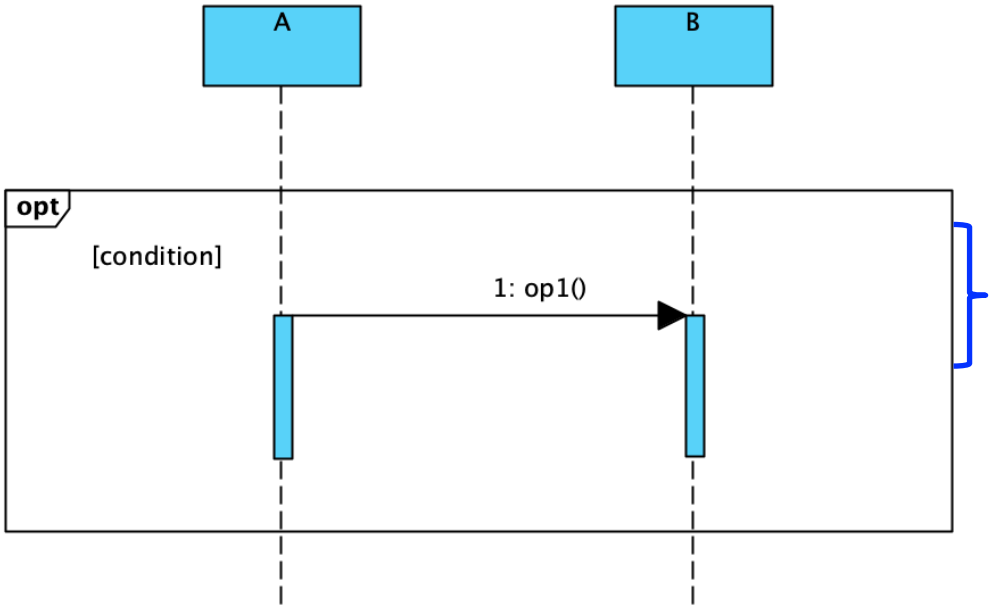
- Combined fragments divide into a sequence diagram different **areas** with **different behavior**.
  - The **operator** defines how its operands execute.
  - The **guard condition** defines whether its operand executes.
  - The **operand** contains the behavior.



Combined Fragments: Operators – *opt*

- **opt** - there is a single operand that executes if the condition is true (like if ... then).
  - It creates a single branch

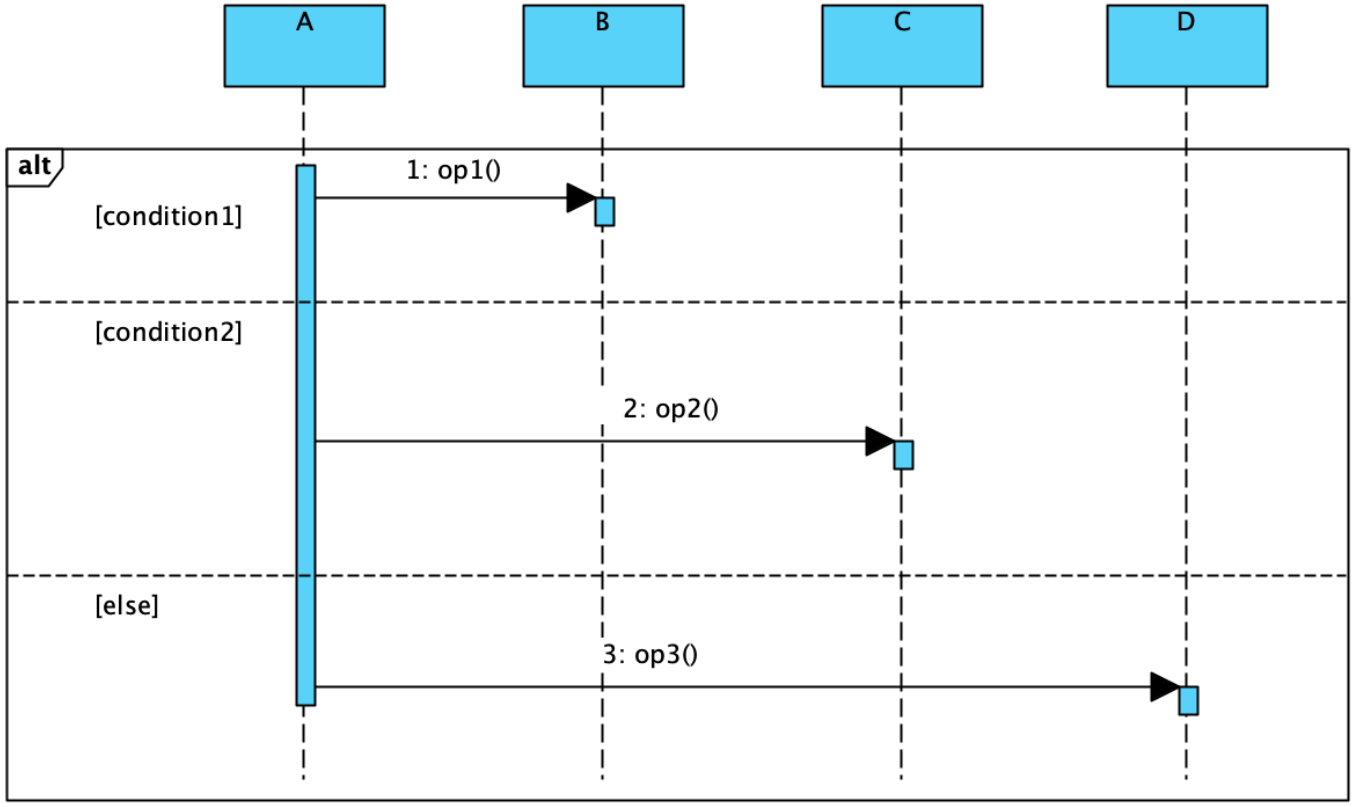
If (condition) then action



Combined Fragments: Operators – *alt*

- **alt** - the operand whose condition is true is executed.
  - it creates multiple branches

If (condition1) then
  operand 1
else if (condition2) then
  operand 2
else
  operand 3



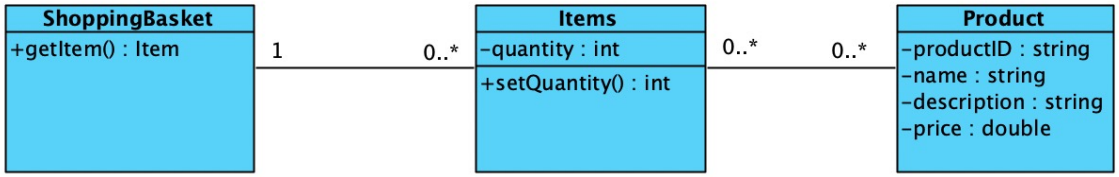
do *op1()* if condition1 is true

do *op2()* if condition2 is true

do *op3()* if none of the other conditions is true

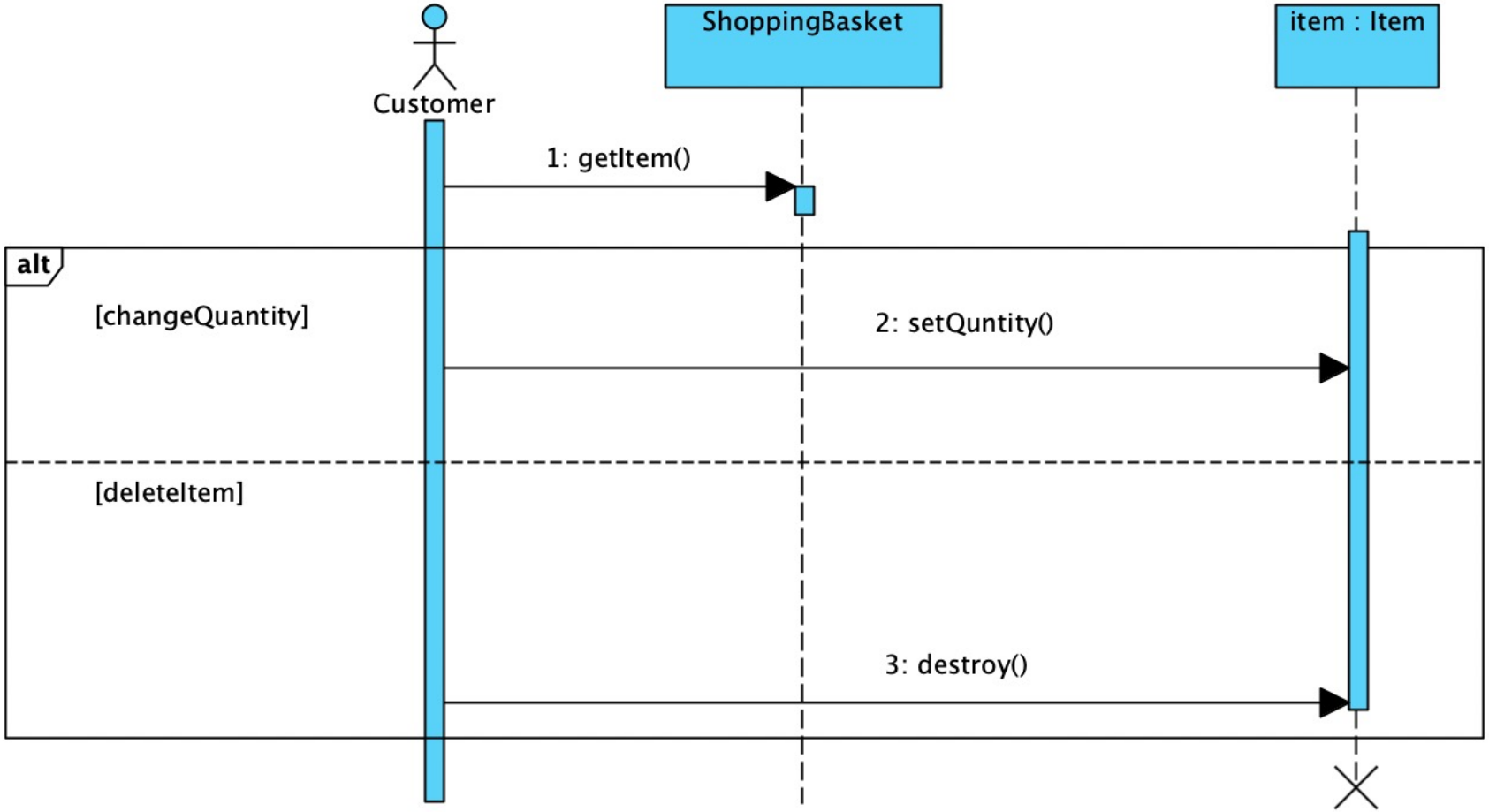
Combined Fragments: *opt* and *alt* – Example

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 The shopping basket contains are available.
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.

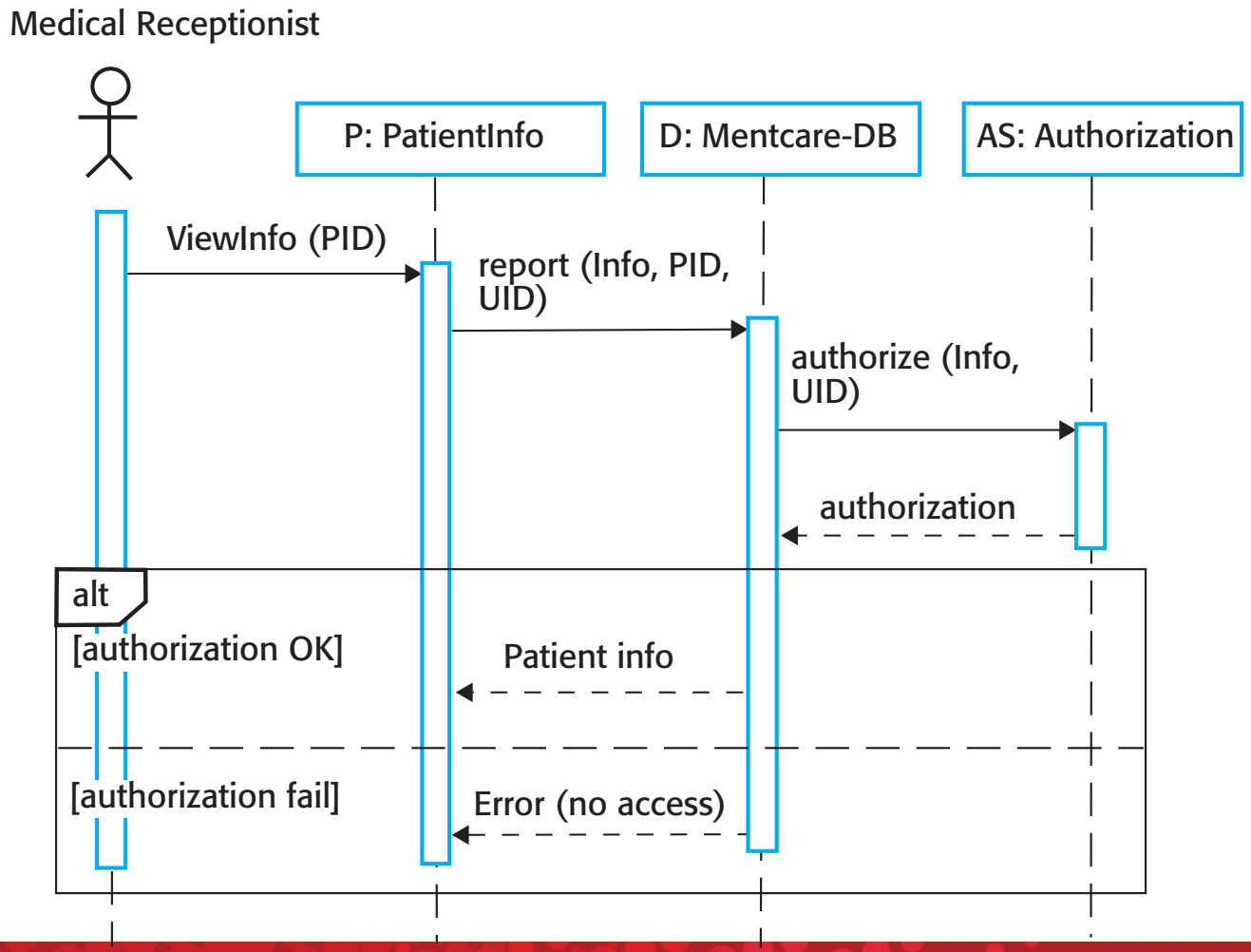


Combined Fragments: Operators – *opt* and *alt*

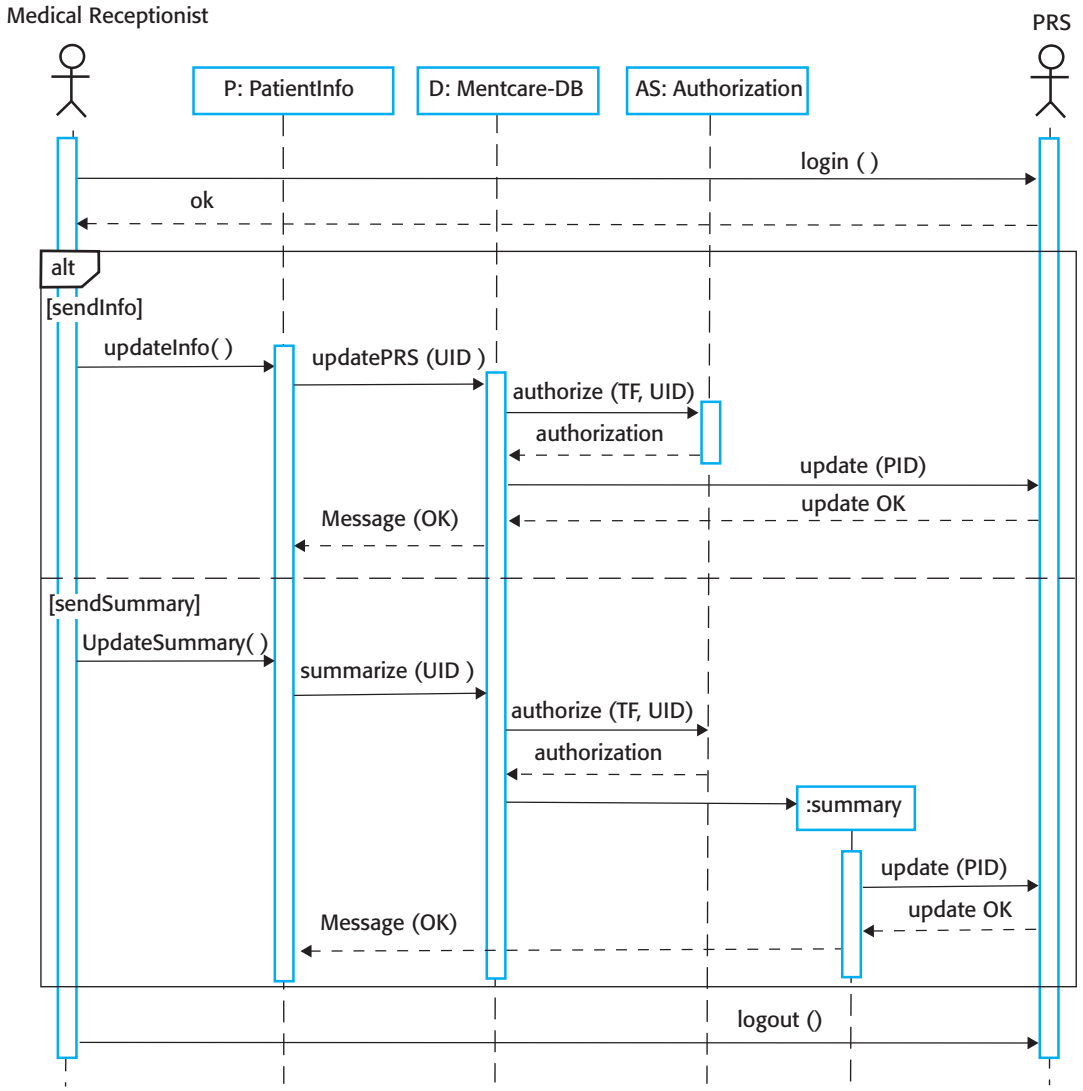
A sequence diagram that realizes the behavior of the **ManageBasket** use case



Sequence diagram for View patient information – *mental health care system*



Sequence diagram for Transfer Data – *mental health care system*



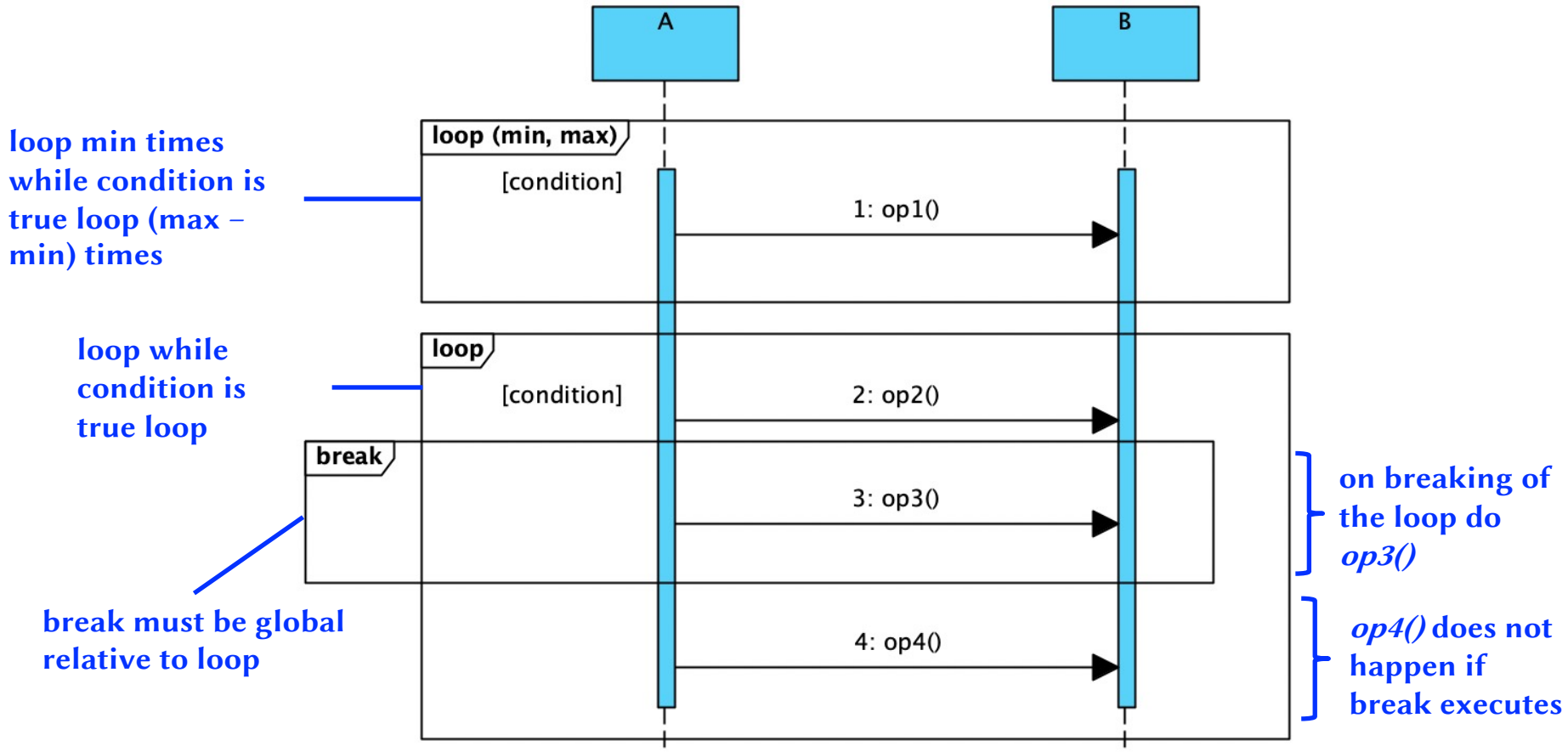
Combined Fragments: Operators – *loop* and *break*

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

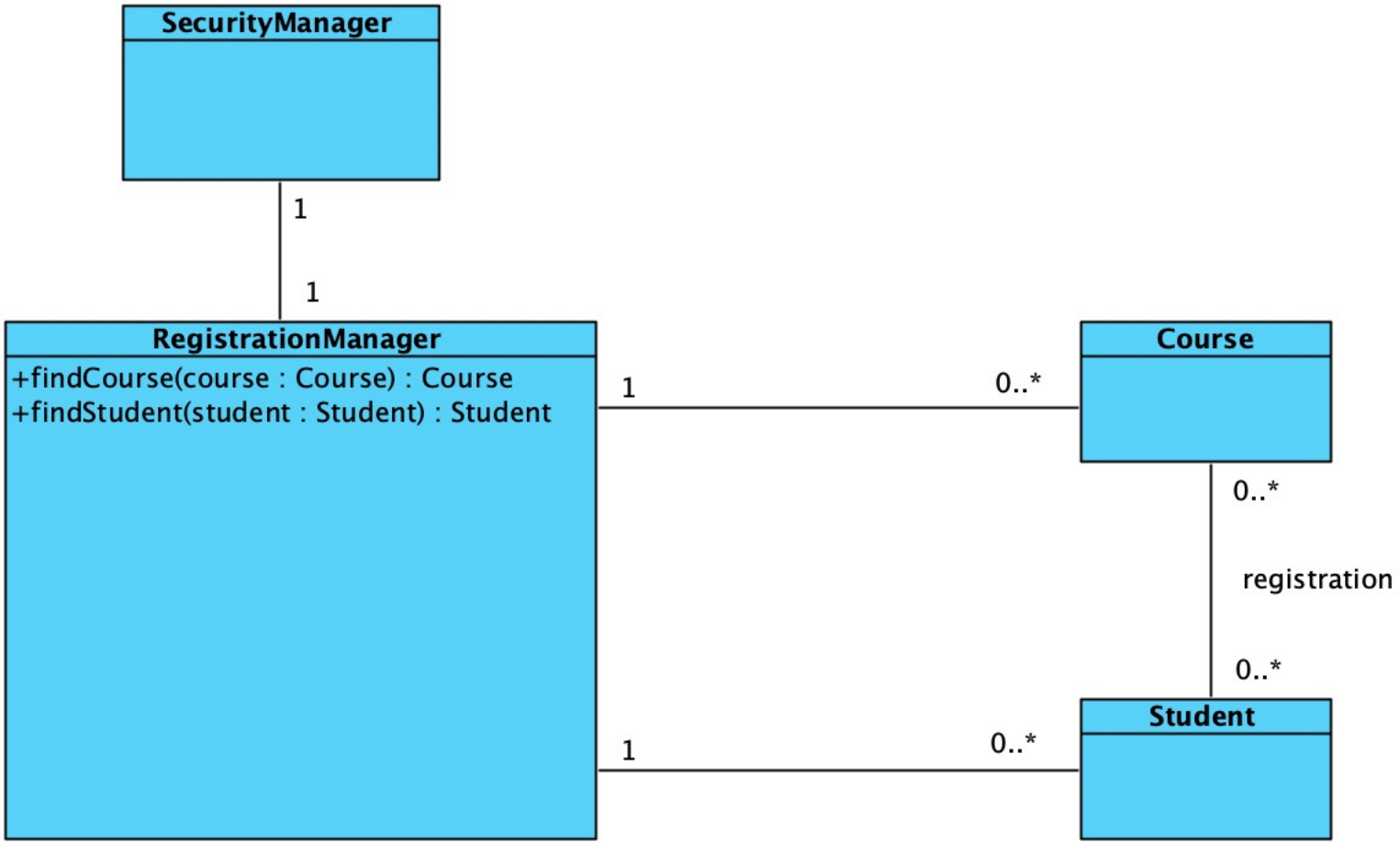
Combined Fragments: Operators – *loop* and *break*

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

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

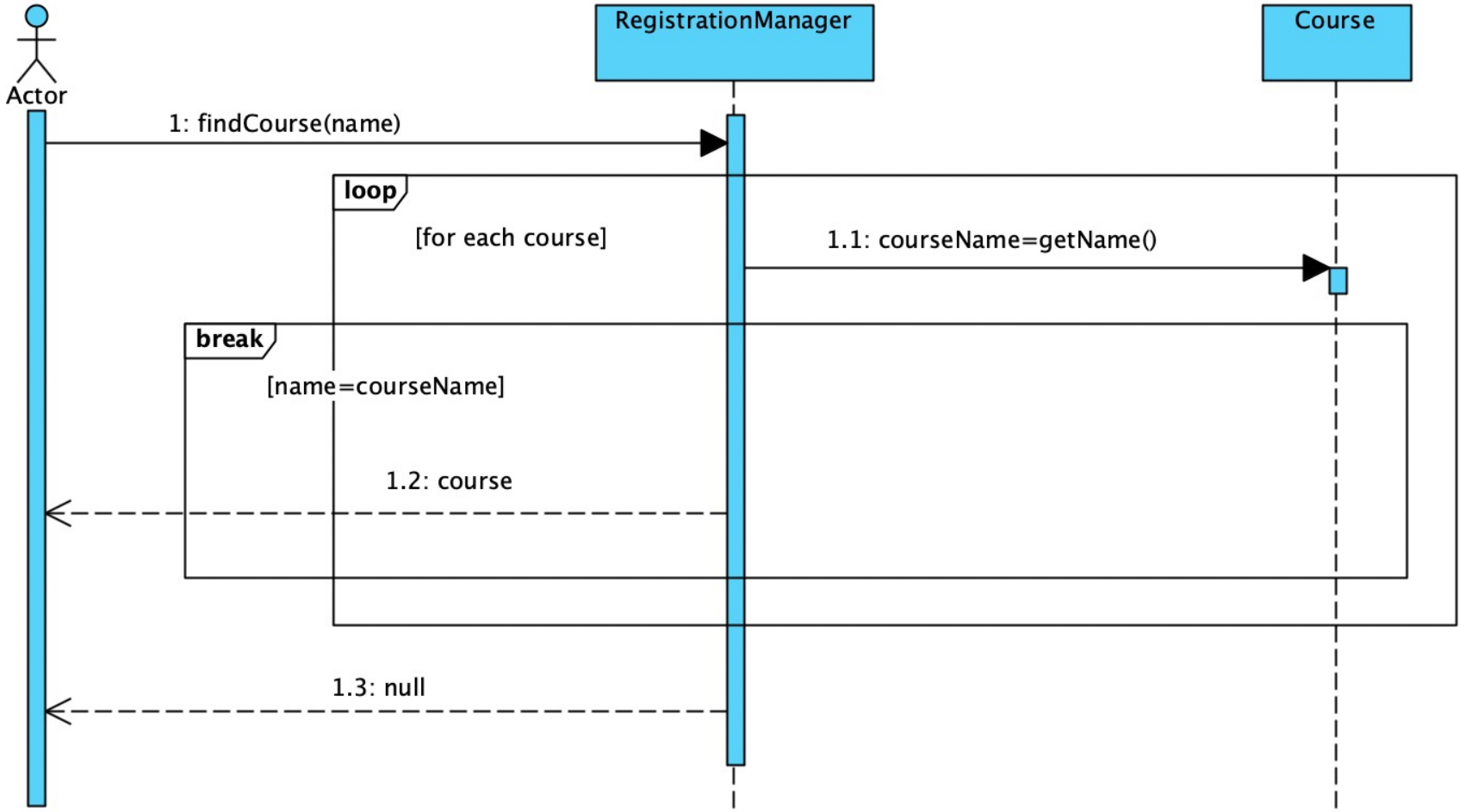


Combined Fragments: *loop* and *break* - Example



Combined Fragments: *loop* and *break* - Example

A sequence diagram that realizes the behavior of the **FindCourse** use case

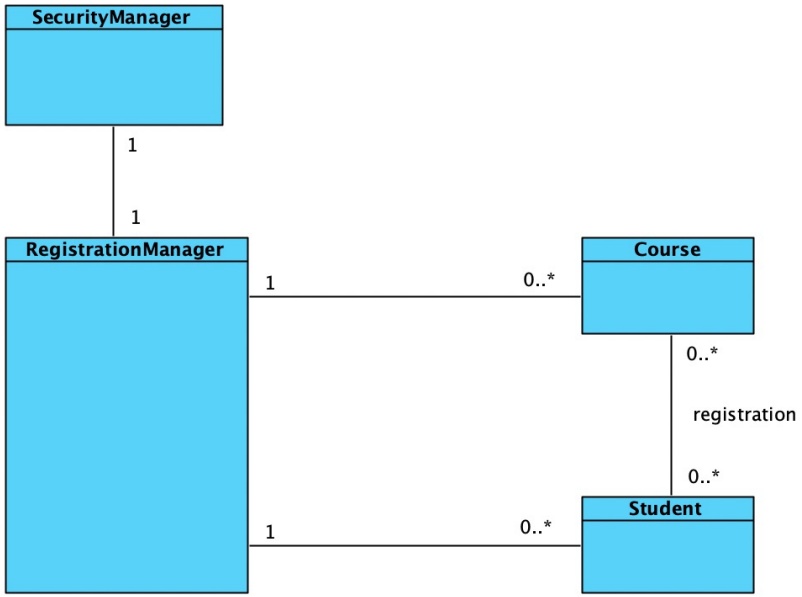


Combined Fragments: Operators – *ref*

- **ref** - the combined fragment refers to another interaction.
  - The flow of the referenced interaction is included in the flow of the referencing interaction.

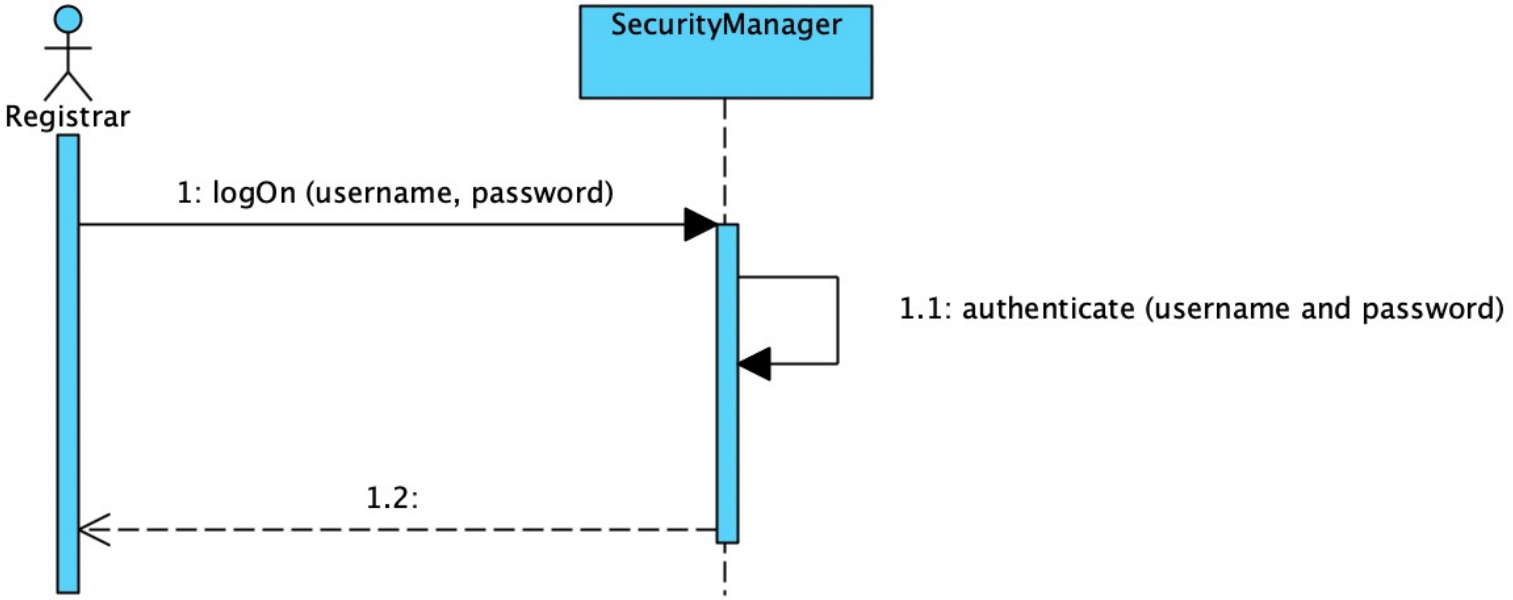
Combined Fragments: Operators – *ref* - Example

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



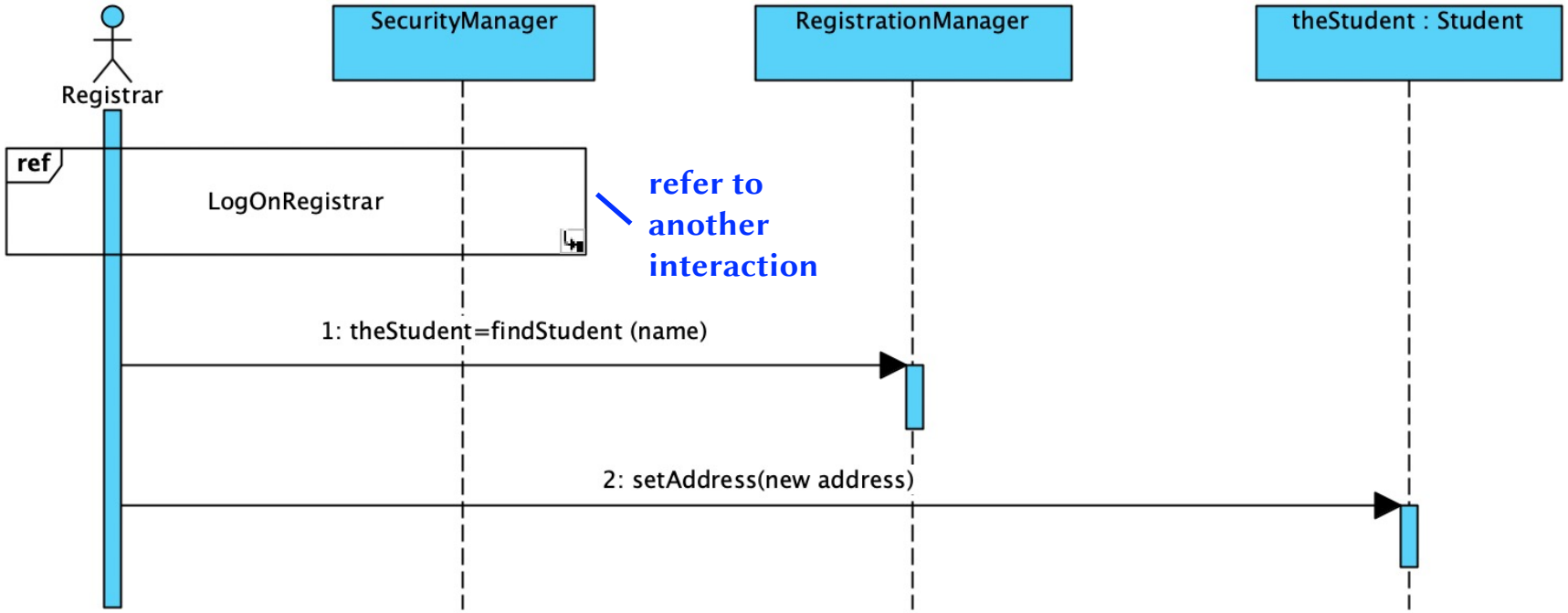
Combined Fragments: Operators – *ref* - Example

A sequence diagram that realizes the behavior of the **LogOnRegistrar** use case



Combined Fragments: Operators – *ref* - Example

A sequence diagram that realizes the behavior of the **ChangeStudentAddress** use case



References

- Arlow, J., Neustadt, I., UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd Ed. Addison-Wesley, 2005.
- UML Distilled, Martin Fowler
- Object-oriented Design course by Raman Ramsin, Sharif University of Technology, Iran, http://sharif.edu/~ramsin/index_files/undergradcourse_OOD.htm
- Sebastian Rodriguez, Software Engineering Fundamentals for IT (2110), RMIT University, Course Materials on RMIT Canvas
- Schaum's Outlines UML (2nd edition)
- Melina Vidoni, Software Engineering Fundamentals (2050), RMIT University, Course Materials on RMIT Canvas
- Halil Ali , Software Engineering Fundamentals (Semester 1, 2020), RMIT University, Course Materials on RMIT Canvas
- Ian Sommerville, Software Engineering, 10th Edition, 2015.

Thanks!

Mojtaba Shahin

mojtaba.shahin@rmit.edu.au