Design Modeling with UML: Design Sequence Diagram

Instructor: Mehroze Khan

Designing the Solution!

- Introduction to Design Phase
- Static vs. Dynamic Design Modelling
- Interaction Diagrams
- Design Class Diagrams

Modelling Dynamic Behavior

- The UML includes interaction diagrams to illustrate how objects interact via messages.
- They are used for dynamic object modeling.
- Static OO design will be presented as Class Diagram.

Sequence Diagram

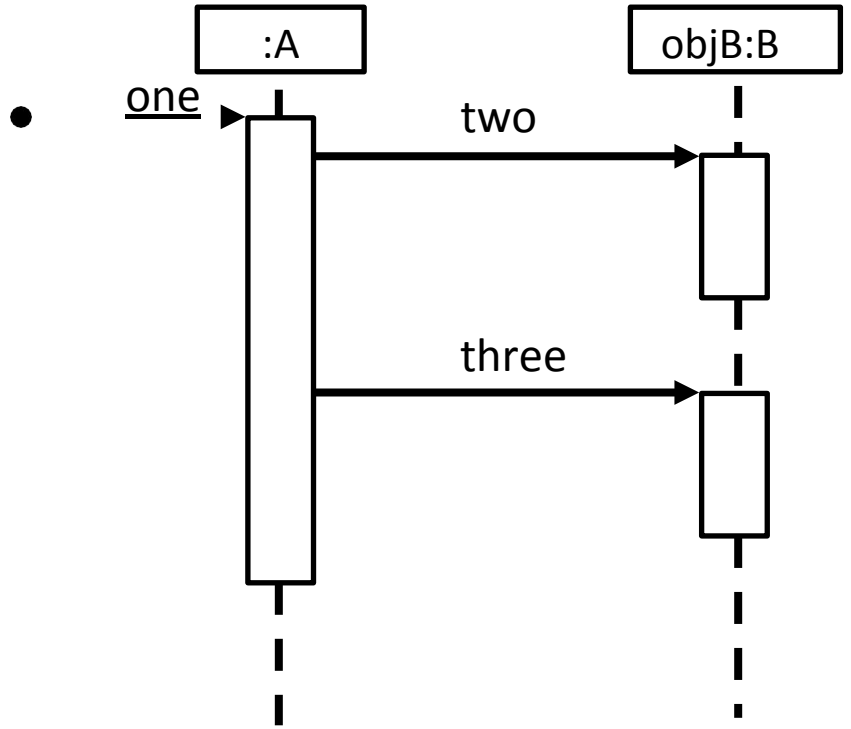
- Sequence diagram simply depicts interaction between objects in a sequential order (i.e. the order in which these interactions take place.)
- Sequence diagrams describe *how* and in what order the objects in a system function

Sequence Diagram

- They illustrate how the different objects in a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed

Interaction diagrams are used to visualize the interaction via messages between objects; they are used for *dynamic object modeling*.

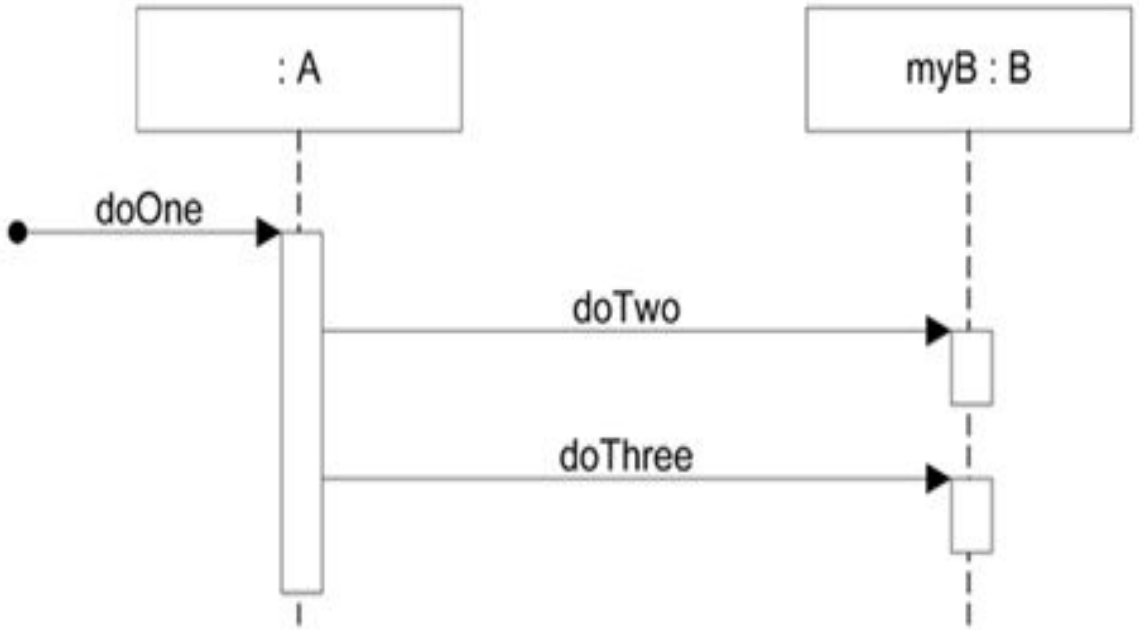
Example of Sequence Diagram



Sequence diagrams

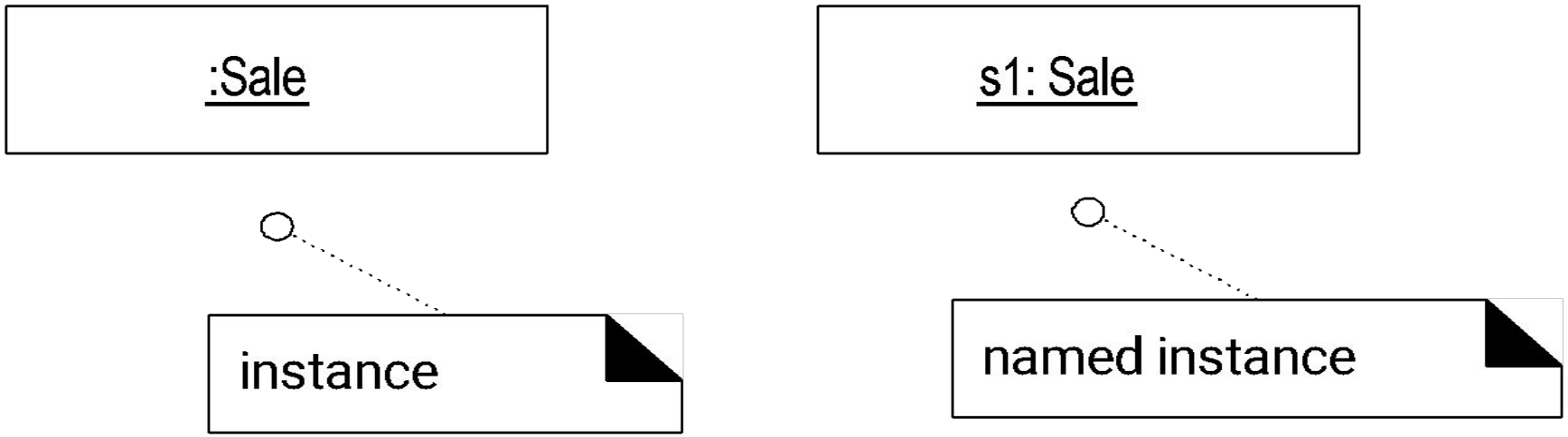
- Sequence diagrams illustrate interactions in a kind of fence format, in which each new object is added to the right

```
public class A
{
   private B myB = new B();
   public void doOne()
   {
       myB.doTwo();
       myB.doThree();
   }
 // …
}
```



Illustrating Classes and Instances

- UML instances and named instances



Format for Interaction Messages

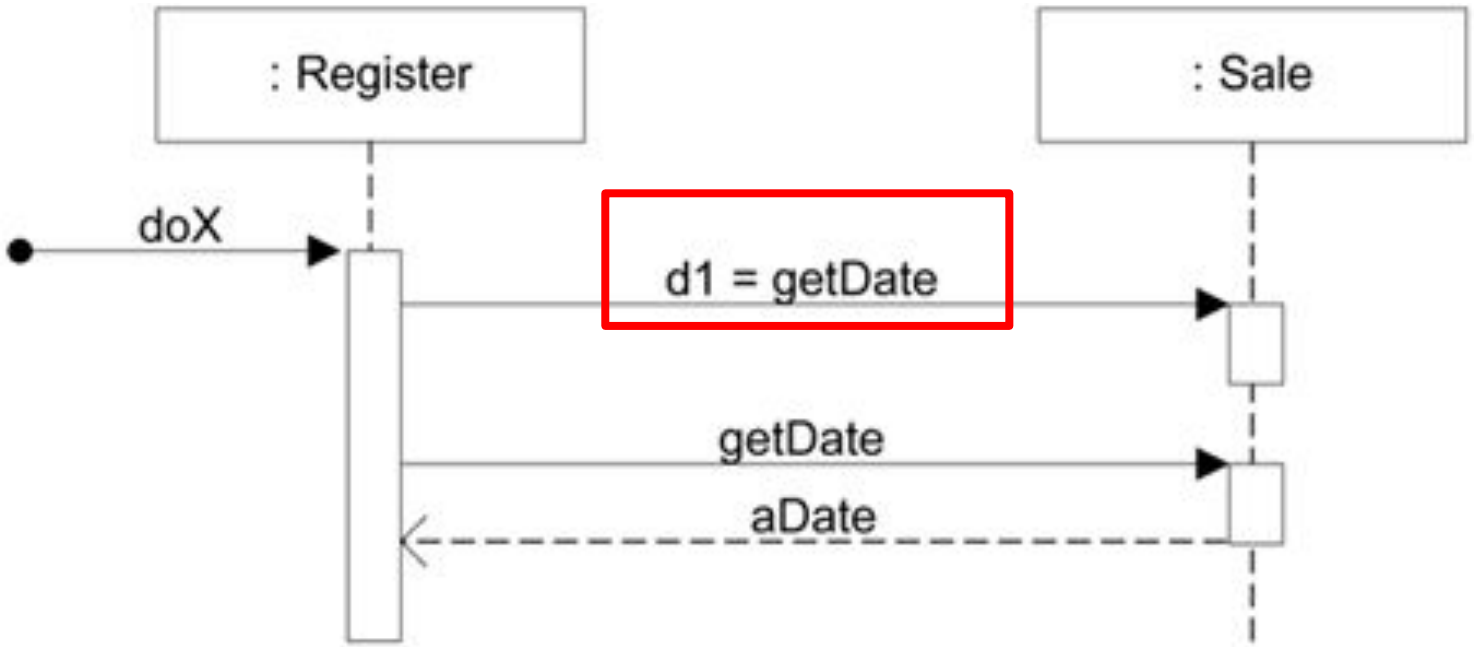
- Interaction diagrams show messages between objects; the UML has a standard syntax for these message expressions

return = message(parameter:parameterType):returnType

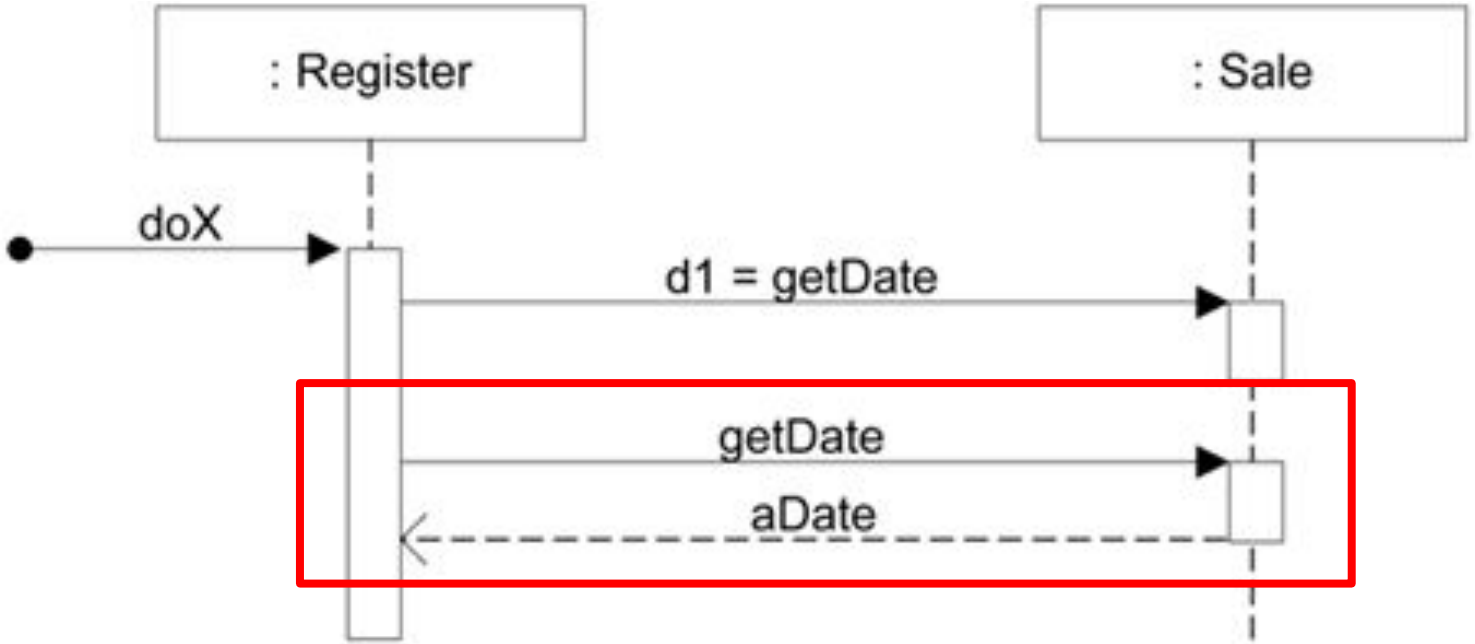
Parentheses are usually excluded if there are no parameters. Type information may be excluded if unimportant.

```
initialize
initialize(code)
d = getProductDescription (id)
d = getProductDescription (id : ItemId)
d = getProductDescription (id : ItemId) : ProductDescription
```

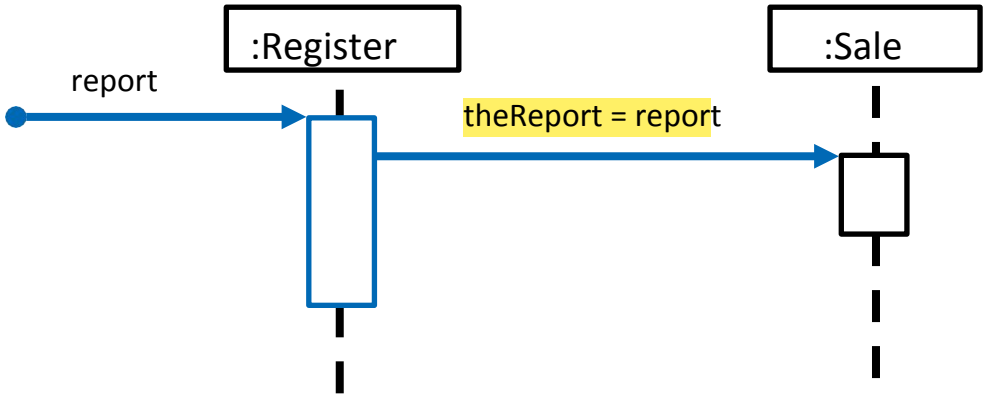
Two ways to show a return result from a message



Two ways to show a return result from a message

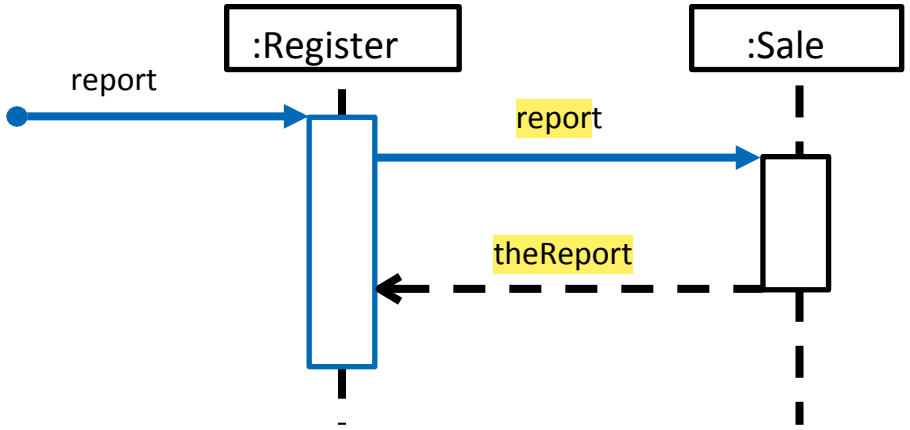


Two ways to show a return result from a message

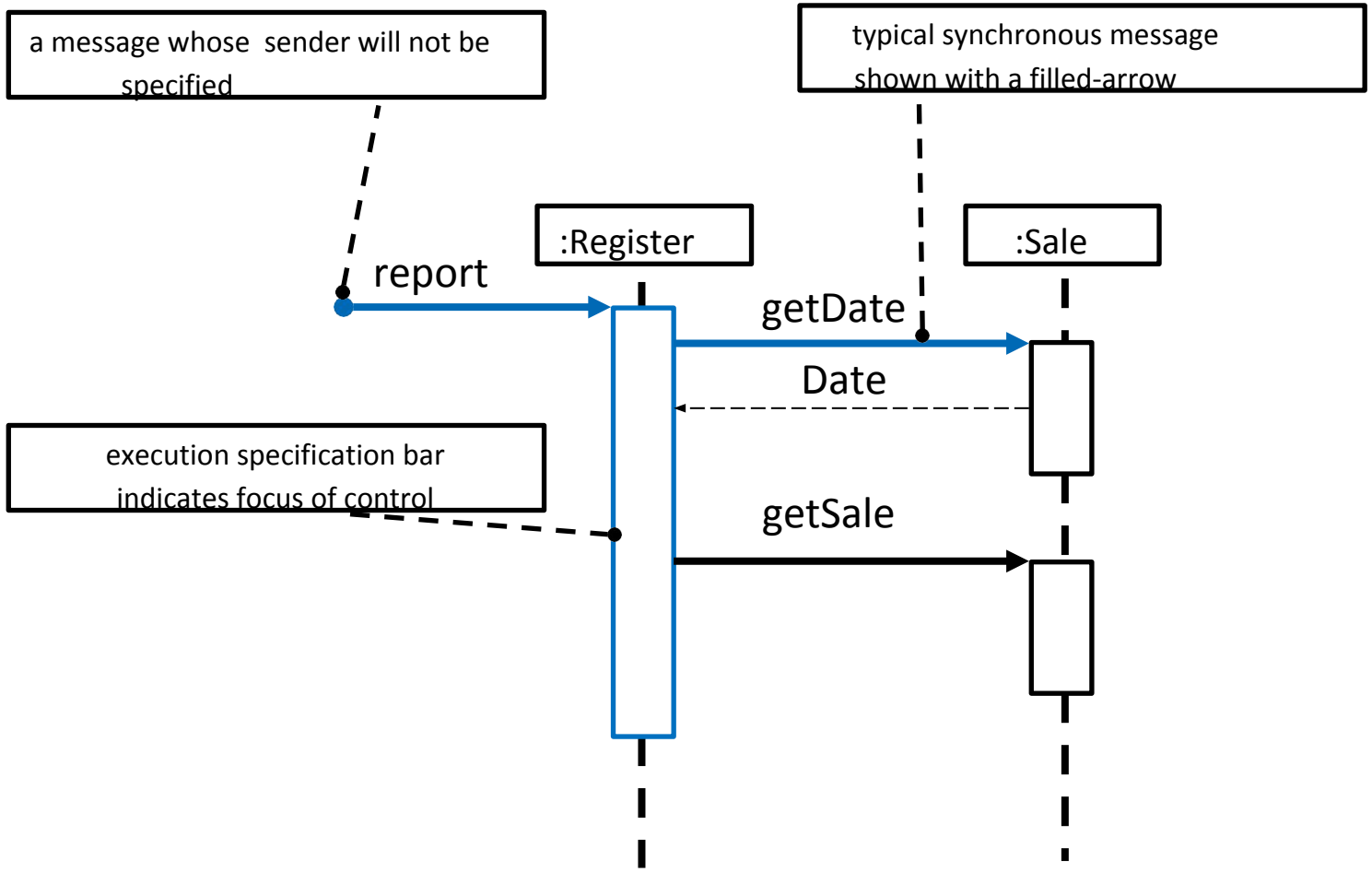


Variant A

Variant B

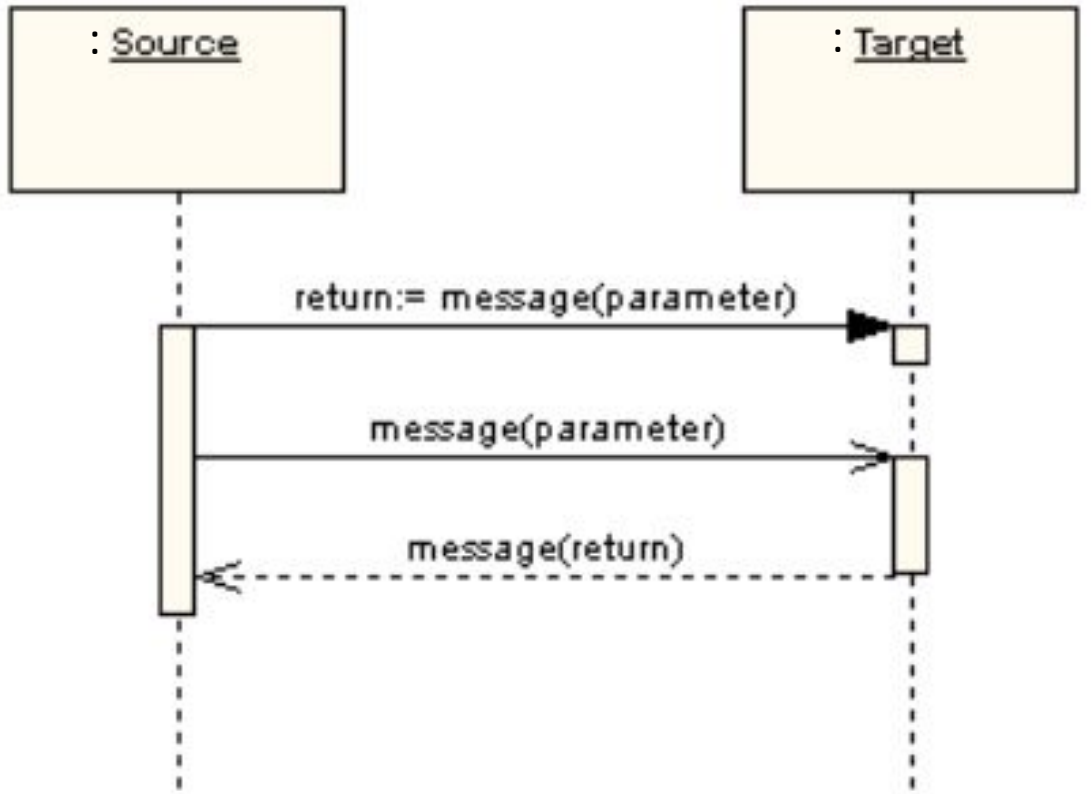


Modeling (Synchronous) Messages



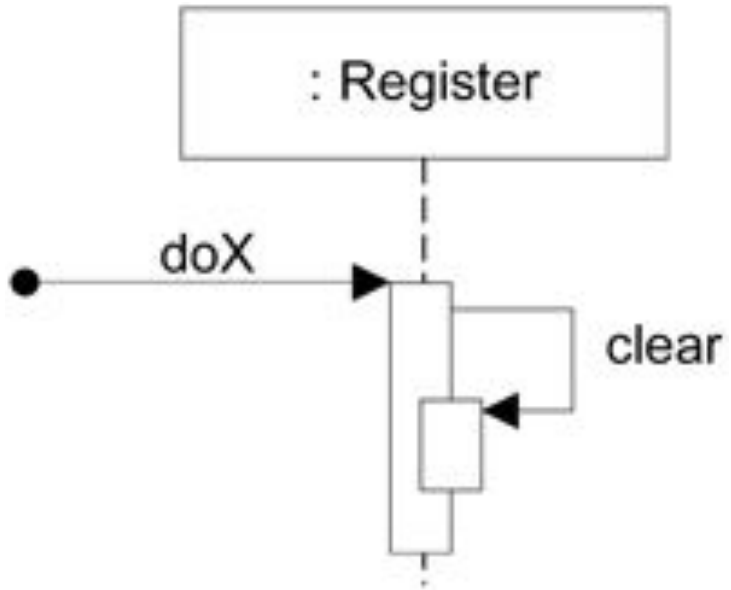
Synchronous and Asynchronous Messages

First message is a **synchronous message** (denoted by the **solid arrowhead**) complete with an **implicit return message**; the second message is **asynchronous** (denoted by **line arrowhead**), and the third is the **asynchronous return message** (denoted by the dashed line).



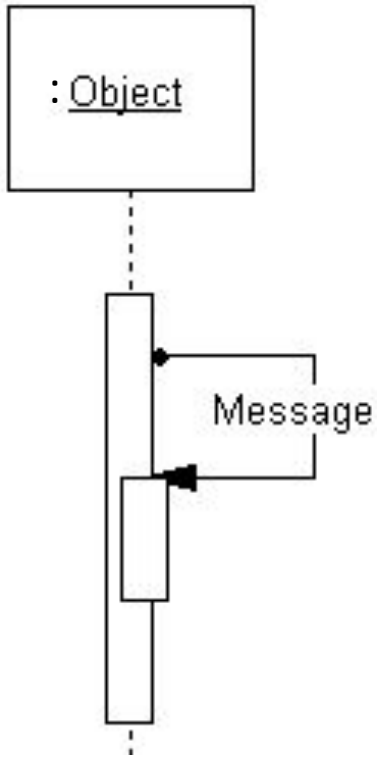
Messages to "self" or "this"

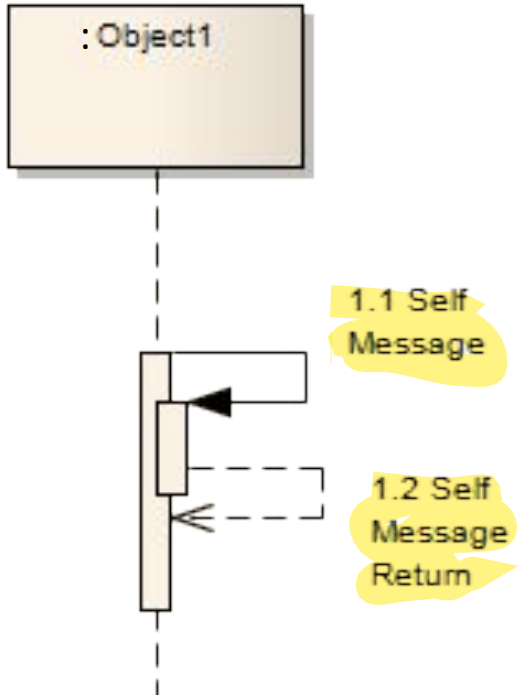
- Messages to "self" or "this"
- A message can be sent from an object to itself. This is illustrated by a link to itself, with messages flowing along the link.

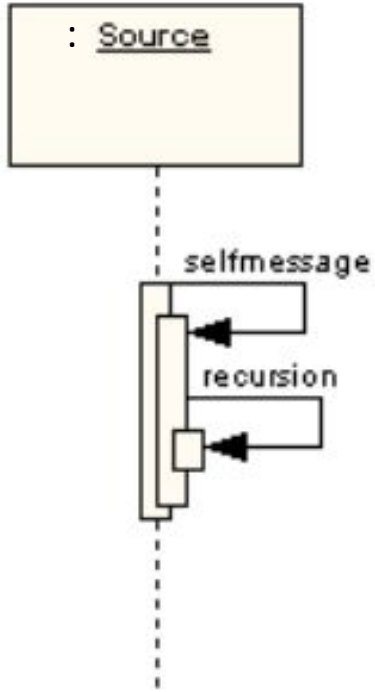


Self Messages

- A self message can represent a **recursive call** of an operation, or one **method calling another method** belonging to the same object. It is shown as creating a **nested focus of control** in the lifeline's execution occurrence.

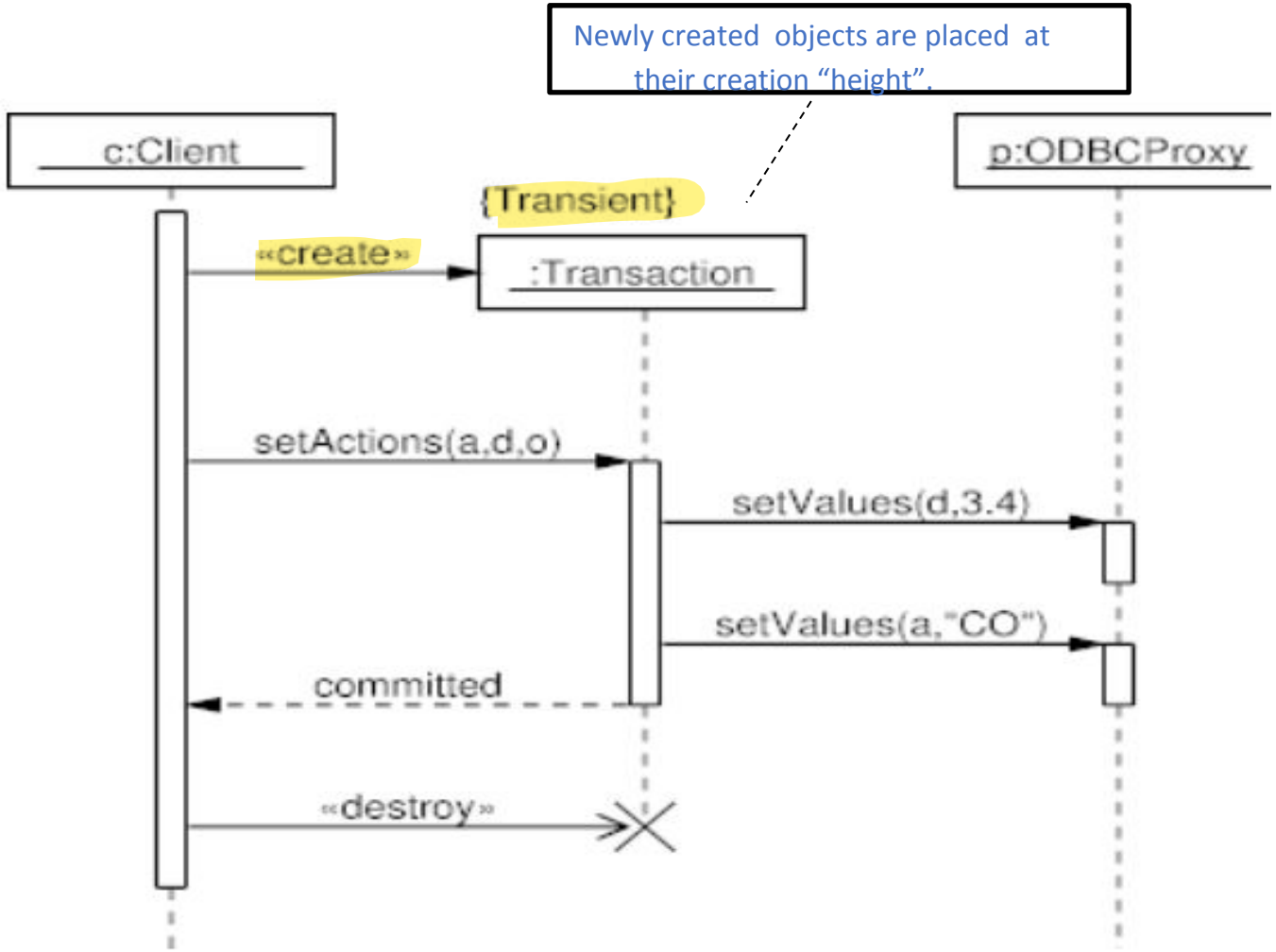






Create and Destroy Messages

- Create messages are used to instantiate a new object
- Delete/Destroy message is used to delete an object



Guards

- When modeling object interactions, there will be times when a **condition** must be met for a message to be sent to the object.
- Guards are used throughout UML diagrams to **control flow**.

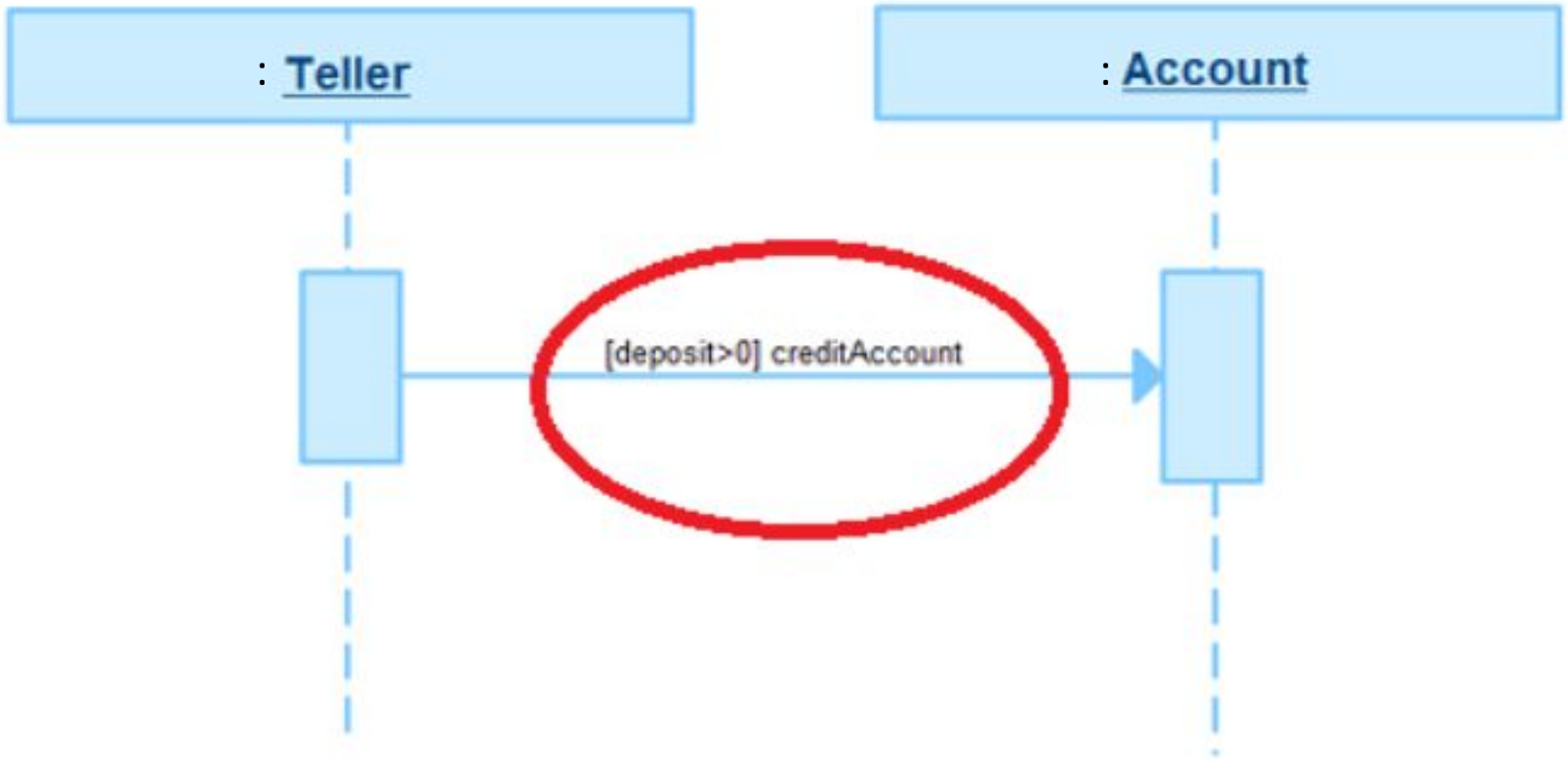


Diagram Frames in UML Sequence Diagrams

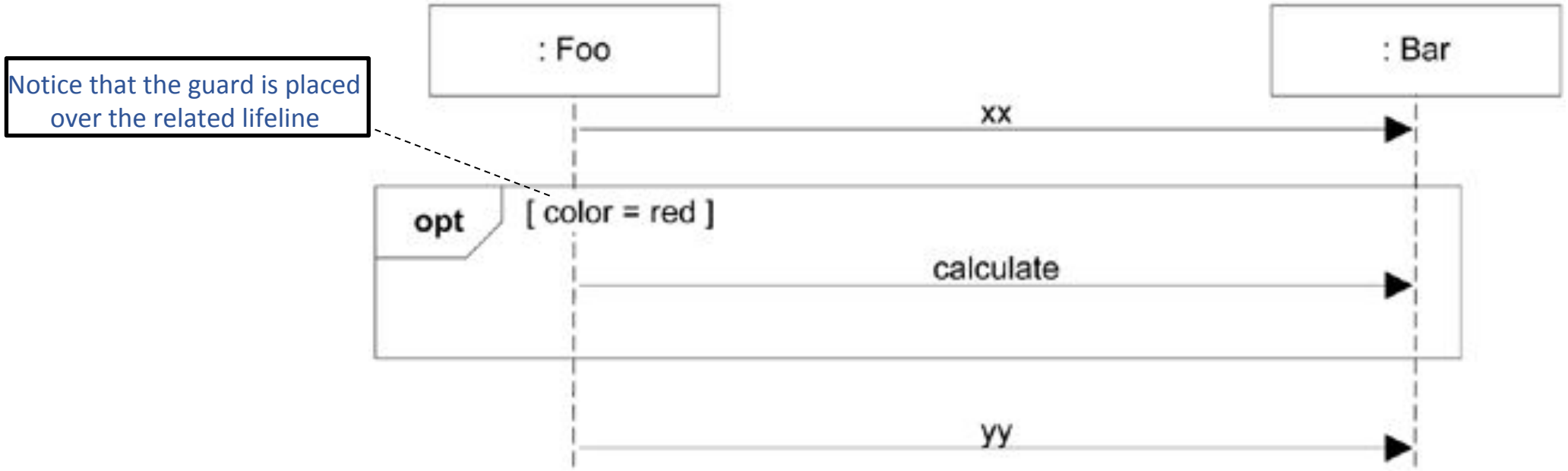
- To support **conditional** and **looping** constructs (among many other things), the UML uses **frames**.
- Frames are regions or fragments of the diagrams; they have an **operator** or **label** (such as loop) and a **guard** (conditional clause).

Frame Operator

Frame Operator	Meaning
alt	Alternative fragment for mutual exclusion conditional logic expressed in the guards.
loop	Loop fragment while guard is true. Can also write loop(n) to indicate looping n times.
opt	Optional fragment that executes if guard is true.
par	Parallel fragments that execute in parallel.

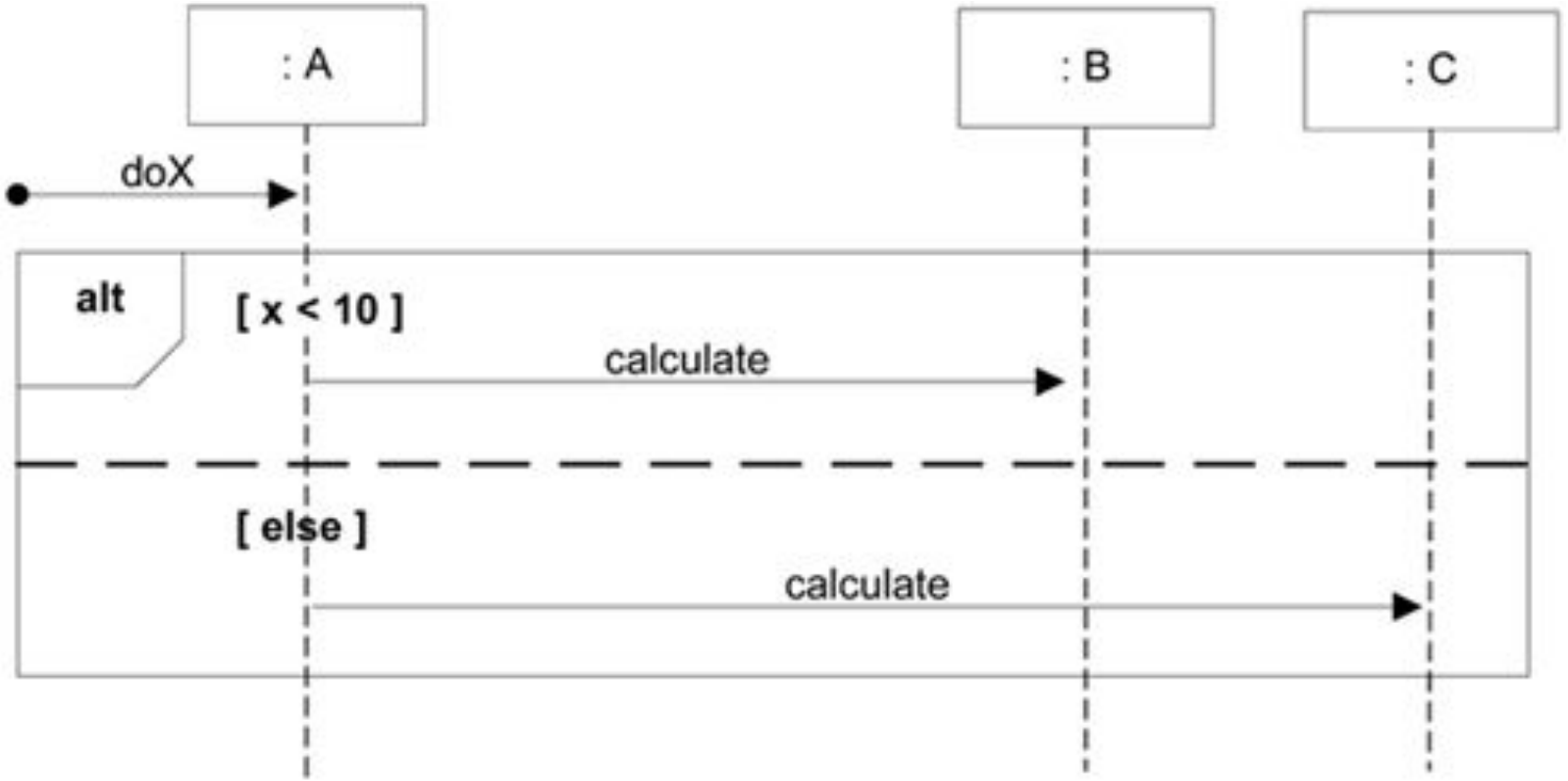
Conditional Messages

- An OPT frame is placed around one or more messages.



Mutually Exclusive Conditional Messages

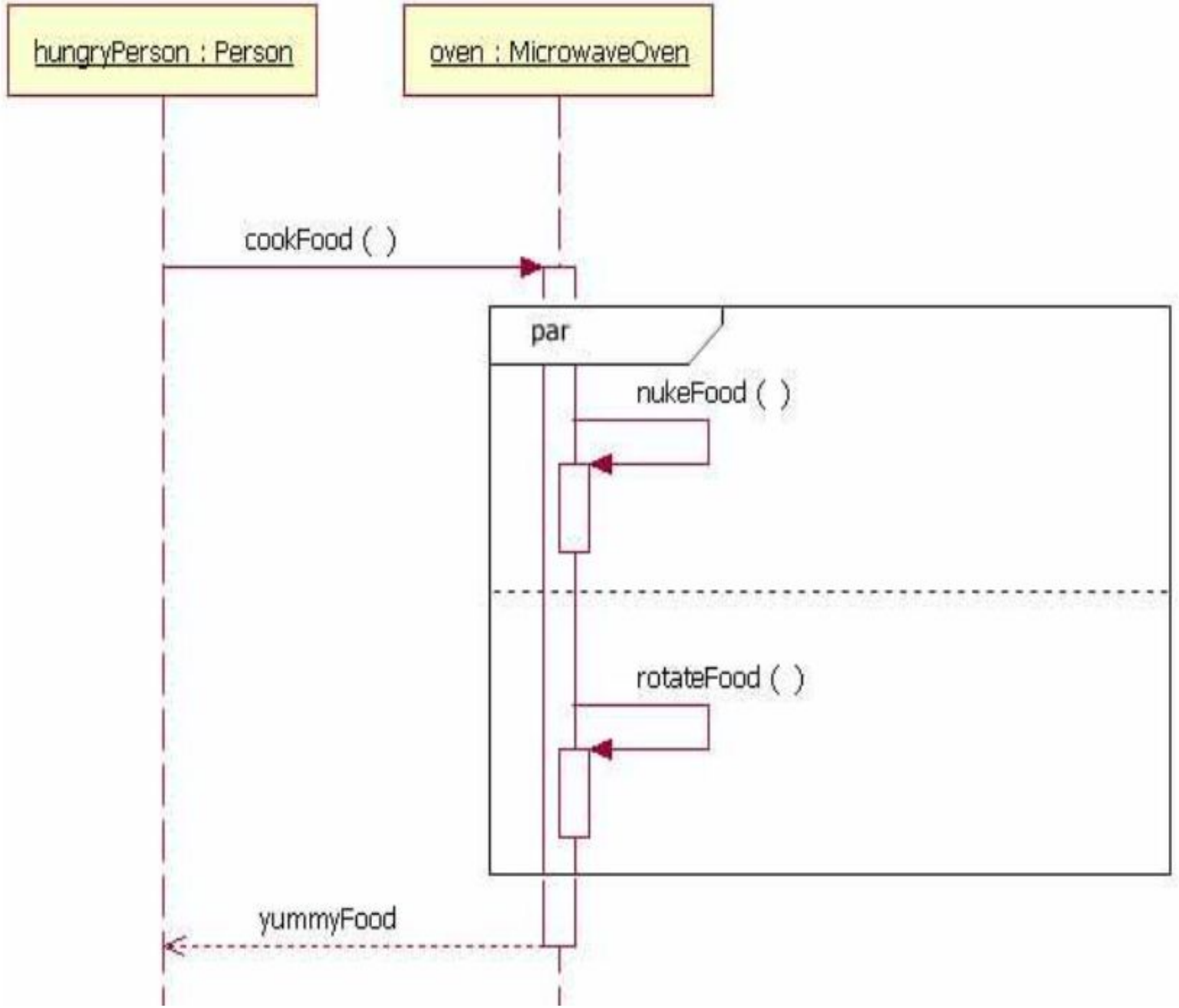
- An ALT frame is placed around the mutually exclusive alternatives



Parallel Frame

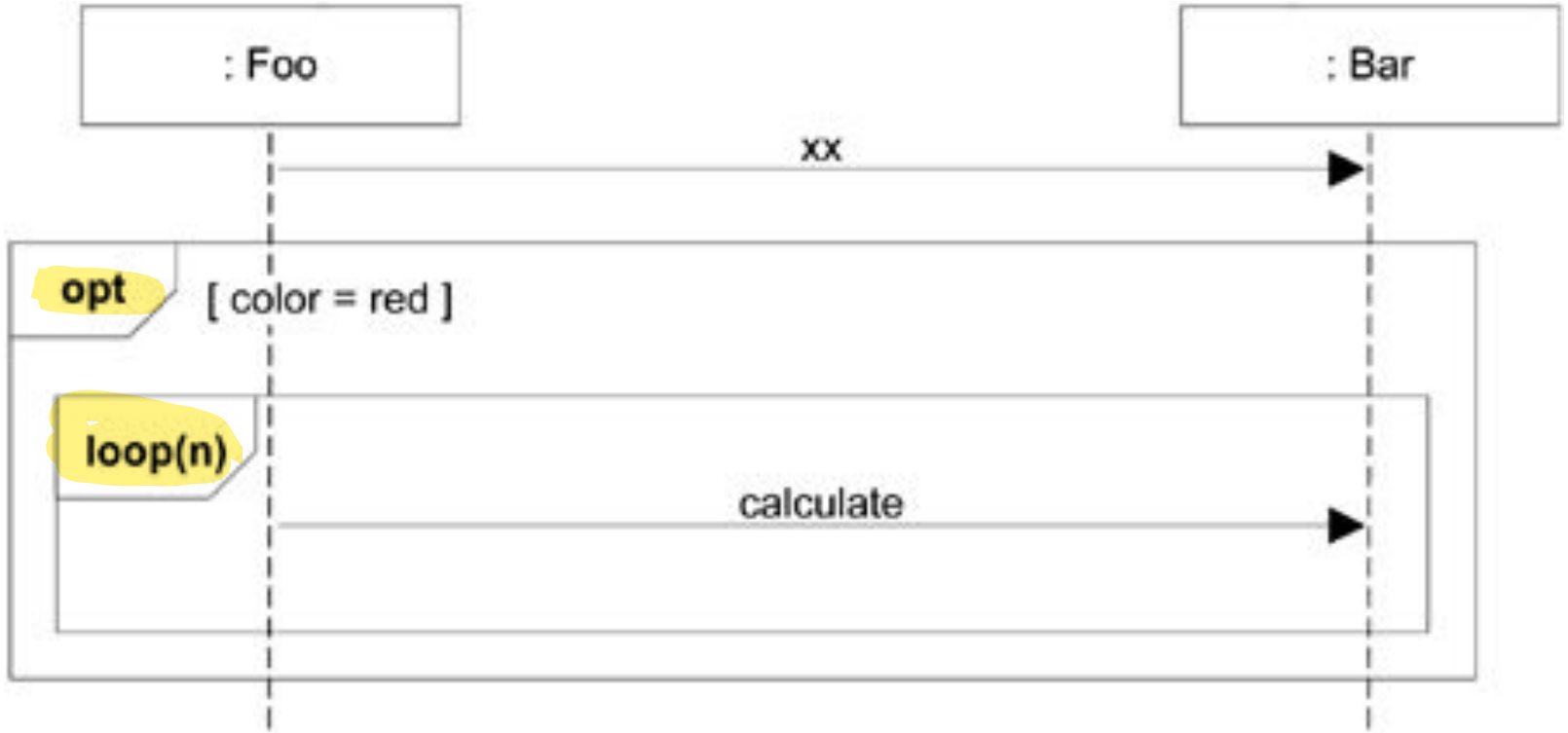
- When the processing time required to complete portions of a **complex task** is longer than desired, some systems handle parts of the processing in parallel.

- The parallel combination fragment element needs to be used when creating a sequence diagram that shows parallel processing activities.

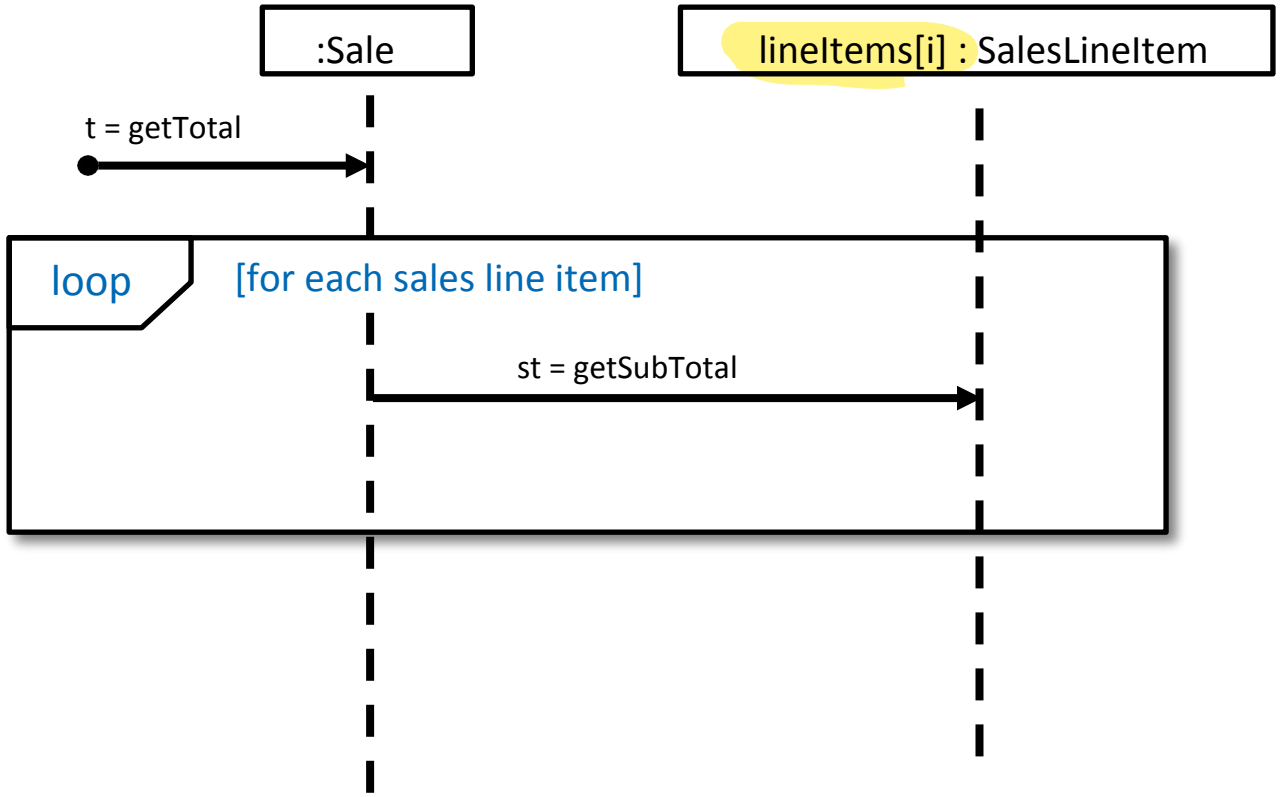


Nesting of Frames

- Frames can be nested



Use a UML loop frame to iterate over a collection

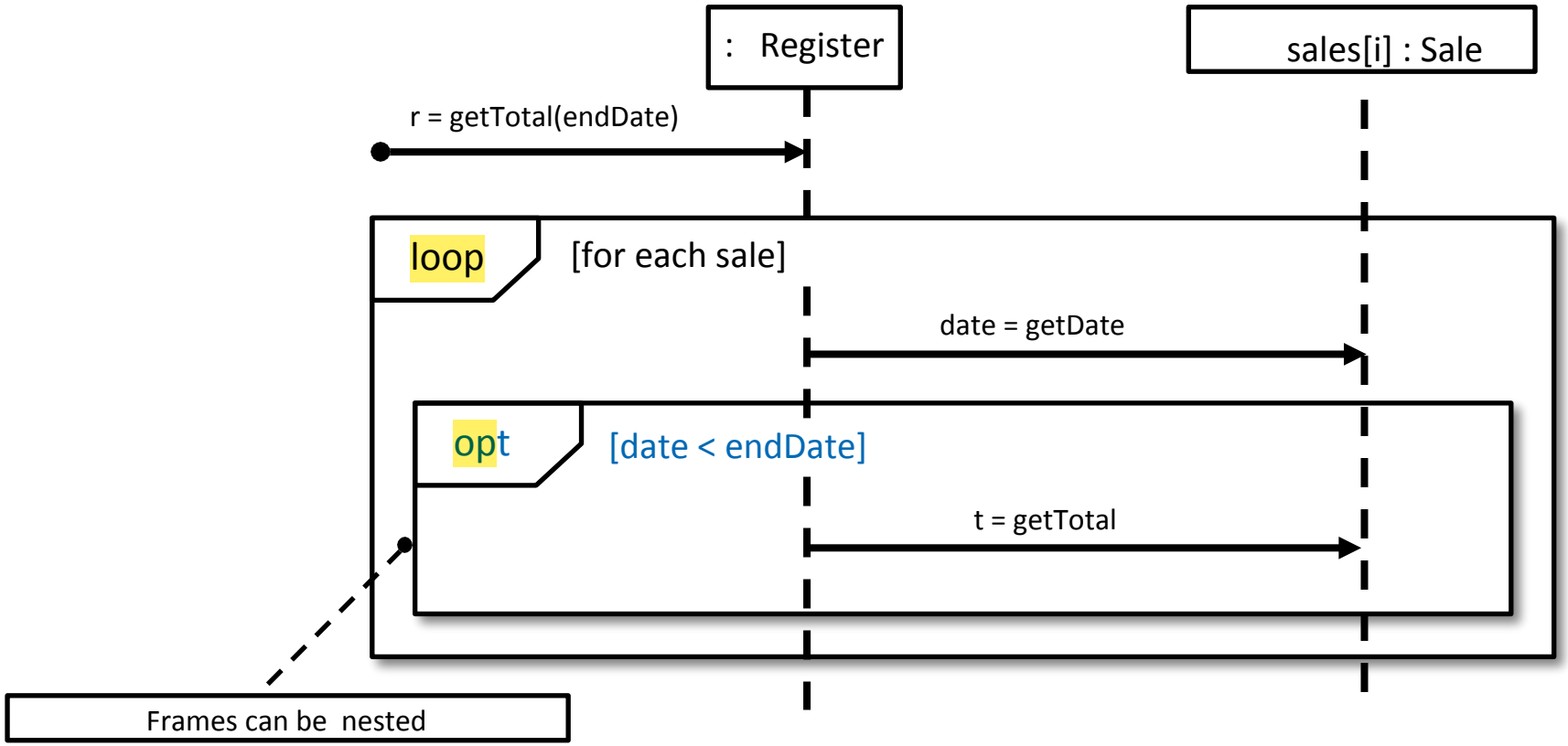


Modeling task: Calculate the total of a sale by summing up the sub totals for each sales line item.

How to model the sending of a message only if a guard condition matches?

```
Modeling task: Get the sum of all sales that happened before a
specific date.
```

Use a UML opt frame to model the sending of a message if the guard condition matches.



References

- Larman, Craig. Applying UML and patterns: an introduction to object-oriented analysis and design and interactive development. Pearson Education India, 2012.

- Object-Oriented Analysis and Design with Applications, Grady Booch et al., 3rd Edition, Pearson, 2007.

- Timothy C. Lethbridge, Robert Laganaiere, Object-Oriented Software Engineering (2nd Edition), McGraw Hill, 2005

- Object-Oriented Modeling and Design with UML, Michael R. Blaha and James R. Rumbaugh, 2nd Edition, Pearson, 2005.