Interaction Diagrams

CSE 3203

Interaction Diagrams

During object design, a logical solution based on the object-oriented paradigm is developed. The heart of this solution is the creation of **interaction diagrams,** which illustrate how objects collaborate to fulfill the requirements.

After—or in parallel with—drawing interaction diagrams, (design) **class diagrams** can be drawn. These summarize the definition of the software classes (and interfaces) that are to be implemented in software.

In terms of the UP, these artifacts are part of the **Design Model.**

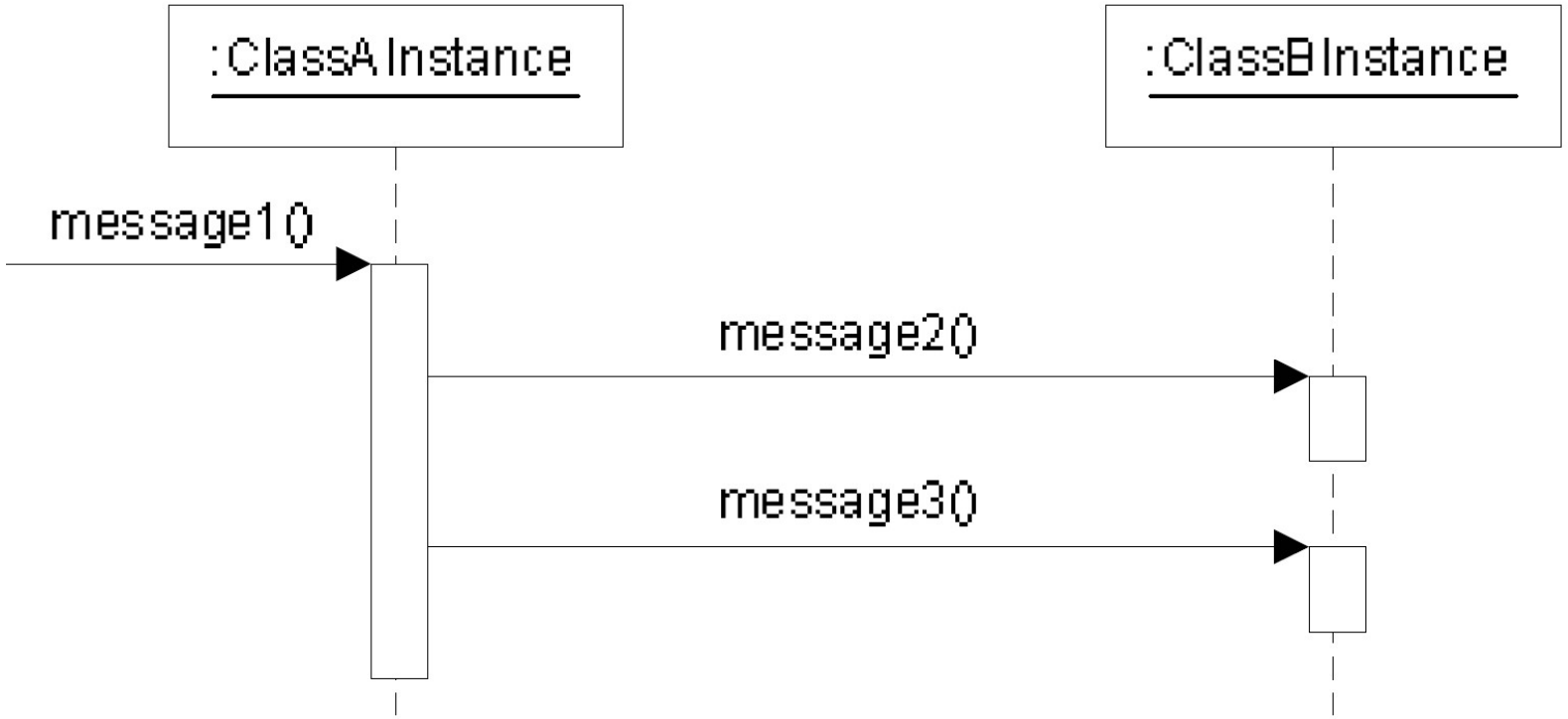
In practice, the creation of interaction and class diagrams happens in parallel and synergistically, but their introduction is linear in this case study, for simplicity and clarity.

Interaction Diagrams

- Interaction diagrams illustrate how objects interact via messages.
- The term interaction diagram, is a generalization of specialized UML diagrams types.
- Interaction diagrams can be classified into two categories.

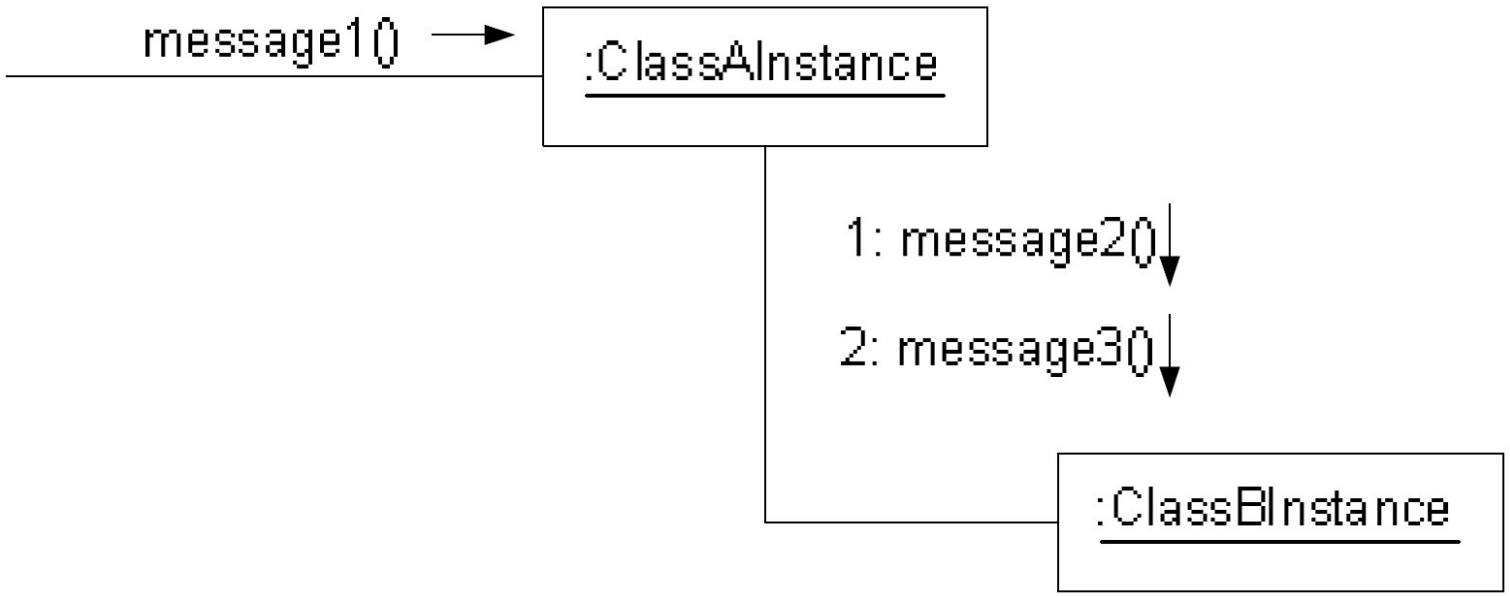
- Can you identify one of the categories of interaction diagrams?

Sequence Diagram!



Collaboration Diagram

- Illustrates object interactions in a graph or network format.
- Objects can be placed anywhere on the diagram.

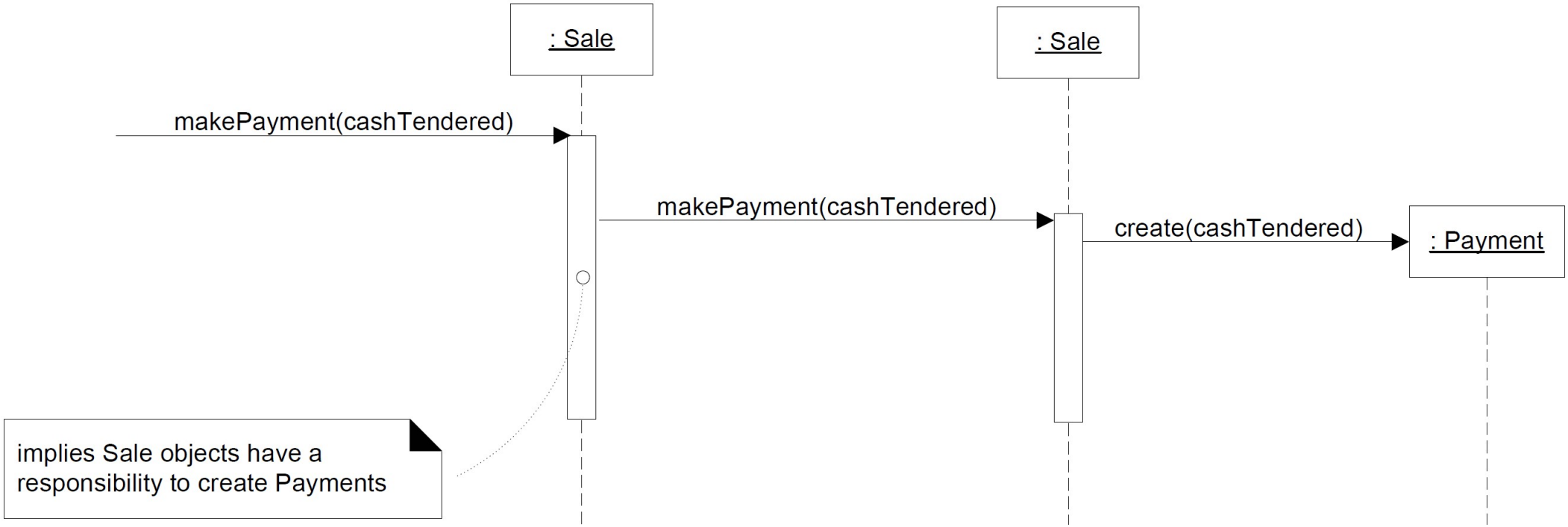


Type of Diagram	Strengths	Weaknesses
Sequence	Clearly shows sequence or time ordering of messages	Forced to extend to the right when adding new objects; consumes horizontal space
	Simple notation	
Collaboration	Space economical – flexibility to add new objects in two dimensions	Difficult to see sequence of messages
	Better to illustrate complex branching, iteration, and concurrent behaviour	More complex notation

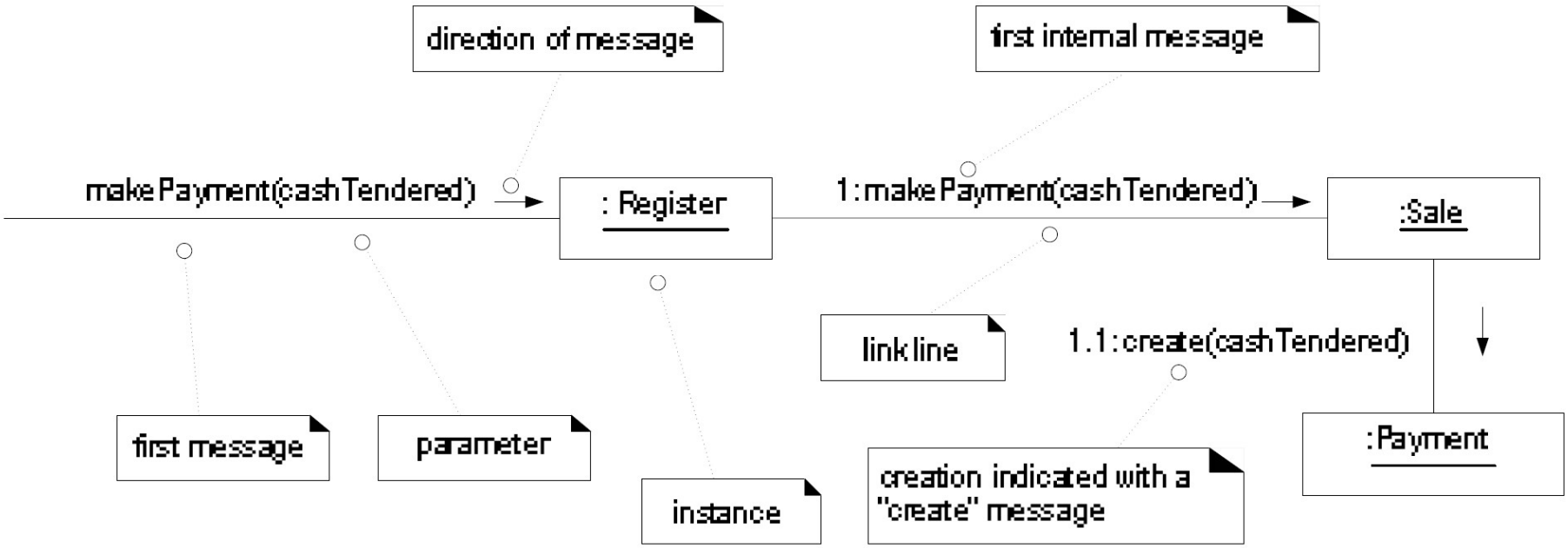
Example

1. The message *makePayment is* sent to an instance of a *Register.* The sender is not identified.
2. The *Register* instance sends the *makePayment* message to a *Sale* instance.
3. The *Sale* instance creates an instance of a *Payment.*

Sequence Diagram



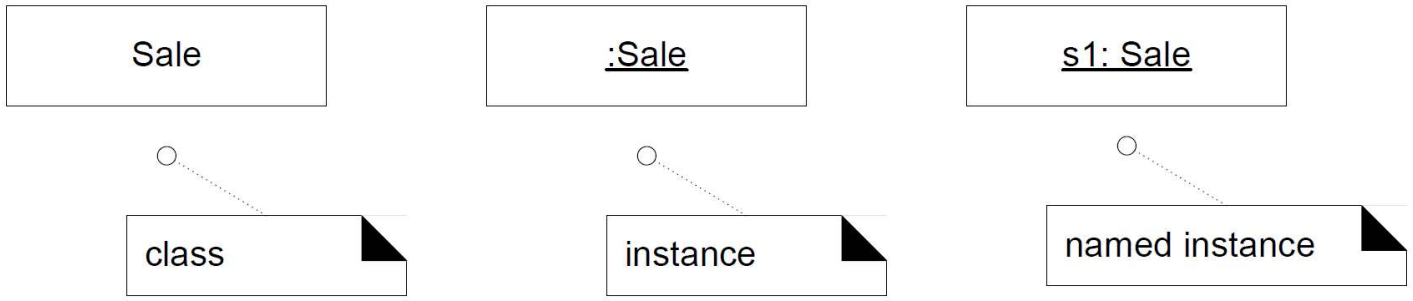
Collaboration Diagram



Notation: Elements

The UML has adopted a simple and consistent approach to illustrate **instances** vs. classifiers (see Figure 15.5):

- *For* any kind of UML element (class, actor, ...), an instance uses the same graphic symbol as the type, but the designator string is underlined.

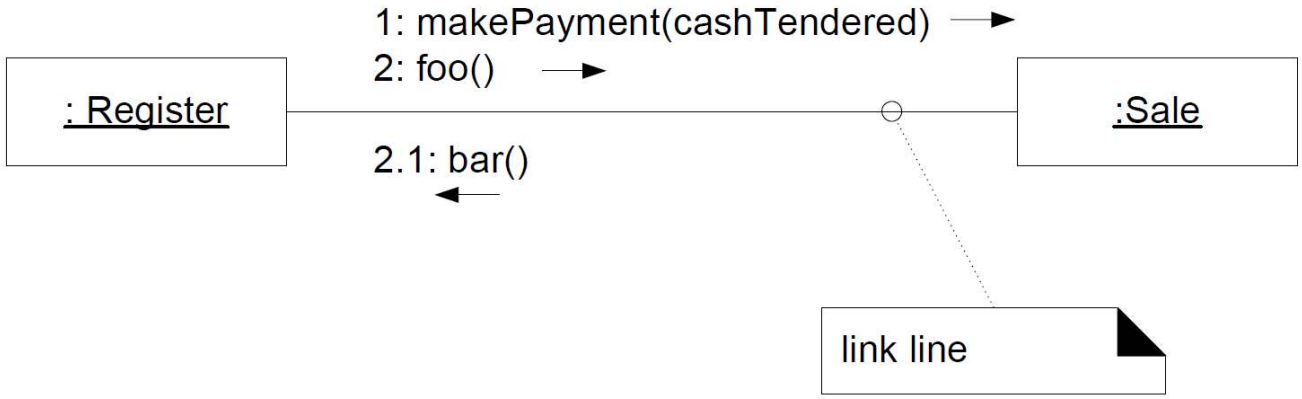


Therefore, to show an instance of a class in an interaction diagram, the regular class box graphic symbol is used, but the name is underlined.

A name can be used to uniquely identify the instance. If none is used, note that a ":" precedes the class name.

Notation: Links

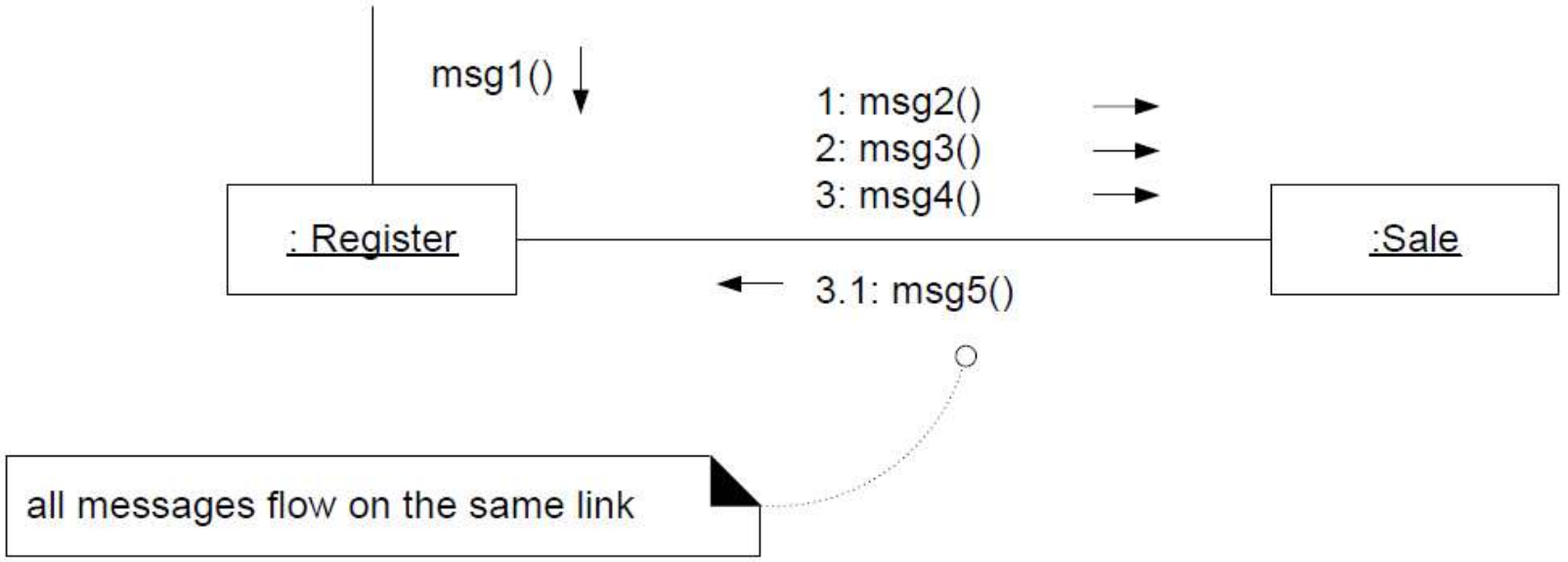
A link is a connection path between two objects; it indicates some form of navigation and visibility between the objects is possible (see Figure 15.6). More formally, a link is an instance of an association. For example, there is a link—or path of navigation—from a *Register* to a *Sale,* along which messages may flow, such as the *makePayment* message.



Note that multiple messages, and messages both ways, can flow along the same single link.

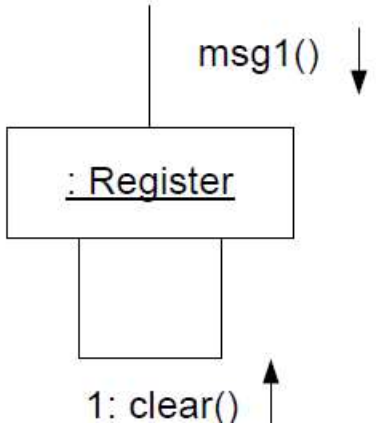
Notation: Messages

- Each message between objects is represented with a message expression and small arrow indicating the direction of the message.



Sending a Message to Self

A message can be sent from an object to itself (Figure 15.8). This is illustrated by a link to itself, with messages flowing along the link.

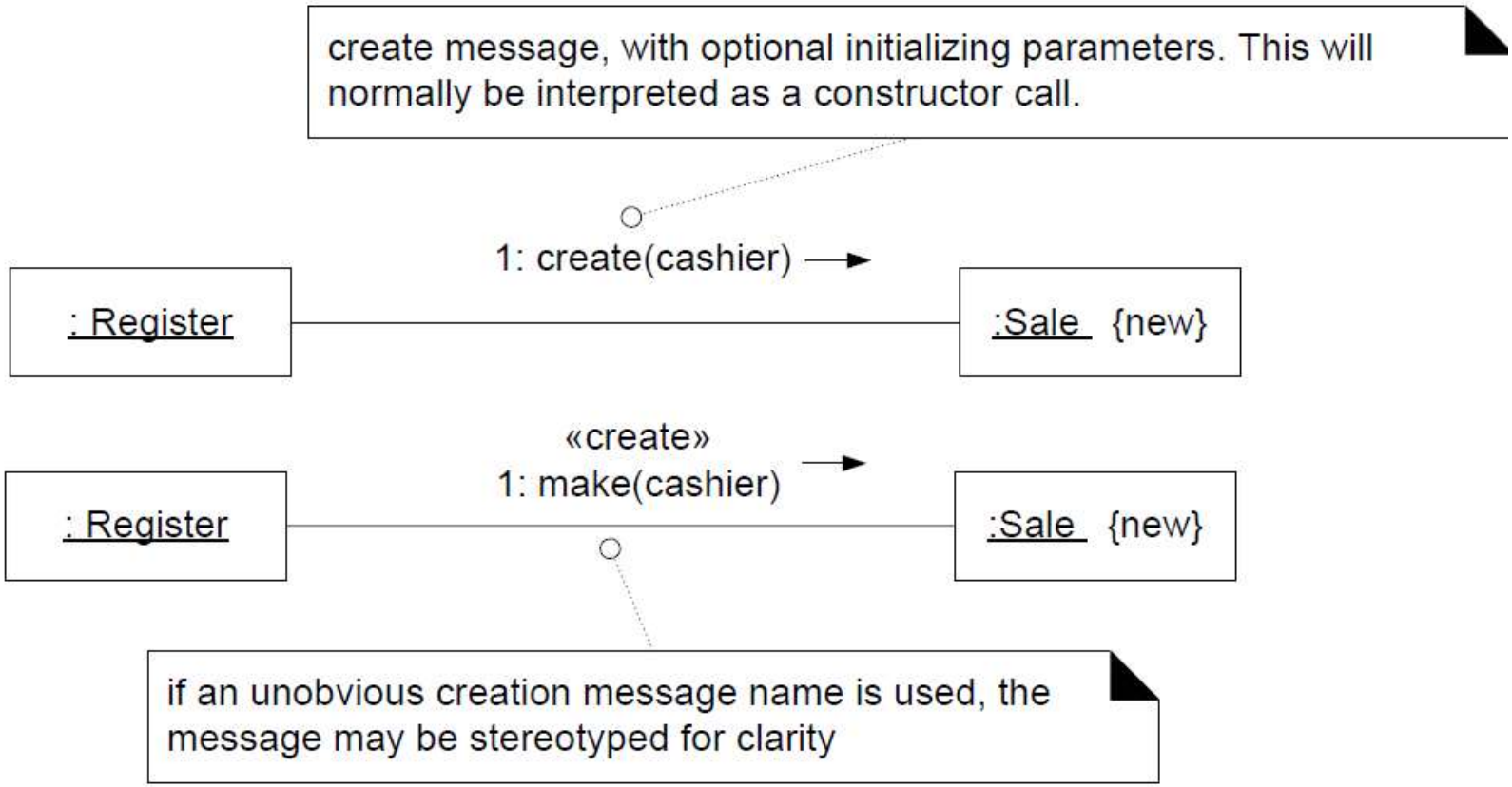


Creation of Instances

Any message can be used to create an instance, but there is a convention in the UML to use a message named *create* for this purpose. If another (perhaps less obvious) message name is used, the message may be annotated with a special feature called a UML stereotype, like so: *«create»*.

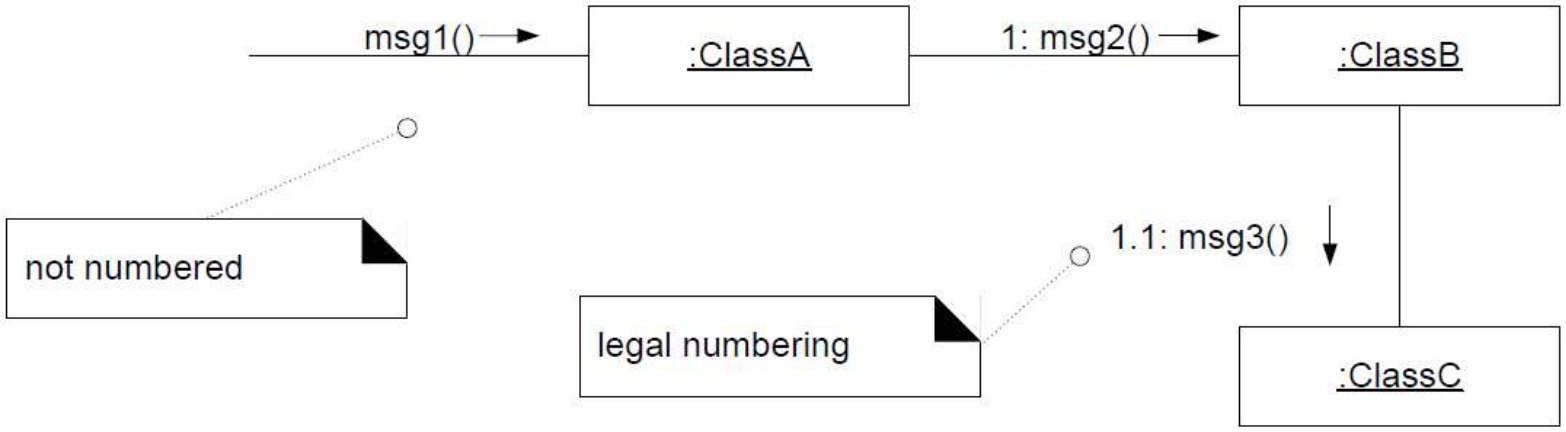
The *create* message may include parameters, indicating the passing of initial values. This indicates, for example, a constructor call with parameters in Java.

Creation of Instances



Message Number Sequencing

1. The first message is not numbered. Thus,*msg1()* is unnumbered.
2. The order and nesting of subsequent messages is shown with a legal num bering scheme in which nested messages have a number appended to them. Nesting is denoted by prepending the incoming message number to the out going message number.



Example of Complex Sequence Numbering