CS5030

Modelling Interactions

Learning objectives

- On completing this lecture and associated reading, you should
  - Understand the purpose of sequence diagrams in UML
  - Know the fundamental constructs of sequence diagrams
  - Be able to apply these constructs in practice

Modelling interactions

- Interactions are part of the behaviour of a system
- Interactions between participants realise a use case, or part of a use case

Interaction diagrams in UML

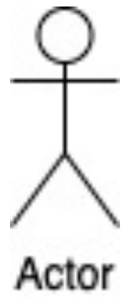
- Sequence diagrams
  - Time-ordered sequence of messages sent between participants in interactions
- Communication diagrams
  - Interactions between participants via messages
- Interaction overview diagrams
  - Overview of control flow (interactions) within a system
- Timing diagrams
  - Detailed timing information about interactions

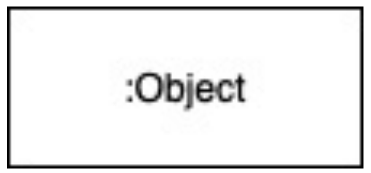
Sequence diagrams

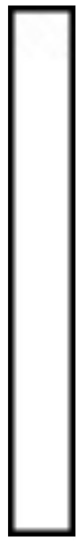
- A sequence diagram shows the order of interactions that take place among participants during a particular use case
  - Time is represented explicitly
  - But focus is on ordering rather than duration

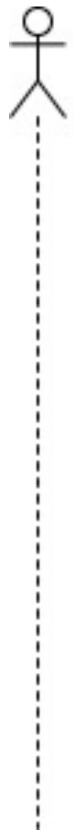
- Two dimensions:
  - Vertical, denoting time;
  - Horizontal, showing the participants involved in the interaction

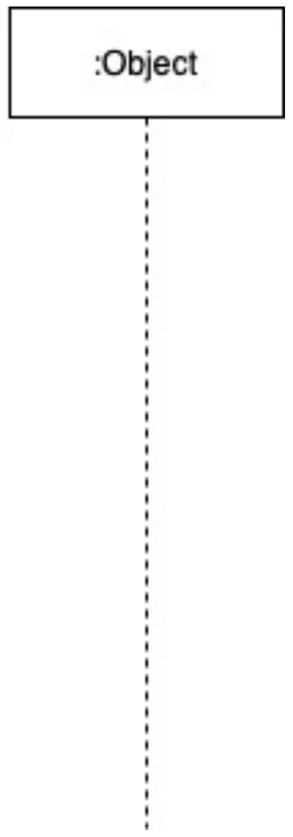
Basic elements (1)






Activation bar


Actor lifeline

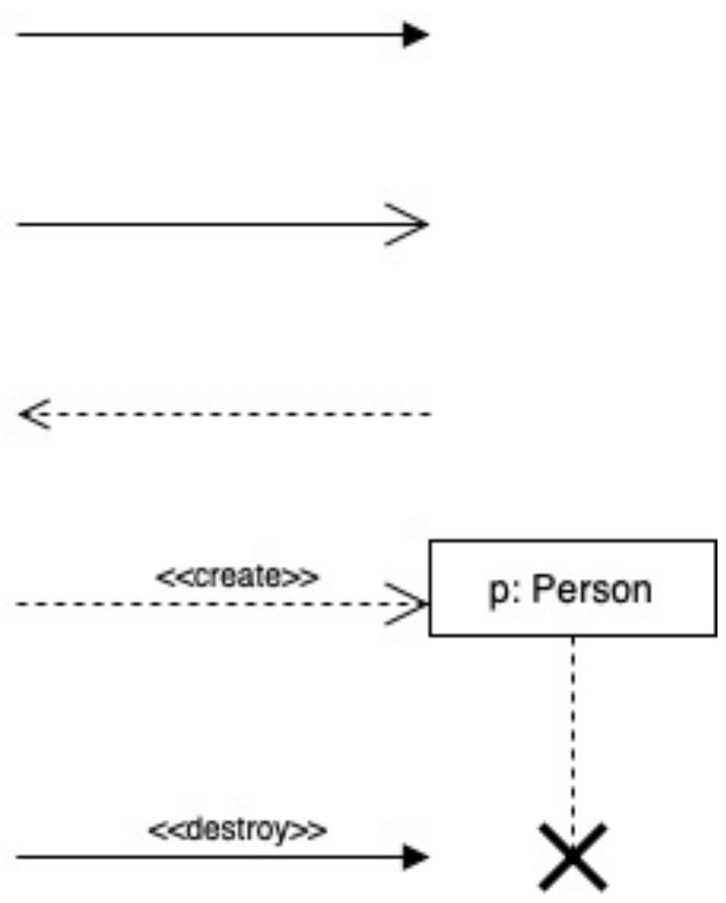


Object lifeline

Basic elements (2)

- An actor is an external entity that interacts with the system
- A lifeline is a named element which represents an individual participant in the interaction
- An activation is the period during which an element is performing an operation
  - The top and the bottom of the bar align with initiation and completion time respectively
- Messages are exchanged by participants to form interactions
  - May take arguments and return a value

Common messages



Synchronous message

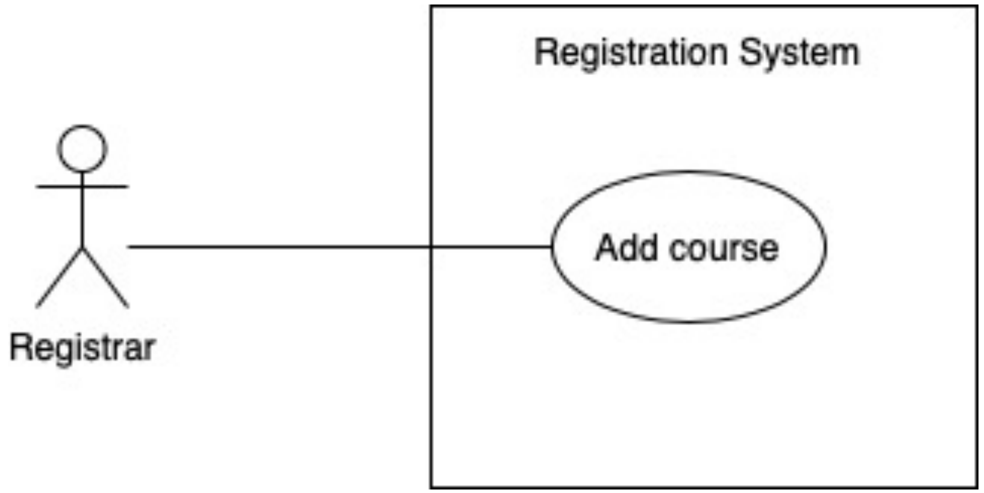
Asynchronous message

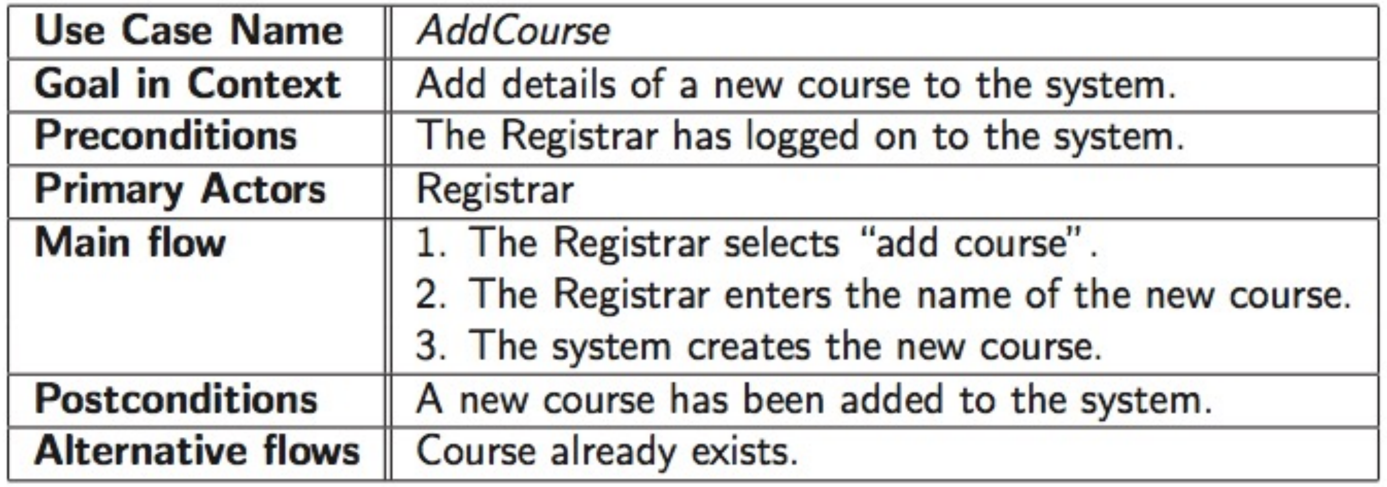
Return message

Participant creation message

Participant destruction message

Example 1: use case

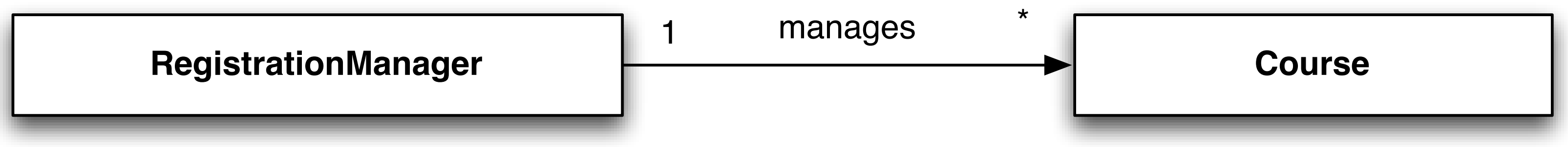




Use Case Name	*AddCourse*
Goal in Context	Add details of a new course to the system.
Preconditions	The Registrar has logged on to the system.
Primary Actors	Registrar
Main flow	1. The Registrar selects “add course”. 2. The Registrar enters the name of the new course. 3. The system creates the new course.
Postconditions	A new course has been added to the system.
Alternative flows	Course already exists.

Example 1: classes

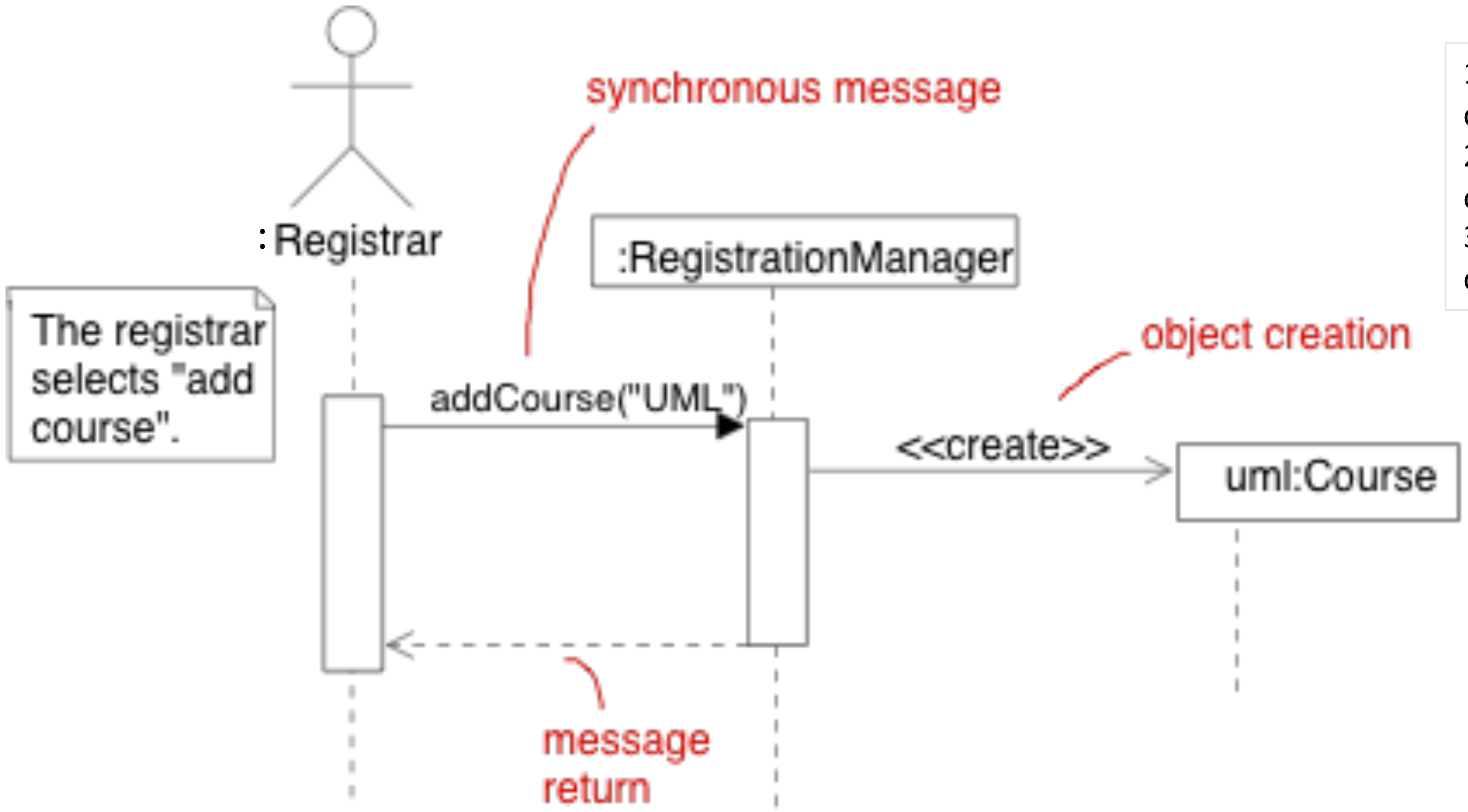
- High-level class diagram



Example: sequence

- The Registrar selects ‘add course’
- The Registrar enters the name of the new course
- The system creates the new course

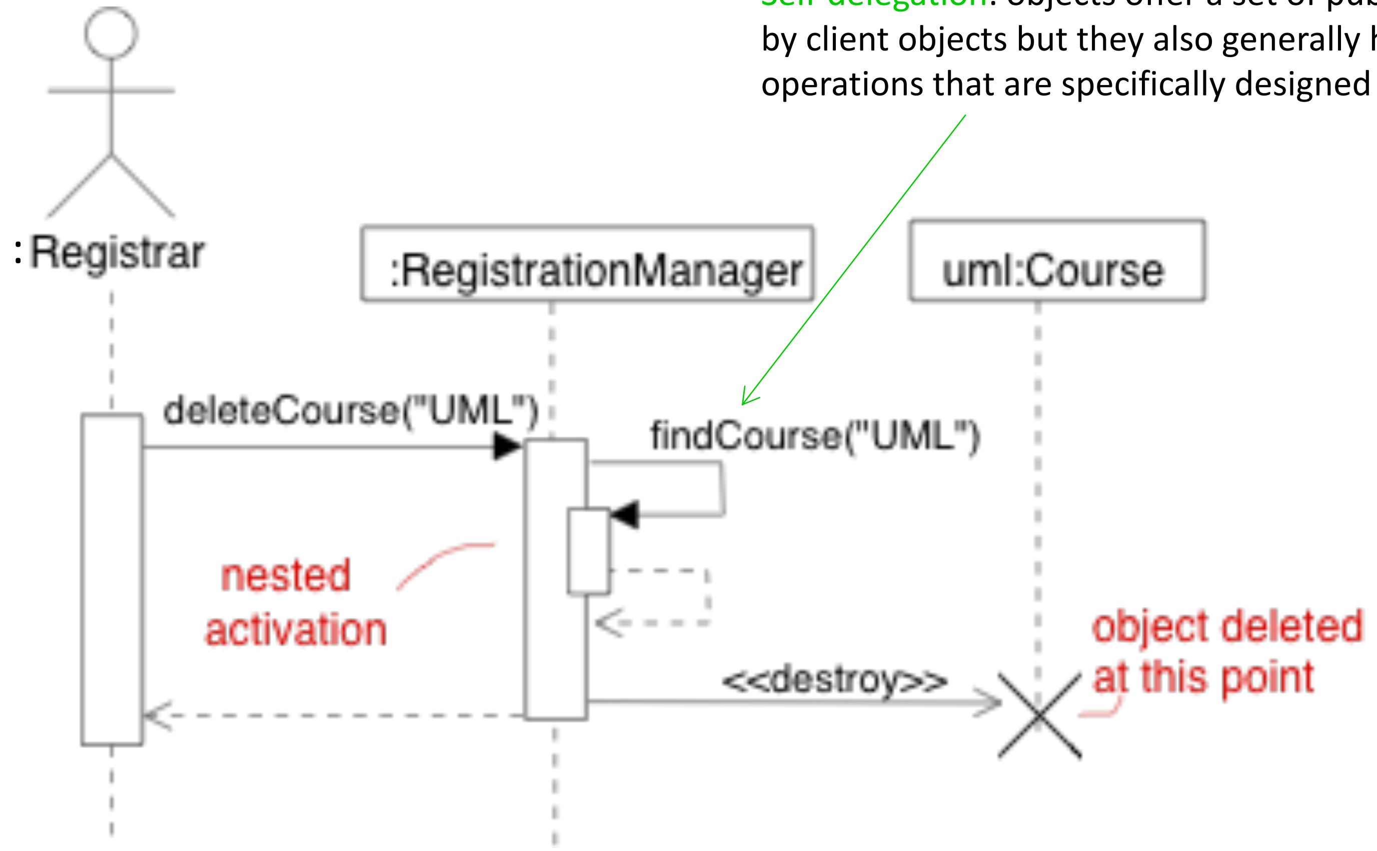
Example 1: sequence diagram



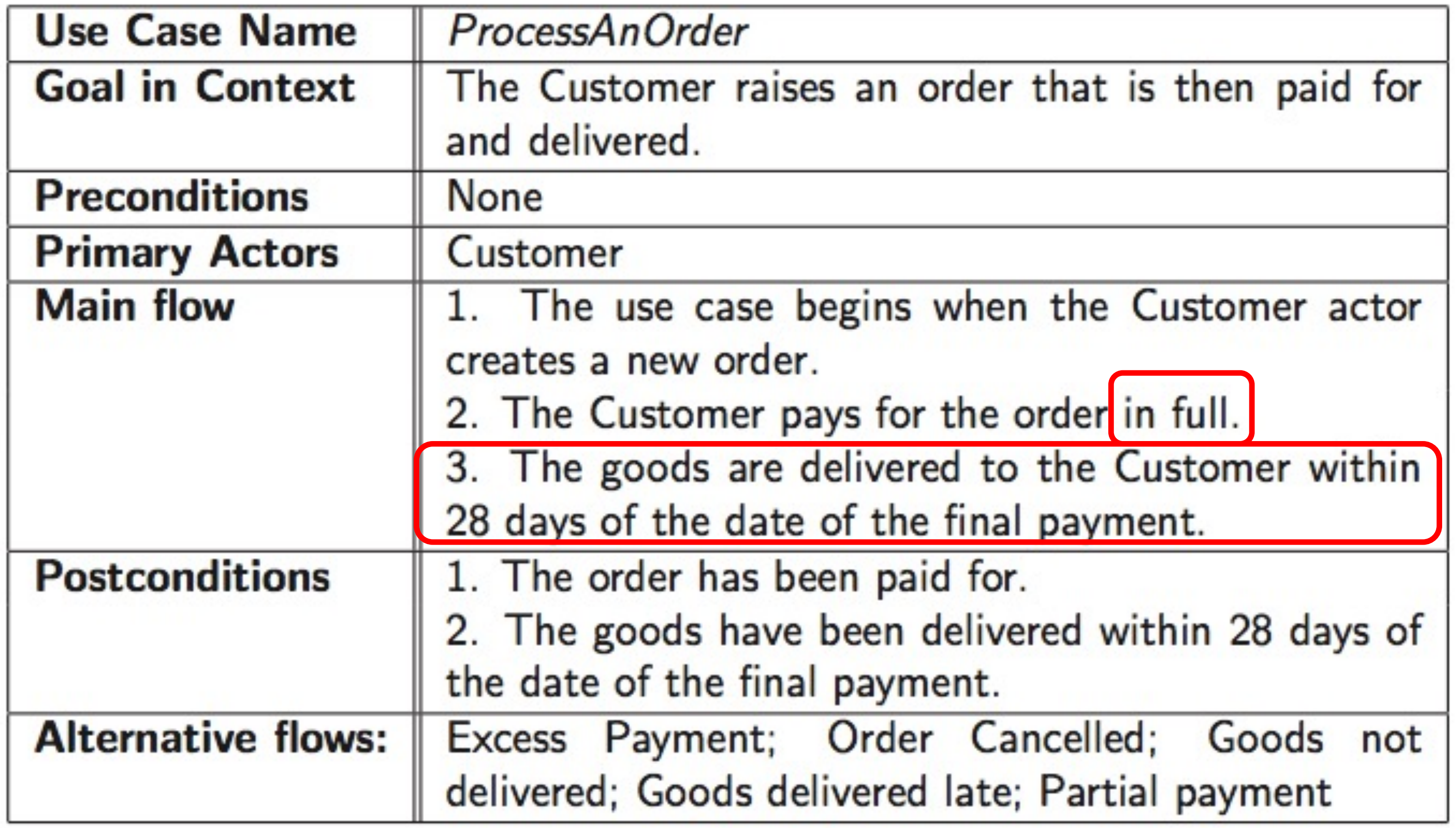
1. The Registrar selects "add course"
2. The Registrar enters the name of the new course
3. The system creates the new course.

Example 2: sequence diagram

Self-delegation: objects offer a set of public operations that can be called by client objects but they also generally have a set of private helper operations that are specifically designed to be called by the object itself

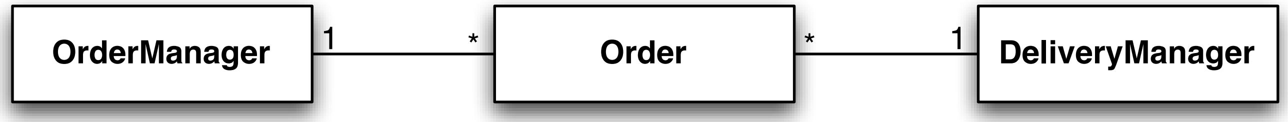


Example 3: use case



Use Case Name	*ProcessAnOrder*
Goal in Context	The Customer raises an order that is then paid for and delivered.
Preconditions	None
Primary Actors	Customer
Main flow	1. The use case begins when the Customer actor creates a new order. 2. The Customer pays for the order in full. 3. The goods are delivered to the Customer within 28 days of the date of the final payment.
Postconditions	1. The order has been paid for. 2. The goods have been delivered within 28 days of the date of the final payment.
Alternative flows:	Excess Payment; Order Cancelled; Goods not delivered; Goods delivered late; Partial payment

Example 3: class diagram



Example 3: sequence

- The use case begins when the Customer creates a new order
- The Customer pays for the order in full
- The goods are delivered to the Customer within 28 days of the date of the final payment

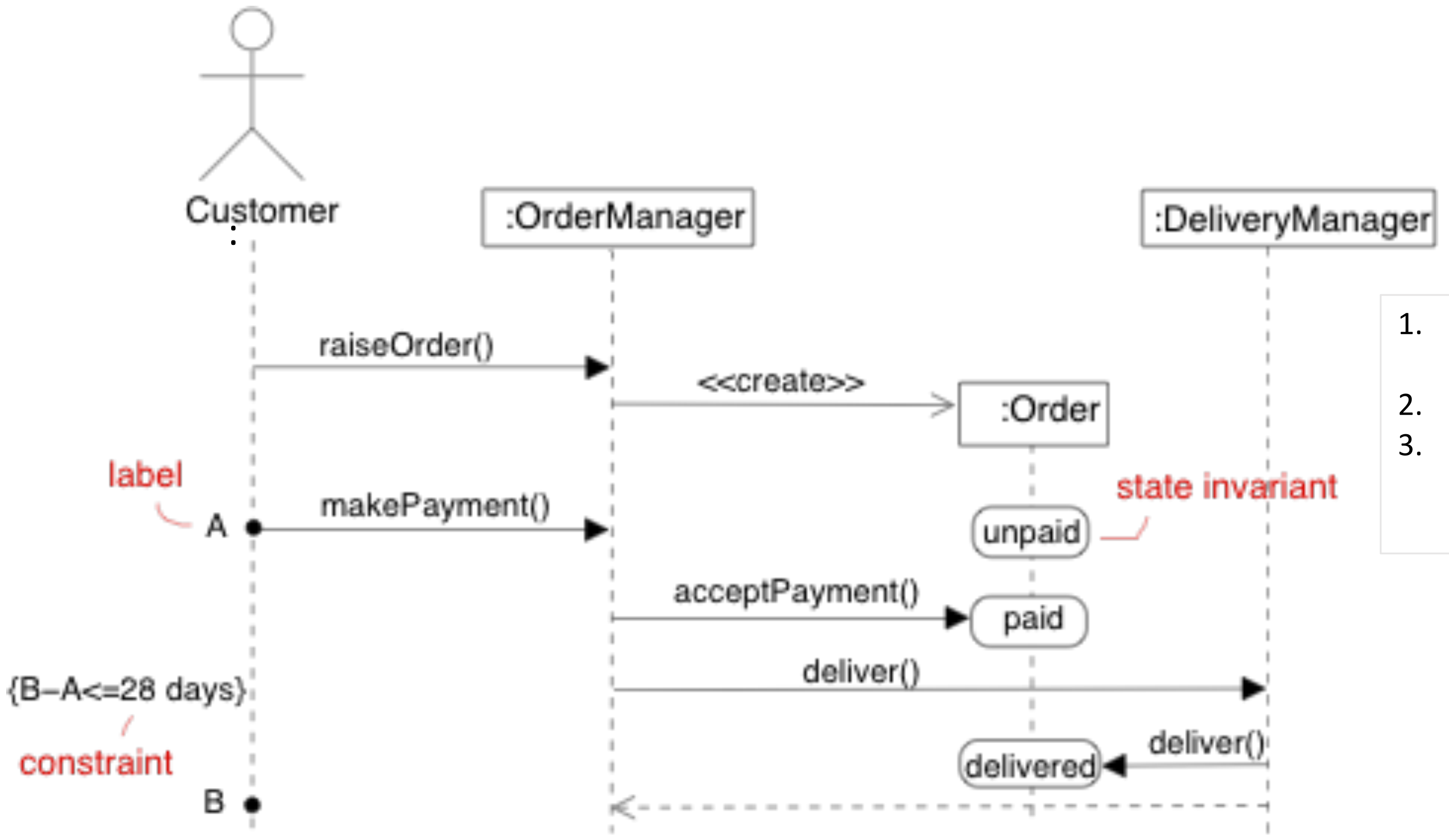
State invariants

- Labels along the lifeline that capture the conditions that must be true for the remainder of the interaction

- State
  - “a … situation during the lifeline of an object during which it satisfies some condition, performs some activity, or waits for some event”

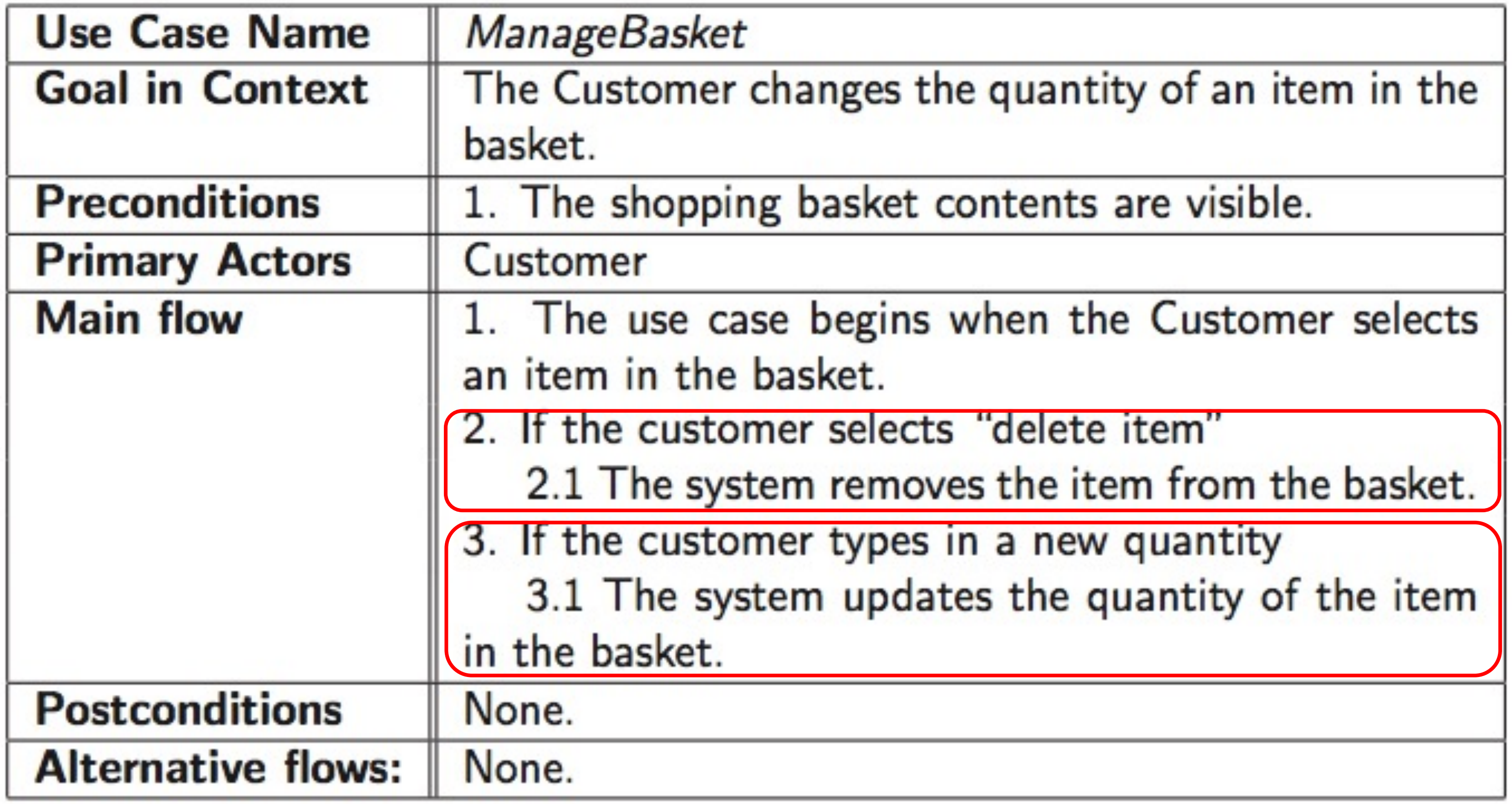
- An object’s state will be changed after receiving a message

Example 3: sequence diagram with state invariants



1. The use case begins when the Customer creates a new order
2. The Customer pays for the order in full
3. The goods are delivered to the Customer within 28 days of the date of the final payment

Example 4: use case – online shop

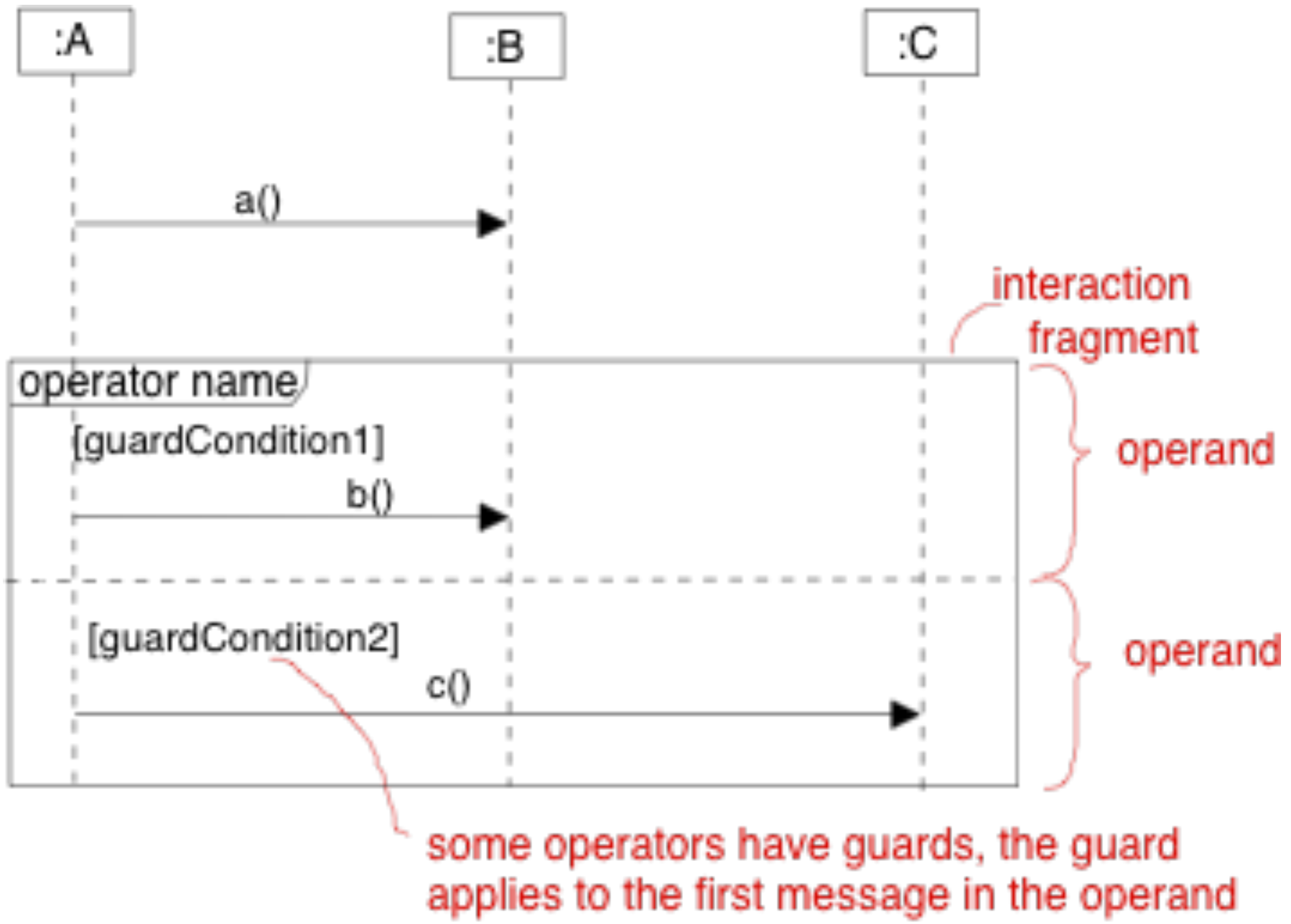


Use Case Name	*ManageBasket*
Goal in Context	The Customer changes the quantity of an item in the basket.
Preconditions	1. The shopping basket contents are visible.
Primary Actors	Customer
Main flow	1. The use case begins when the Customer selects an item in the basket. 2. If the customer selects "delete 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

- Sequence diagrams may be divided into areas, each of which is called a combined fragment
    - Logical grouping
- Each combined fragment has one operator, one or more operands, and zero or more guard conditions
    - Operand: a sequence of messages
    - Operator: how its operands are executed
    - Guard condition: whether their operands are executed

Fragments and operators



Common operators (1)

- alt: a choice of behaviour
  - At most one of the operands will execute
  - The operand that executes must have a guard expression that evaluates to be true at this point in the interaction
- opt: a choice of behaviour
  - where either the (sole) operand happens or nothing happens
- ref:
  - The fragment refers to another interaction

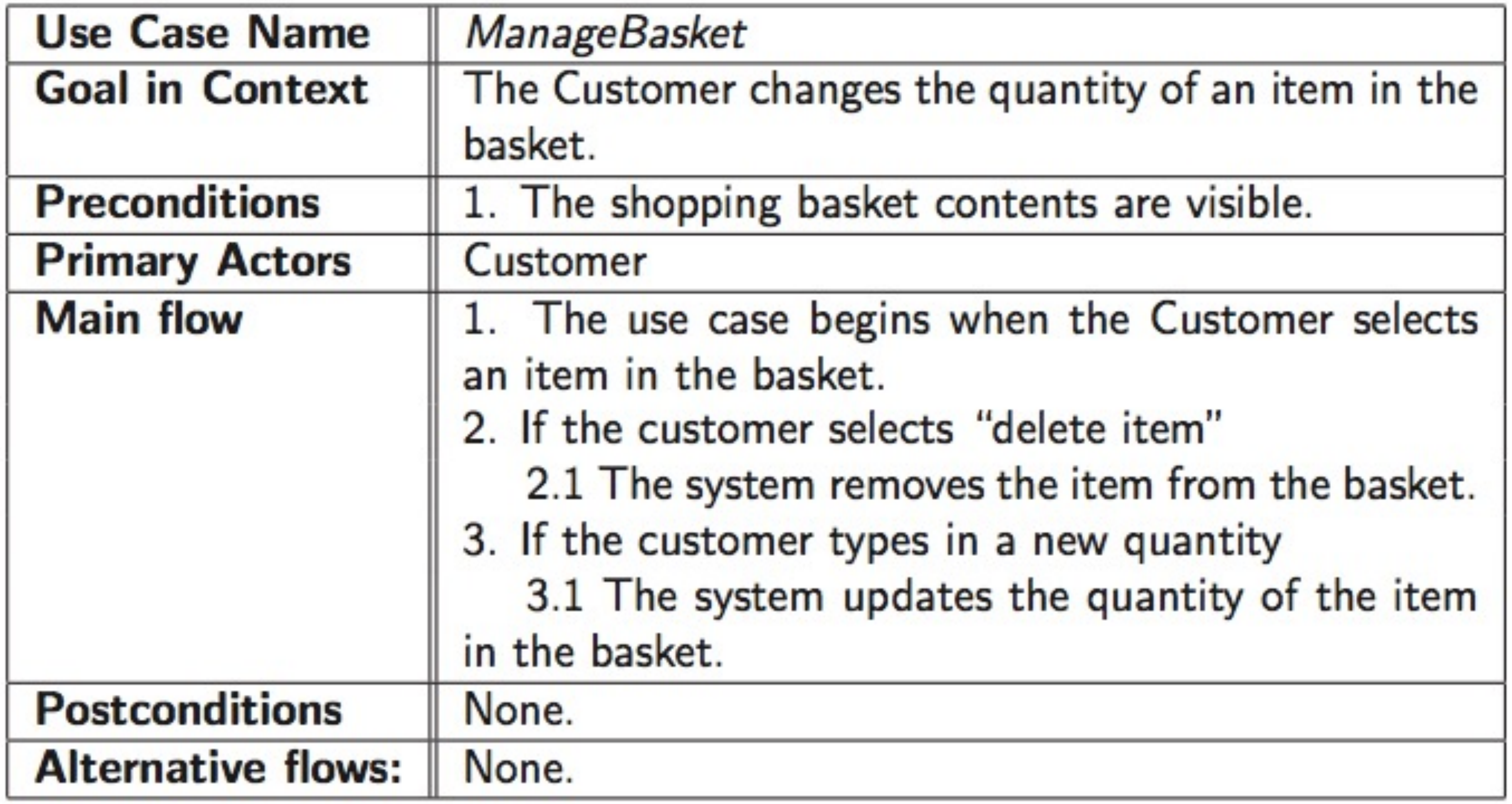
Common operators (2)

- loop: iteration
  - should be repeated a certain number of times, indicated by an interval or a boolean condition
  - e.g. loop (1, 10) or loop [count ≥ 0]
- break:
  - If the guard condition is true, the operand is executed but not the rest of the enclosing interaction
- neg: show invalid interactions
  - to show interactions that must not happen

Common operators (3)

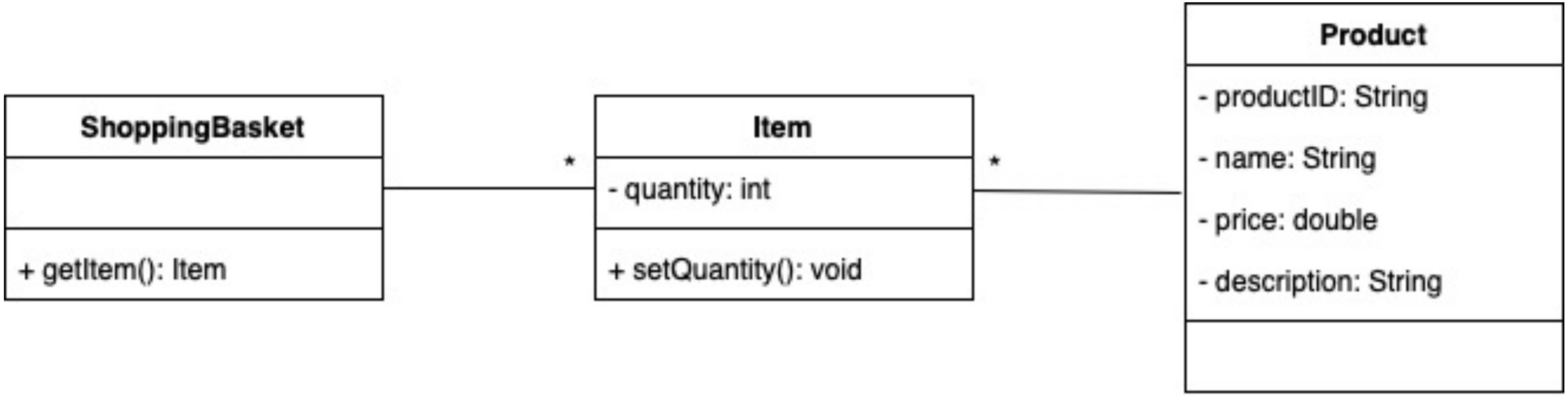
- par: parallel
  - between the behaviours of the operands
  - event occurrences of the different operands can be interleaved in any way as long as the ordering imposed by each operand as such is preserved
- assert: assertion
  - The operand is the only valid behaviour at that point in the interaction; any other behaviour would be an error
  - To indicate that some behaviour must occur at a certain point in the interaction

Example 4

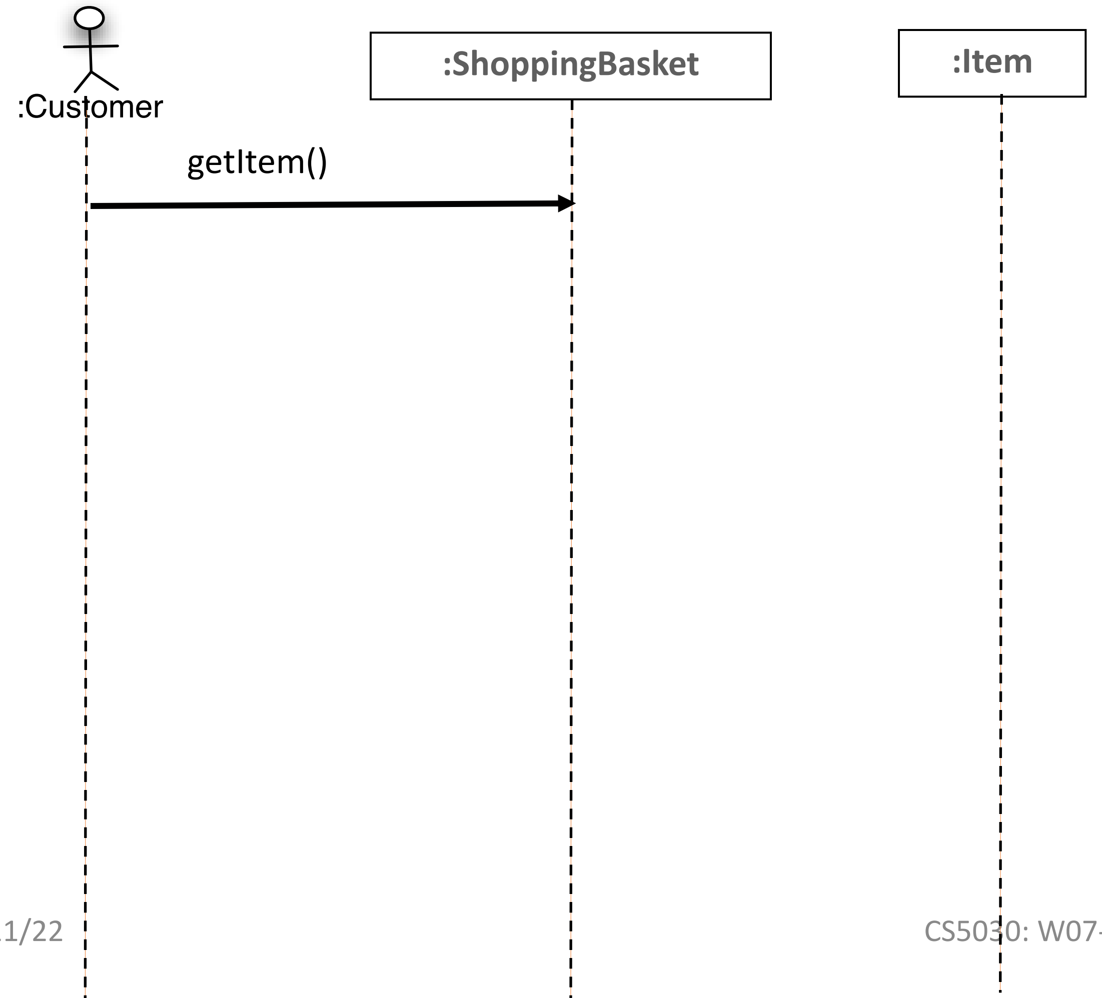


Use Case Name	*ManageBasket*
Goal in Context	The Customer changes the quantity of an item in the basket.
Preconditions	1. The shopping basket contents are visible.
Primary Actors	Customer
Main flow	1. The use case begins when the Customer selects an item in the basket. 2. If the customer selects "delete 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.

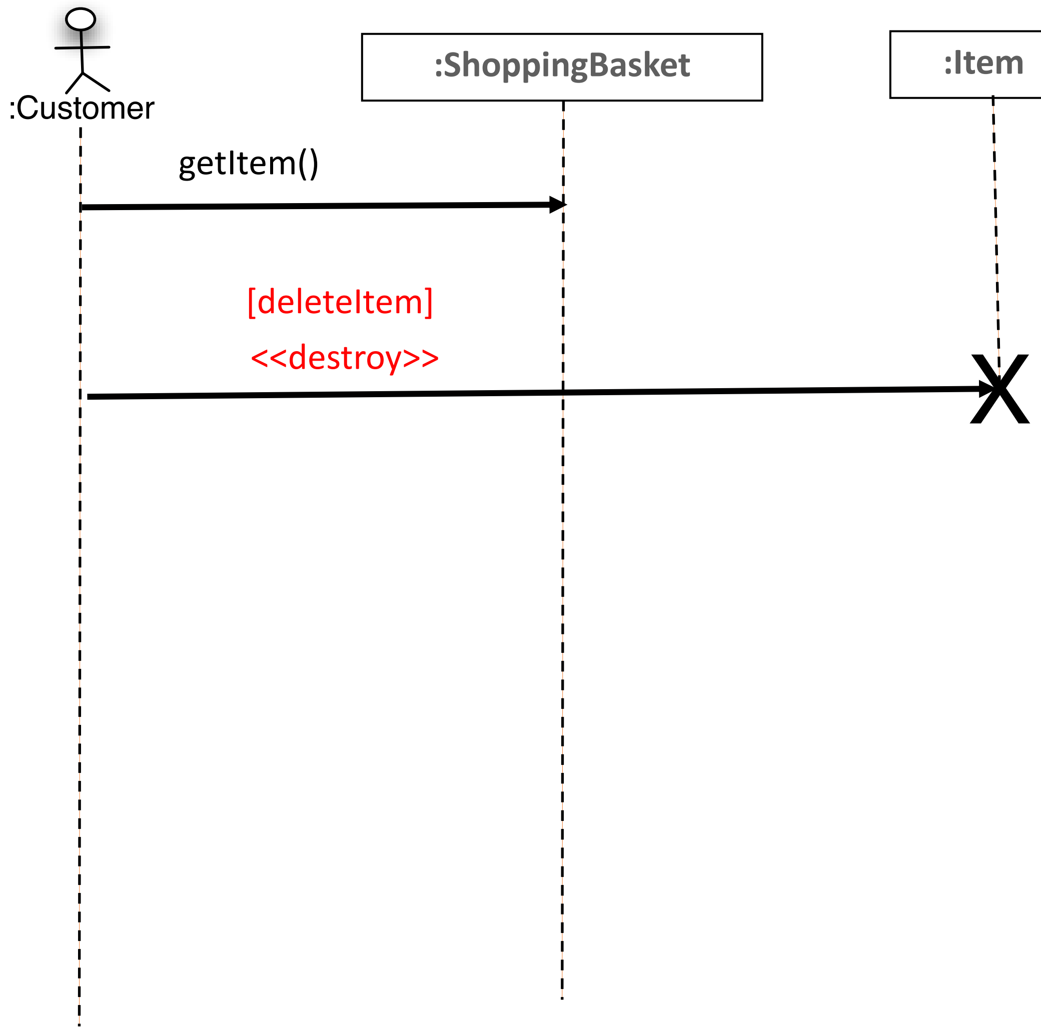
Example 4: class diagram



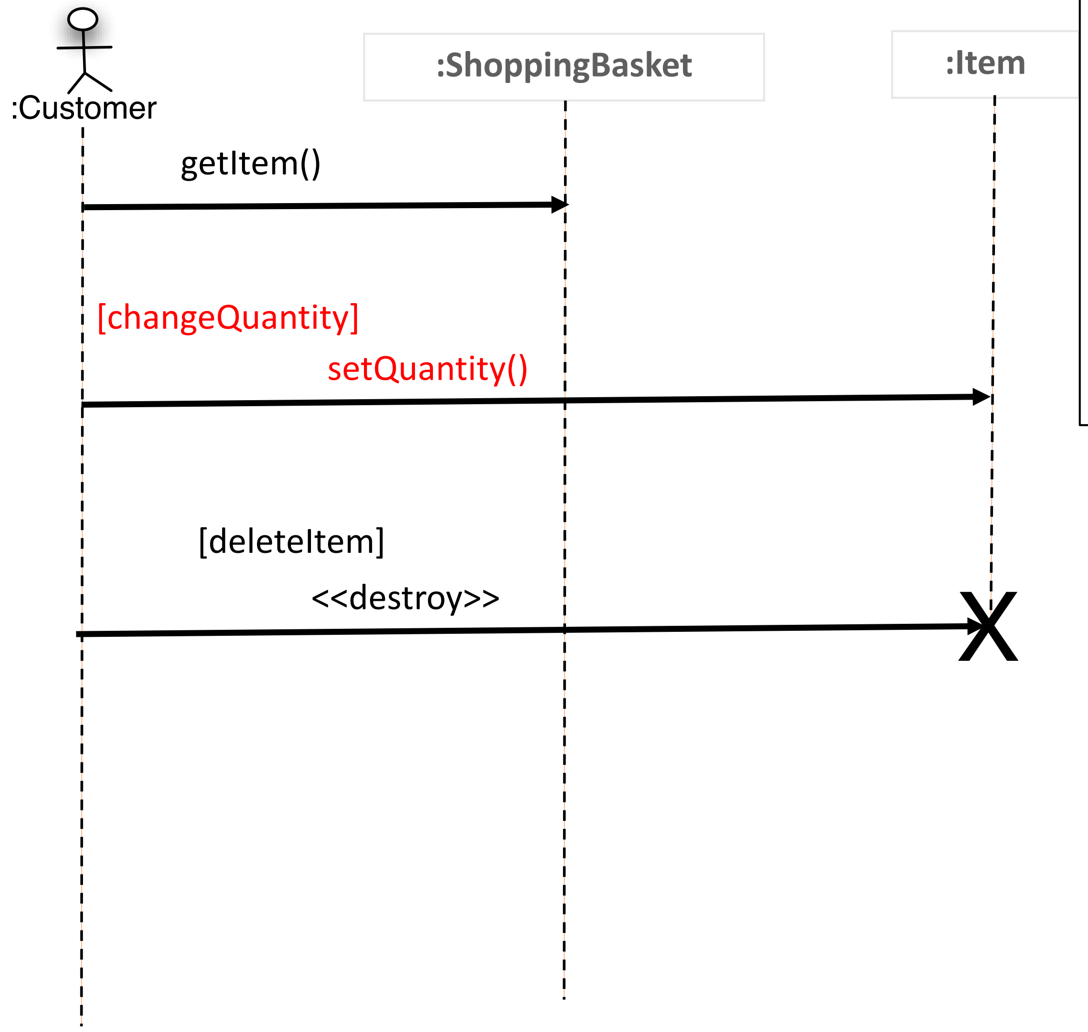
Example 4: sequence diagrams



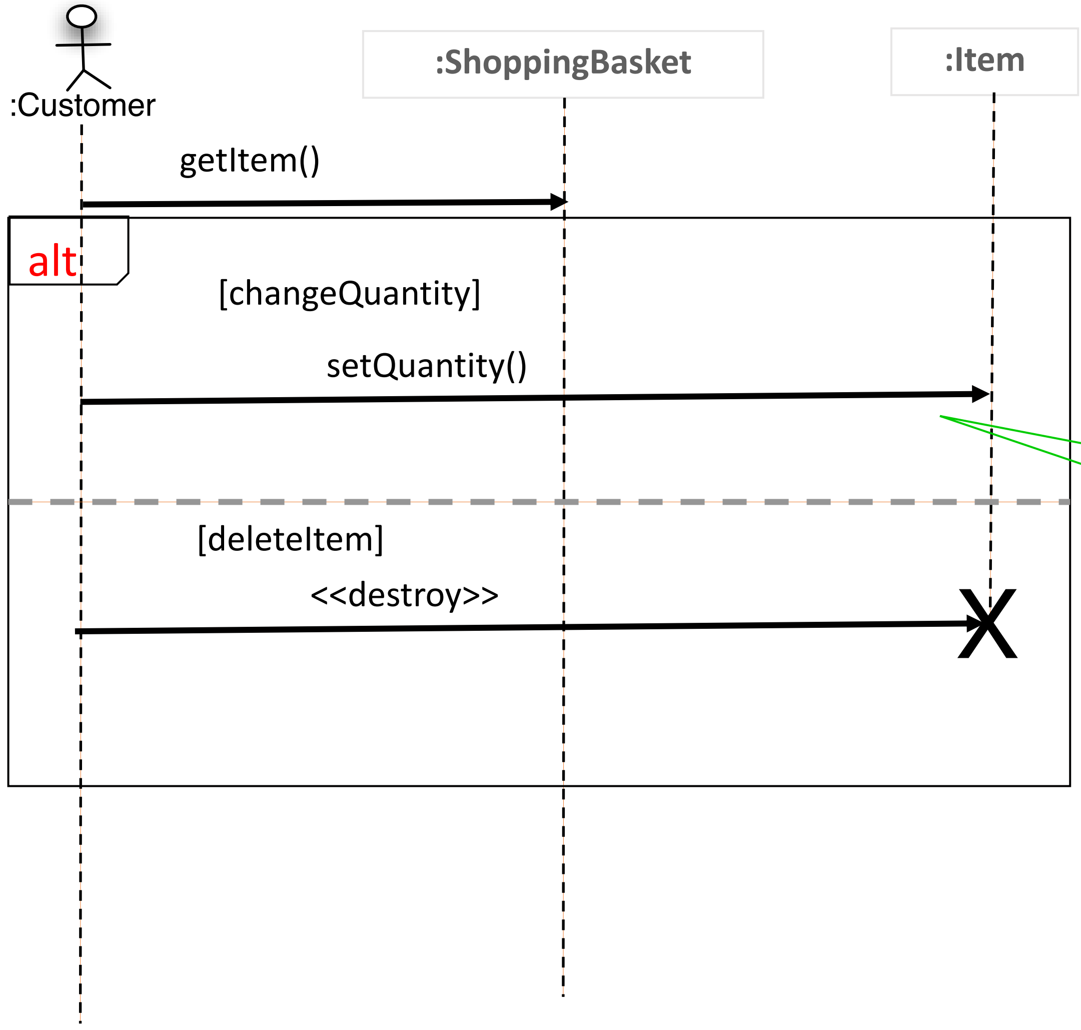
1. The use case begins when the Customer selects an item in the basket
2. If the customer selects "delete item"
   2.1 The system removes the item from the basket
3. If the customer types in a new quantity
   3.1 The system updates the quantity of the item in the basket.



1. The use case begins 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.

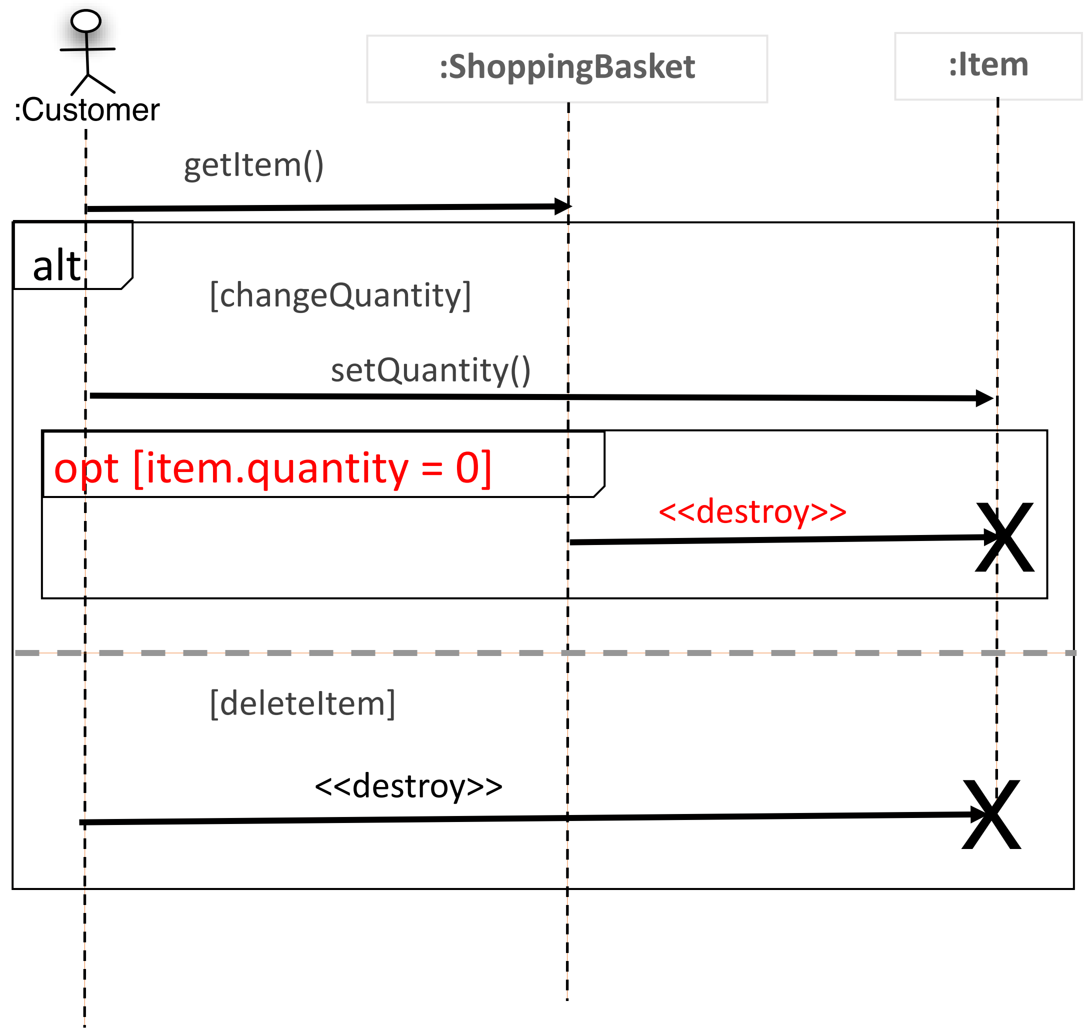


1. The use case begins when the Customer selects an item in the basket
2. If the customer selects "delete item"
   2.1 The system removes the item from the basket
3. If the customer types in a new quantity
   3.1 The system updates the quantity of the item in the basket.



1. The use case begins 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.

What if the Customer sets the quantity to be 0?



1. The use case begins 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.

Sequence diagrams - uses

- Sequence diagrams are useful when modelling
  - High-level interactions between objects in a system
  - Interactions between objects that realise a use case or operation
  - Single or generic interactions/scenarios
  - Real-time systems

Key points

- Use case realisation refines a specification of a use case into class diagrams and interaction diagrams
- Sequence diagrams model time-ordered sequence of messages sent between participants in the interaction
- Each use case realisation deals with exactly one use case