Sequence Diagram

SIMANTA MITRA

After this lesson, you will know

- what are sequence diagrams
- how to read sequence diagrams
- how to draw sequence diagrams

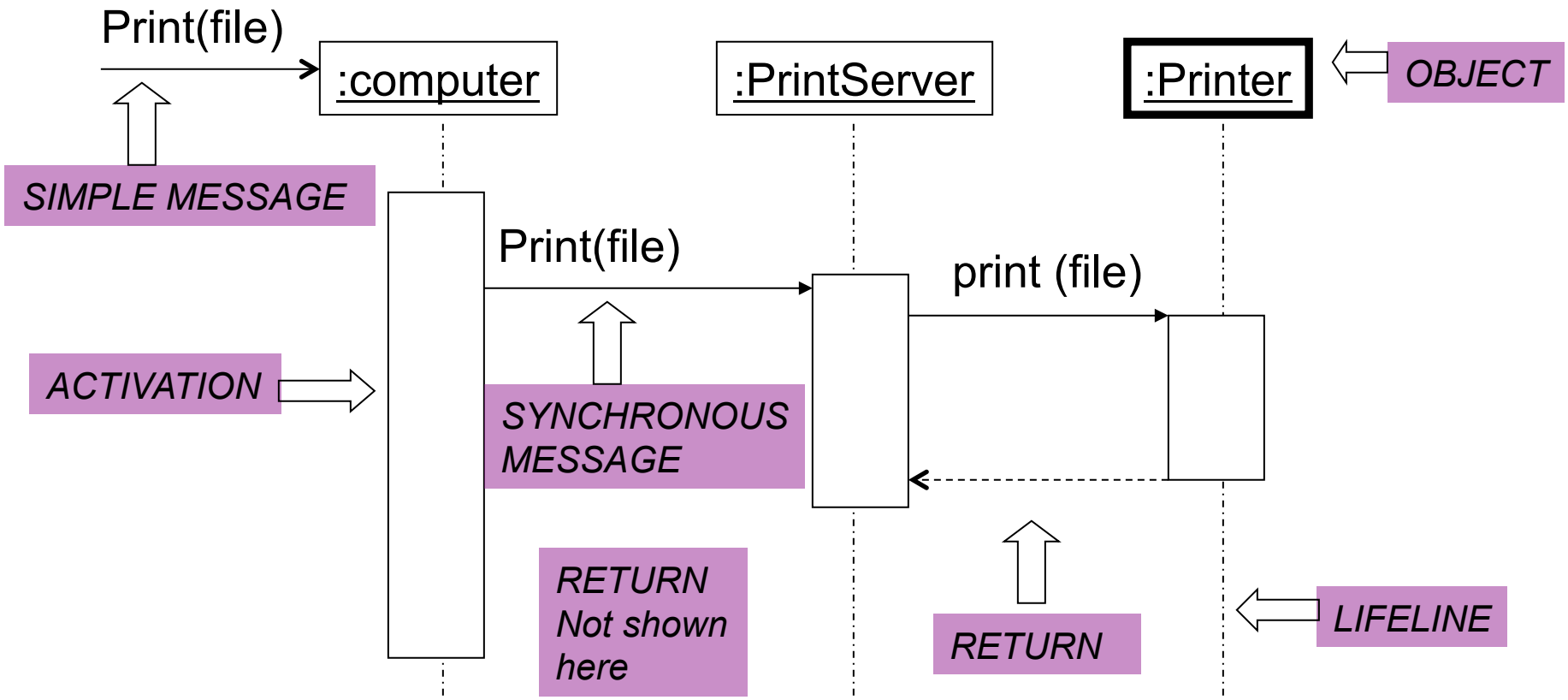
WHAT IS A SEQUENCE DIAGRAM?

Flowcharts are used to understand how procedural code works.

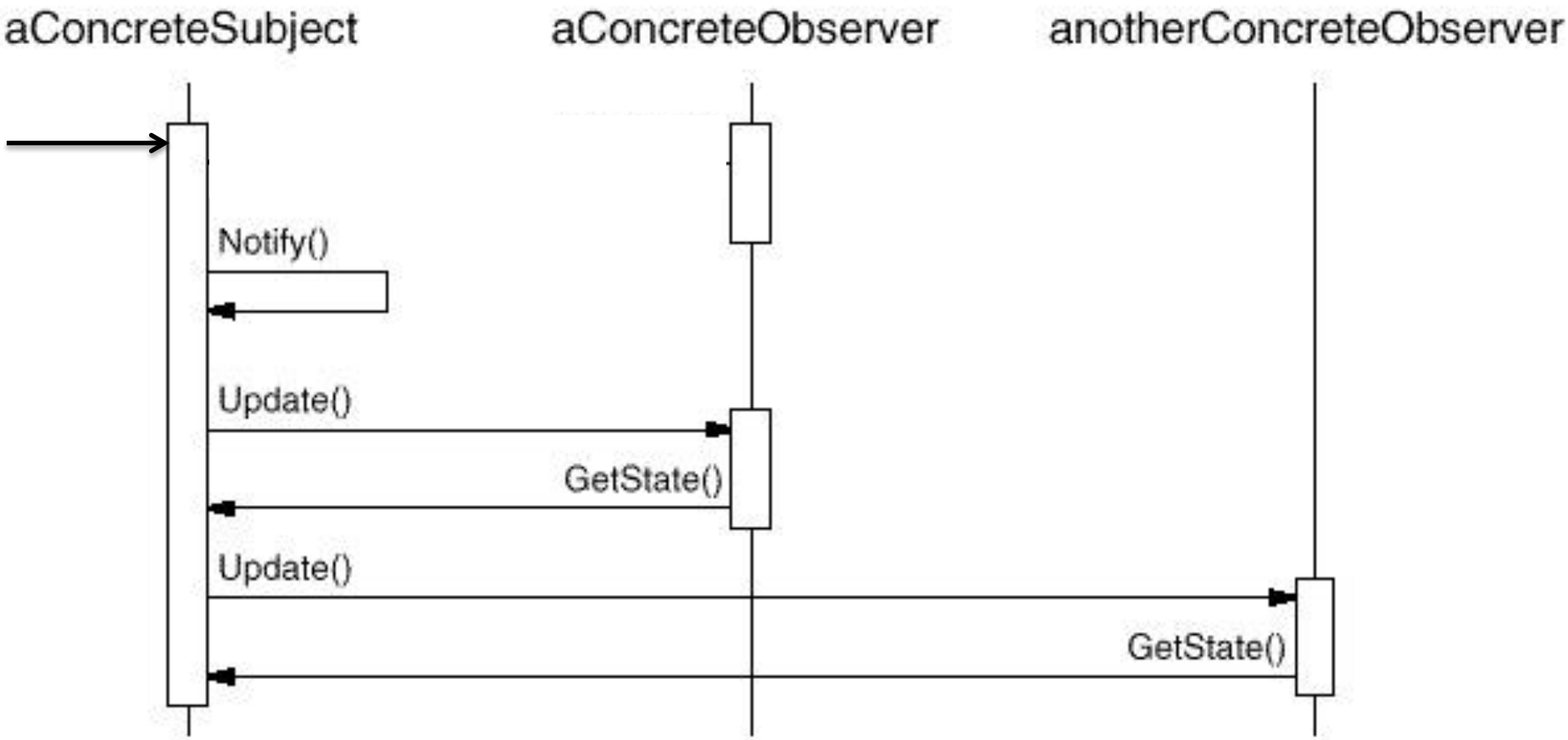
Sequence diagrams are used to understand how objects INTERACT with each other to accomplish a task.

A sequence diagram shows interactions between objects over time.

Example Sequence Diagram

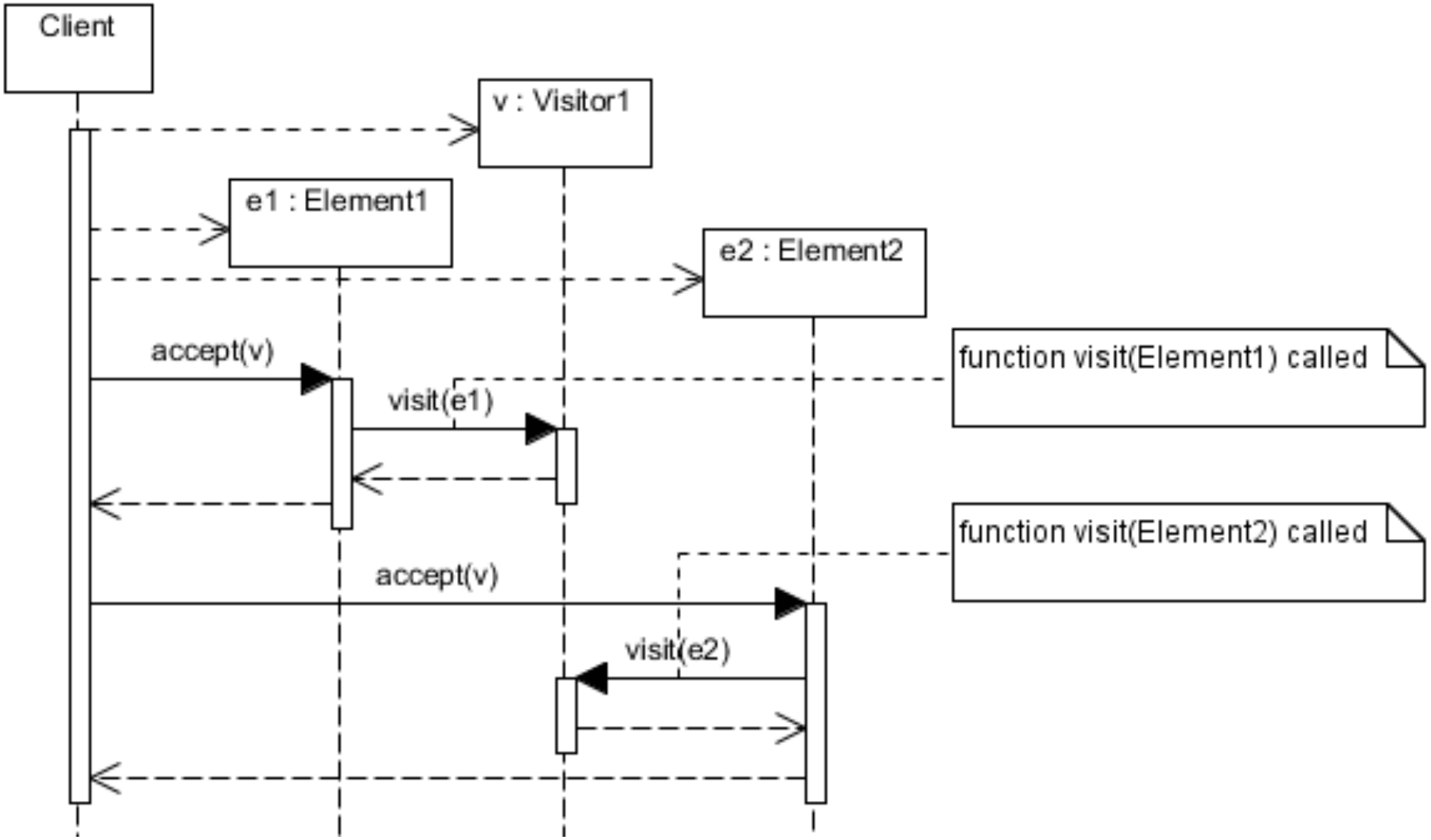


Observer Pattern



Note: there are errors in the diagram (but it gives the overall idea)

Visitor pattern



Note: there are errors in the diagram (but it gives the overall idea)

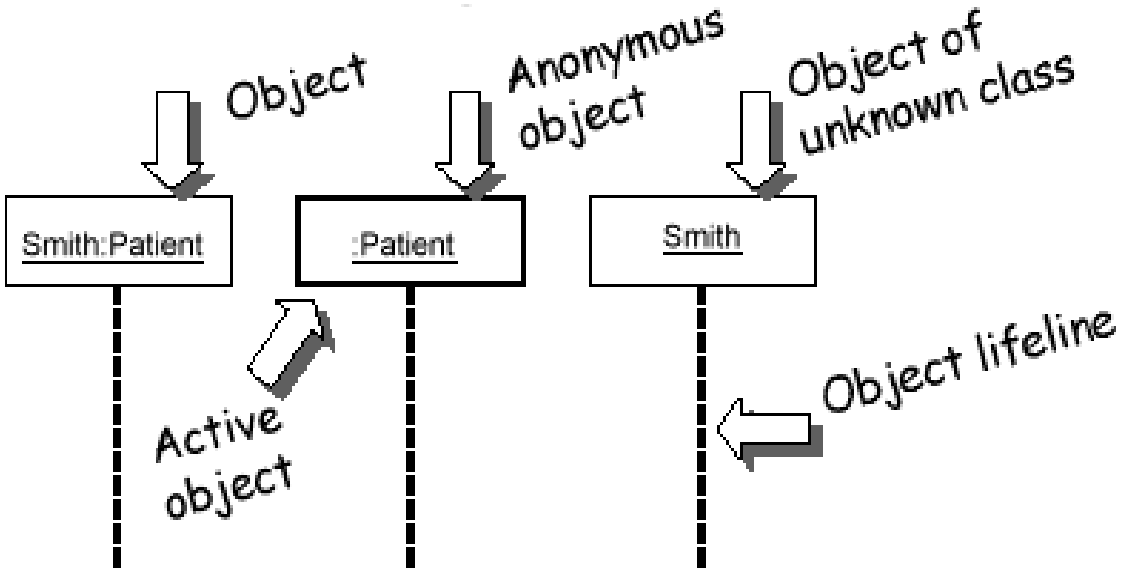
BASIC SEQUENCE DIAGRAMS

Five Elements of a basic seq diagram

1. *objects* – and not FUNCTIONS
2. *activation* only when object's method is on stack (i.e. activated)
3. messages are named
4. correct *arrowhead* (synchronous vs async)
5. *return* labeled with value (if needed)

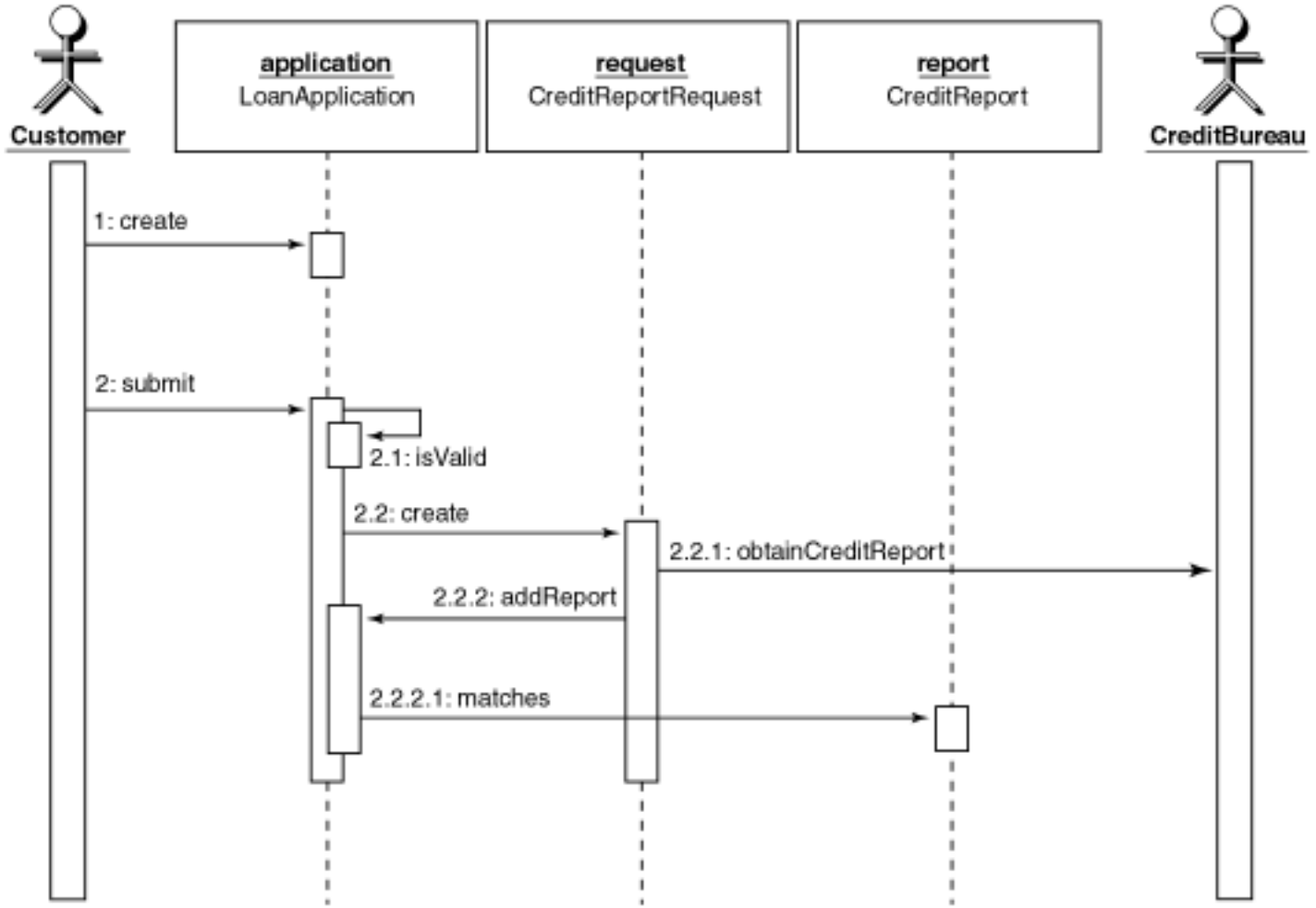
Objects

- Rectangles with object type, optionally preceded by object name and colon
  - write object's name if it clarifies the diagram
  - object's "**life line**" represented by dashed vertical line



Name syntax: <objectname>:<classname>

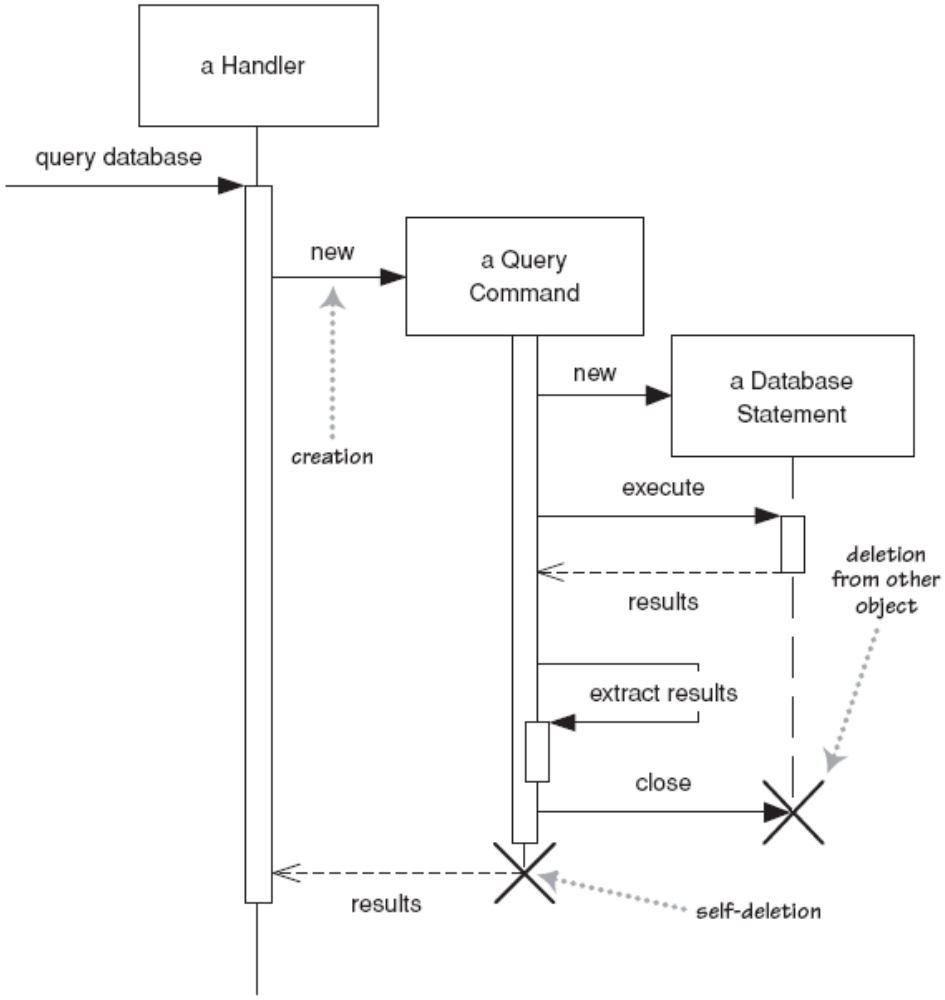
Actors in seq diagrams



Lifetime of objects

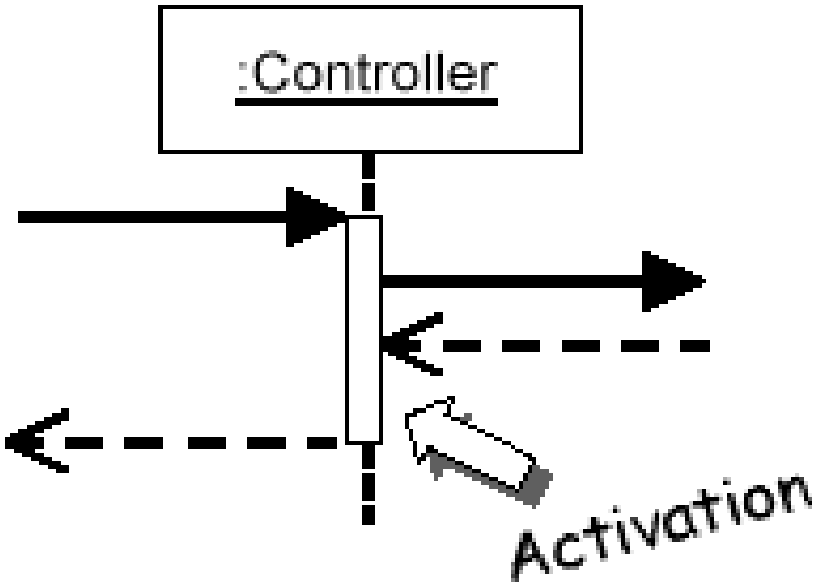
- creation: arrow with '**new**' written above it
  - notice that an object created after the start of the scenario appears lower than the others

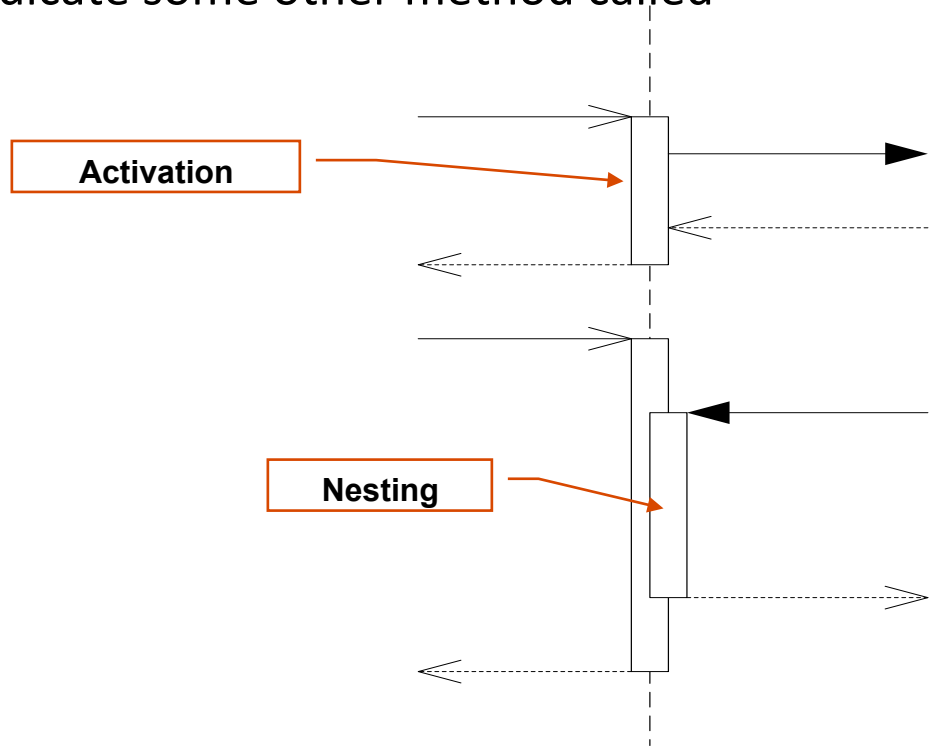
- deletion: an **X** at bottom of object's lifeline
  - Java doesn't explicitly delete objects; they fall out of scope and are garbage-collected



Activation: i.e. method calls

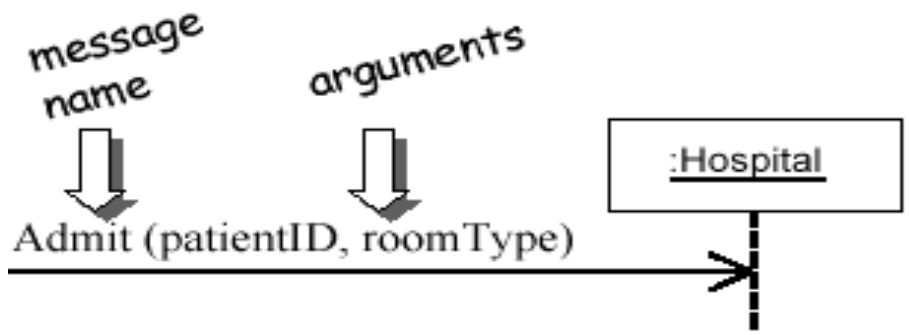
- **Activation**: thick box over object's life line; drawn when object's method is on the stack
  - object is running its code, or it is on the stack waiting for another object's method to finish
  - nest to indicate recursion OR to indicate some other method called



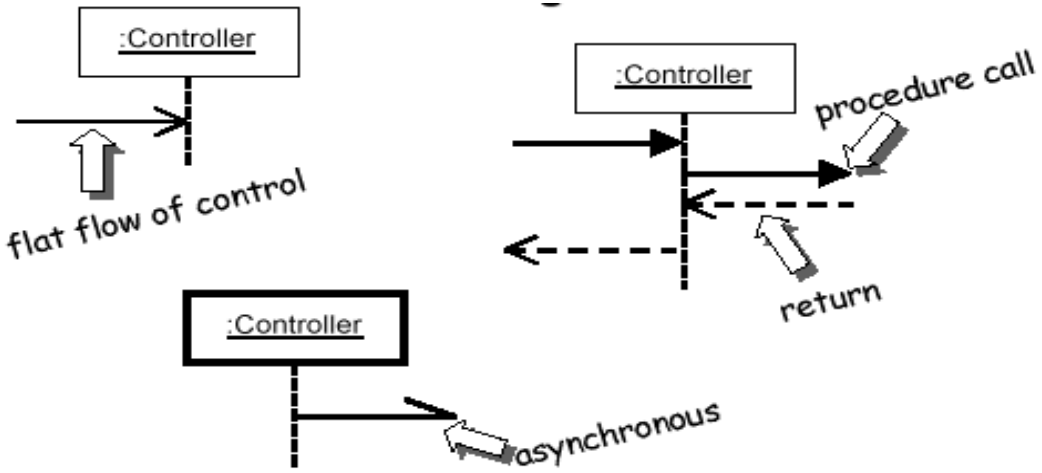


Messages, arrowheads, return messages

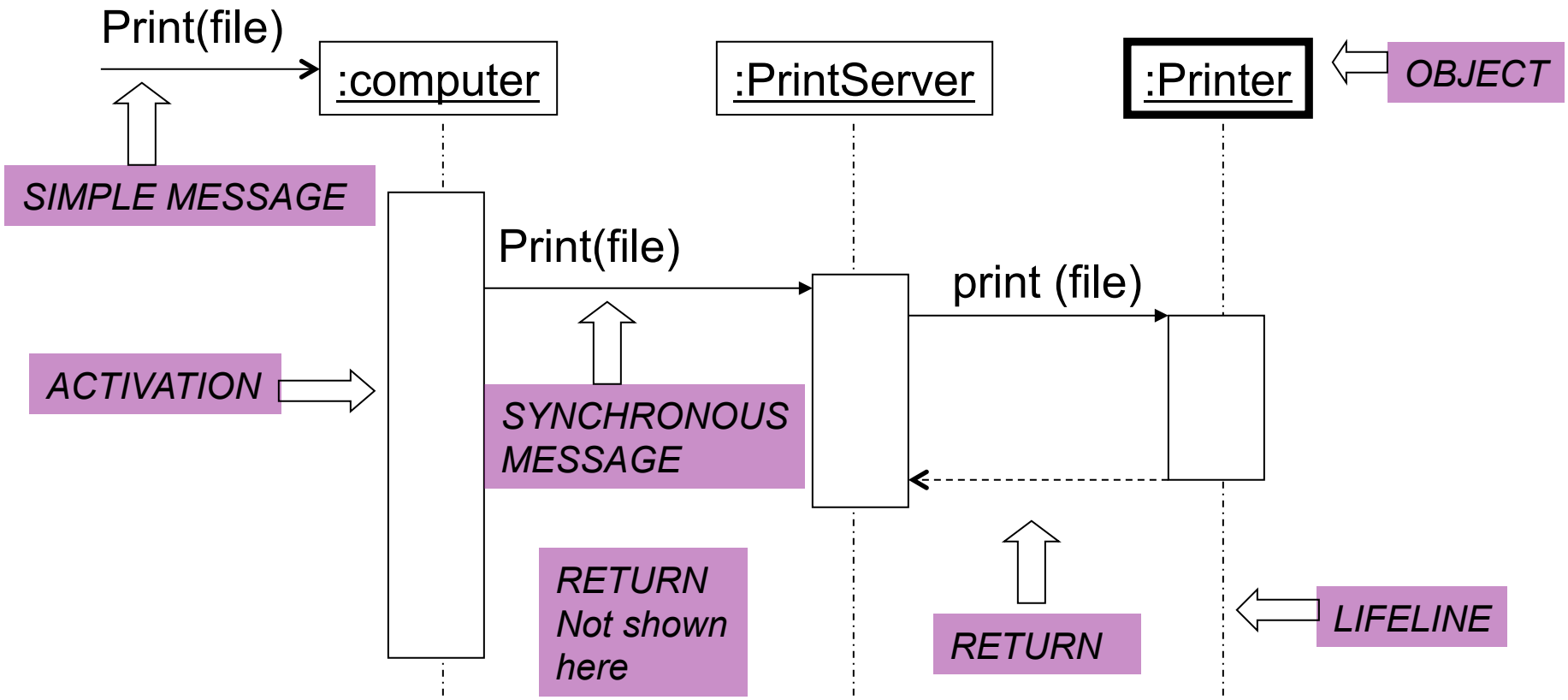
- message indicated by **horizontal arrow** to other object
- write message name and arguments above arrow



- different arrowheads for normal / concurrent (asynchronous) methods
- arrow back indicates return (usually dashed)



Example Sequence Diagram



Self Check

how are these represented in sequence diagrams

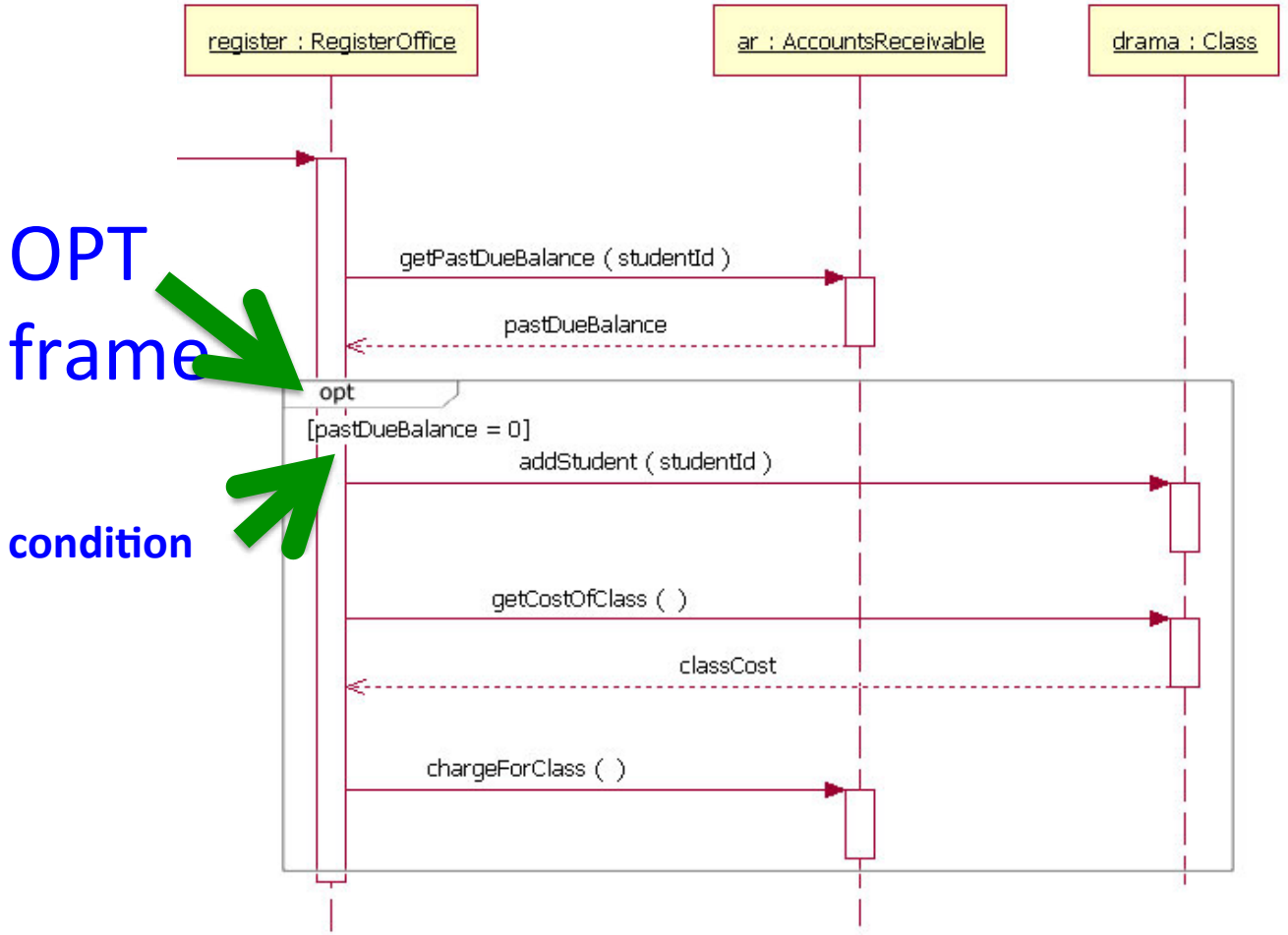
1. object
2. lifeline
3. activation
4. creation of object
5. deletion of object
6. active object
7. message (or method call)
8. nested calls
9. synchronous message
10. asynchronous message
11. return message
12. actors

ADVANCED SEQUENCE DIAGRAMS

Use a Frame or box around part of a sequence diagram to show

- if-then (use OPT frame)
- if-then-else (use ALT frame)
- loop (use LOOP frame)
- method (use REF frame)

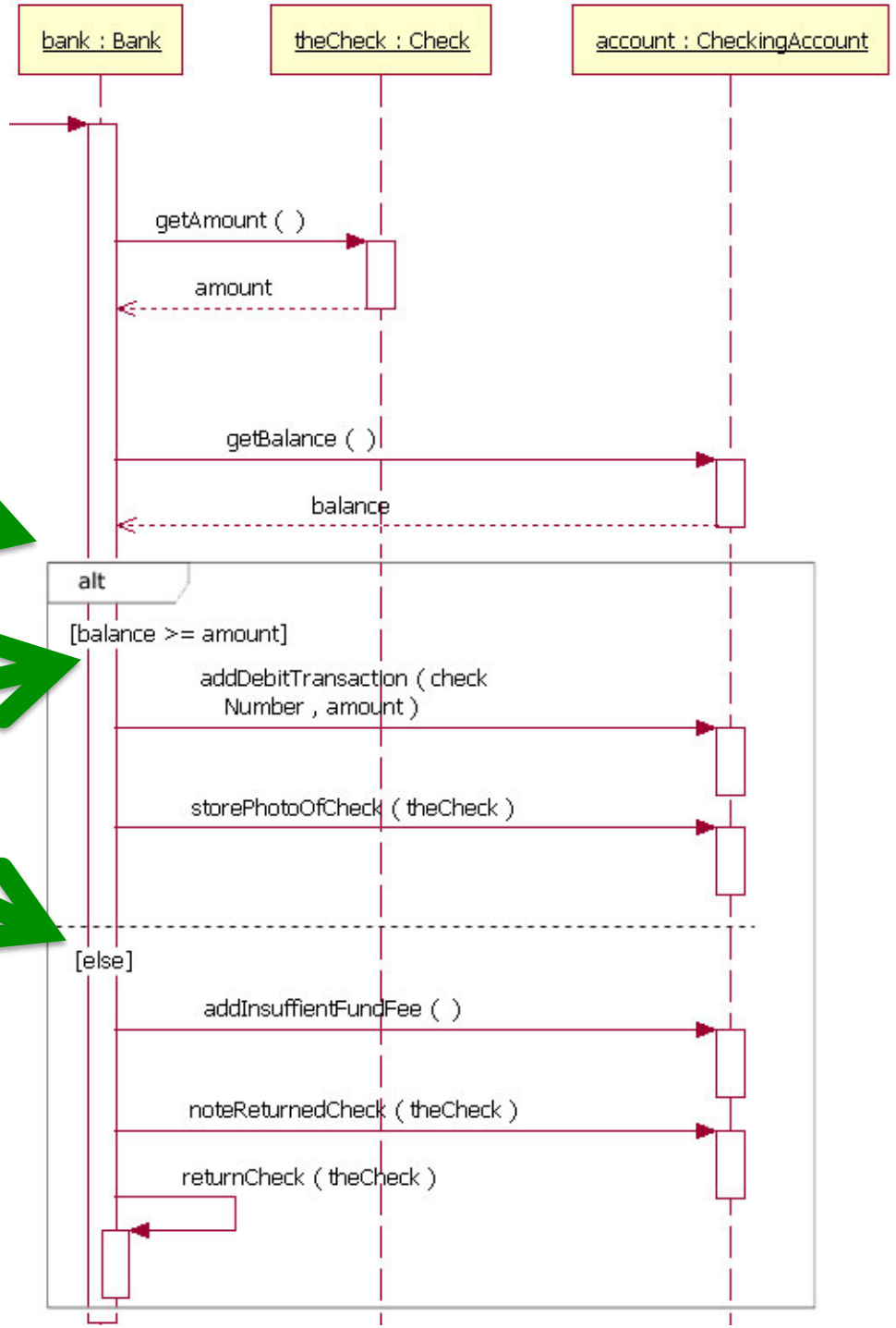
```
ASSUME GIVEN OBJECTS   AccountsReceivable ar, Class drama,  StudentID studentId;

pastDueBalance = ar.getPastDueBalance(studentId);
if (pastDueBalance == 0) {
   drama.addStudent(studentId);
   classCost = drama.getCostOfClass();
   ar.chargeForClass(studentId);
}
```



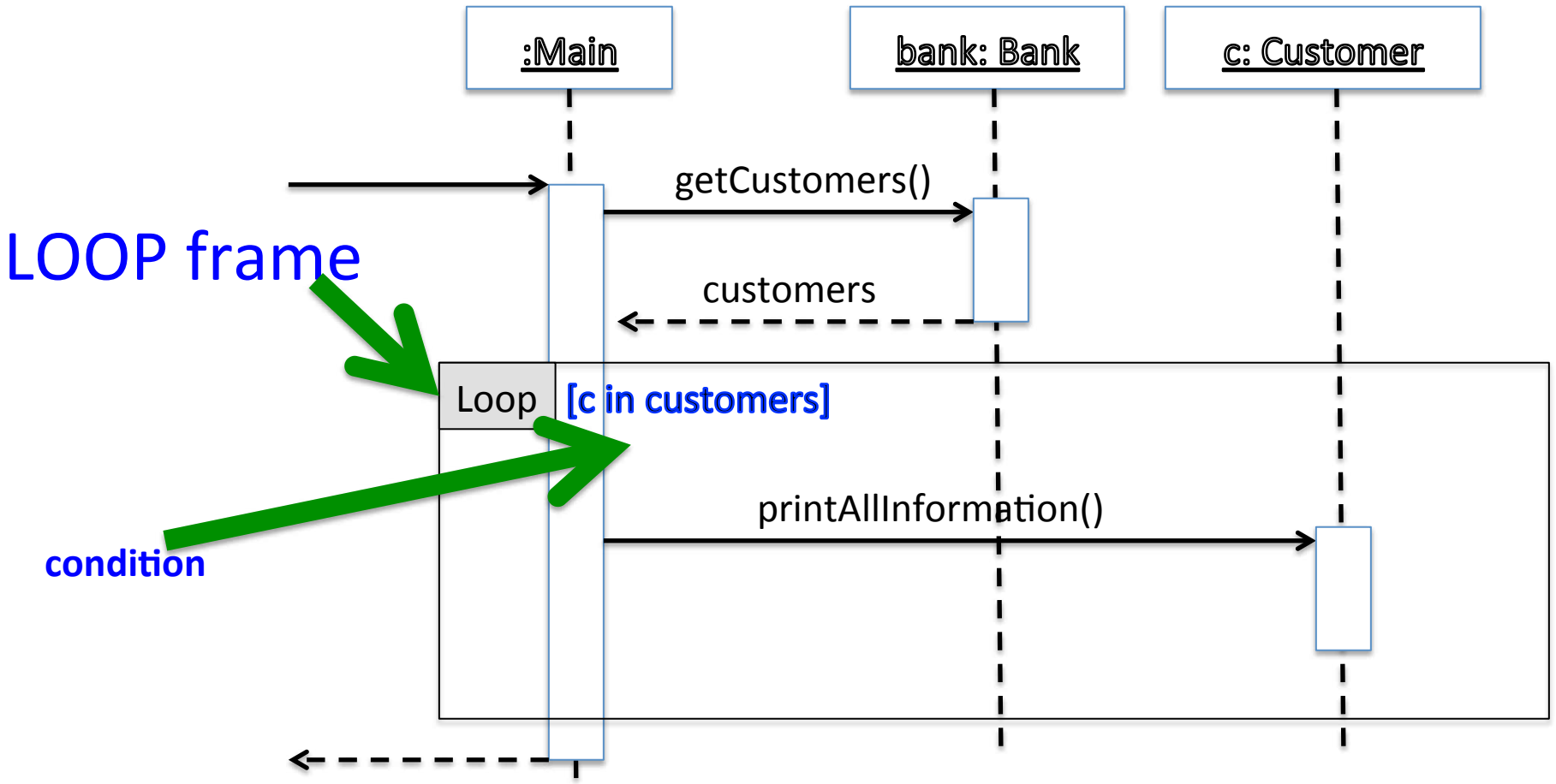
IF THEN ELSE

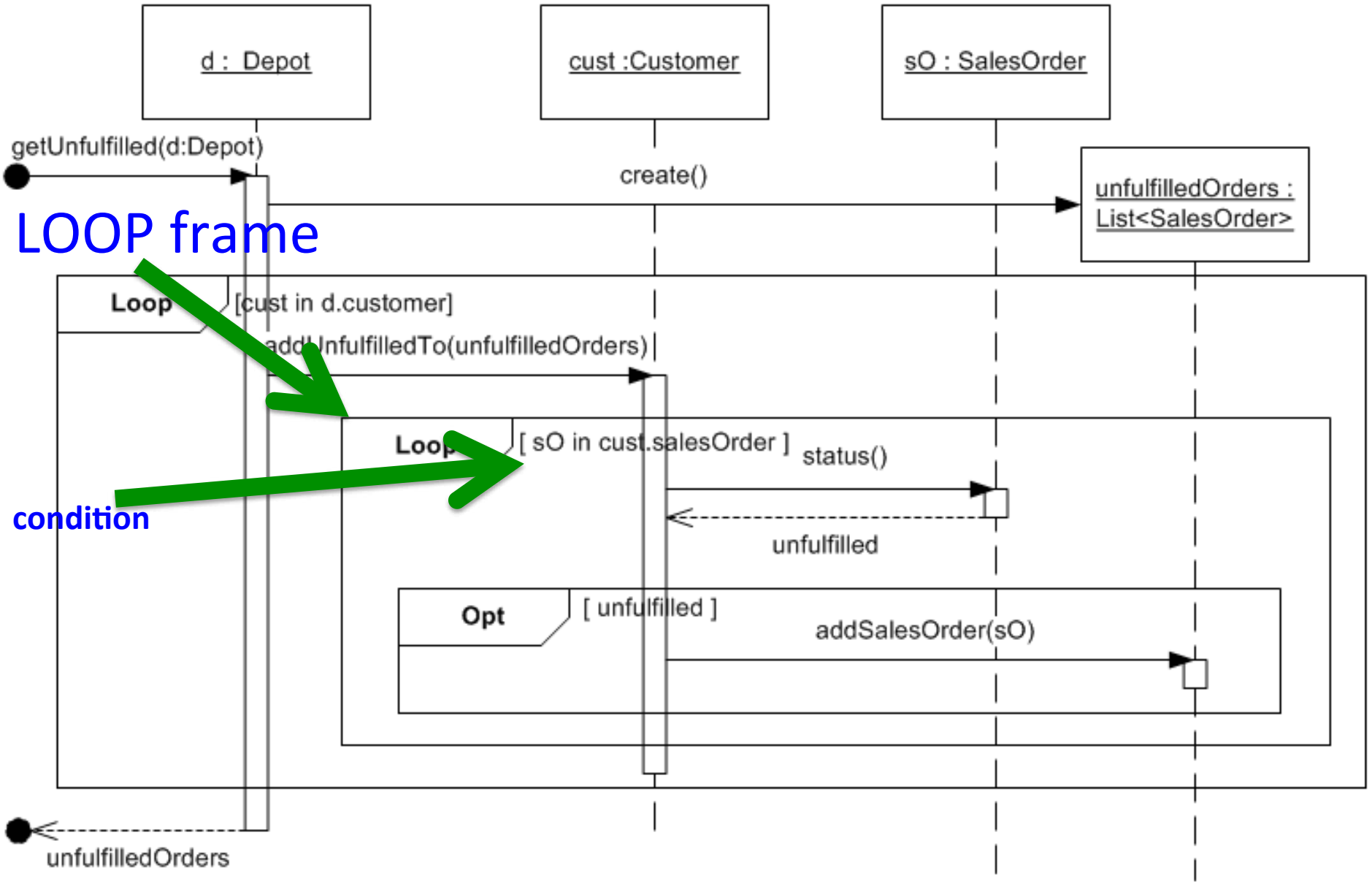
ALT frame

condition



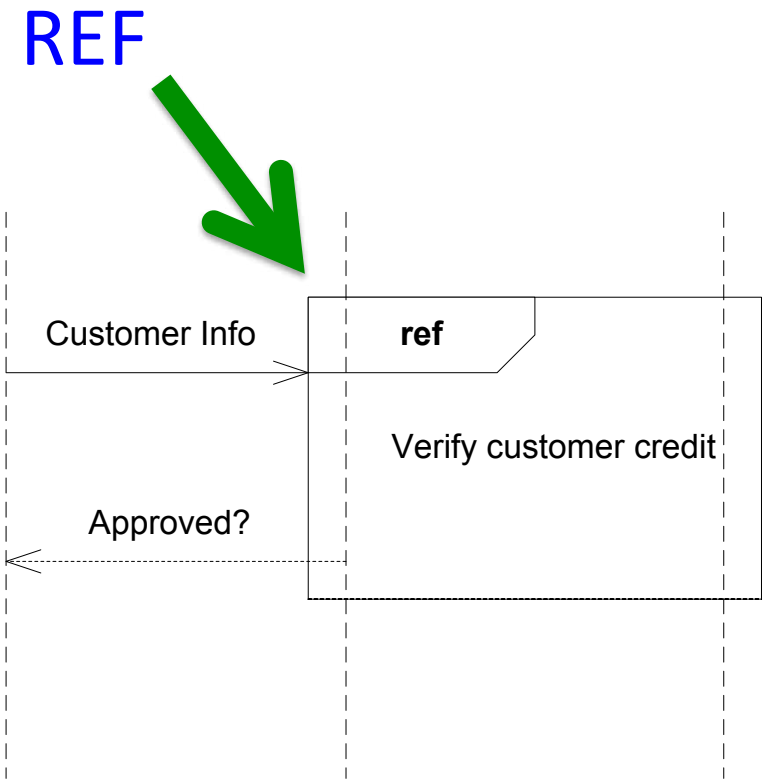
```
//loop over customers in bank

List customers = bank.getCustomers();

for (Customer c :  customers) {
   c.printAllInformation ();
}

return;
```

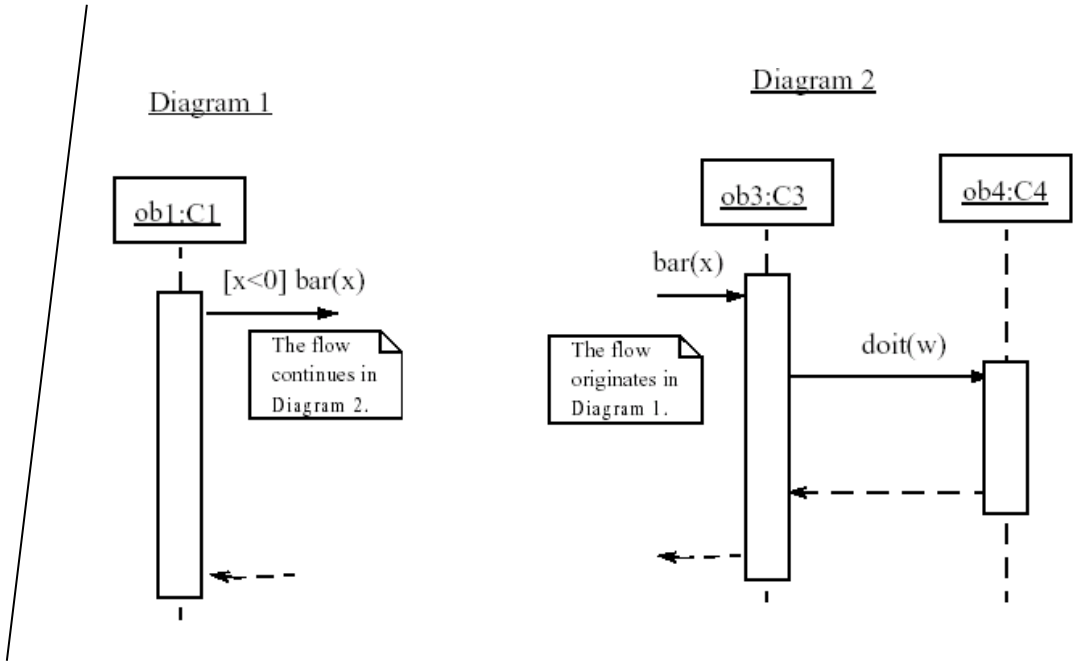



d : Depot
cust :Customer
sO : SalesOrder
getUnfulfilled(d:Depot)
create()
unfulfilledOrders : List<SalesOrder>
LOOP frame
Loop
[cust in d.customer]
addUnfulfilledTo(unfulfilledOrders)
Loop
[sO in cust.salesOrder]
status()
condition
unfulfilled
Opt
[unfulfilled]
addSalesOrder(sO)
unfulfilledOrders

linking sequence diagrams

- if one sequence diagram is too large or refers to another diagram, indicate it with either:
  - an unfinished arrow and comment
  - a "ref" frame that names the other diagram



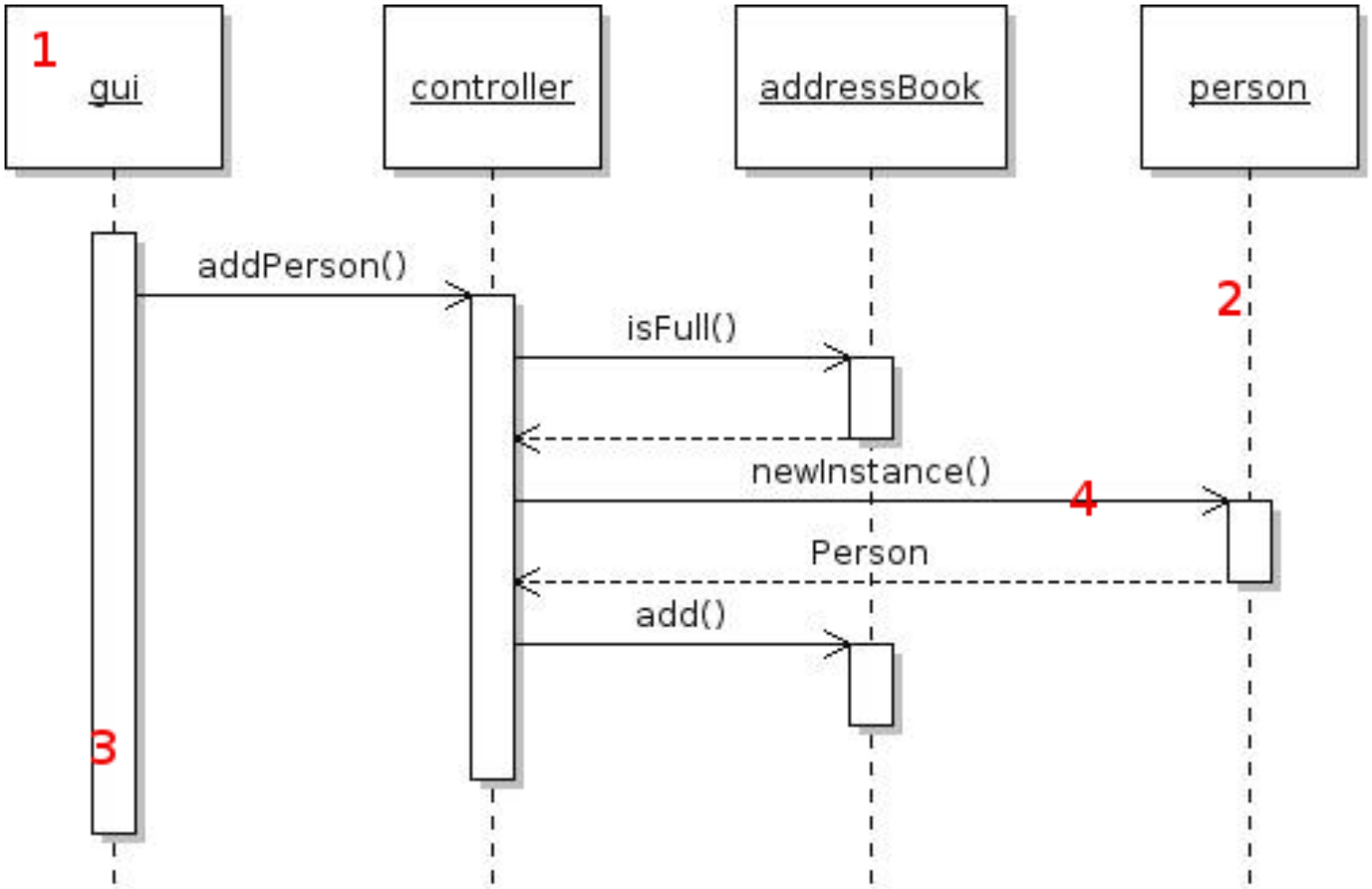


To summarize, we learnt

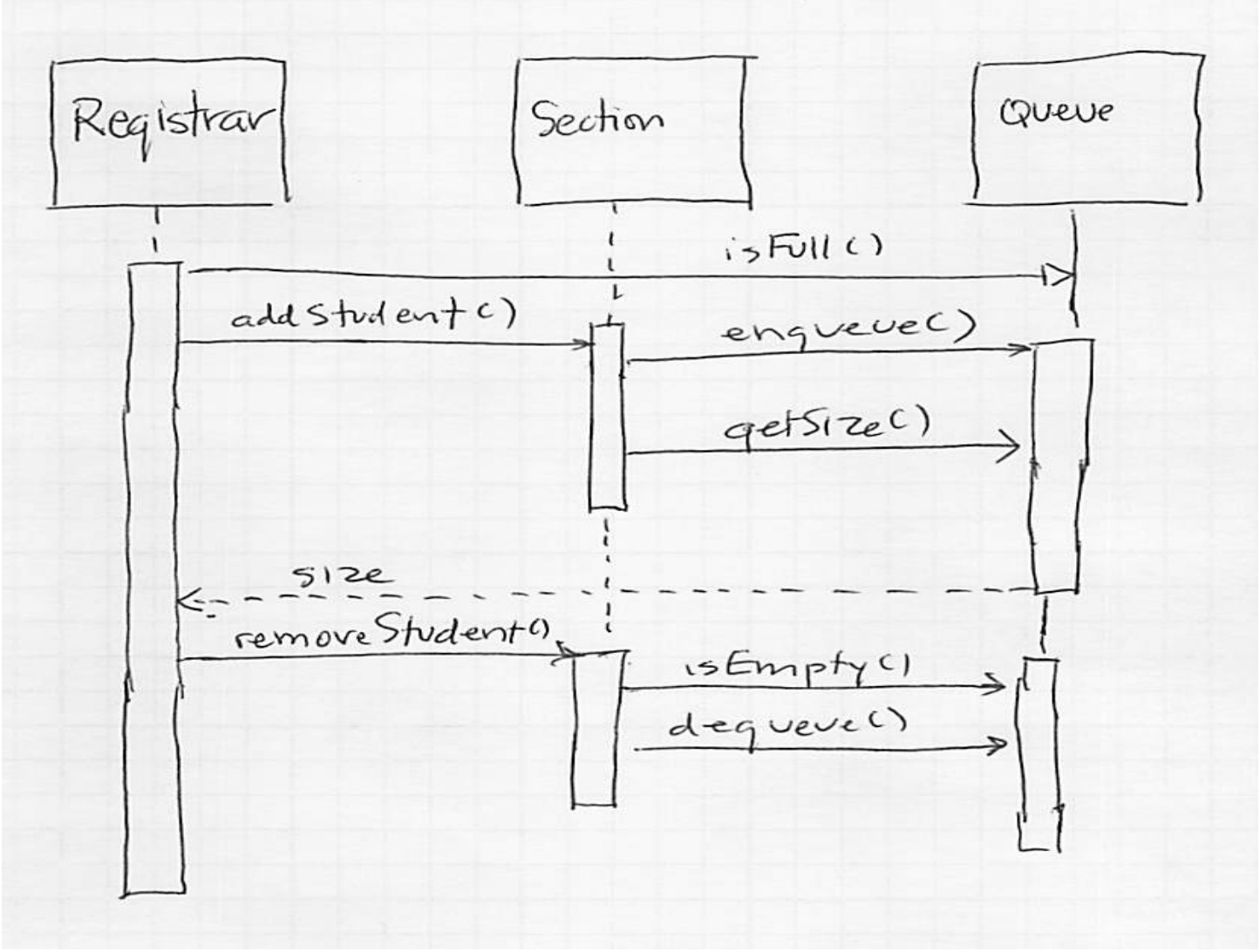
- what are sequence diagrams
- how to read/draw sequence diagrams
  - objects, lifeline, activation, messages
  - opt, alt, loop, ref frames

Self Check

Exercise: name the parts



Exercise: identify errors



Exercise: draw seq diagram

Assume that the getUnfulfilled method is being invoked for a particular Depot object. Draw a sequence diagram showing how the objects interact to accomplish the task. Do show the List as a separate object too.

```
class Depot {
  Customer [] customers;
  List <SalesOrder> getUnfulfilled() {
      List <SalesOrder> unfulfilled = new List <SalesOrder>();
      for (customer in customers) {
          customer.addUnfulfilledOrders(unfulfilled);
      }
      return unfulfilled;
   }
}
class Customer {
  private SalesOrder order;
  void addUnfulfilled(List<SalesOrder> unfulfilled) { boolean fillStatus = order.getStatus();
      if (!fillStatus) { unfulfilled.add(order);}
  }
}
class SalesOrder {   boolean getStatus() { ...}  }
```

Exercise: draw seq diagram

- Draw a sequence diagram that captures ALL the information in the below description. Assume Teller is "automated".

- 1) A **customer** deposits a cheque to the teller.
- 2) The teller asks the customer for her account number.
- 3) After getting that item of information, the teller queries the customer information on his computer.
- 4) The computer searches the database and returns the information to the teller.
- 5) While the computer is searching, the teller asks the customer for her ID.
- 6) The teller checks the information returned by the computer for “holds” on the account.
- 7) If there are no “holds” on the account, the teller checks the information on the ID with that returned by his computer and the signature on the cheque with the one shown on her computer and then enters the deposit amount on the computer.
- 8) While the computer is updating the database, the teller writes out a receipt and gives it to the customer