Object Oriented Analysis

- OOA consists of three steps
  - Use case modelling
  - Class modelling
  - Dynamic modelling

Class relationships

- **3 relationships**
  - **Use or awareness (associations)**
  - **Aggregation (or containment)**
  - **Inheritance (or specialisation)**

- Use - most general relationships
- Class A uses Class B if :
  - **an operation of Class A receives or returns an object of Class B**
  - **In the process of an operation of A, an object of Class B must be inspected or created**
  - **Objects of A contain objects of Class B or reference to objects of Class B**

- **Aggregation:**
  - **Objects of Class A contains objects of Class B or reference to objects of Class B, then A uses B for aggregation**
  - **has-a (/part-of) relationship**
  - **cardinality, eg 1:1 or 1:n relationship**

- **Inheritance:**
  - **is-a relationship**
  - **specialisation**

- OOA consists of three steps
  - Use case modelling
  - Class modelling
  - Dynamic modelling

Class Diagram so Far

- **The class diagram so far only indicates associations between classes, their names, direction and cardinality (multiplicity)**
- **We have an initial set of operations derived for classes but we are not yet certain whether these operations are sufficient (they almost certainly are not!)**

- **It is still undefined which operation traverses along which association and it is equally undefined which operation has to use other operations in order to implement the behaviour associated with it**
- **This is why there are a lot of question marks attached to associations**

Object Oriented Analysis Model

- **During the analysis stage we use two important representations to show dynamic behavior:**
  - **‘object interaction diagrams’ (or simply ‘interaction diagrams’) known in UML as:**
    - **`sequence diagrams' showing the interaction of a set of objects in temporal order**
    - **Collaboration diagrams (not dynamic - similar to CRD)**
  - **`state transition diagram', describes the temporal evolution of an object of a given class in response to other objects**

Sequence Diagram

- **Interacrion diagrams provide the essential first step in clarifying the messages passed between all the objects involved.**
- **At this point the use cases take on a central role, by providing views of exactly how parts of the system should interact.**

Sequence Diagram

- **A sequence diagram is constructed for each usecase.**
- **Then, the class diagram is checked against them in order to make sure that:**
  - **all the operations required for the implementation of the use cases have been identified in the class diagram**
  - **the objects can identify each other based on the associations which appear in the class diagram**
- **Missing operations and associations are introduced to the class diagram**

Sequence Diagram

- **A sequence diagram shows a set of objects and the temporal order in which operations are invoked between them**
- **The representation of such interactions is essential because the isolated behaviours of individual objects do not give a complete view of a complex system.**

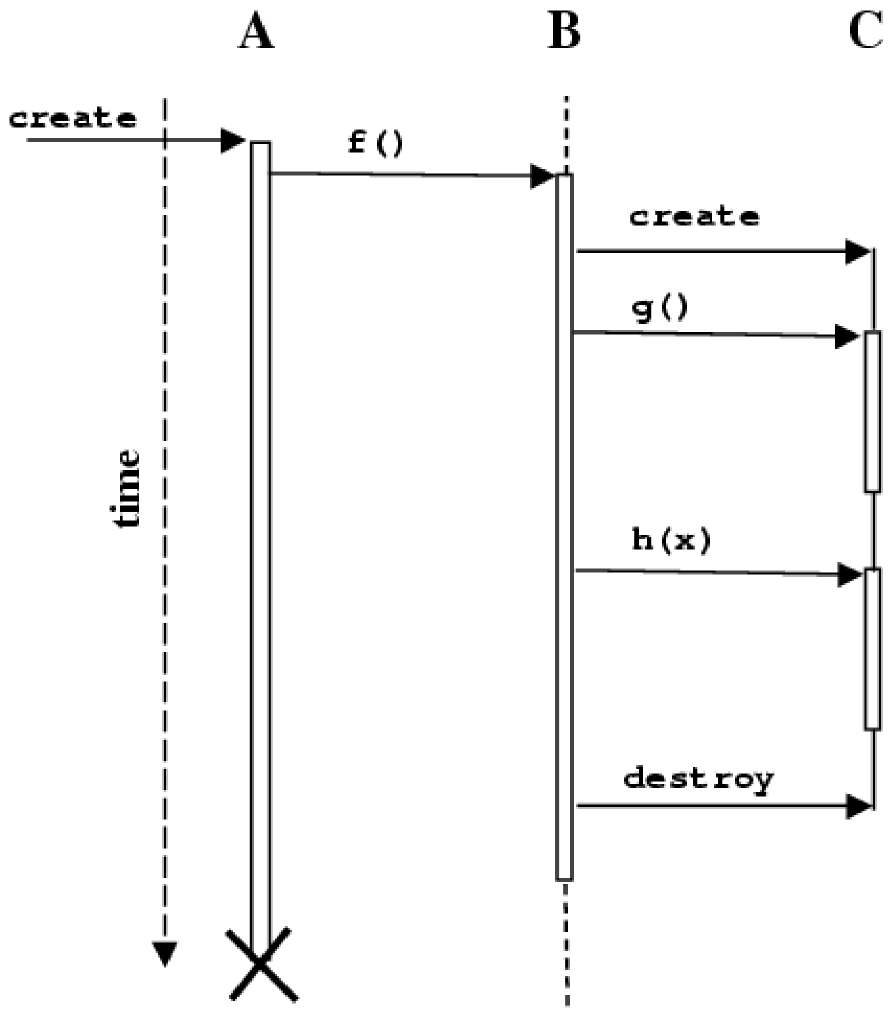
- **The next slide outlines the notation provided in UML for sequence diagrams**

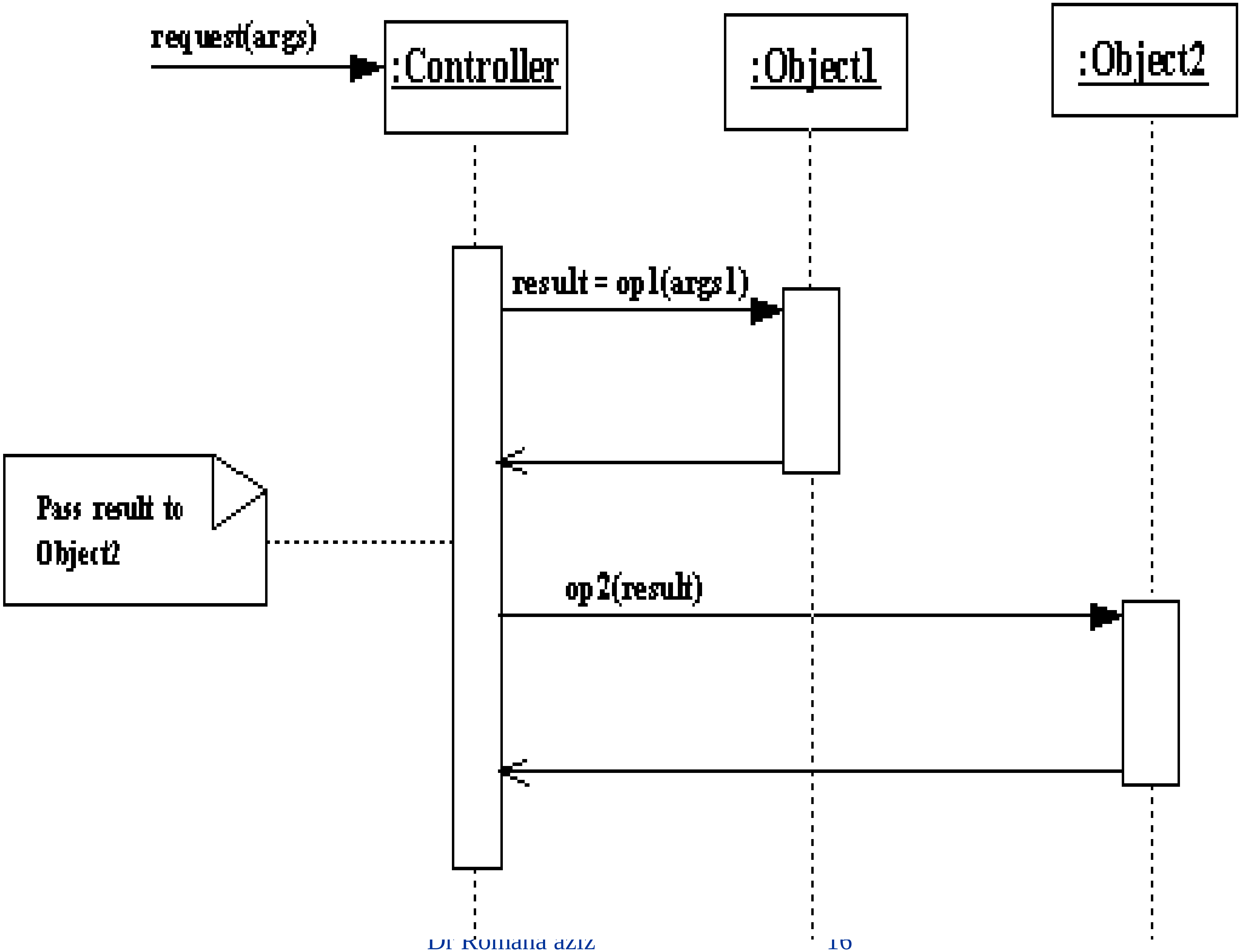
SEQUENCE DIAGRAM IN UML: The notation

- Shows interactions among a set of objects in temporal order

Notational conventions:

- Objects: the existence of an object is shown as a vertical line (called 'lifeline')
- Messages: marked by horizontal lines labelled with the name of the message and its argument values
- Object destruction: shown by a big X




request(args)
:Controller
:Object1
:Object2
result = op1(args1)
Pass result to Object2
op2(result)

- **Sequence diagrams do not show actual execution times, rather they show that a computation has begun but not yet completed**
- **In other words, the vertical axis is not a linear measure of time**
- **In the previous example the heights of the vertical boxes denoting execution activity are not significant**.

- **These boxes only have meaning to show the relative occurrence of an object's execution in the sequence**
- **Thus, it is not correct to say that the execution times of Object1 and Object2 are approximately the same because the vertical boxes denoting their executions have approximately the same height.**

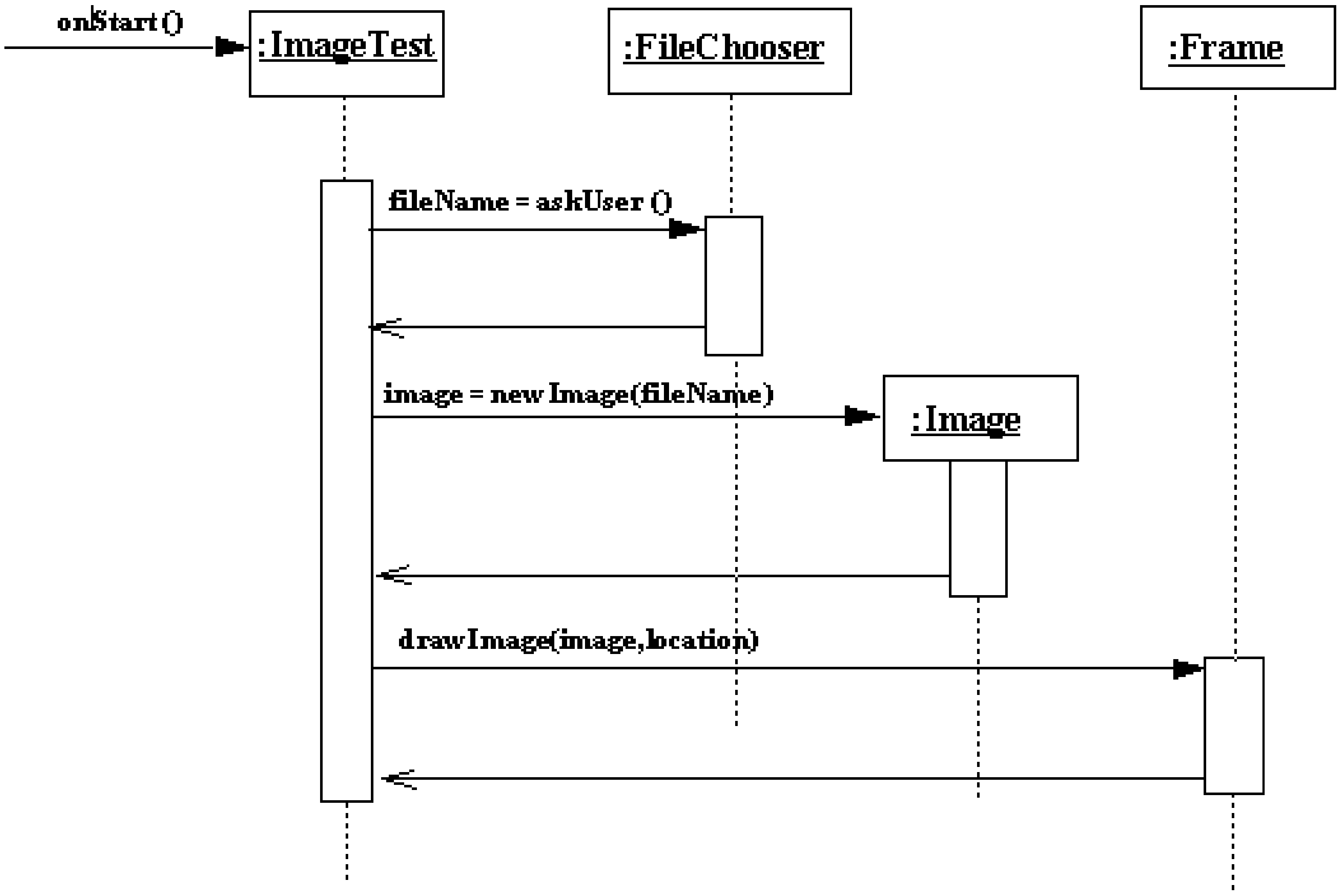
- **Moreover in the previous example after invoking the op1 method of Object1, the Controller object is still depicted as active even though execution control will pass to Object1. The Controller object is still active after this invocation because the computation of its request method has begun but has not yet completed.**

A Sequence of Interactions with GUI Objects

- **The sequence of operations necessary to display an image (e.g., a ".GIF" image) are used to illustrate a simple system based on sequenced interaction. The steps to display the image are:**
  - **ask the user for the name of the file containing the image**
  - **create the image using the contents of the named file**
  - **draw the image in a Frame**

A Sequence of Interactions with GUI Objects

- **The structure of a simple system to perform this sequence of steps is shown in next slide as a UML sequence diagram.**
- **It is possible, of course, to add other steps (such as verifying that the named file exists and does, in fact, contain an image)**
- **While the exact steps may change, the central idea is that the system is built around a sequence of actions taken by independent objects**


onStart ()
:ImageTest
:FileChooser
:Frame
fileName = askUser ()
image = new Image(fileName)
:Image
drawImage(image,location)

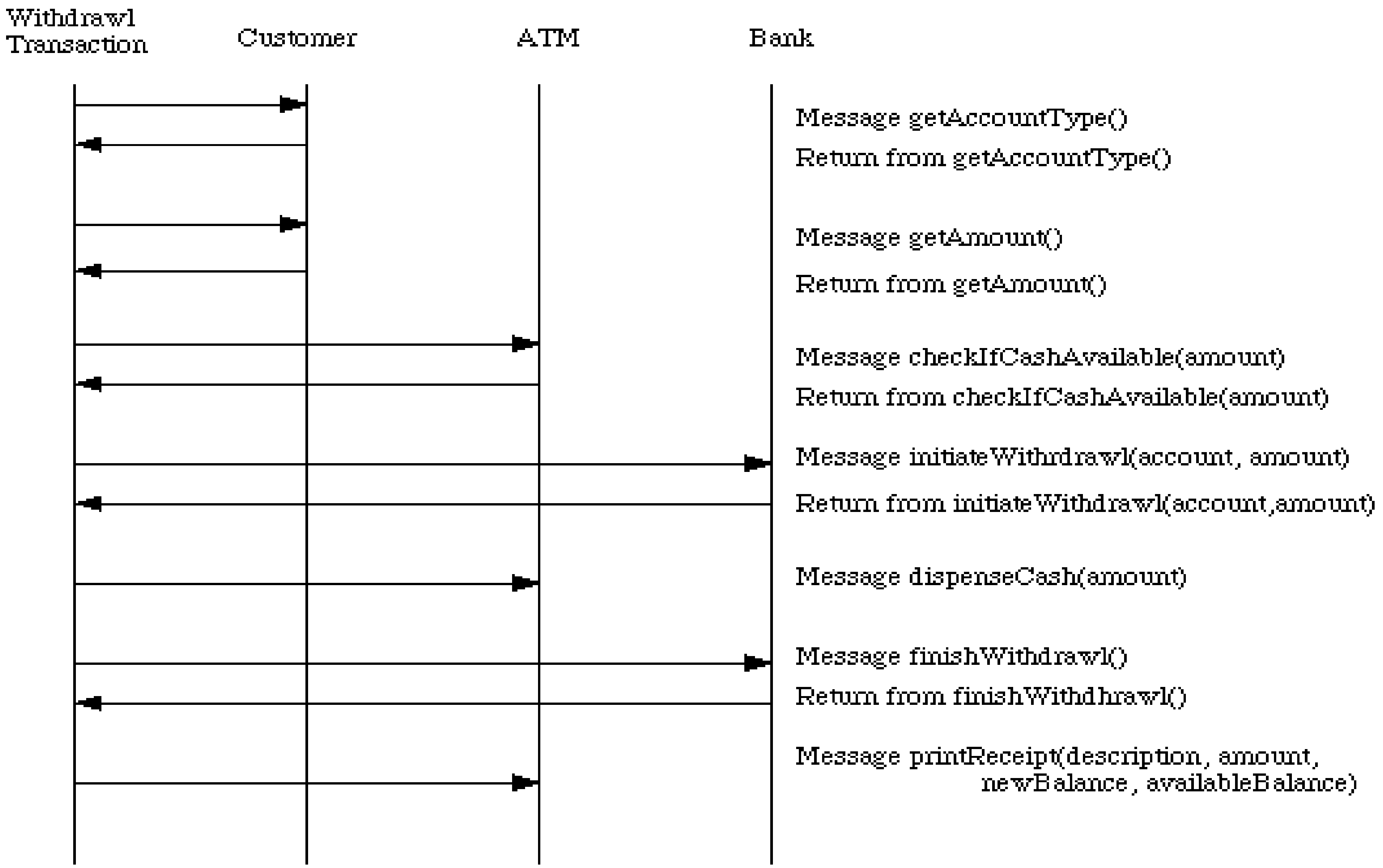
ATM Interaction Diagram

- An interaction diagram is an analysis tool that can be used when an object interacts with various other objects to perform some task.
- One place this occurs in the ATM is during the execution of a transaction, when the transaction must interact with the Customer (via the ATM) to get specific information (such as choice of account(s), amount), with the Bank, and with various components of the ATM (e.g. cash dispenser, printer).
- The diagram on following slide is an interaction diagram for a WithdrawlTransaction.

Cash Withdrawl Transaction Use Case

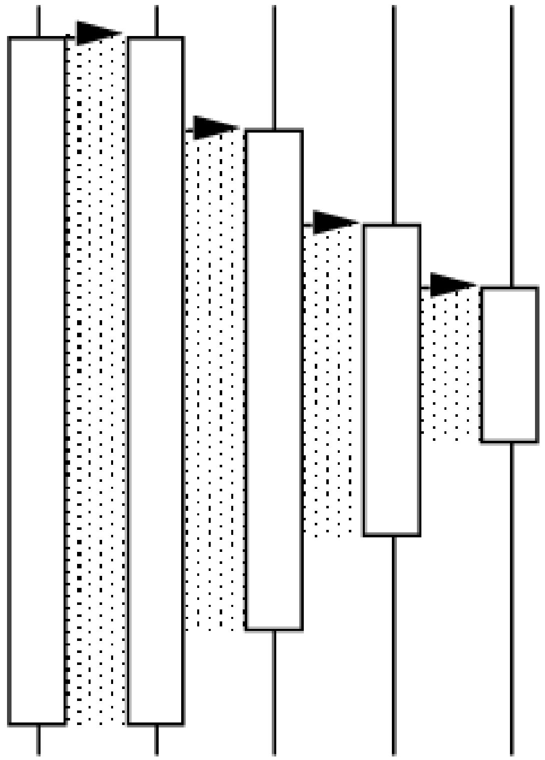
- **A cash withdrawl transaction is started from within a session when the customer chooses cash withdrawl from the menu of possible transaction types. The customer chooses a type of account to withdraw from (e.g. checking) from a menu of possible accounts, and then chooses a dollar amount from a menu of possible amounts. The system verifies that it has sufficient money on hand to satisfy the request. If not, it reports a failure to the session, which initiates the Failed Transaction to report the problem. If there is sufficient cash, it sends the customer's card number, PIN, chosen account and amount to the bank, which either approves or disapproves the transaction. If the transaction is approved, the machine dispenses the correct amount of cash and issues a receipt. If the transaction is disapproved due to an incorrect PIN, the Incorrect PIN extension is executed. All other disapprovals are reported to the session, which initiates the Failed Transaction. The bank is notified whether or not an approved transaction was completed in its entirety by the machine; if it is completed then the bank completes debiting the customer's account for the amount.**

Interaction Diagram for a Withdrawl Transaction


Withdrawl Transaction
Customer
ATM
Bank
Message getAccountType()
Return from getAccountType()
Message getAmount()
Return from getAmount()
Message checkIfCashAvailable(amount)
Return from checkIfCashAvailable(amount)
Message initiateWithrdrawl(account, amount)
Return from initiateWithdrawl(account,amount)
Message dispenseCash(amount)
Message finishWithdrawl()
Return from finishWithdhrawl()
Message printReceipt(description, amount, newBalance, availableBalance)

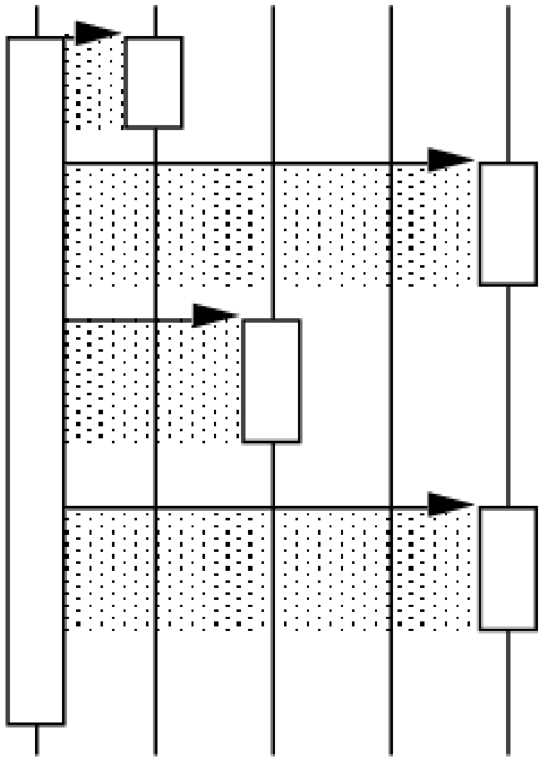
PATTERNS OF SEQUENCE DIAGRAM STRUCTURES

"Stair" decentralised



for well-structured sequence of operations

"Fork " centralised

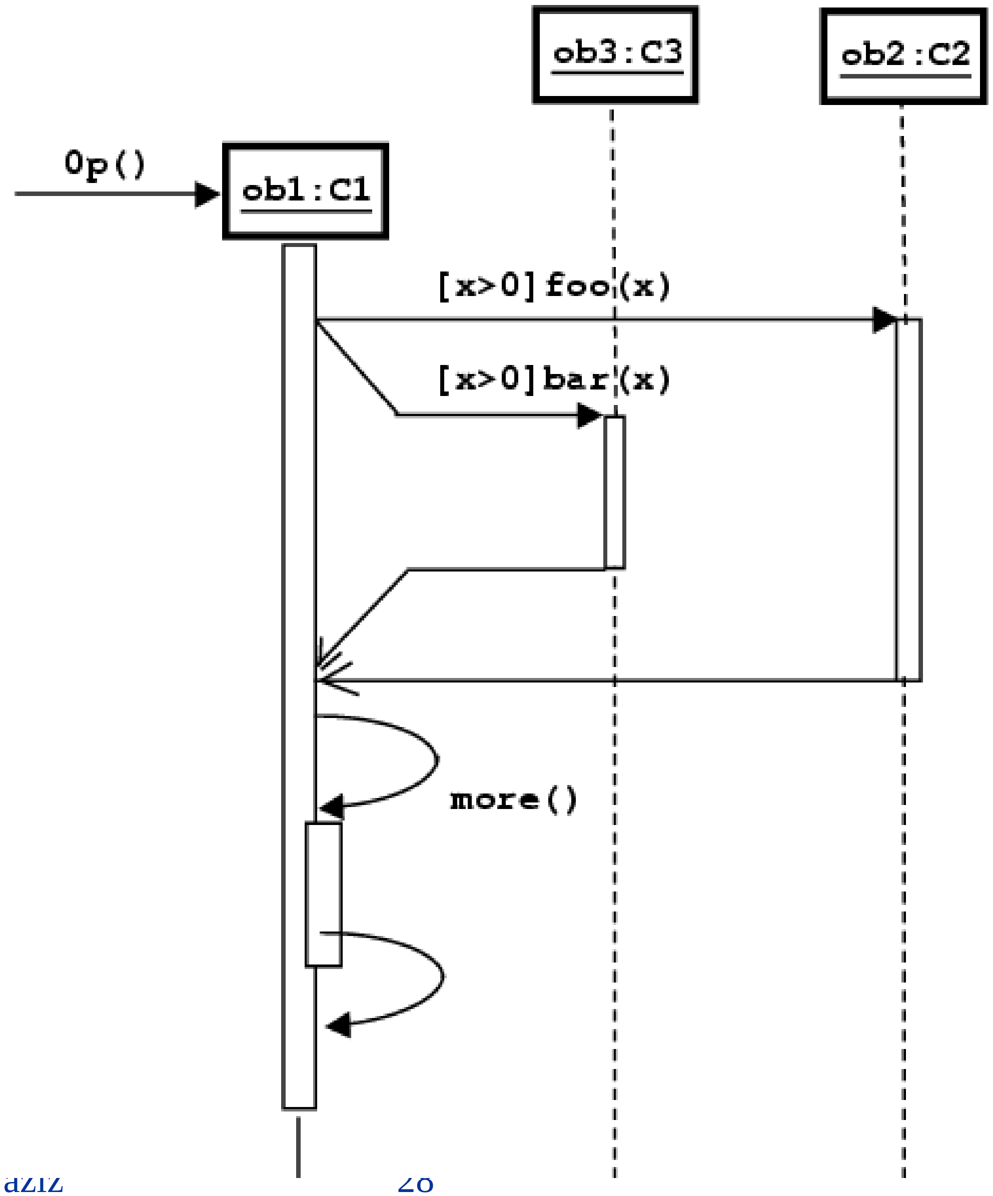


for variable sequence of operations

- Decentralised control is appropriate if operations have a strong connection with hierarchical or fixed temporal relationships (as in `Returning item').
- Centralised pattern is more suitable if operations can change order; new operations can be inserted.
- Next, we review how control flow directives can be included in sequence diagrams...

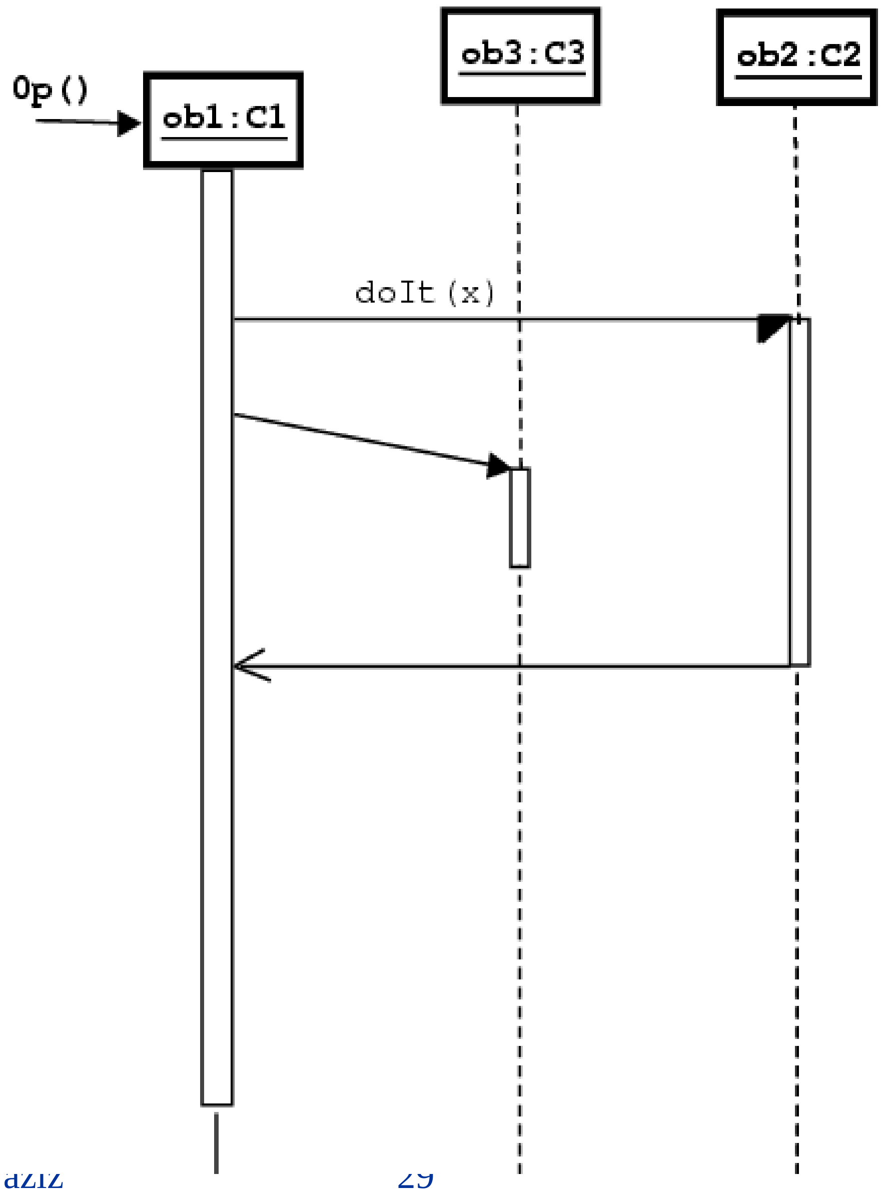
SEQUENCE DIAGRAM IN UML: More notational conventions

- **Conditional branching:** multiple arrows leaving a single point, labelled by a guard condition in brackets and return arrows
- **Recursion:** message to itself and new activation box drawn slightly to the right of the first one



SEQUENCE DIAGRAM IN UML: More notational conventions (cont'd)

- **Asynchronous messages:** drawn with a half arrow head
- **Return messages:** drawn with a non solid arrow
- **'Long' messages:** drawn with arrows slanted downward



0p()
ob1:C1
ob3:C3
ob2:C2
doIt(x)