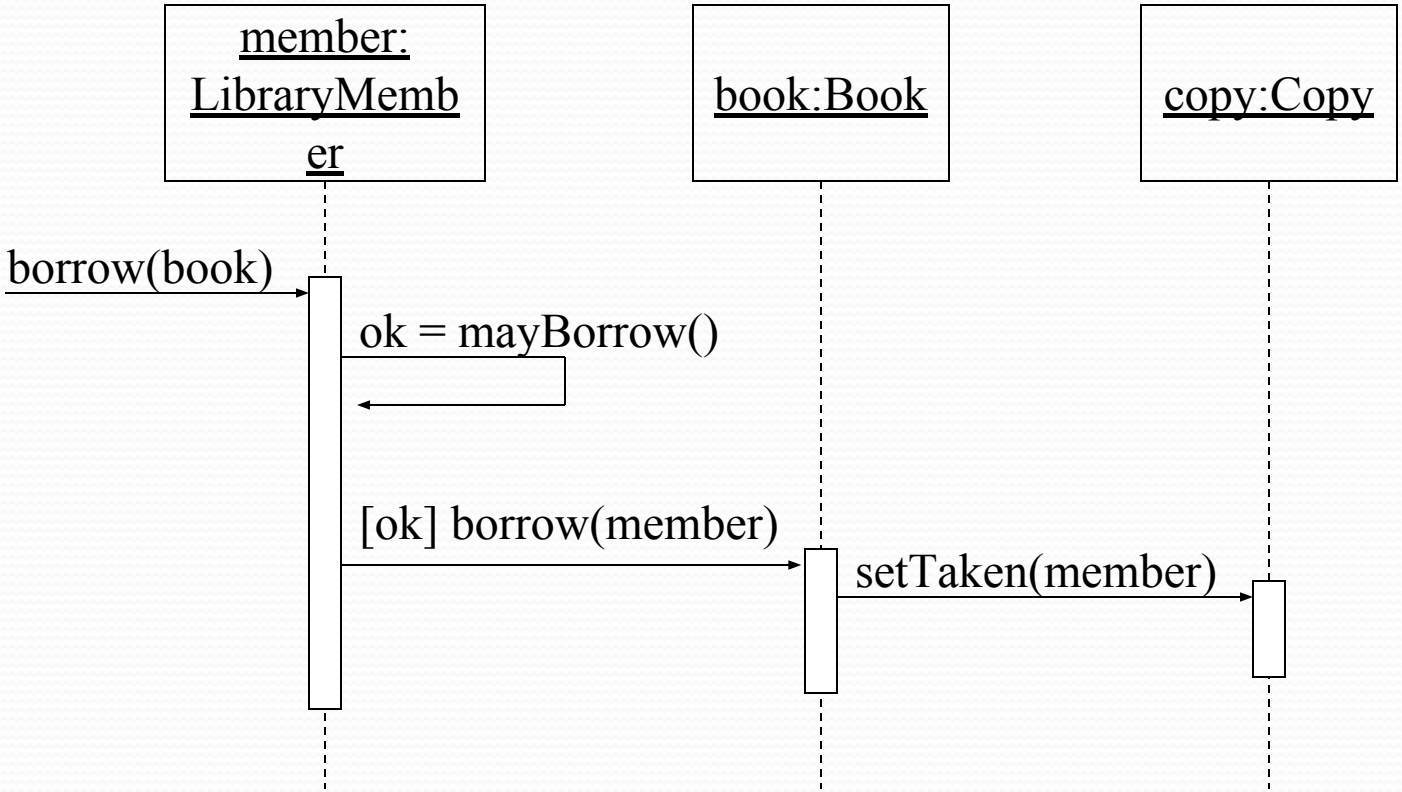
Sequence diagram

Dr. Taimoor Khan
Taimoor.khan@nu.edu.pk

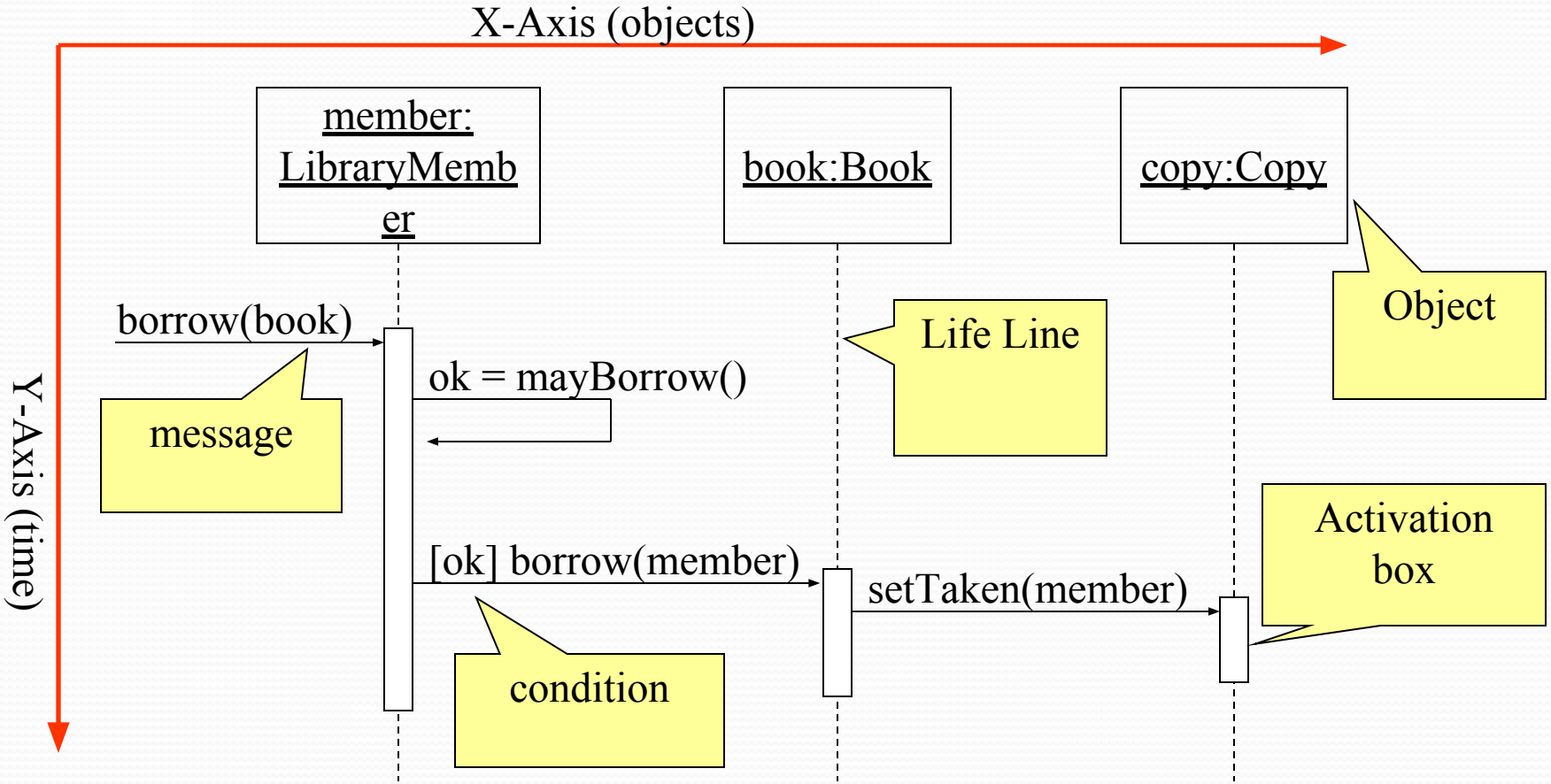
Sequence Diagrams

- Illustrates how objects interacts with each other.
- Emphasizes time ordering of messages.
- Can model simple sequential flow, branching, iteration, recursion and concurrency.

A Sequence Diagram

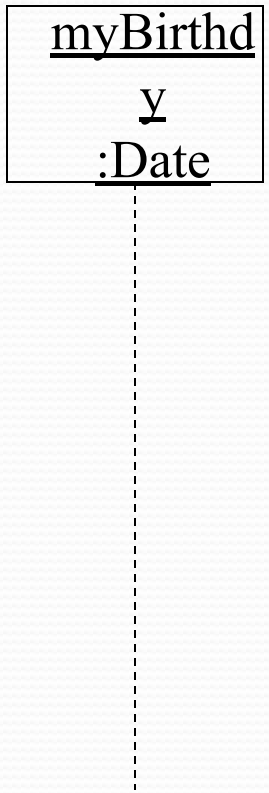


A Sequence Diagram



Object

- Object naming:
  - syntax: *[instanceName][:className]*
  - Name classes consistently with your class diagram
  - Include instance names when objects are referred to in messages
- The *Life-Line* represents the object's life during the interaction



Messages

- An interaction between two objects is performed as a message sent from one object to another
- If object obj_1 sends a message to another object obj_2 some link must exist between those two objects

Messages (Cont.)

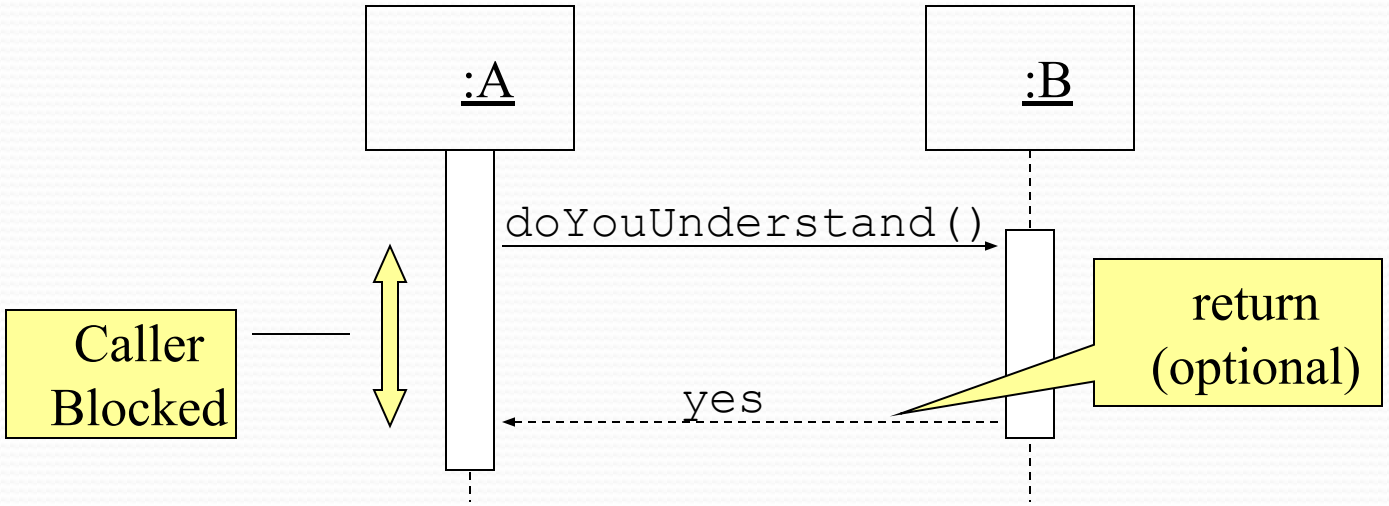
- A message is represented by an arrow between the life lines of two objects.
  - Self calls are also allowed
  - The time required by the receiver object to process the message is denoted by an *activation-box*.
- A message is labeled at minimum with the message name.
  - Arguments and control information (conditions, iteration) may be included.

Return Values

- Optionally indicated using a dashed arrow with a label indicating the return value.
  - Don't model a return value when it is obvious what is being returned, e.g. getTotal()
  - Model a return value only when you need to refer to it elsewhere, e.g. as a parameter passed in another message.
  - Prefer modeling return values as part of a method invocation, e.g. `ok = isValid()`

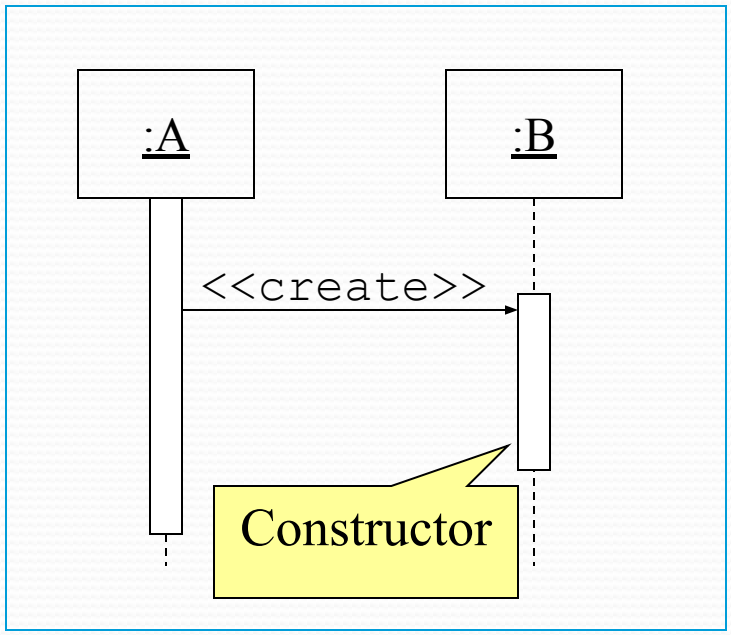
Synchronous Messages

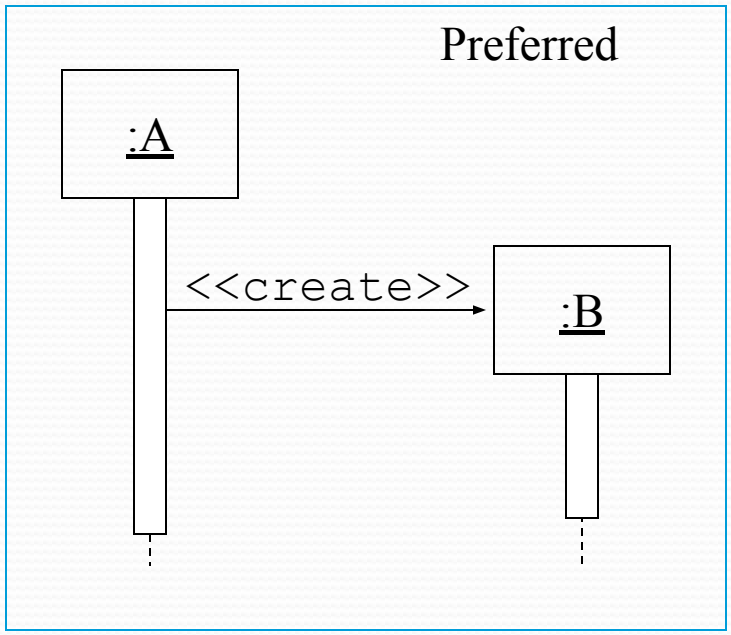
- Nested flow of control, typically implemented as an operation call.
  - The routine that handles the message is completed before the caller resumes execution.



Object Creation

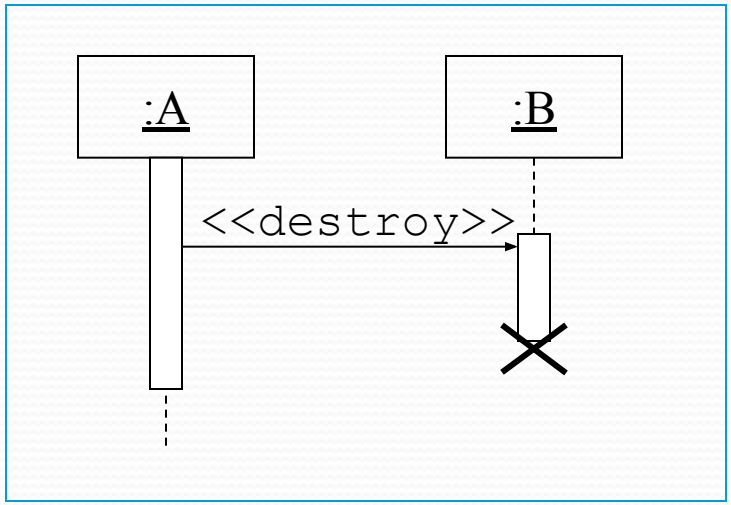
- An object may create another object via a `<<create>>` message.





Object Destruction

- An object may destroy another object via a `<<destroy>>` message.
  - An object may destroy itself.
  - Avoid modeling object destruction unless memory management is critical.



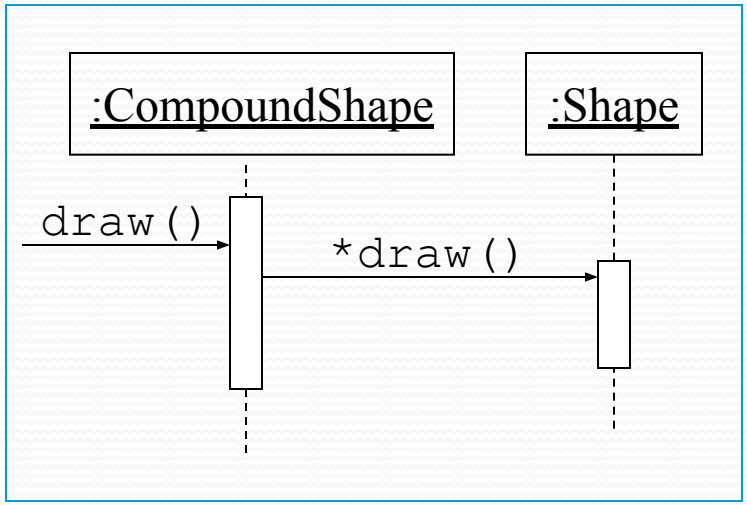
Control information

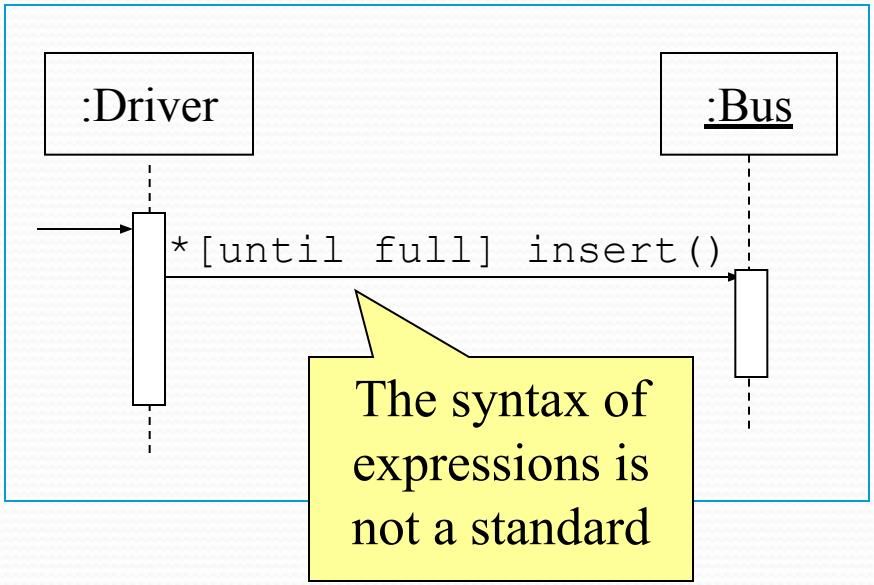
- Condition
  - syntax: ‘[‘ expression ’]’ message-label
  - The message is sent only if the condition is true
  - example:

- Iteration
  - syntax: * [‘[‘ expression ‘]’] message-label
  - The message is sent many times to possibly multiple receiver objects.

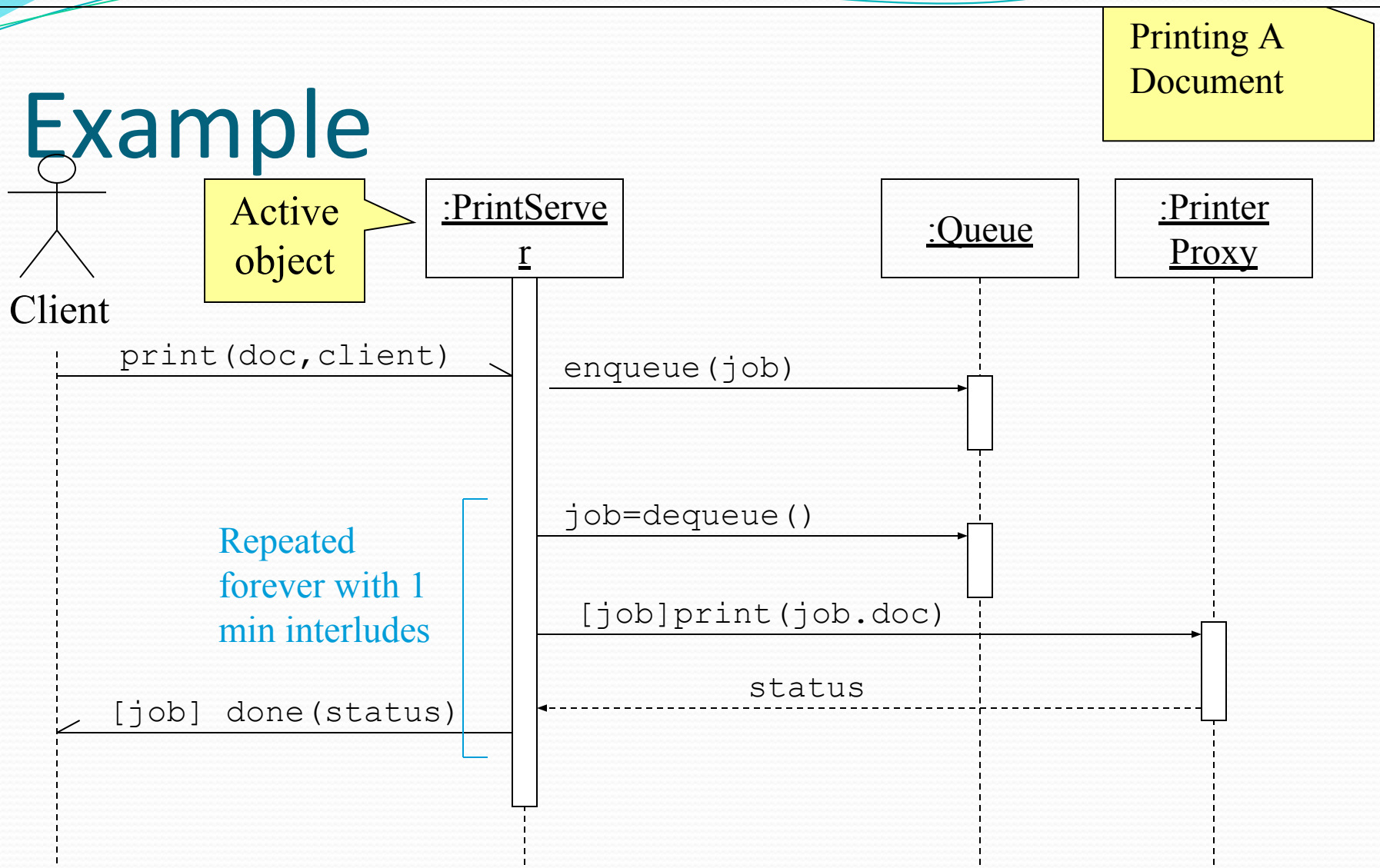
Control Information (Cont.)

- Iteration examples:





Example


Printing A Document
Client
Active object
:PrintServer
:Queue
:Printer Proxy
print(doc,client)
enqueue(job)
job=dequeue()
Repeated forever with 1 min interludes
[job]print(job.doc)
status
[job] done(status)