High Level Design-2

- Interaction diagrams
  - Communication diagrams
    - Links, messages, message numbers, iteration, conditions

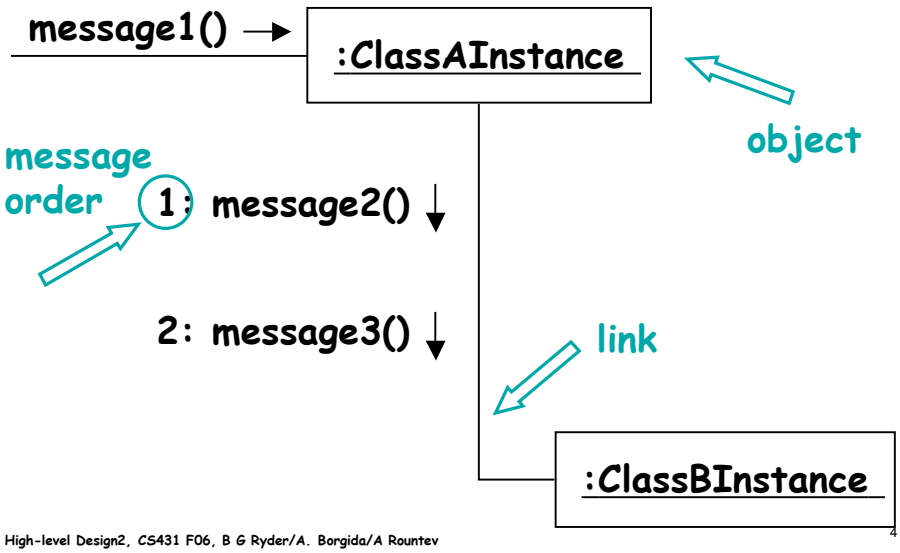
UML Interaction Diagrams

- Key tool in object-oriented design
- Show objects and their interactions
- While creating the interaction diagrams, the designer makes decisions about object responsibilities and object interactions
- Interaction diagrams require creativity
  - Harder than use cases and domain models

Communication vs. Sequence Diagrams

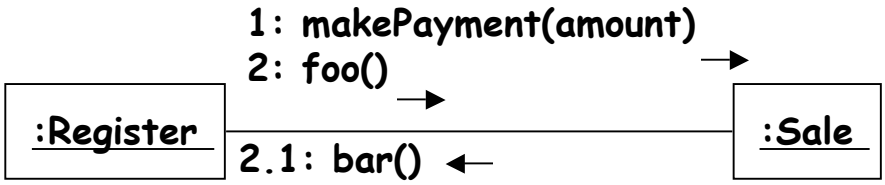
- Represent the same information
  - Can be used interchangeably
- Advantages and disadvantages
  - Collaboration diagrams: harder to keep track of the flow of control, but save space
  - Sequence diagrams are more "verbose"
- In practice people seem to prefer sequence diagrams

A Simple Communication Diagram



Links

- Links: shows that messages may flow between two objects
  - Many messages may flow along the same link
  - Messages may flow in both directions



Messages

"return:=message(parameter:Type):ReturnType"

- UML notation for messages
- Sometimes there is no return value

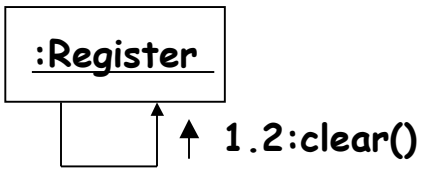
```
activate(level:StartLevel,num_users:Integer)
```

- Type information may be excluded if obvious or unimportant

```
spec:=getProductSpec(id)
spec:=getProductSpec(id:ItemID)
spec:=getProductSpec(id:ItemID):ProductSpec
```

Messages, cont.

- A small arrow indicates direction
- Sequence number describes the ordering
  - e.g., 1, 2.1, 3.4.1
- Messages to "self" (`this`) are possible



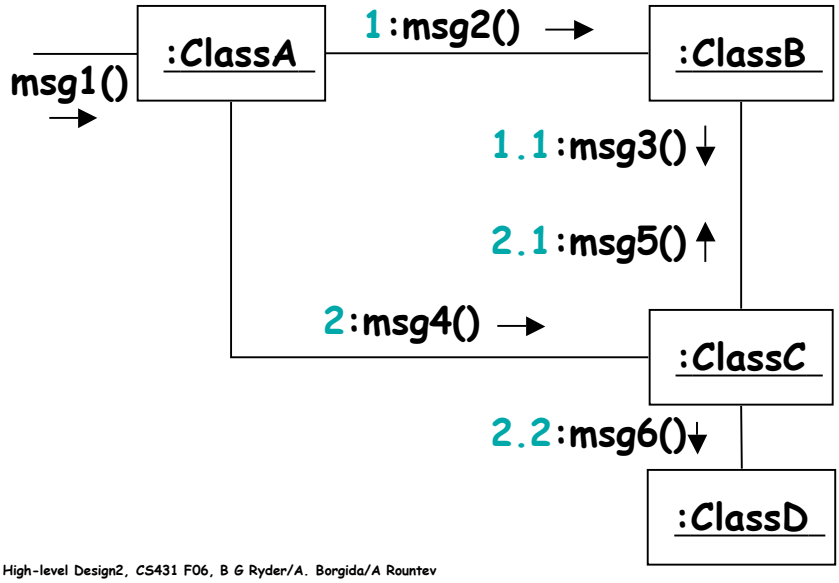
Creation of Instances

- UML convention: message named create
- May include parameters
  - Indicates the passing of initial values
- {new} may optionally be added

Message Numbers

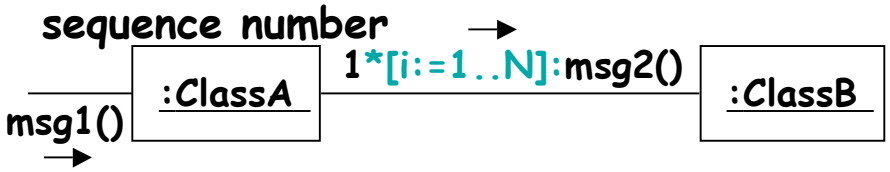
- The first message is not numbered
  - Represents an action/event that triggers the collaboration
- If a message m with number x.y.z is received, the messages that are sent during the processing of m are numbered x.y.z.1, x.y.z.2, etc.
  - And the rule is applied recursively to these outgoing messages

Example



Iteration

- An iteration clause is appended to the sequence number



- If the details of the iteration clause are not important: just "*"
  - e.g. 3.1*:update()

Iteration over a Collection

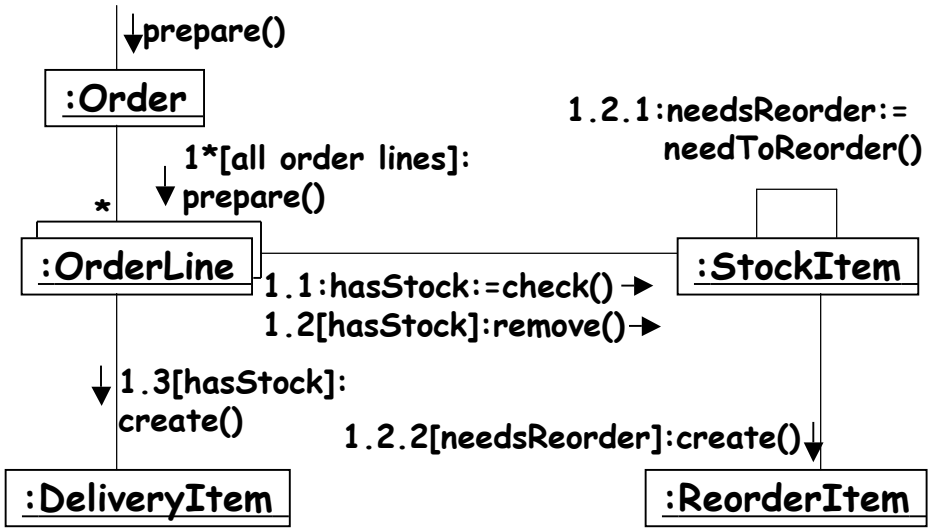
- Very common: iterate over all objects in a collection (e.g. list, map, etc.)
  - e.g. in Java: done through `java.util.Iterator`
  - UML: the set of objects is called a multiobject
- The message is sent to each object in the collection (not to the collection object)

Conditional Messages

- A condition clause is appended to the sequence number

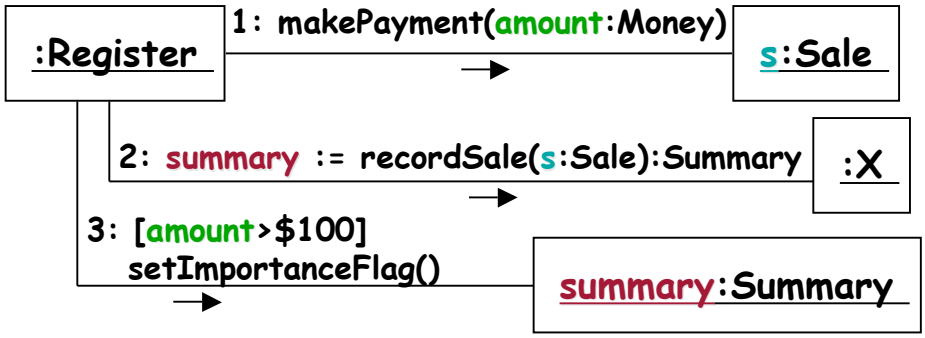
- The message is sent only if the condition is true

Another Example



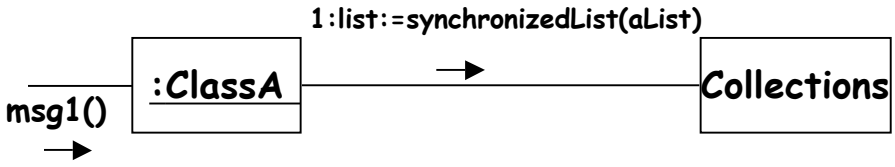
Names of Objects

- Representation of parameters and return values that are references to objects



Messages to Classes

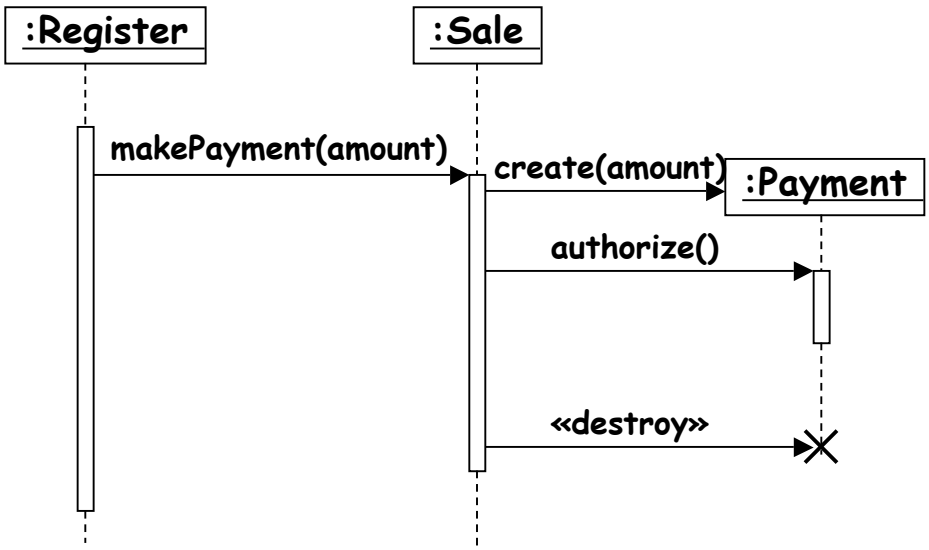
- It is possible to send a message to a class rather than to an object
  - This invokes a class operation
  - e.g. in Java/C++ such operation are implemented by static methods



Sequence Diagrams

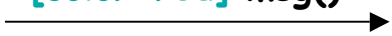
- Examples of how to do what we did with communication diagrams, with sequence diagrams instead
  - Go over yourselves and bring questions to class

Object Creation and Destruction



Conditions and Iteration

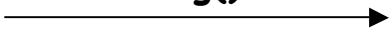
- A condition indicates when a message in sent [color=red] msg()



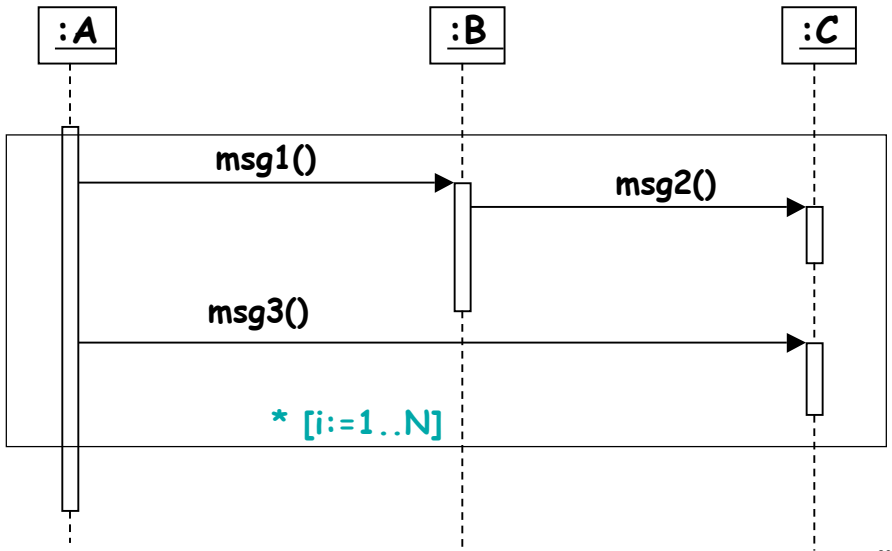
- Iteration for one message

*[i:=1..N]:msg()

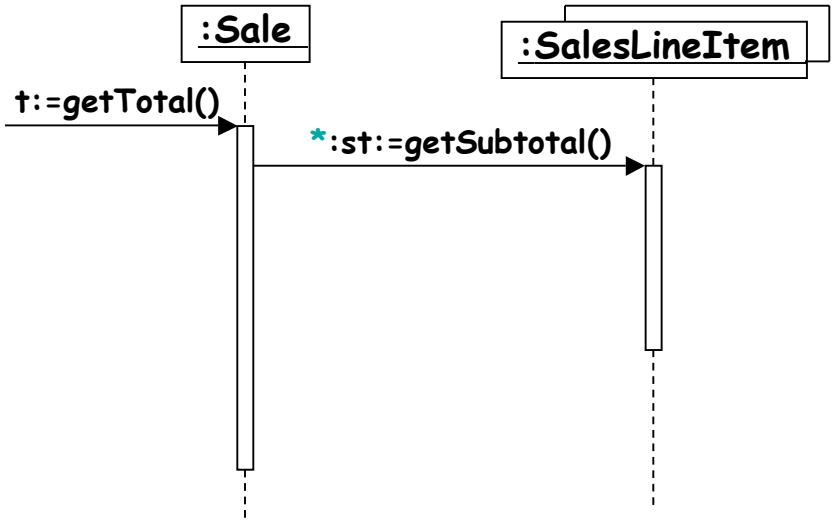
*:msg()



Iteration for Multiple Messages

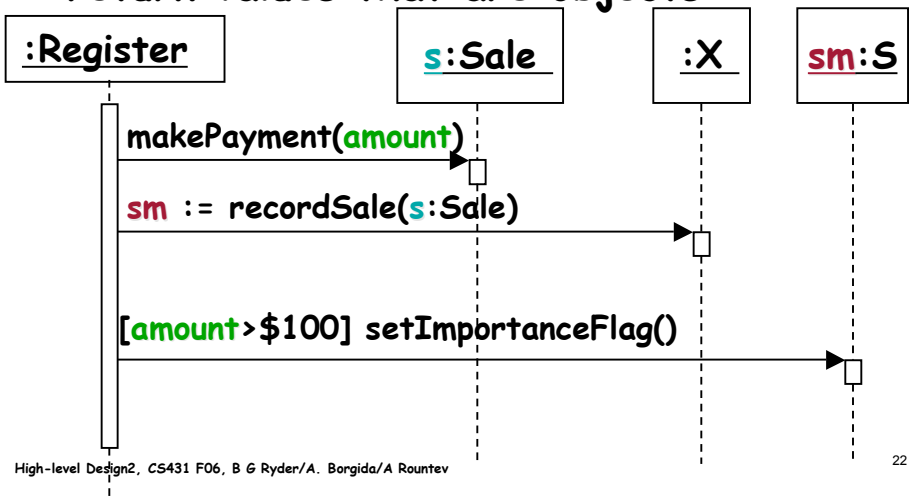


Iteration for a Multiobject



Names of Objects

- Representation of parameters and return values that are objects



Messages to Classes