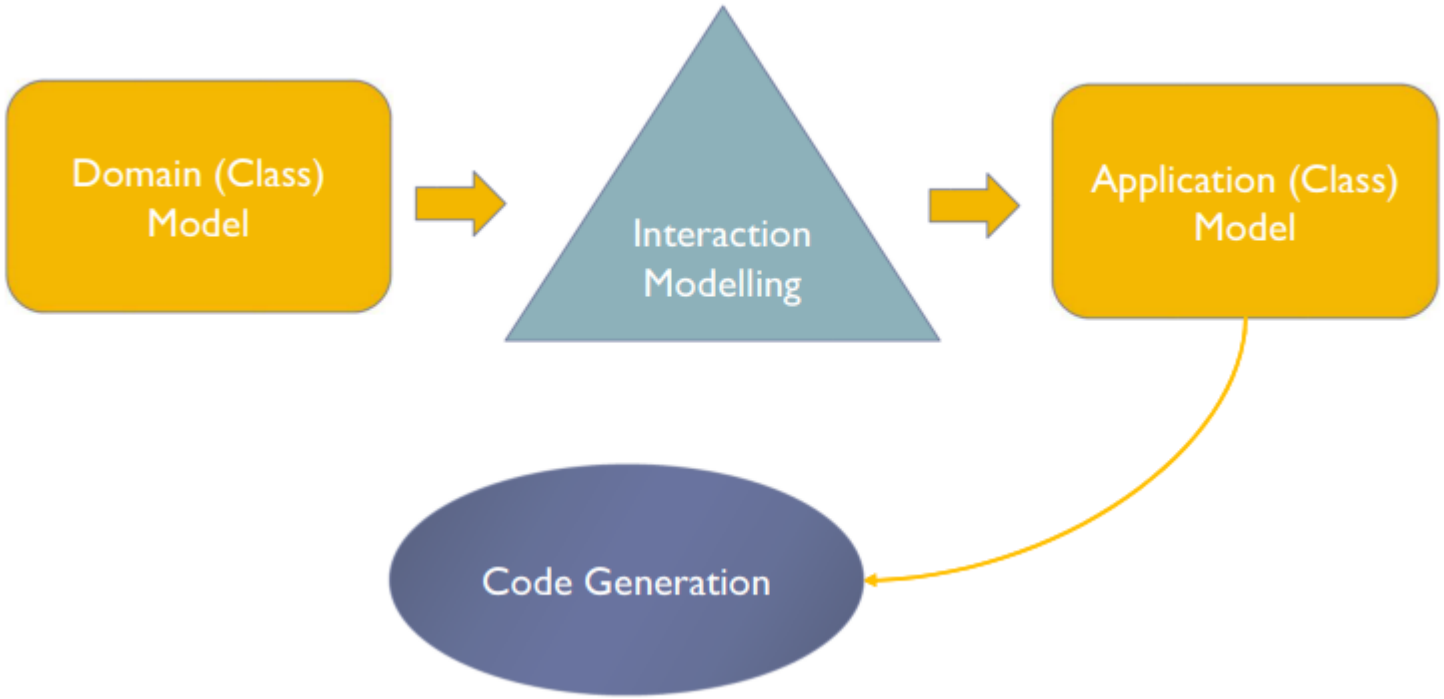
Interaction Modelling: Use-case & Sequence models

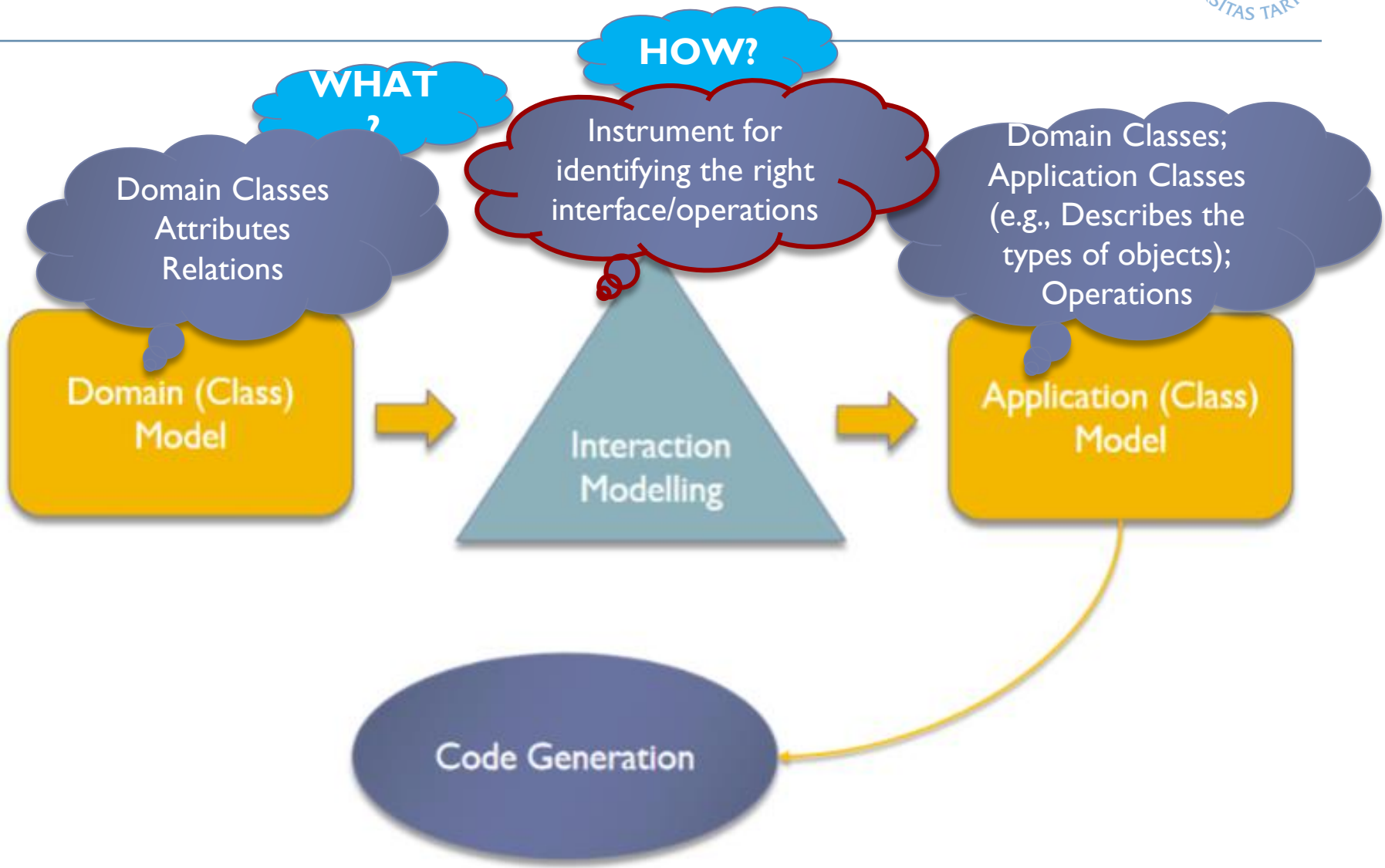
Anastasija Nikiforova

Institute of Computer Science

Based on materials developed by Marlon Dumas, Fabrizio Maria Maggi, Luciano García-Bañuelos

Software Development Methodology




WHAT
?
HOW?
Instrument for identifying the right interface/operations
Domain Classes
Attributes
Relations
Domain Classes; Application Classes (e.g., Describes the types of objects); Operations
Domain (Class) Model
Interaction Modelling
Application (Class) Model
Code Generation

Interaction modelling: overview

Interaction modelling: overview

HOW DO OBJECTS INTERACT?

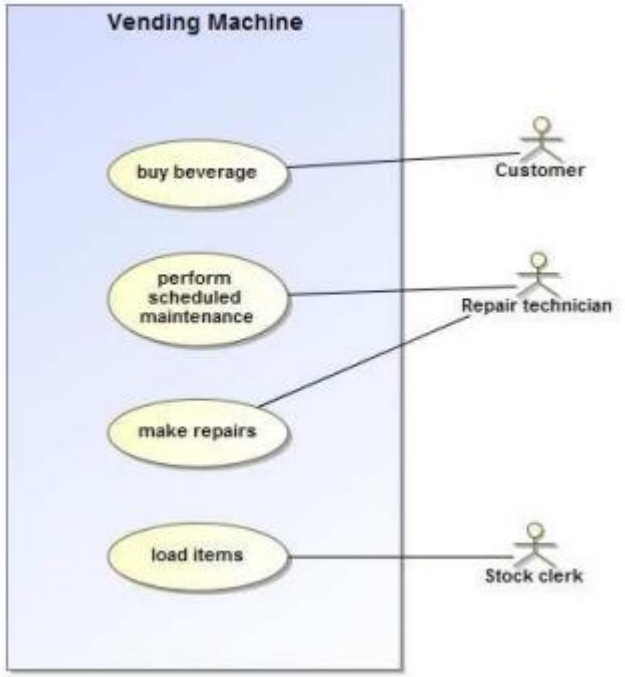
Interaction modelling: output

Interaction modelling: output


HOW?

Use case diagrams

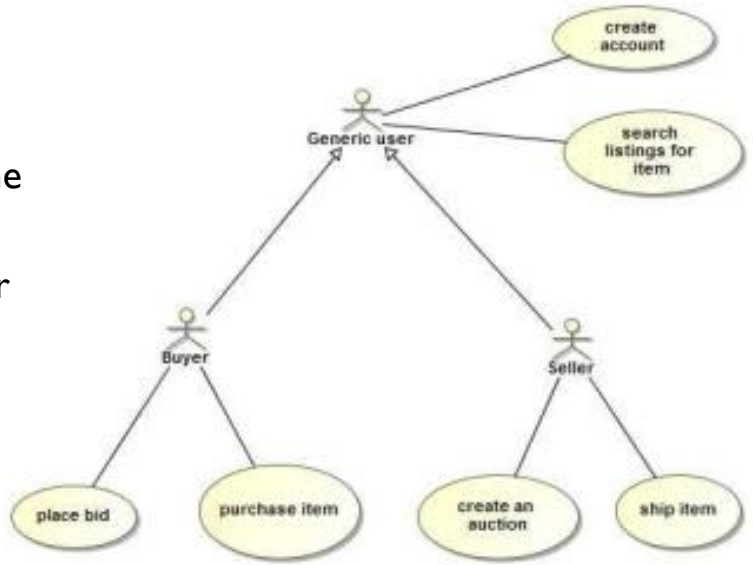
- UML has a graphical notation for summarizing use cases into use case diagrams
- A rectangle contains the use cases for a system with the actors listed on the outside
- The name of the system is written near a side of rectangle
- A name within an ellipse denotes a use case
- A "stick man" icon denotes an actor with the name placed below the icon
- Solid lines connect use cases to participating actors



Actor generalization

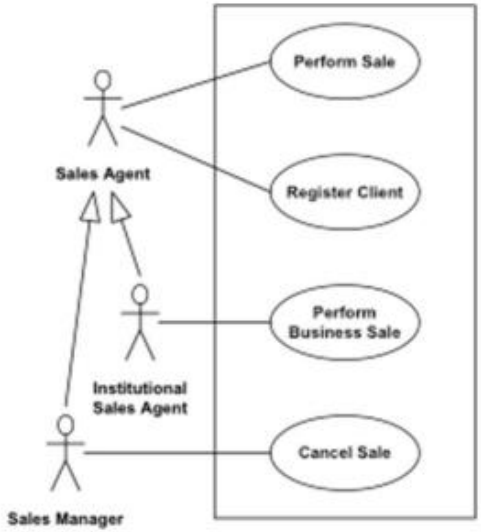
Generalization works in use case diagrams as well!

- The child actor inherits all use case associations from the parent
- Actor generalization should be used of the specific actor has more responsibility than the generalized one (i.e., associated with more use cases)
  - Example: *requirements management use case diagram*
  - duplicate behavior in both the buyer and seller, which includes "create an account" and "Search listings"
  - **Rather than having duplication**, use a **more general user** that has this behavior and then **the actors will "inherit" this behavior from the general user**



*Actor generalization should be used i**f the specific actor has more responsibility than the generalized one** (i.e. associated with more Use Cases)*

Actor generalization



*Actor generalization should be used i**f the specific actor has more responsibility than the generalized one** (i.e. associated with more Use Cases)*

Use case relationships -> linking use cases

- **For large applications complex use-cases can be built from smaller pieces!!!**

- **Linking enables flexibility in requirements specification**
  - Isolating functionality
  - Enabling functionality sharing
  - Breaking functionality into manageable chunks

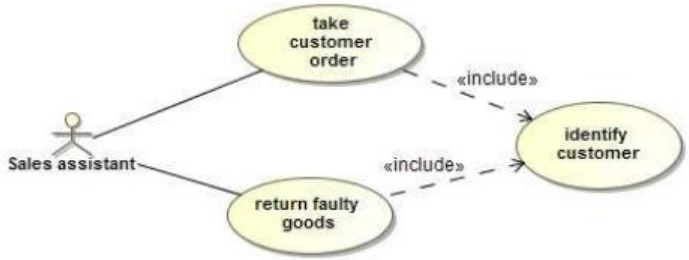
- Three linking mechanism are available in UML:
  - *Include*
  - *Extend*
  - *Generalization*

- And one grouping mechanism:
  - *Package*

Include

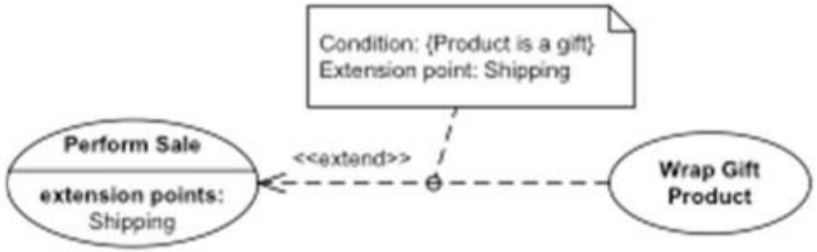
- **A use case can make use of another smaller use case**

- **Include** is used when:
  - **Decomposing complicated behavior**
  - **Centralizing common behavior // incorporates the behavior of another use case (e.g., subroutines)**
- Factoring a use case into pieces is appropriate **when the pieces represent significant behavior units**
- **The base use case explicitly incorporates the behavior of another use case at a location specified in the base**



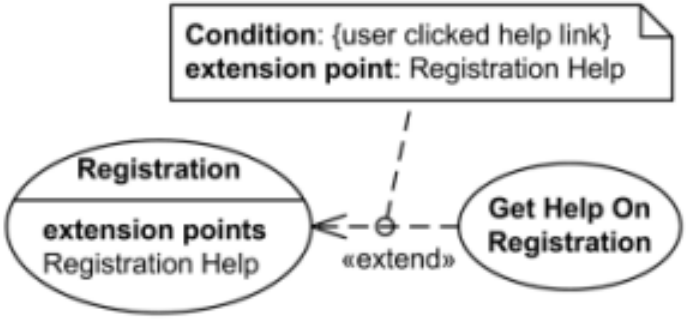
Extends

- **The base use case can incorporate another use case at certain points, called extension points**
- Adds an "extra behaviour" to a base use cases used **in the situation in which some initial capability is defined and later features are added modularly**
- **Base use case is meaningful on its own, it is independent of the extension**.
- Extension **typically defines optional behavior that is not necessarily meaningful by itself**
- Note the direction of the arrow
  - **The base use-case does not know which use-case extends it**



Extends

- Use case "Registration" is meaningful on its own.
- It could be optionally extended with "get Help On registration"

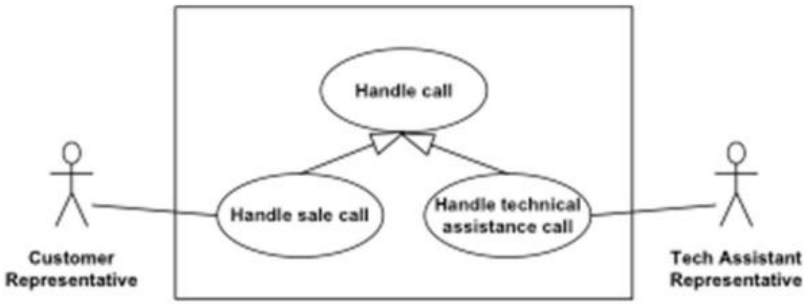


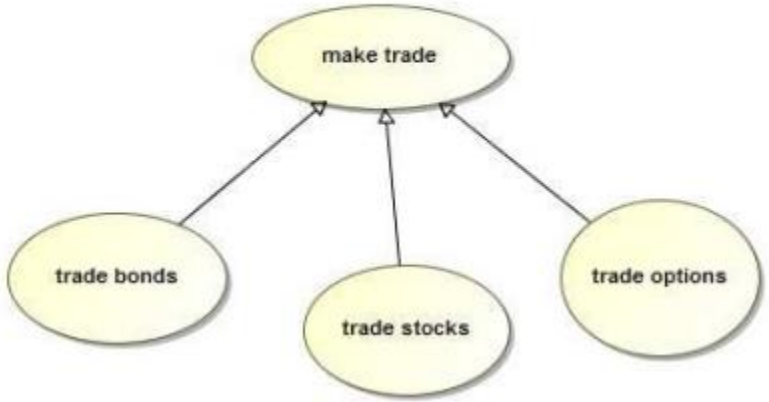
- **Extension points:** specify the location at which the behavior of the base use case may be extended.
- Extension points can have a condition attached.
- The extension behavior occurs only if the condition is true when the control reaches the extension point.

Generalization

- **The child Use case inherits the behavior of the parent Use case:**
  - **The interaction** (described in the textual description)
  - **Use case links** (associations, include, extend, generalization)

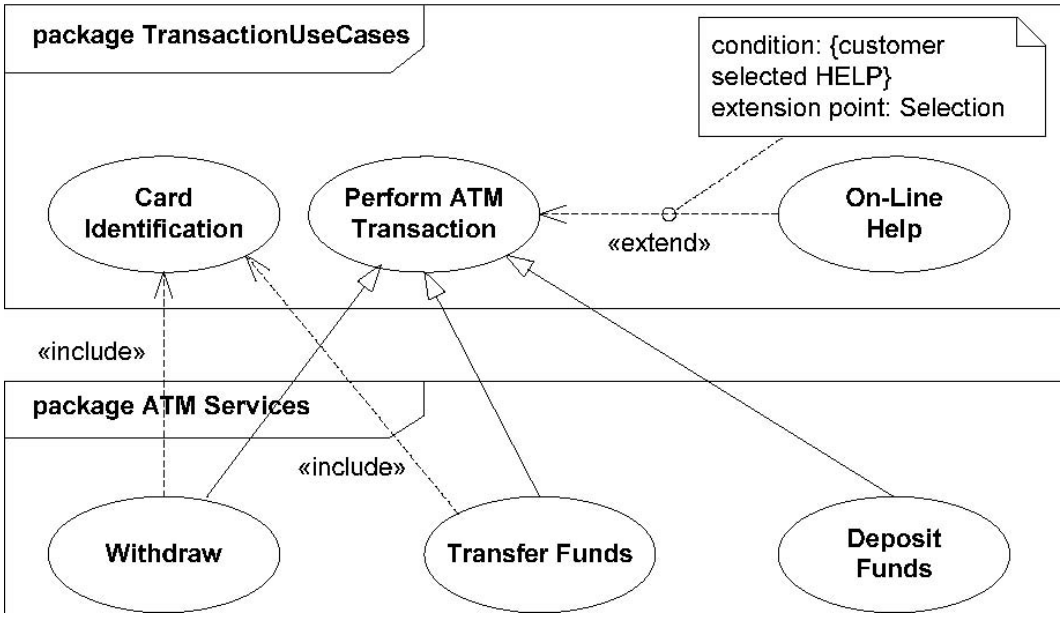
- **The child Use case can substitute the parent Use case**
  - Overriding occurs through the **textual description**





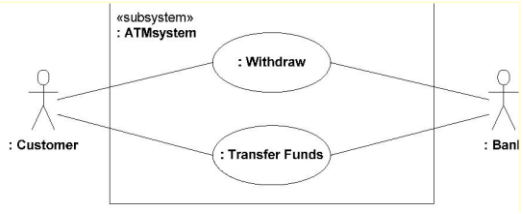
Packaging: use cases organized in packages

- A **use-case package** is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
- The detailed behavior defined by a use case is notated according to the chosen description technique, in a separate diagram or textual document. Operations and attributes are shown in a compartment within the use case.



Extension points may be listed in a compartment of the use case with the heading extension points.

The description of the locations of the extension point is given in a suitable form, usually as ordinary text, but can also be given in other forms, such as the name of a state in a state machine, an activity in an activity diagram, a precondition, or a postcondition.



https://www.site.uottawa.ca/~tcl/gradtheses/mnojoumian/ThesisFiles/FinalSpec/UML/16.3.6.html

Use case description

Name
Actors
Triggers
Preconditions
Post-conditions
Success scenario
Alternative flows

may also have additional row "Stakeholders" (after "Actors")

may also have additional row "Exceptions" (sometimes "Alternative flow OR Exceptions")

Use case description

Name
Actors
Triggers
Preconditions
Post-conditions
Success scenario
Alternative flows

What starts the use-case?

Examples:

Customer reports a claim

Customer inserts card

System clock is 10:00

Use case description

Name
Actors
Triggers
Preconditions
Post-conditions
Success scenario
Alternative flows

What the system needs to be true before running the use-case.

Examples

User account exists

User has enough money in his/her account

There is enough disk space

Use case description

Name
Actors
Triggers
Preconditions
Post-conditions
Success scenario
Alternative flows

A post-condition is the outcome of the use-case.

Examples

Money was transferred to user account

User is logged in

The minimal things a system ensures (even in case of failures)

Example

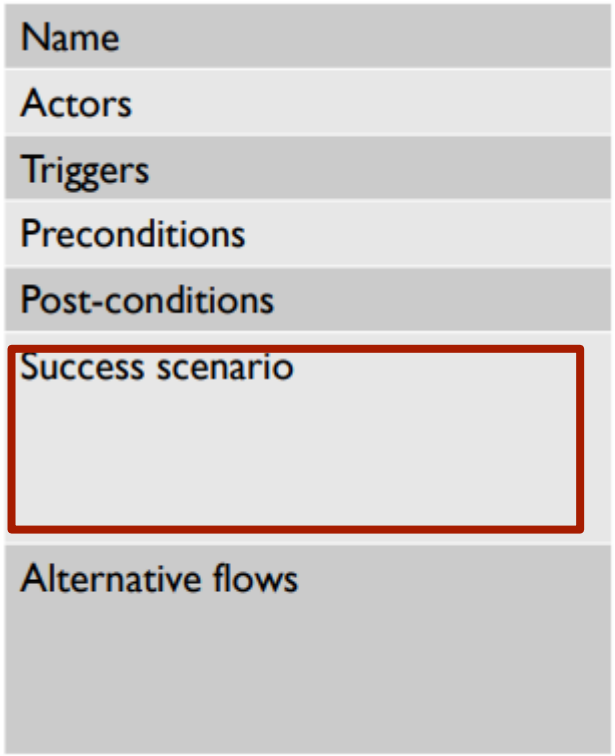
Money is not transfer unless the customer …

What happens when the Use Case concludes successfully?

Example

The money is transferred

Use case description



- **The success scenario is the main storyline of the use-case**
- **It is written under the assumption that everything is okay, no errors or problems occur, and it leads directly to the desired outcome of the use-case**

- **It is composed of a sequence of action steps**

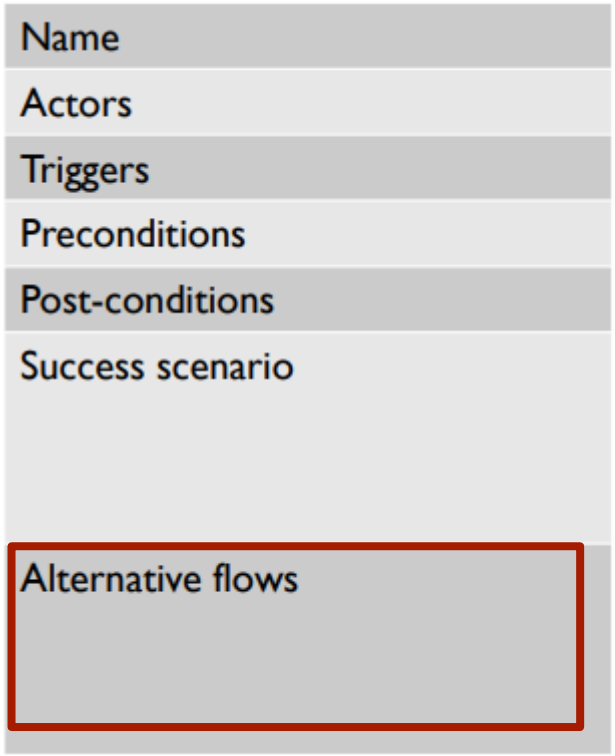
  Example

  1. Administrator enters course name, code and description

  2. System validates course code

  3. System adds the course to the db and shows a confirmation message

Use case description



- **Used to describe exceptional functionality**
  - **Errors**

    Case did not eject properly

    Any network error occurred during steps 4-7

    Any type of error occurred
- **Unusual or rare cases**
  - *Credit card is defined as stolen*
  - *User selects to add a new word to the dictionary*
- **Endpoints**
  - *The system detects no more open issues*
- **Shortcuts**
  - *The user can leave the use-case by clicking on the "esc" key*

Alternatives to use cases

- **User stories (agile methods)**
  - Written in the language of the user
  - Captures: Who, what, why

    Example:

    As a student, I want to search all courses in my Masters that are offered in the current semester and that I have not yet passed, in order to register to some of them.

    What is the difference between them? → User Story vs Use Case for Agile Software Development

- **Storyboards (i.e. "detailed" user stories)**
  - Sketch of how a user will perform a task
  - Shows the interactions and relevant objects (or screens) at each step
  - Used in UI Design (but not only)

To sum up

- Use case models include
  - Use case diagrams
  - (Textual) use case descriptions

- Simple use case template available in the "Lectures" section of course web page
- Guidelines given in Section 7.1.4 Guidelines for Use Case Models pages 135-136 of the book "Object-oriented modeling and design with UML"

Interaction Modelling: Sequence Diagrams

Anastasija Nikiforova

Institute of Computer Science

Based on materials developed by Marlon Dumas, Fabrizio Maria Maggi, Luciano García-Bañuelos

Interaction modelling: Detailing Use Cases with Scenarios

Use Case: Buy a beverage

Summary: The vending machine delivers a beverage after a customer selects and pays for it.

Actors: Customer

Preconditions: The machine is waiting for money to be inserted.

Description: The machine starts in the waiting state in which it displays the message "Enter coins." A customer inserts coins into the machine. The machine displays the total value of money entered and lights up the buttons for the items that can be purchased for the money inserted. The customer pushes a button. The machine dispenses the corresponding item and makes change, if the cost of the item is less than the money inserted.

Exceptions:

Canceled: If the customer presses the cancel button before an item has been selected, the customer's money is returned and the machine resets to the waiting state.

Out of stock: If the customer presses a button for an out-of-stock item, the message "That item is out of stock" is displayed. The machine continues to accept coins or a selection.

Insufficient money: If the customer presses a button for an item that costs more than the money inserted, the message "You must insert $*nn.nn* more for that item" is displayed, where *nn.nn* is the amount of additional money needed. The machine continues to accept coins or a selection.

No change: If the customer has inserted enough money to buy the item but the machine cannot make the correct change, the message "Cannot make correct change" is displayed and the machine continues to accept coins or a selection.

Postconditions: The machine is waiting for money to be inserted.

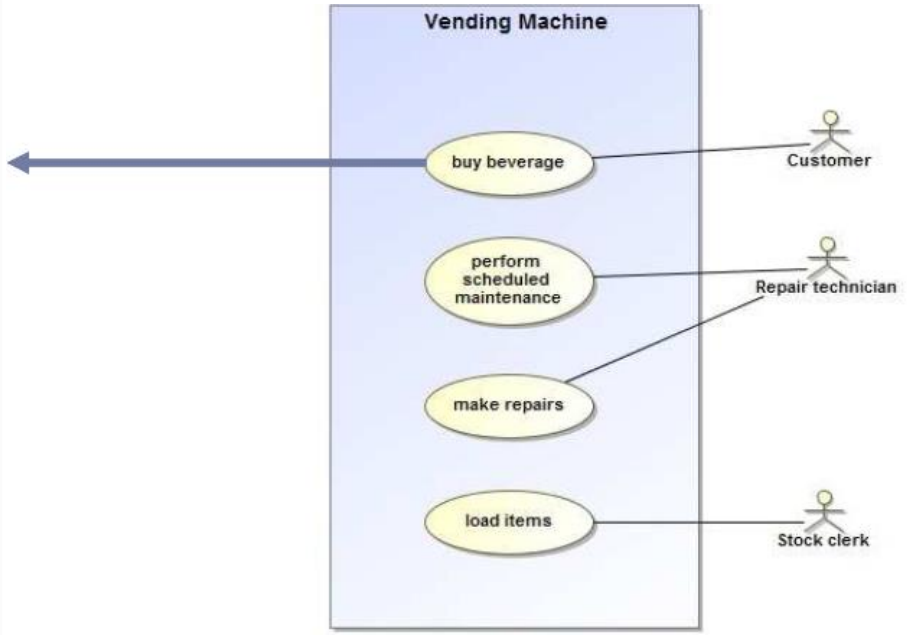
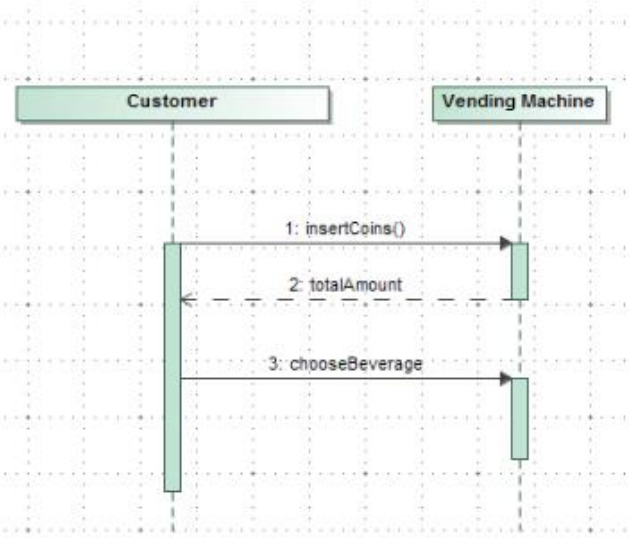
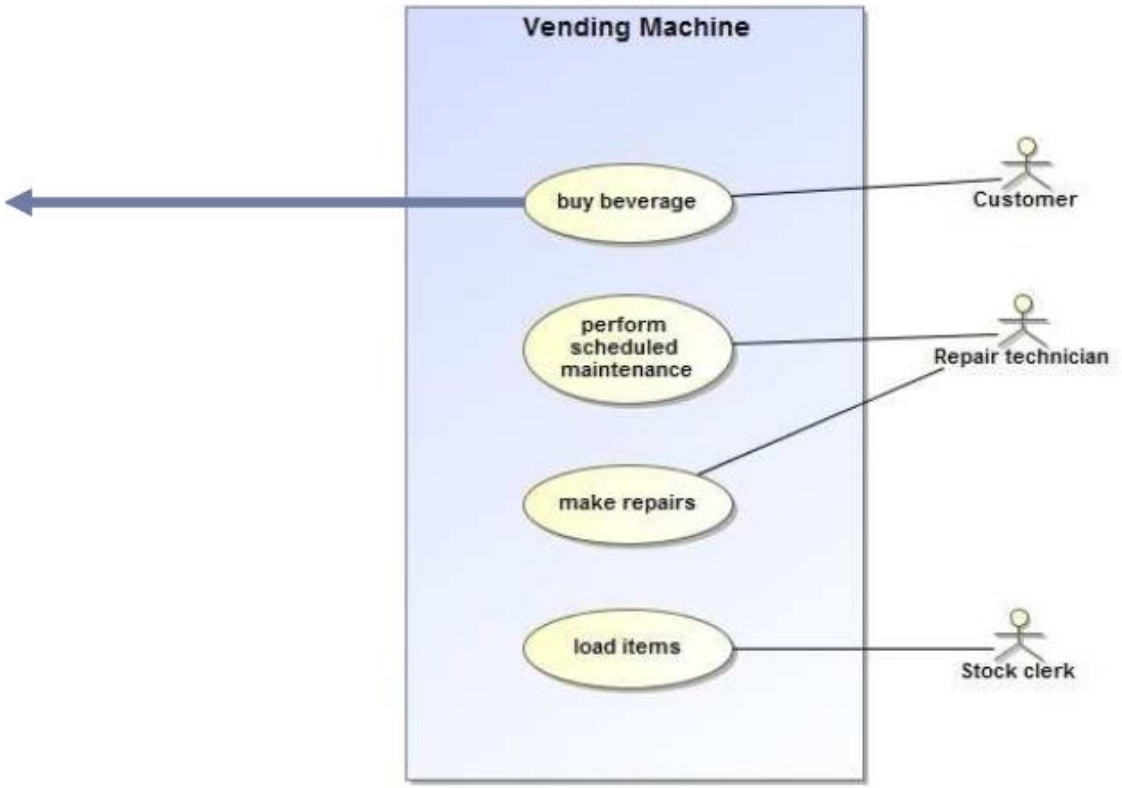


Figure 7.2 Use case description. A use case brings together all of the behavior relevant to a slice of system functionality.

Interaction Modelling





Sequence Diagrams as a support to define and document interfaces

!!! The creation of a sequence diagram should be driven

by the objective of writing interfaces* for the classes of a domain model by reasoning on their interactions in the implementation of the required functionalities

***In UML modeling, **interfaces are model elements that define sets of operations that other model elements, such as classes, or components must implement**. An implementing model element realizes an interface by overriding each of the operations that the interface declares.

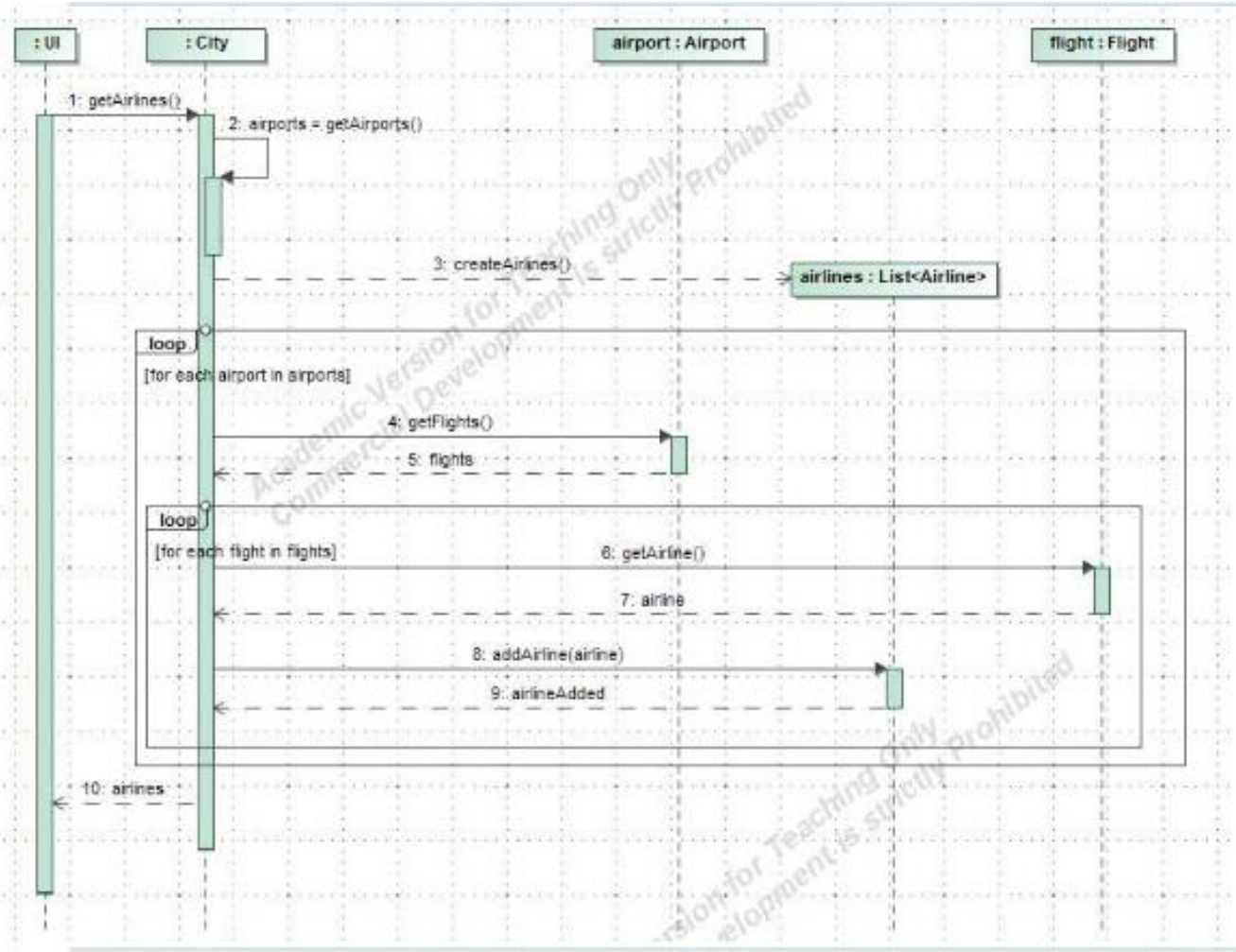
Interfaces - IBM Documentation

Sequence Diagrams as a support to define and document interfaces

- A sequence diagram **is used to support the definition of the interfaces** through the **identification of the operations that the classes need to expose for implementing the required functionalities**

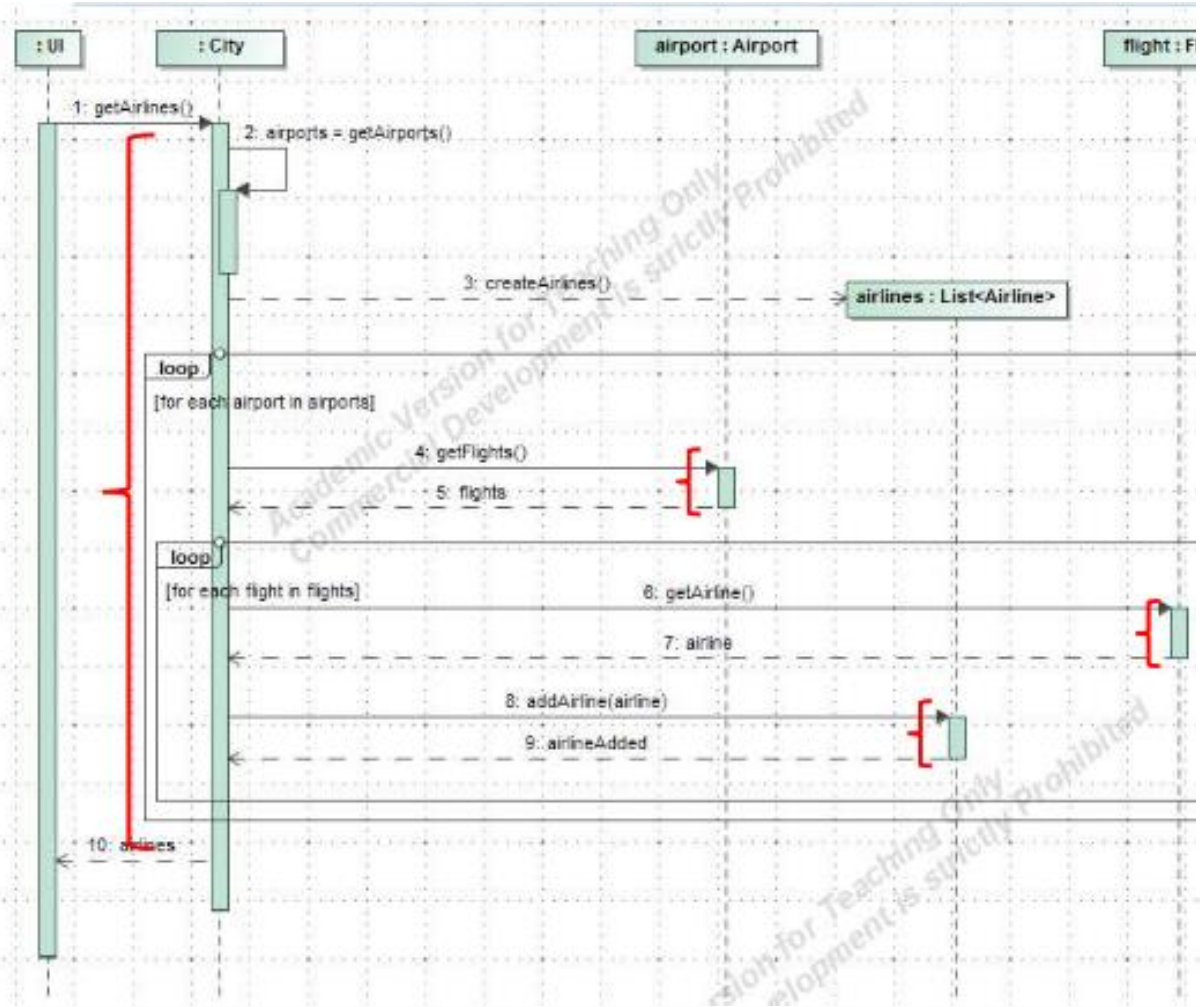
- A sequence diagram **is used to document where the identified operations come into play in the implementation of the required functionalities**

Sequence Diagrams



Sequence diagrams show operation calls

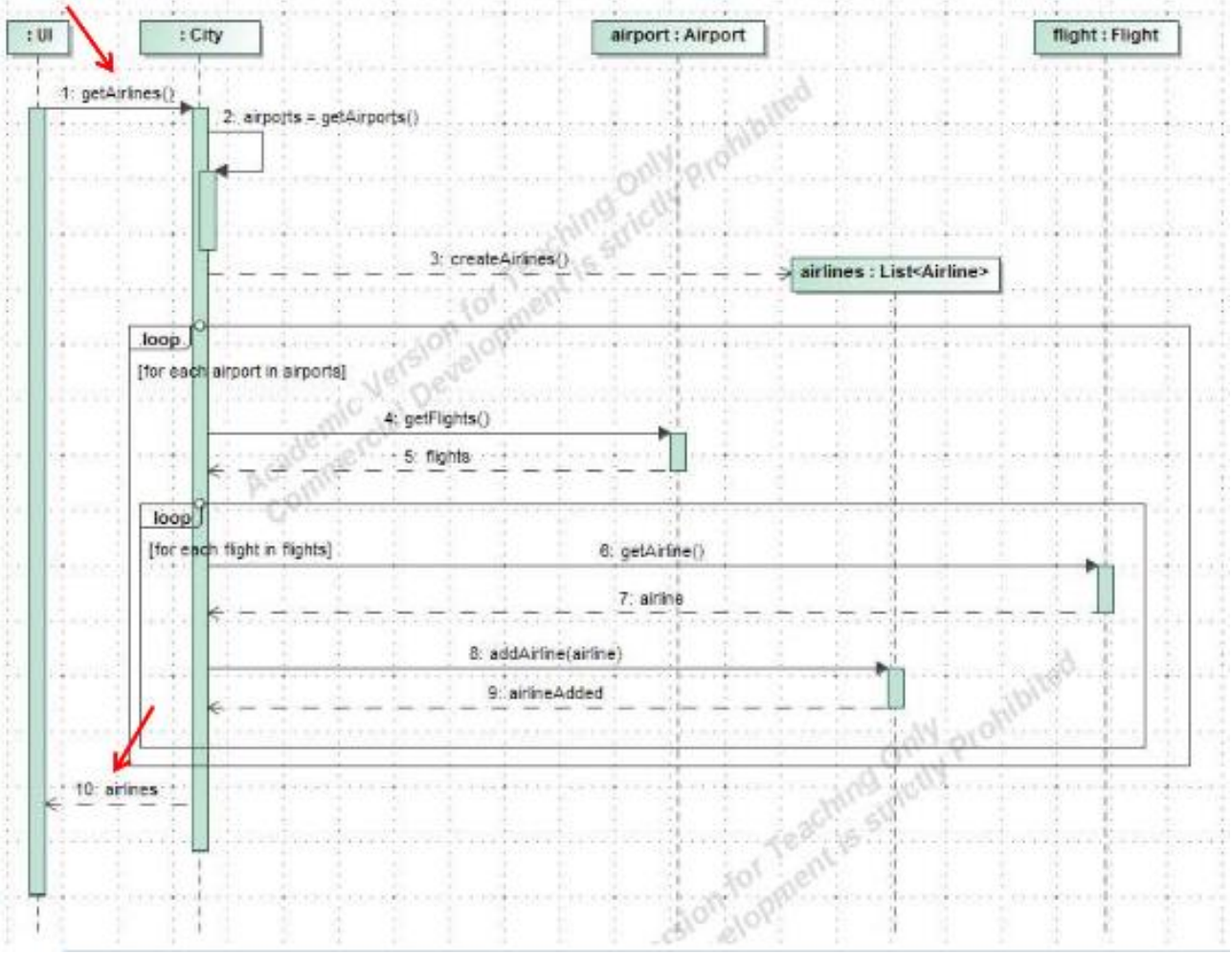
Sequence Diagrams



The period of time of an object's execution (a thin rectangle) is called **activation** OR focus of control OR **execution specification** OR execution occurrence.

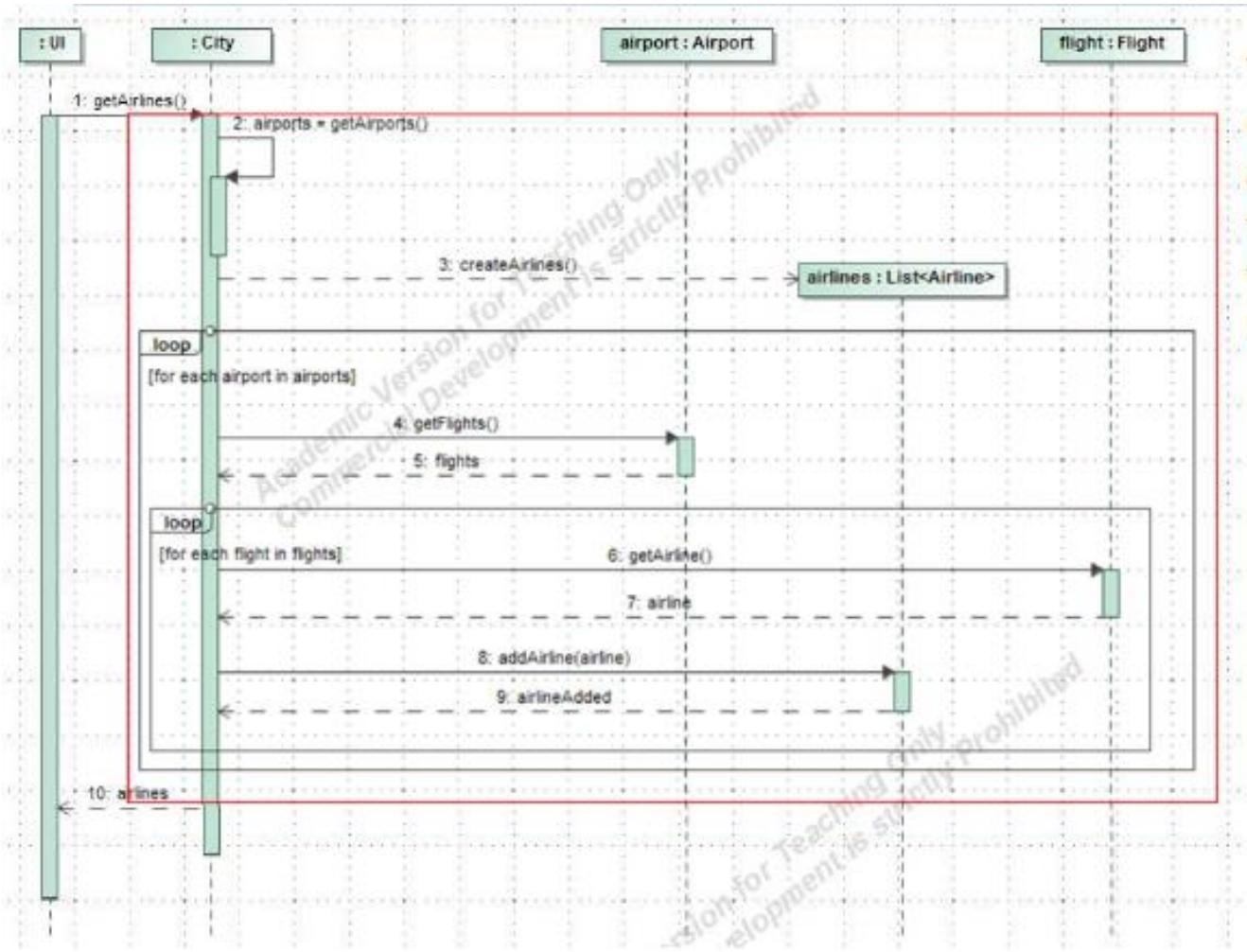
An activation shows the time period during which a call of an operation is processed including the time, when the called operation invoke others operations

Sequence Diagrams



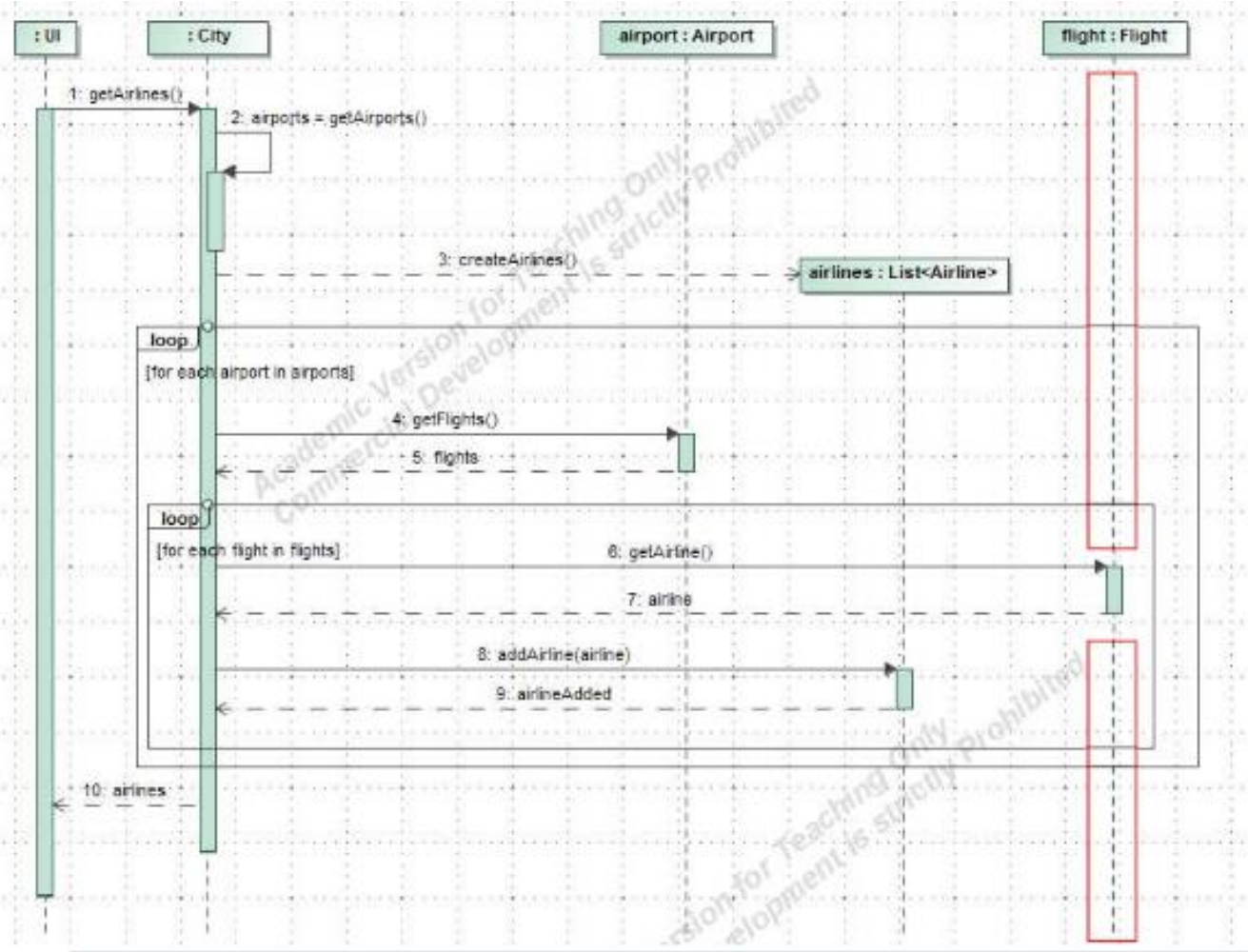
An activation has a call arrow coming into its top and a return arrow leaving its bottom

Sequence Diagrams



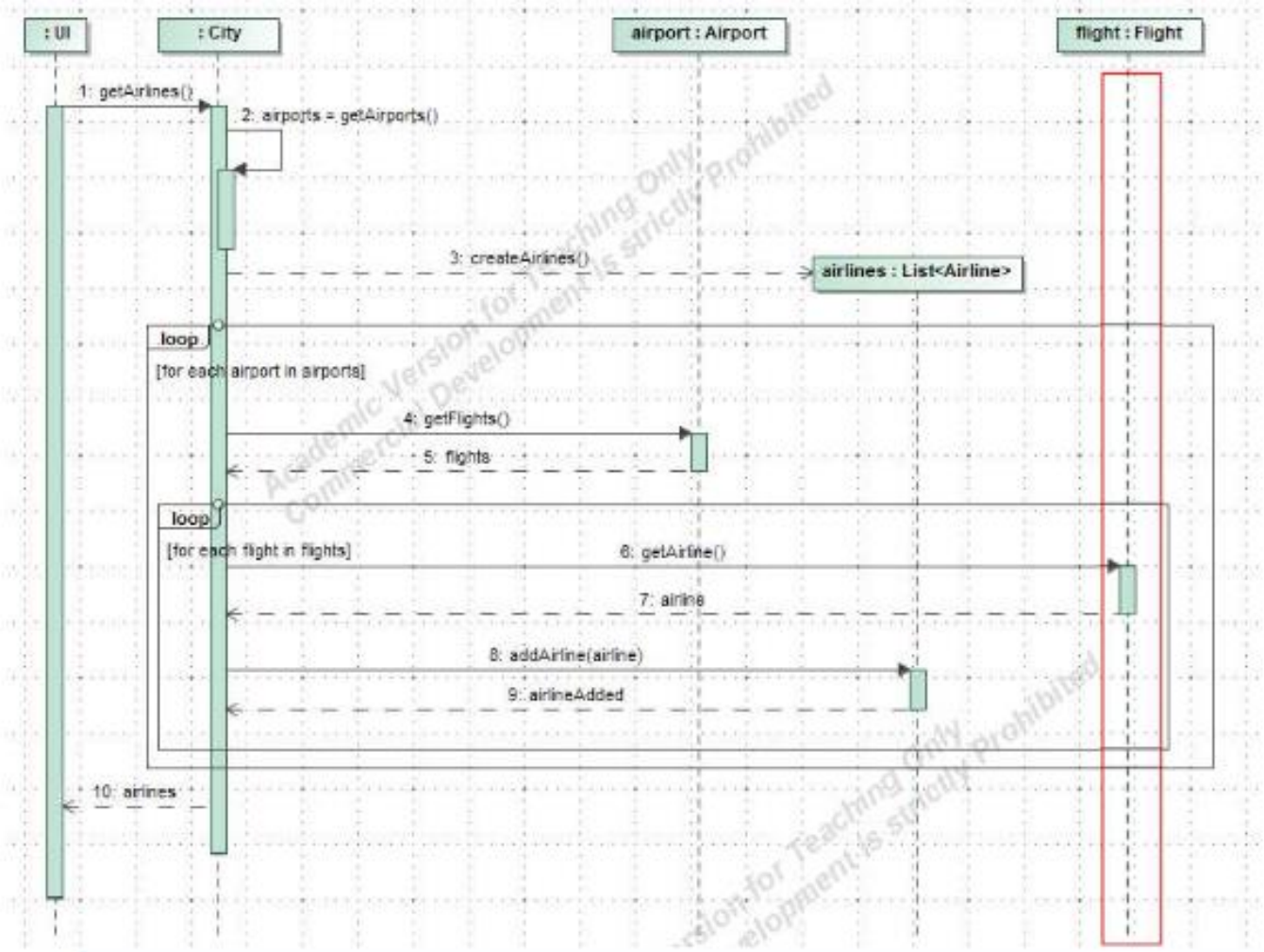
The body of the operation is made of all the interactions that occur between the call arrow and the return arrow

Sequence Diagrams



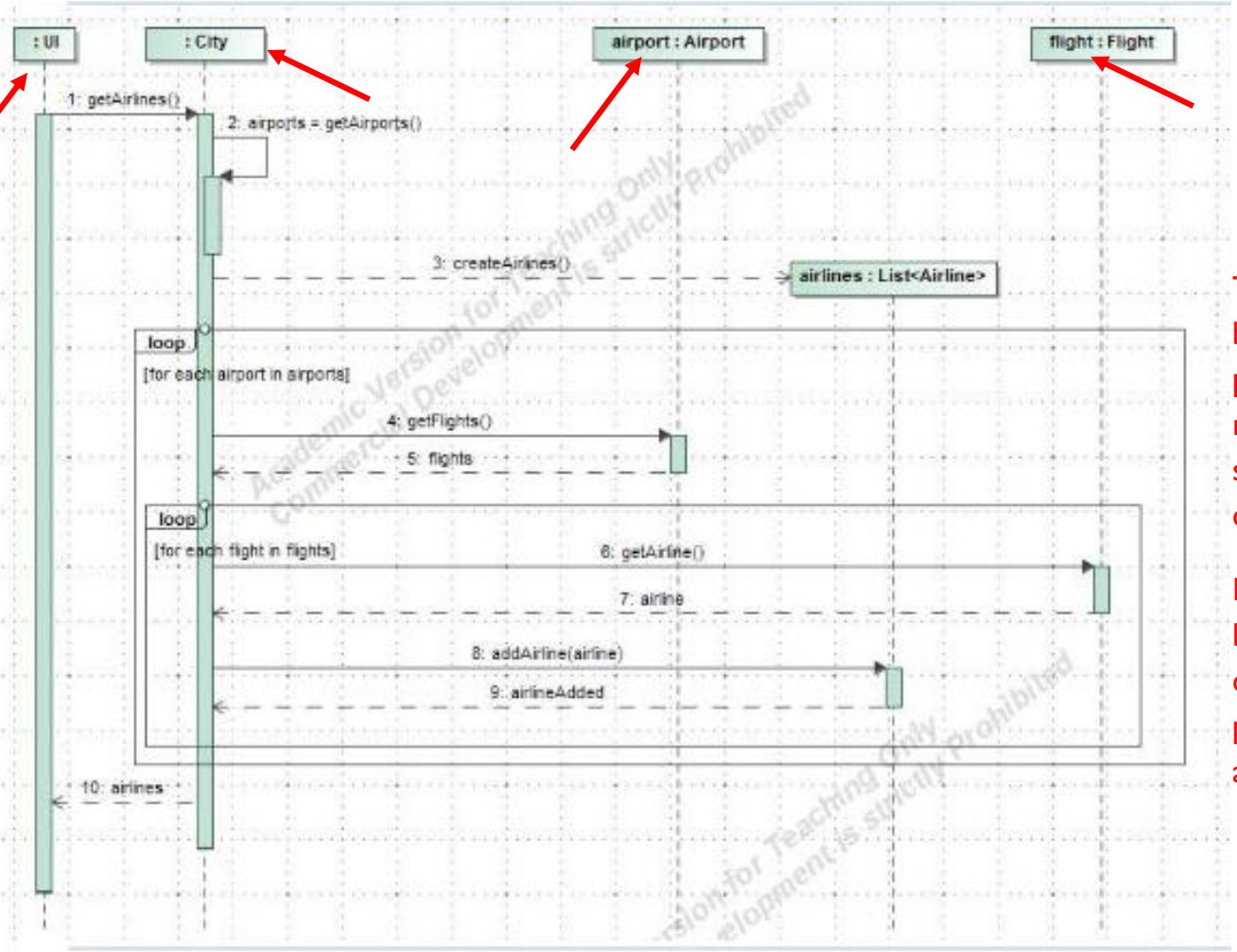
The period of time when an object exists but is not active is shown as a dashed line

Sequence Diagrams



The entire period of time when an object exists is called *lifeline*

Sequence Diagrams

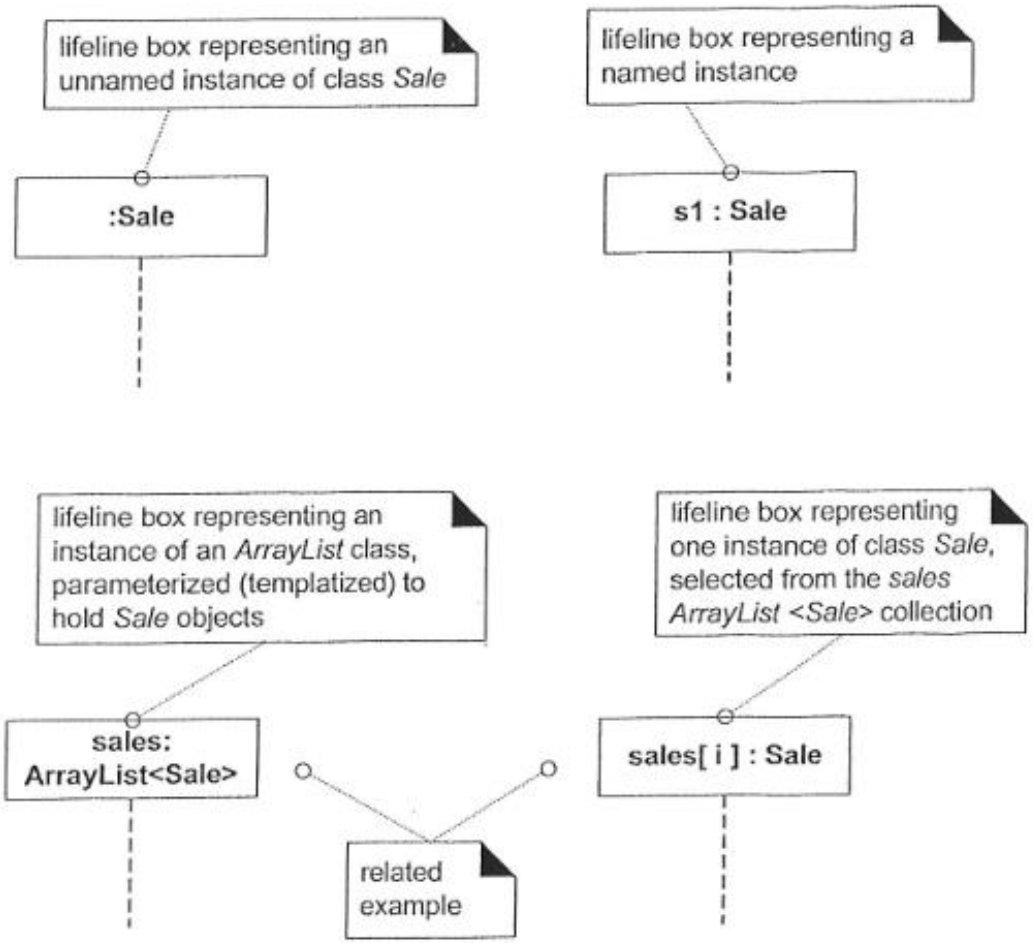


Lifeline box representing (1) an unnamed instance of class, (2) – named instance

Their precise UML definition is subtle, but informally they represent the **participants** in the interaction-related parts defined in the context of some structure diagram, such as a class diagram.

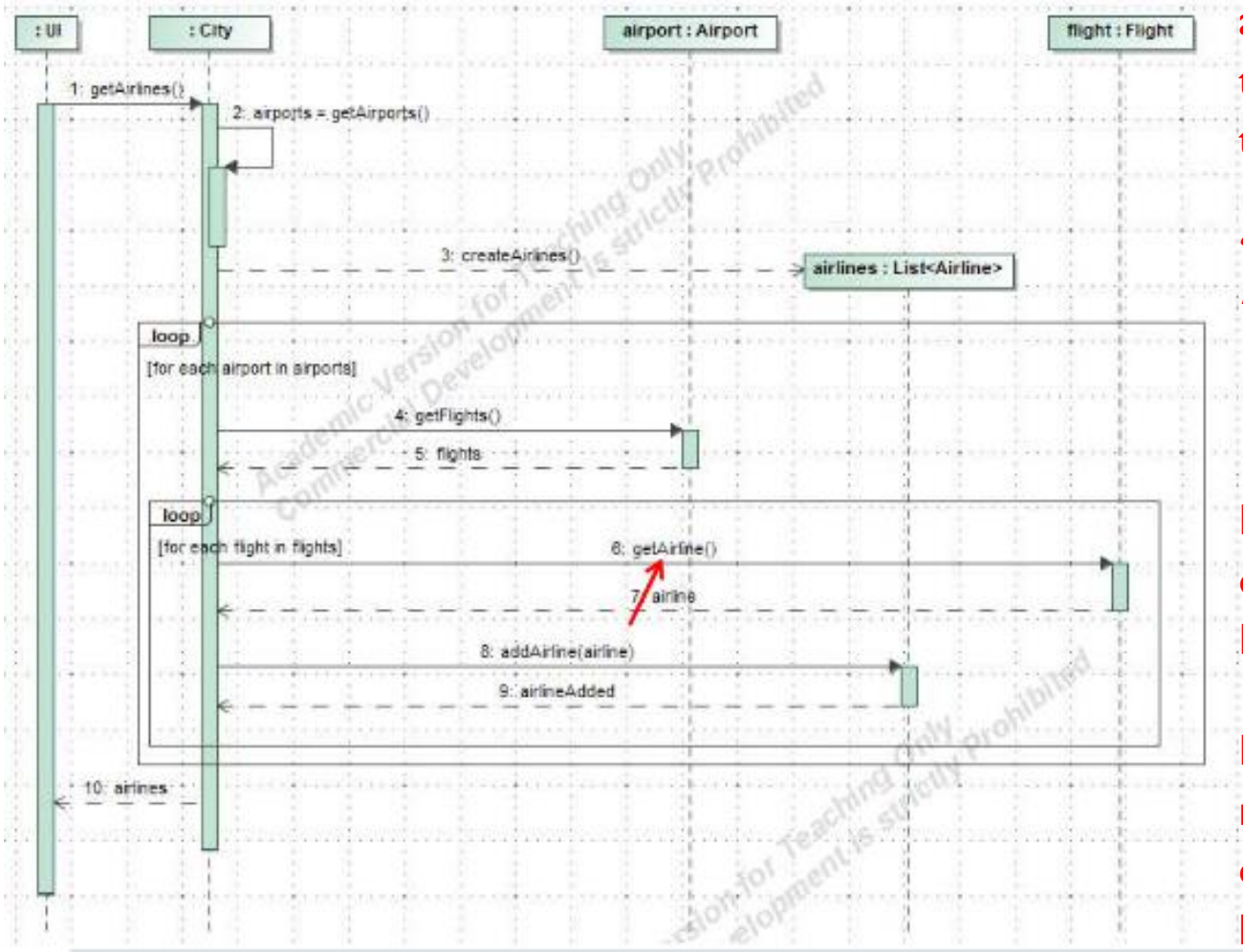
Not precisely accurate to say that a lifeline box equals an instance of a class, but informally & practically, the participants will often be interpreted as such

Sequence Diagrams



Lifeline box may represent more than only name or unnamed instance of a class (depends on participants in interactions)

Sequence Diagrams



The notation for a call is an arrow from the calling activity to the activation created by the call

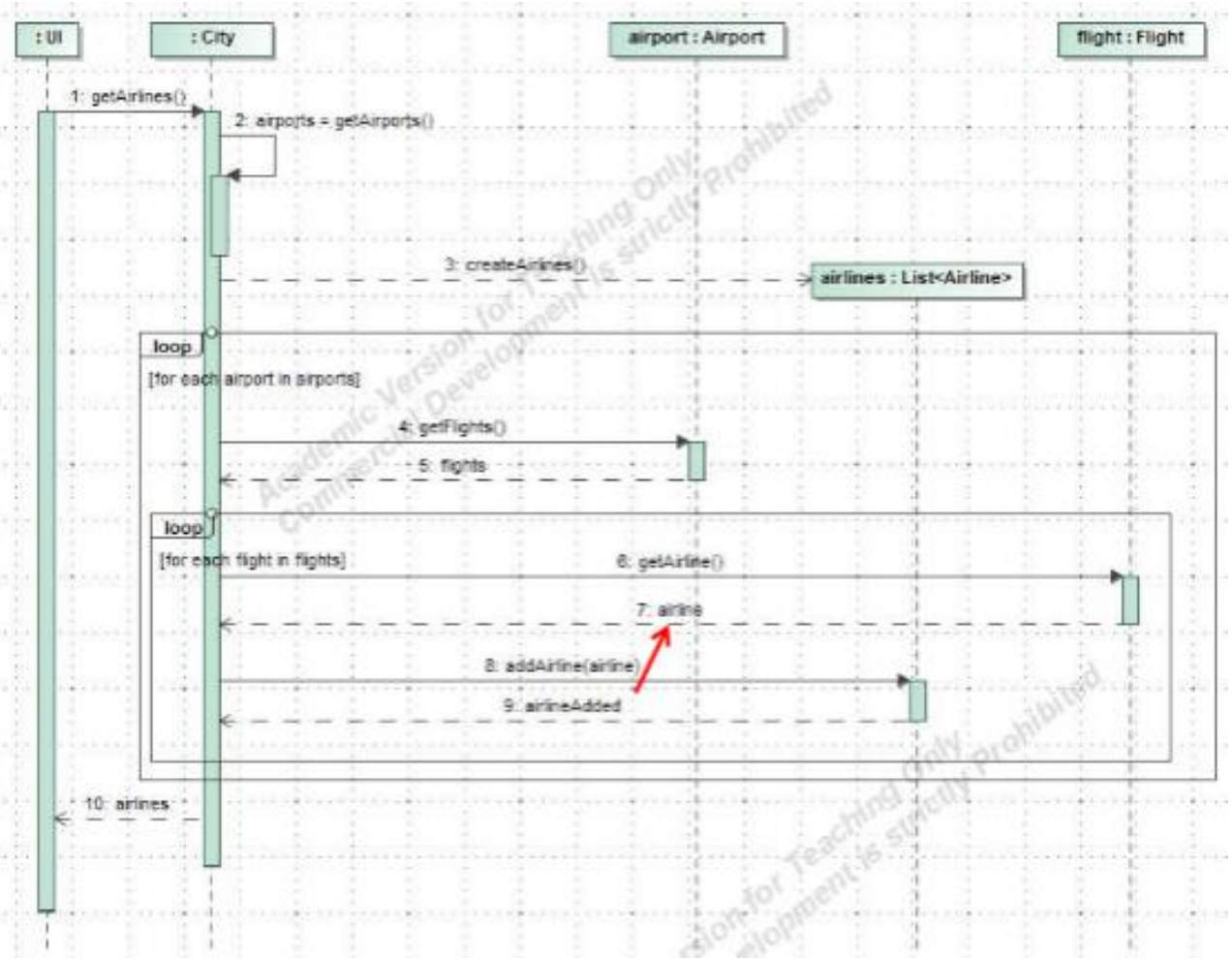
also "message" (e.g. MagicDraw), in some cases "asynchronous message" or "call message")

Defines a particular **communication between Lifelines of an Interaction**.

Is **a kind of message that represents an invocation of operation of target lifeline**.

What is Sequence Diagram? (visual-paradigm.com)

Sequence Diagrams



A return of a call is a dashed arrow from the bottom of the called activation to the calling activation

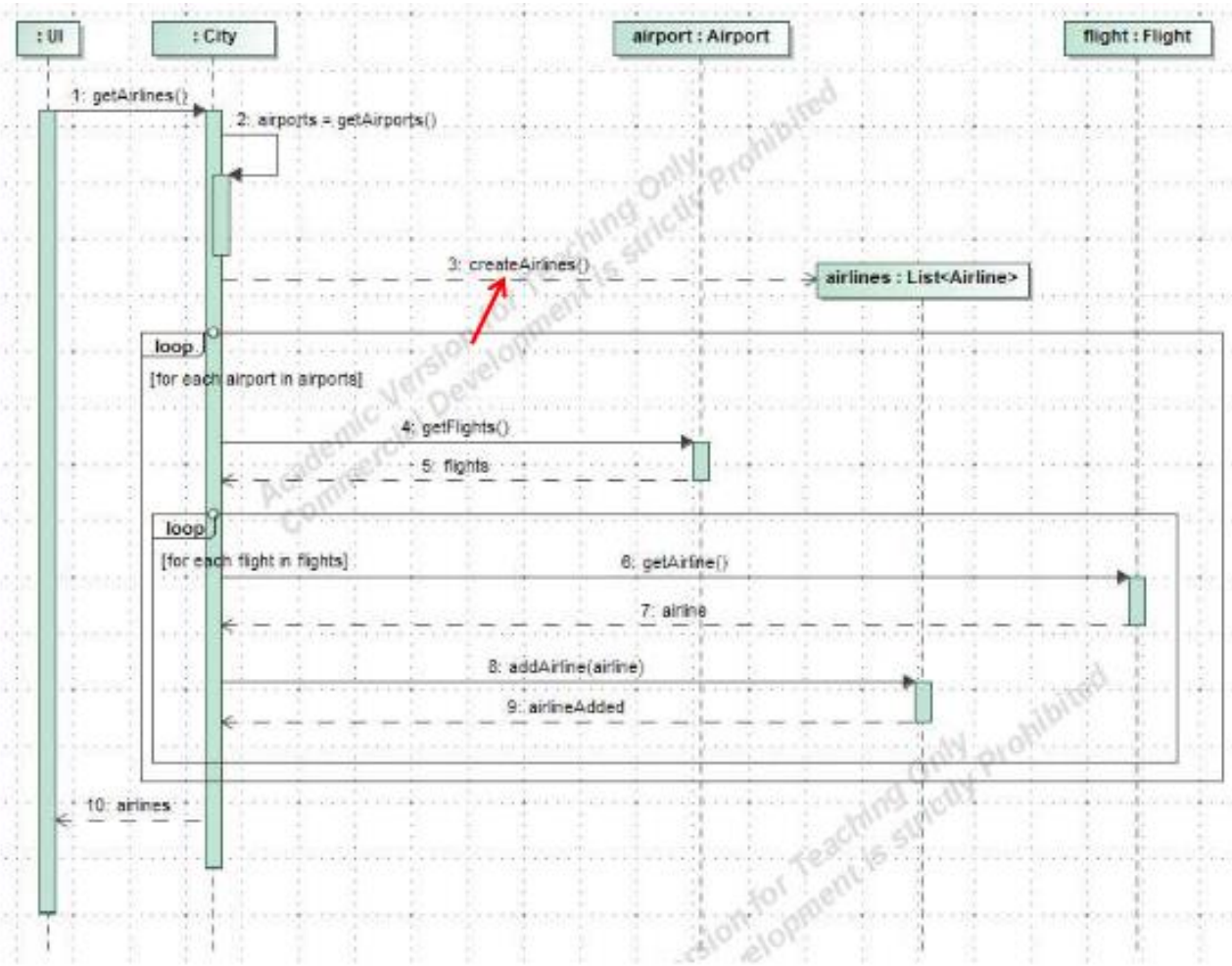
(also "reply message" (e.g., MagicDraw), "object creation message")

Defines **a particular communication** between Lifelines of an Interaction.

Is **a kind of message that represents the pass of information back to the caller of a corresponded former message**.

What is Sequence Diagram? (visual-paradigm.com)

Sequence Diagrams



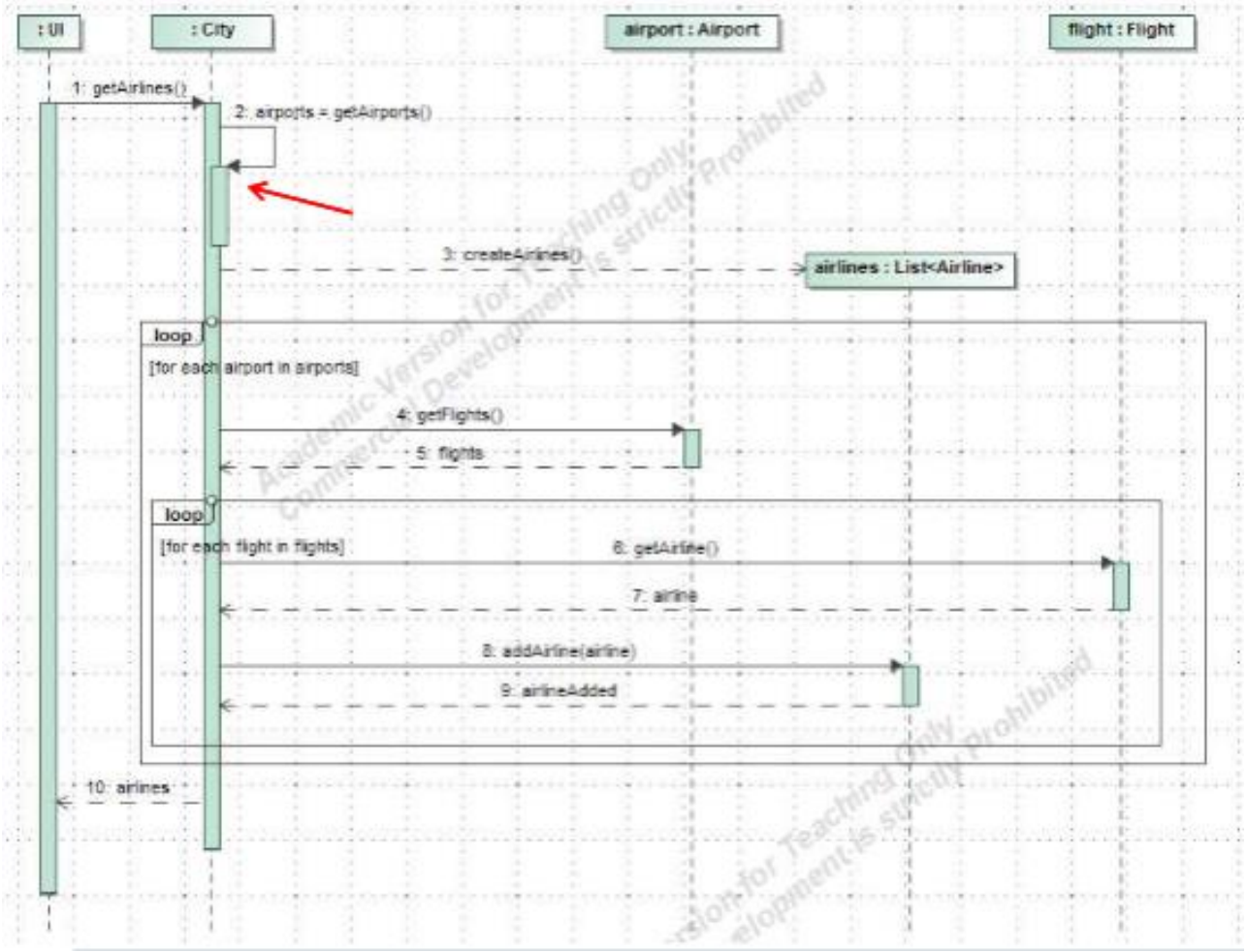
If an object does not exist at the beginning of the sequence diagram, it must be created. UML shows creation by placing the object symbol at the head of the dashed arrow representing the call that creates the object.

I.e., defines a particular communication between Lifelines of an Interaction.

Is **a kind of message that represents the instantiation of (target) lifeline.**

What is Sequence Diagram? (visual-paradigm.com)

Sequence Diagrams



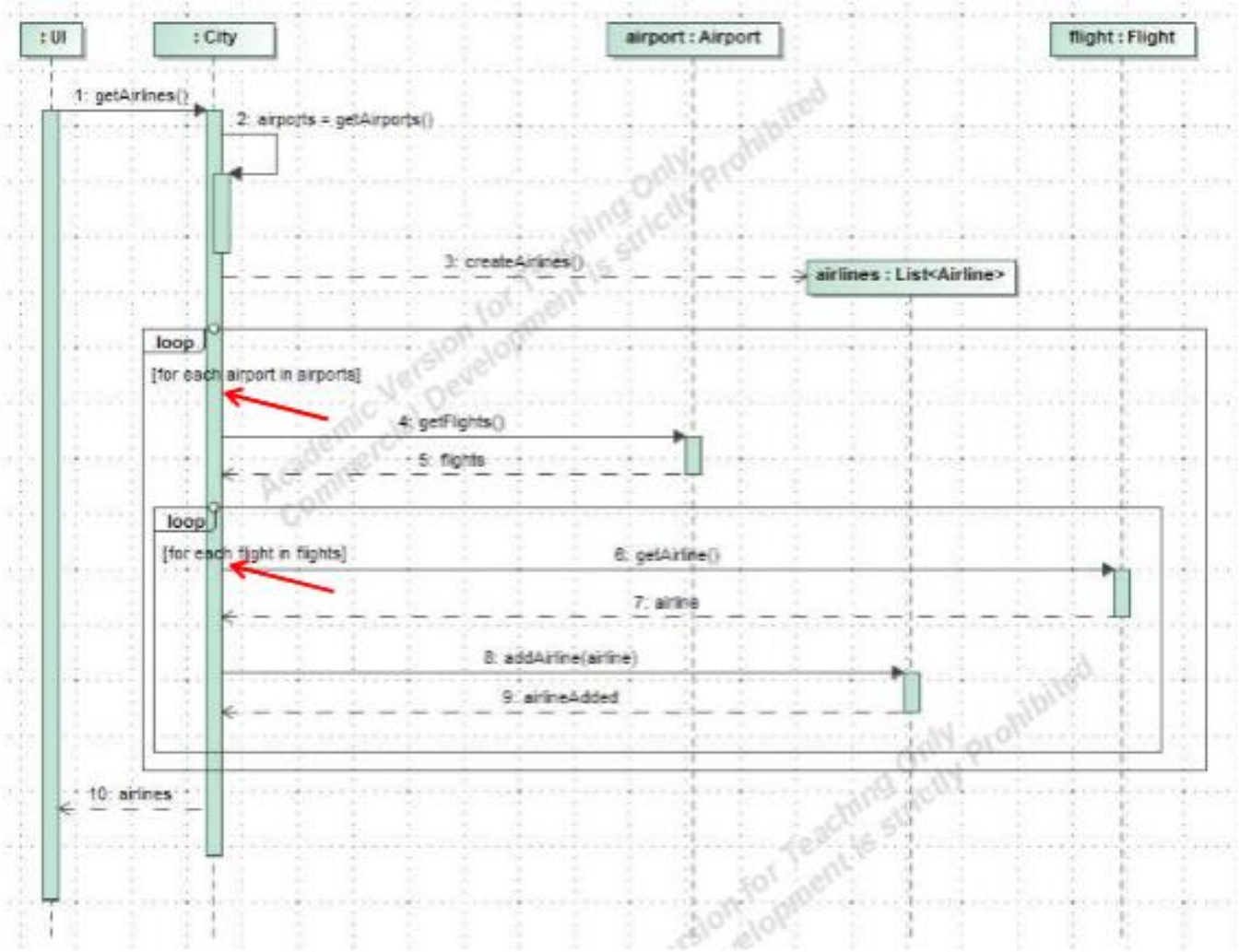
An object can call its own operations ("self" or "this" calls),

Advanced Sequence Diagrams

- Sequence diagrams also provide "**frame operators**"

Frame operator	**Meaning**
loop	Alternative fragment for mutual exclusion conditional logic expressed in the guards.
alt	Loop fragment while guard is true. Can also wríLe loop(n) to indicate looping n times. There is discussion that the specification will be enhanced to define a FOR loop, such as loop(i, 1, 10)
opt	Optional fragment that executes if guard is true

Sequence Diagrams: loop



Loops

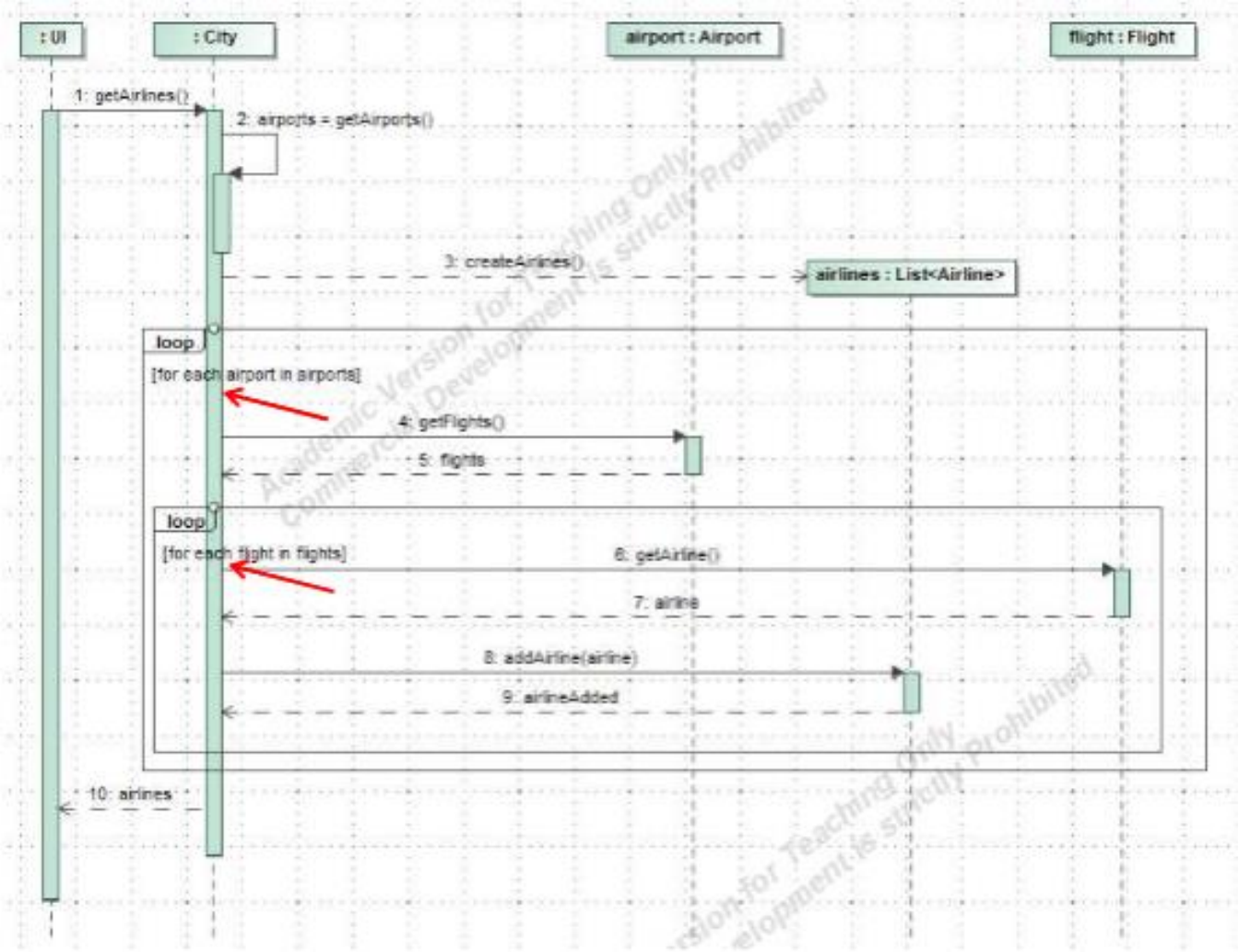
To support conditional and looping constructs, the UML uses **frames**.

Frames are regions or fragments of the diagrams that have:

(1) operator or **label** (such as loop)

(2) a **guard** (conditional clause)

Sequence Diagrams: loop



Loops

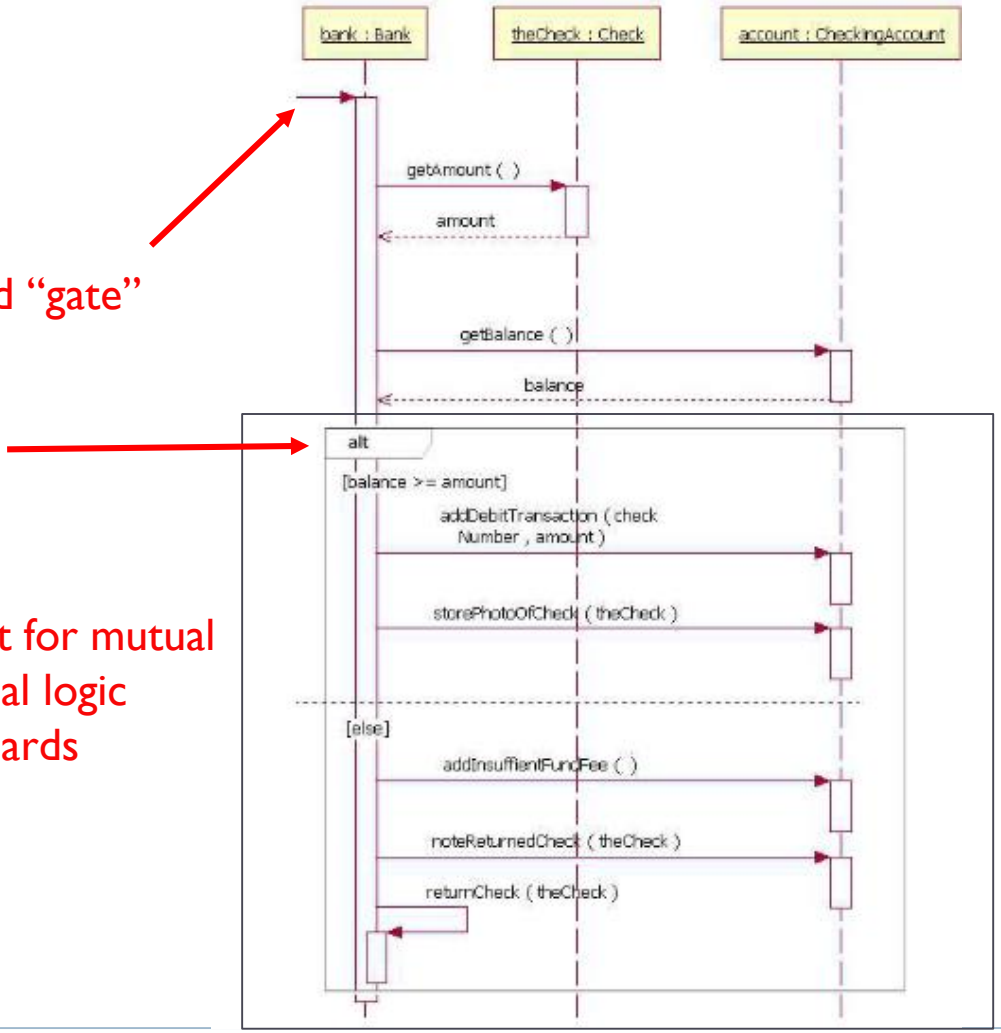
To support conditional and looping constructs, the UML uses **frames**.

Frames are regions or fragments of the diagrams that have:

(1) operator or **label** (such as loop)

(2) a **guard** (conditional clause)

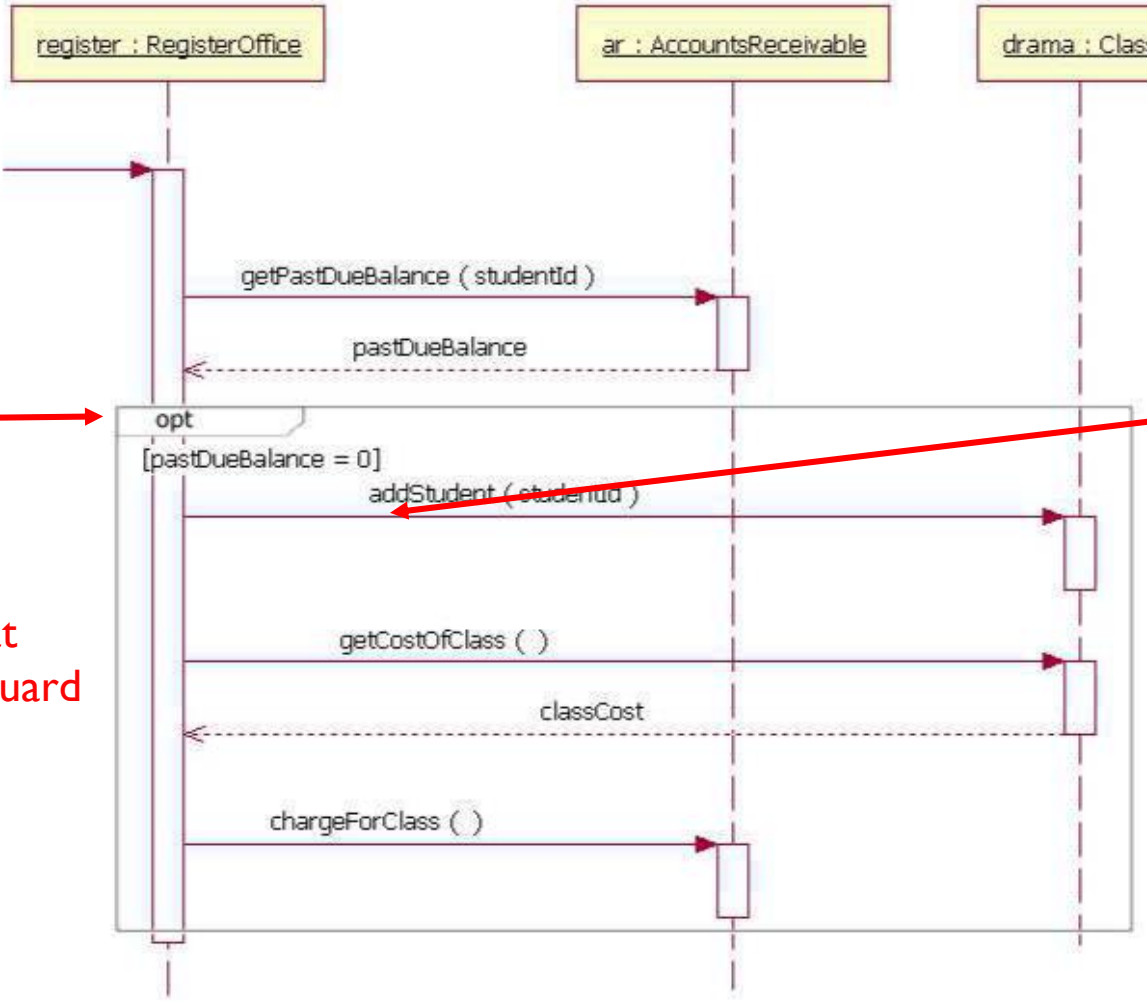
Advanced Sequence Diagrams: alt



also called "gate"

alt
alternative fragment for mutual exclusion conditional logic expressed in the guards

Advanced Sequence Diagrams: opt



opt
Optional fragment that executes if guard is true

Also called diagram frames or interaction frames

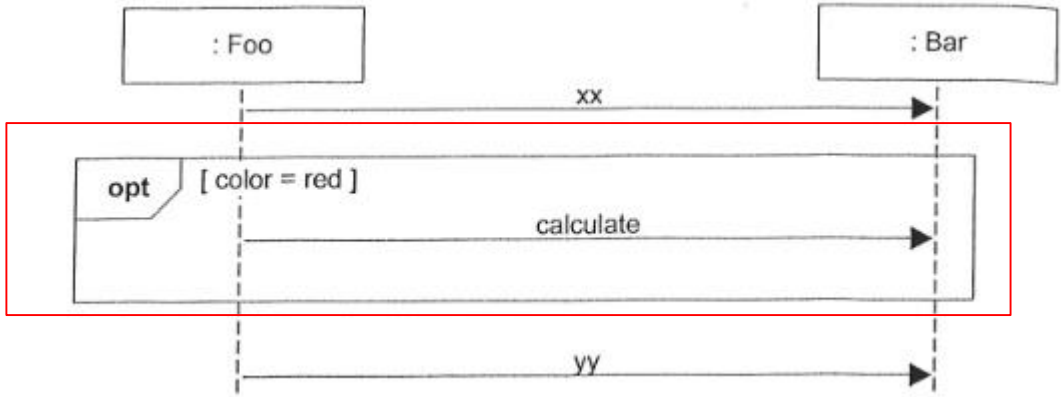
The **[boolean test]*** guard should be placed over the lifeline to which it belongs

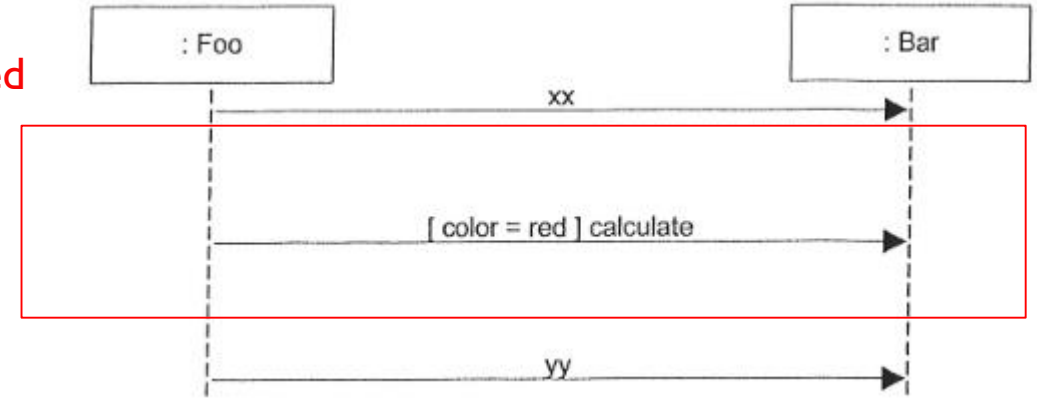
* e.g., [color = red]

Advanced Sequence Diagrams: opt

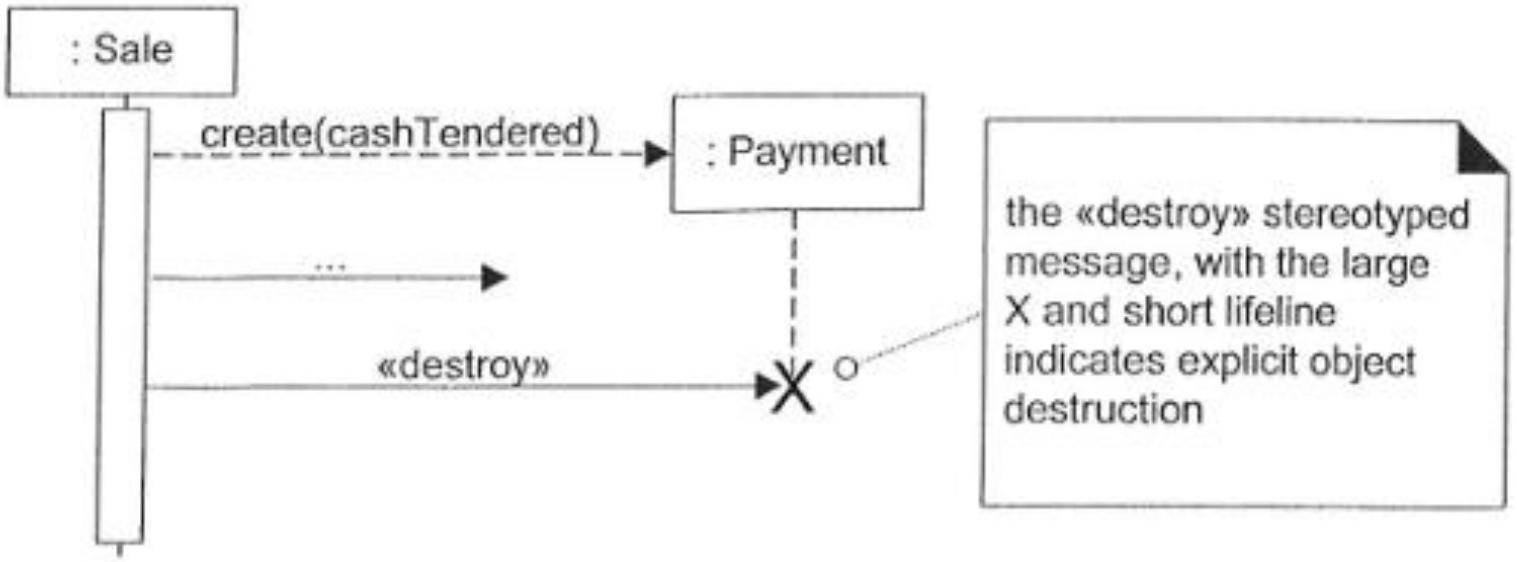
Heavyweight to show a single conditional message ⇒

older UML versions used simpler approach for single conditional message.
Not legal in UML 2, but still is used





Advanced Sequence Diagrams: destroy



In some circumstances it is desirable to show explicit destruction of an object.

For example, when using language that does not have automatic garbage collection, or when you want to especially indicate an object is no longer usable (such as a closed database connection).

Sequence Diagrams at a glance

Dimensions: object and time

Object Dimension

- The horizontal axis shows the elements that are involved in the interaction
- Conventionally, the objects involved in the operation are listed from left to right according to when they take part in the message sequence.

Time Dimension

- The vertical axis represents time proceedings (or progressing) down the page.
- Time in a sequence diagram is all a **about <u>ordering</u>, not duration** ⇒ the vertical space in an interaction diagram is not relevant for the duration of the interaction.

Source: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/