# XIII. Sequence and Collaboration Diagrams

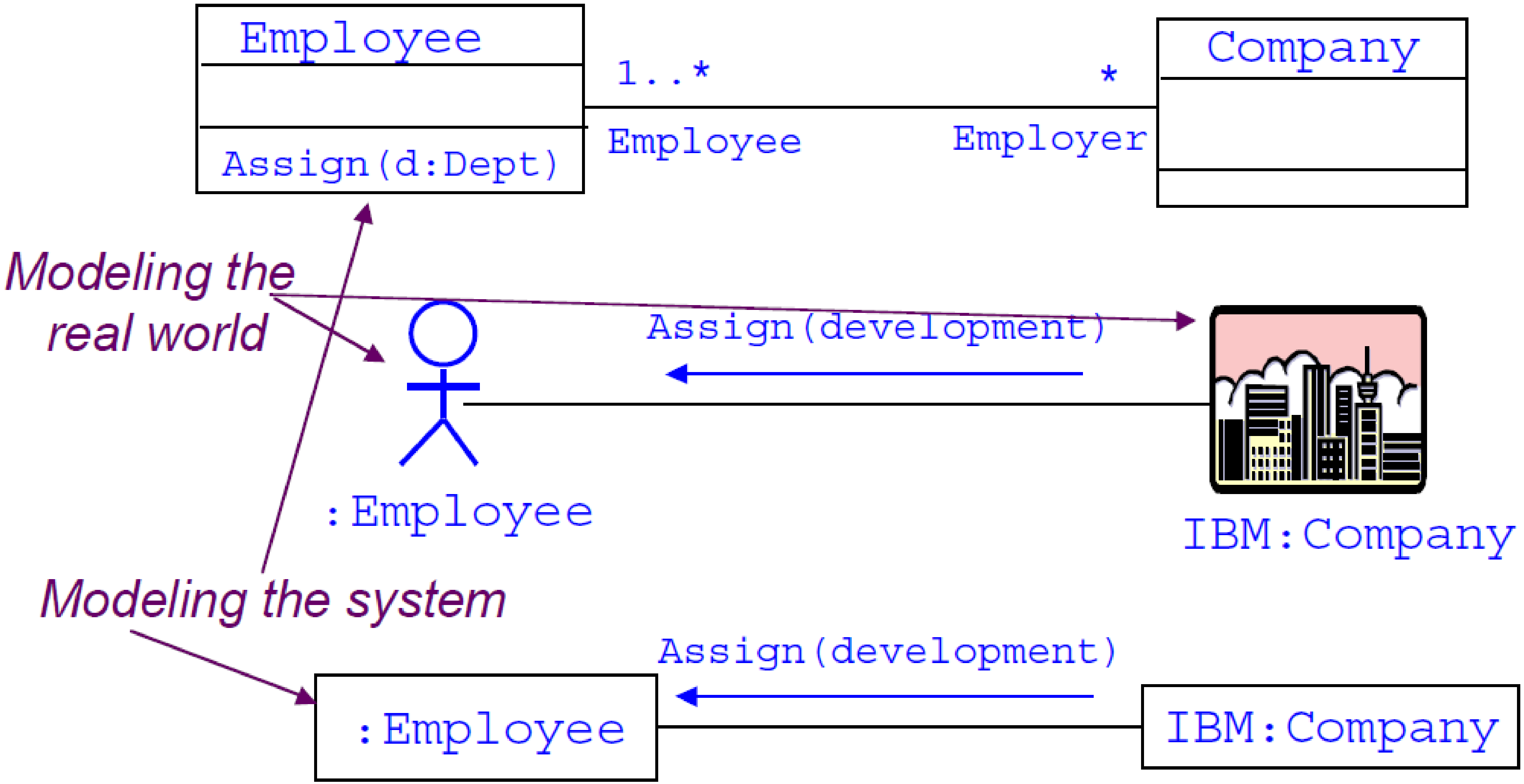
***Interaction Diagrams***
***Sequence Diagrams***
***Examples***
***Collaboration Diagrams***

***Presentation: N.C. Danh***

# *Interaction Diagrams*

- *Interactions among actors (people/objects) are modeled by* ***interaction diagrams.***
- *An* ***interaction*** *involve the exchange of messages between two or more actors.*
- *There are two types of interaction diagrams:*
  - ✓*Sequence diagrams;*
  - ✓*Collaboration diagrams.*

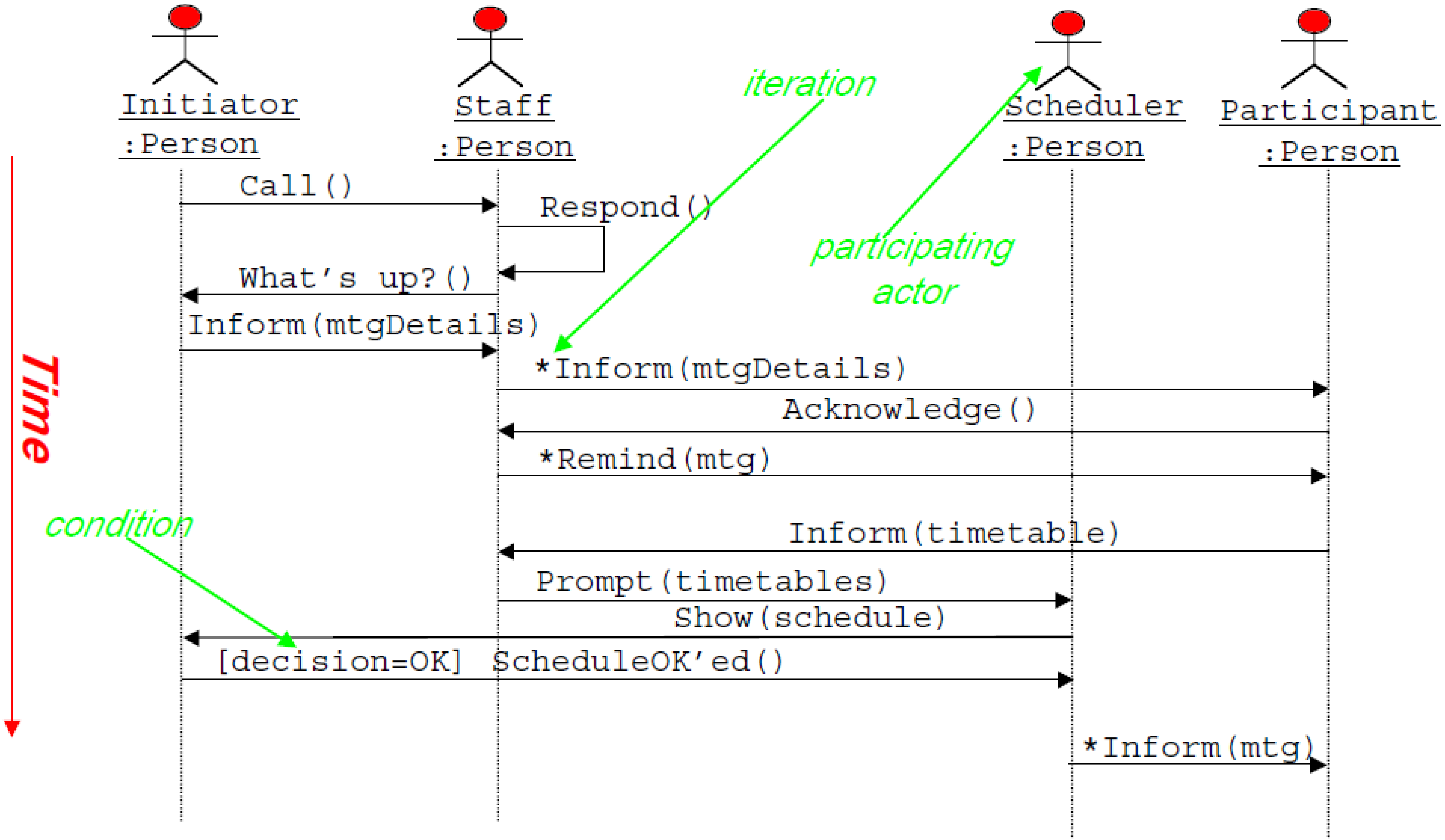
# The Nature of an Interaction



# *Sequence Diagrams*

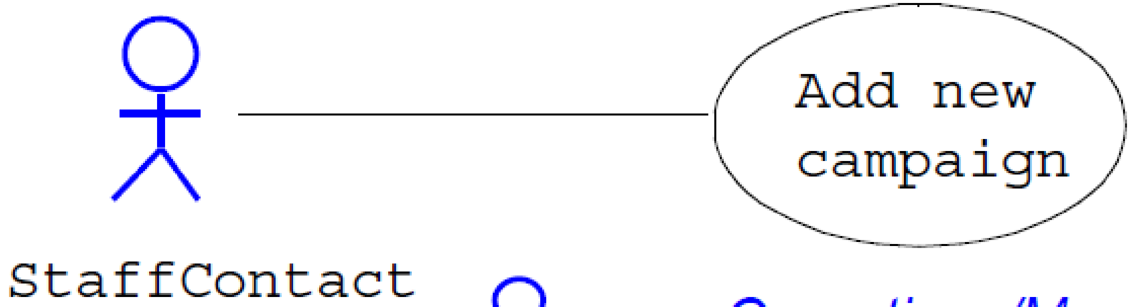
- *Sequence diagrams describe in detail how actors use use cases; they can also model external business processes.*
- *Interactions consist of one or more* ***messages.*** *Interactions may be synchronous, or asynchronous.*
- *Sequence diagrams defined during requirements analysis should* ***not****:*
  - ✓ *include design objects;*
  - ✓ *specify message signatures in any detail.*

# The Basic Idea



# Example: Add a New Campaign

- *Getting back to the use case “Add a new campaign”*



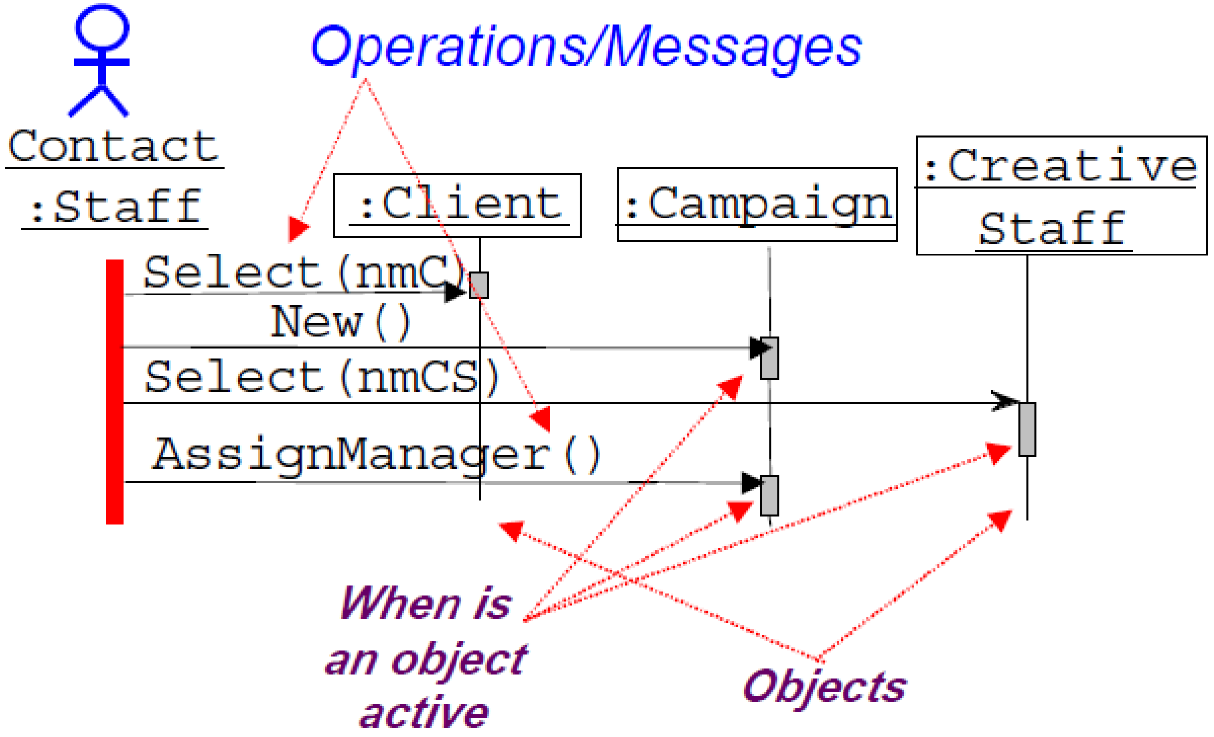
***Add new campaign***

✓ *Find client by name*

✓ *Create new campaign*

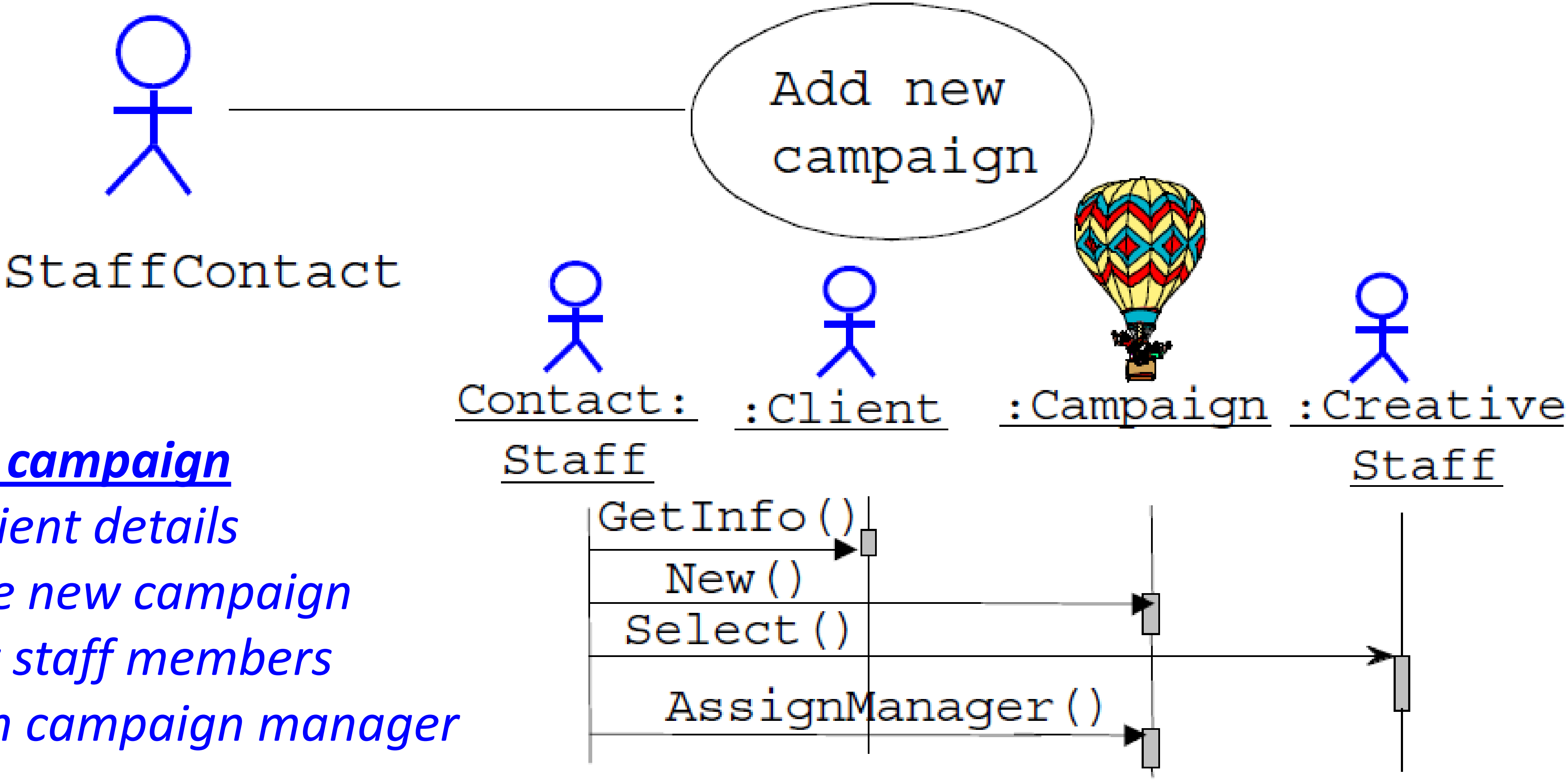
✓ *Find creative staff member by name*

✓ *Assign campaign manager*



# Add New Campaign

- *This describes a business process, no system involved.*



***Add new campaign***

✓ *Get client details*

✓ *Create new campaign*

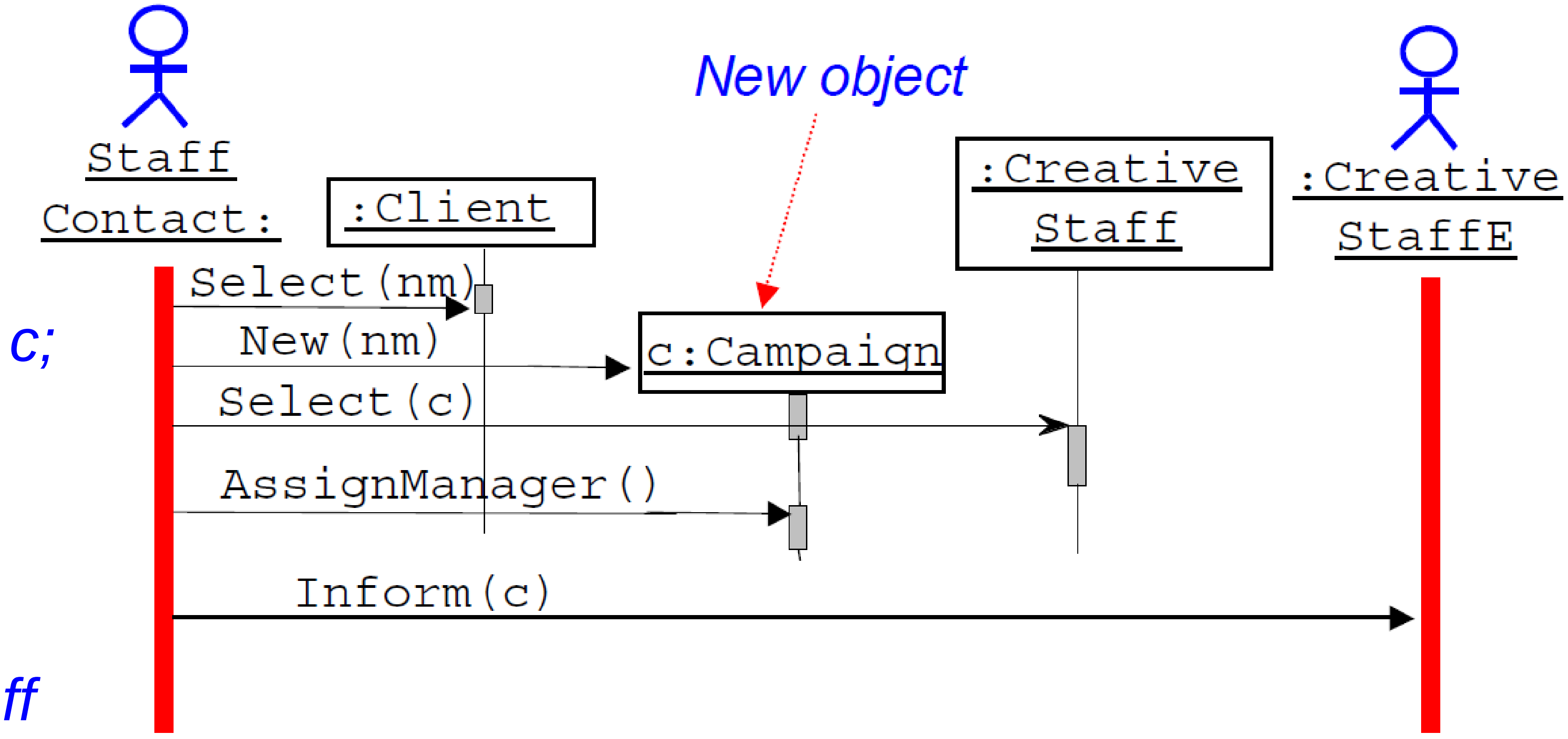
✓ *Select staff members*

✓ *Assign campaign manager*

# A More Realistic Example

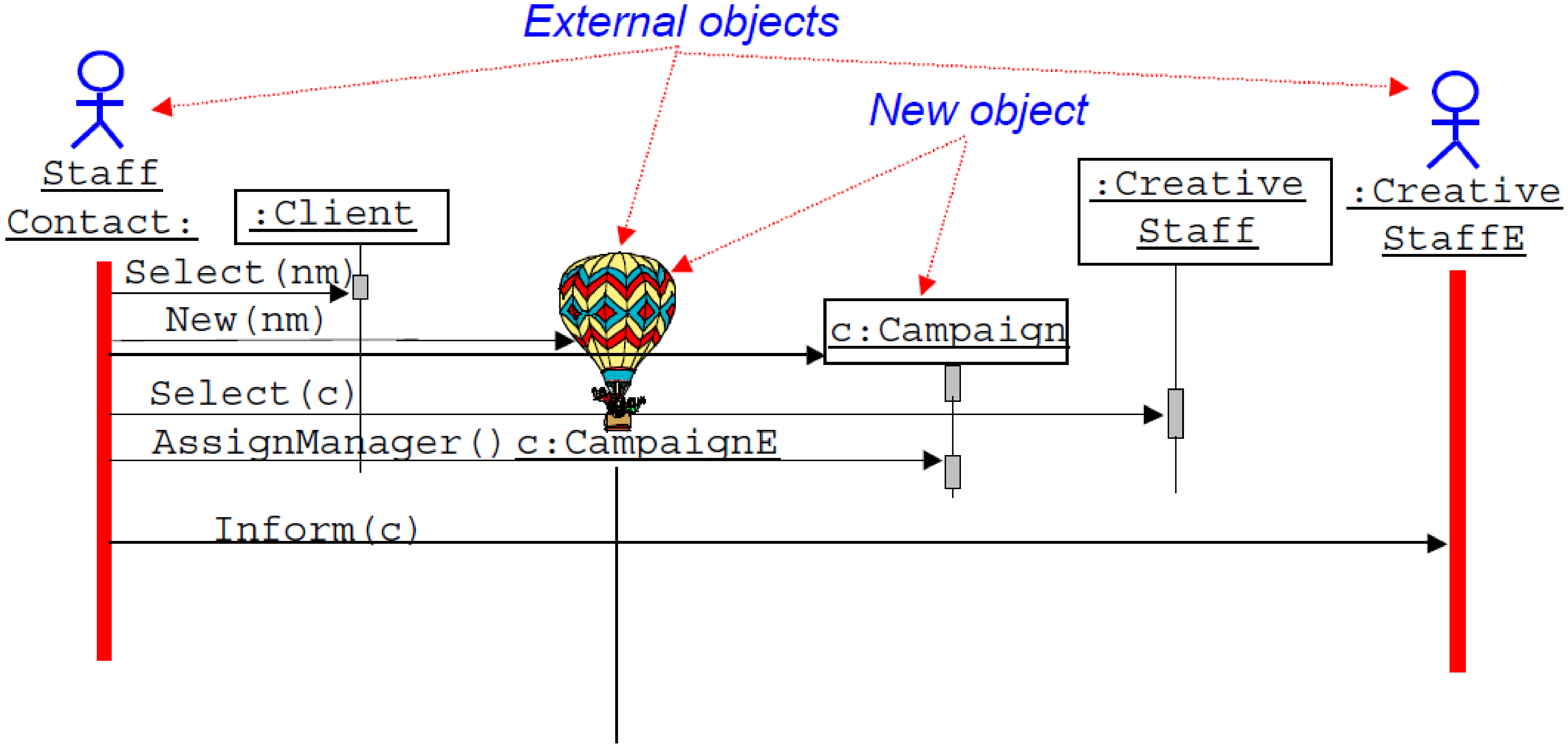
**Add new campaign**

- ✓ *Find client by name;*
- ✓ *Create new campaign c;*
- ✓ *Assign creative staff member to c;*
- ✓ *Assign campaign manager;*
- ✓ *Inform the creative staff person.*

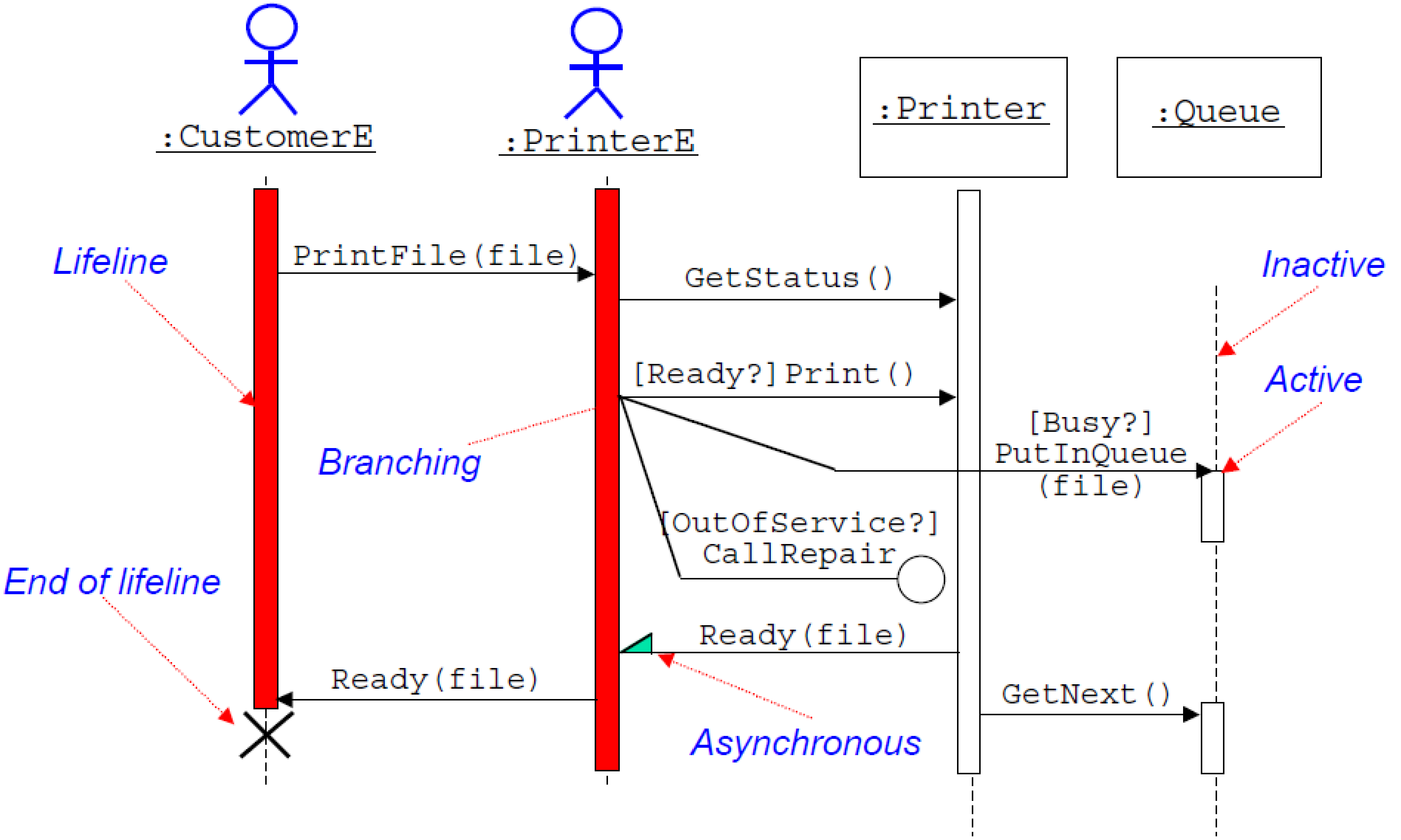


- *This describes a business process involving two people and three system objects.*

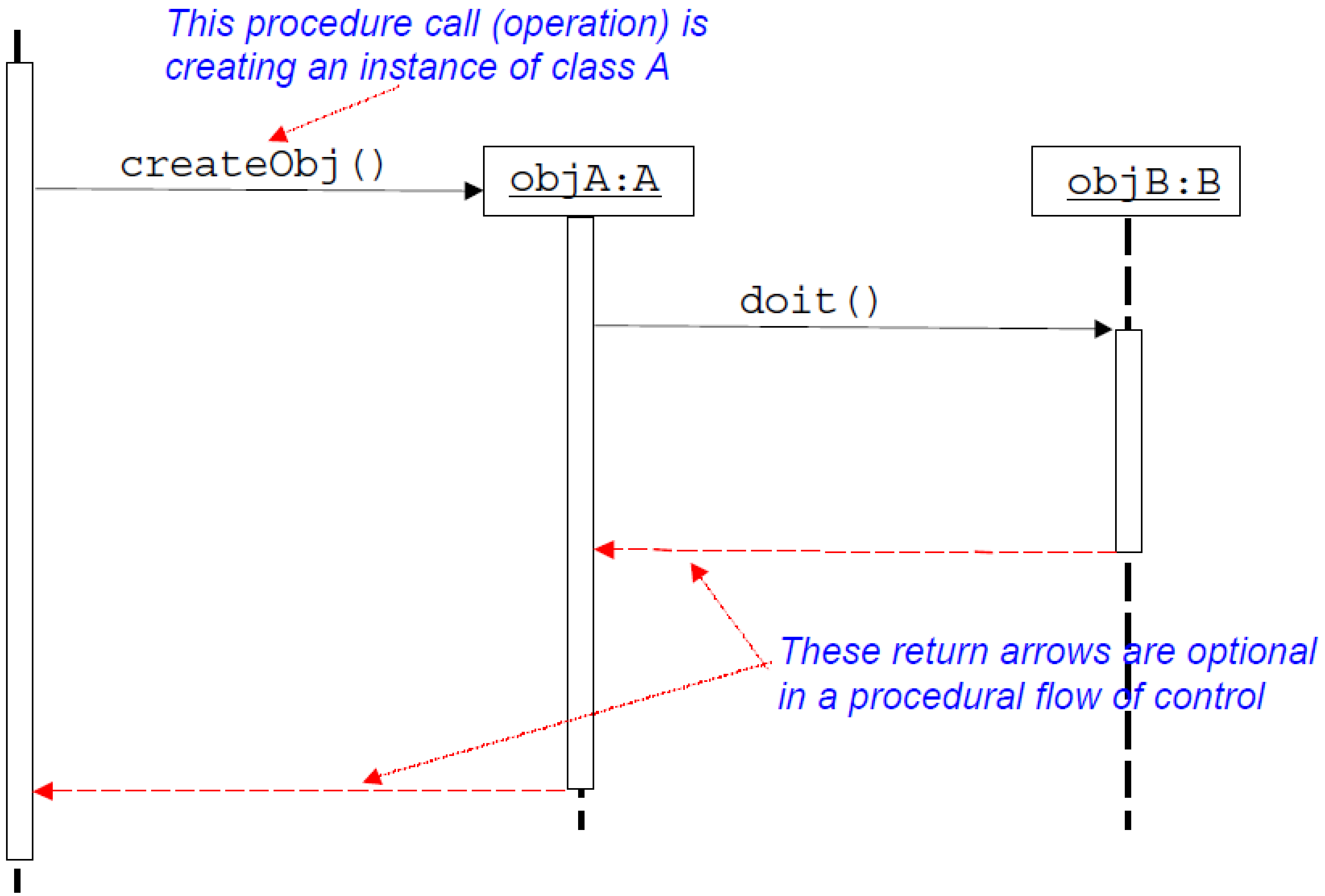
# An Even More Realistic Example



# Another Example: Print Shop



# Flow of Control



This procedure call (operation) is creating an instance of class A

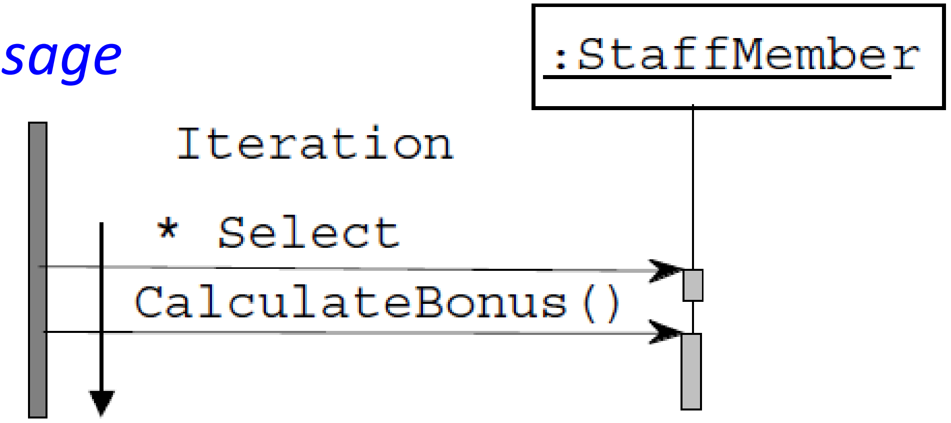
These return arrows are optional in a procedural flow of control

# Iteration

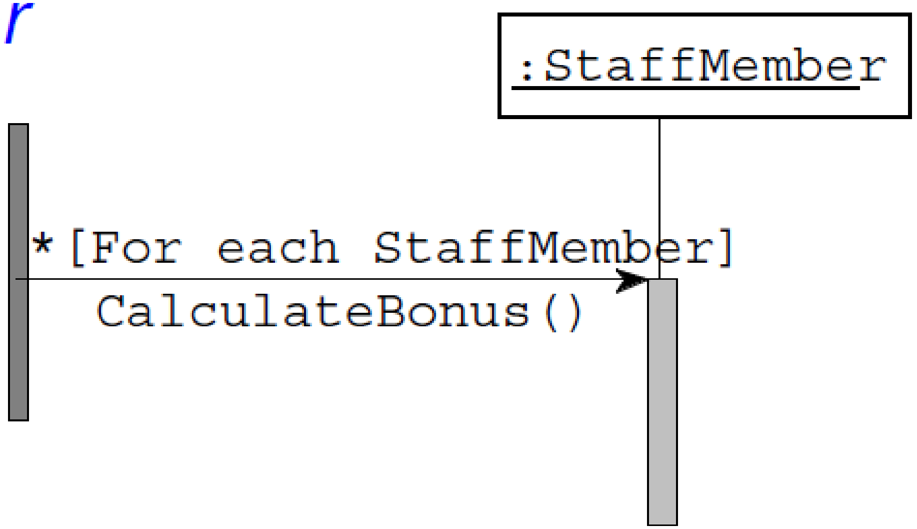
- *Iteration shown with an asterisk.*
  - ✓ *Each* StaffMember *will be selected in turn*
  - ✓ *Once selected, the* CalculateBonus *message will be sent to the one currently selected*
- *There is only one loop!*

```
Calculate Staff Bonuses

Description
Start
  For Each StaffMember
    Select next Staff Member
    Calculate Bonus for Staff
    Member
```



*or*

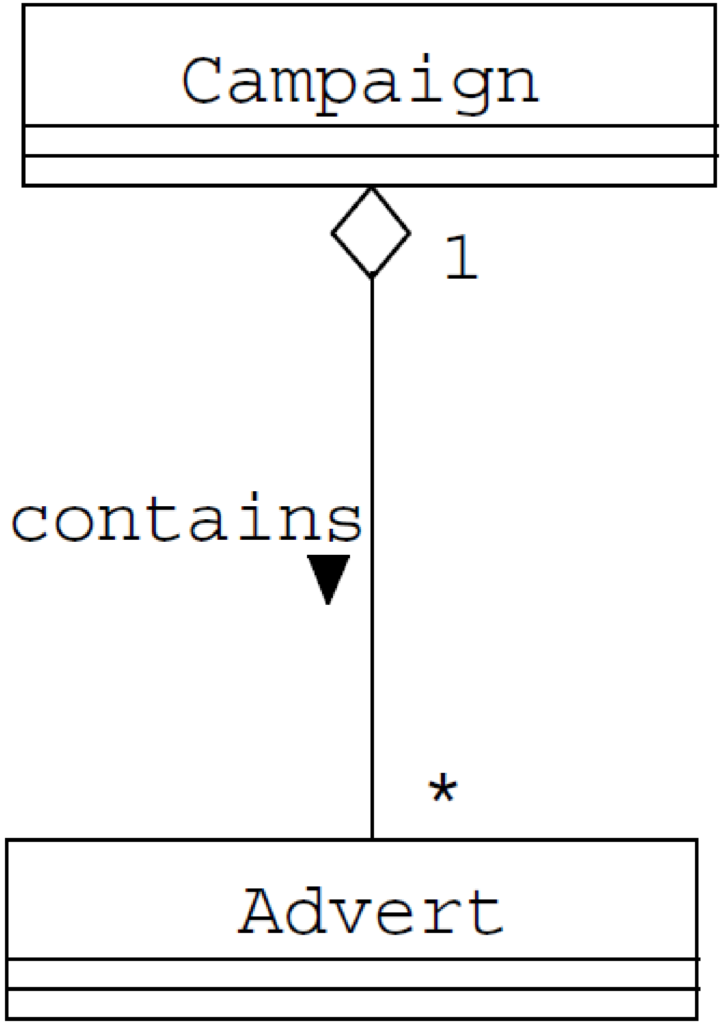


# *Drawing Sequence Diagrams*

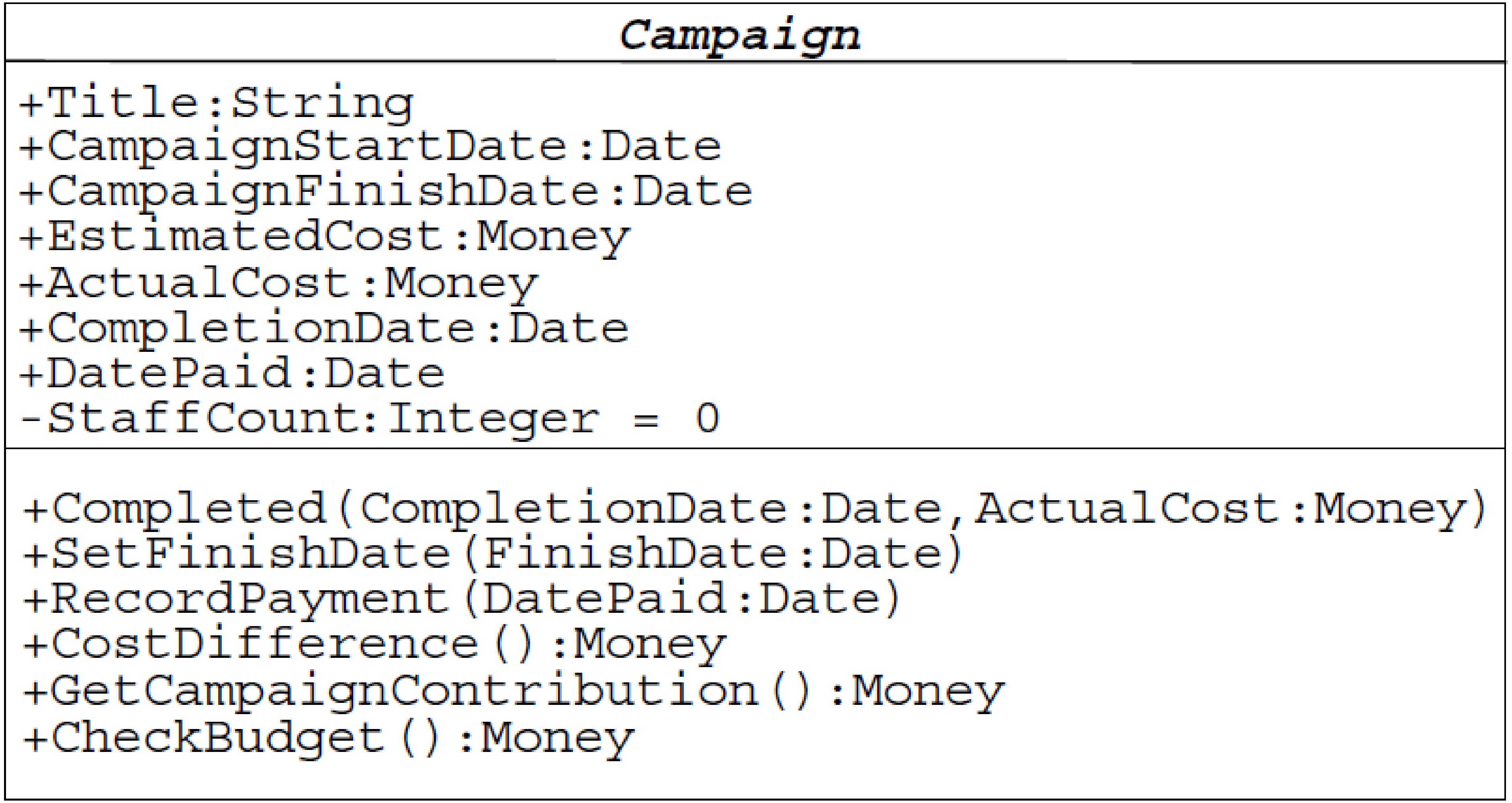
- *For a use case, identify participating actors.*
- *Imagine that there is a use case required by Agate called* `Check Campaign Budget.`
- `Campaign` ***has an*** `EstimatedCost` ***attribute and*** `Advert` ***has an*** `EstimatedCost` ***attribute.***
- *The purpose of the use case is to check that the total estimated cost of all the adverts is less than that for the campaign as a whole.*
- *...Which objects are involved here?*

# Campaign and Advert

*Class diagram showing aggregation*



# The Campaign Class

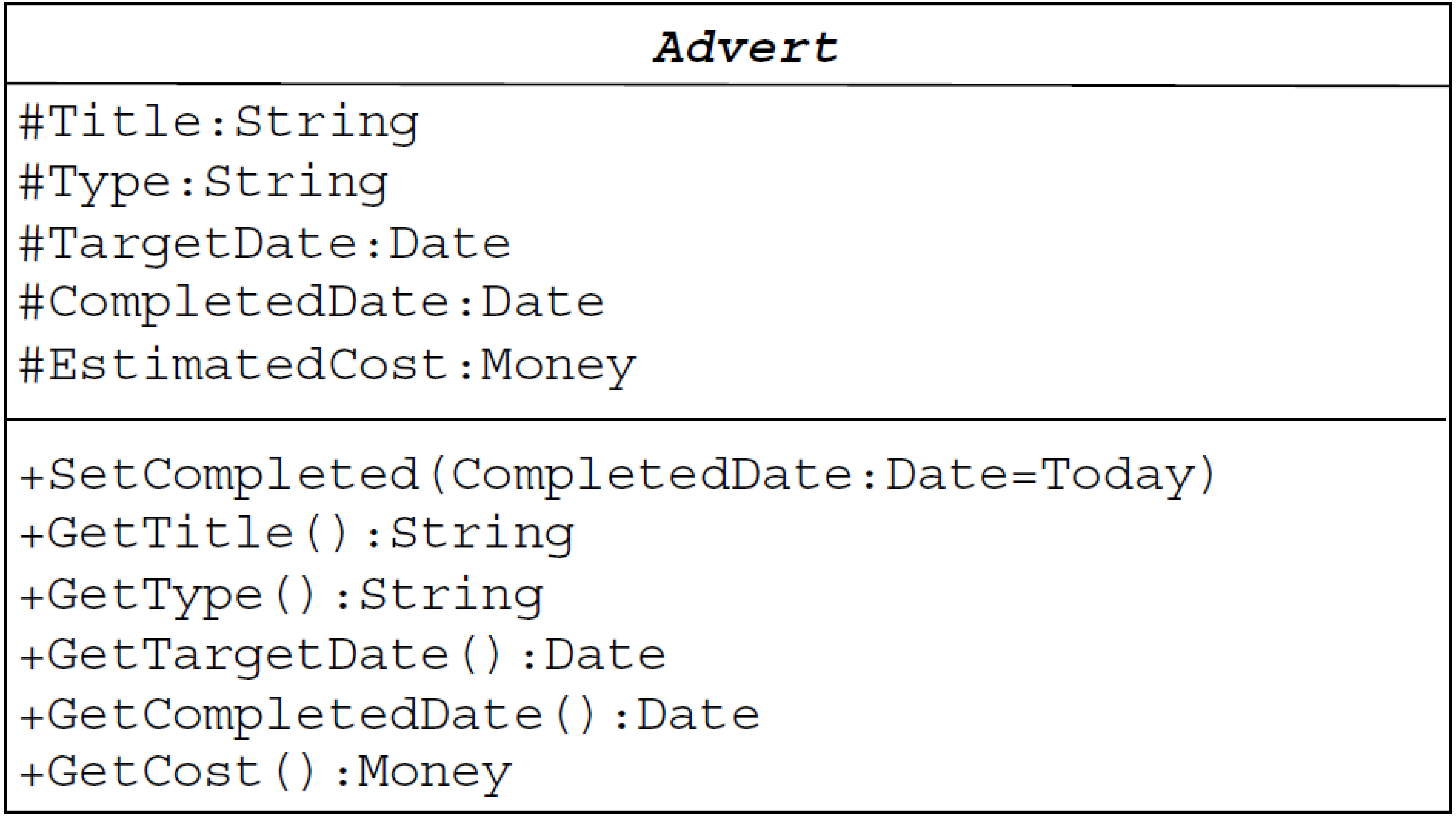


*Campaign*

+Title:String
+CampaignStartDate:Date
+CampaignFinishDate:Date
+EstimatedCost:Money
+ActualCost:Money
+CompletionDate:Date
+DatePaid:Date
-StaffCount:Integer = 0

+Completed(CompletionDate:Date,ActualCost:Money)
+SetFinishDate(FinishDate:Date)
+RecordPayment(DatePaid:Date)
+CostDifference():Money
+GetCampaignContribution():Money
+CheckBudget():Money

# The Advert Class



***Advert***

#Title:String
#Type:String
#TargetDate:Date
#CompletedDate:Date
#EstimatedCost:Money

+SetCompleted(CompletedDate:Date=Today)
+GetTitle():String
+GetType():String
+GetTargetDate():Date
+GetCompletedDate():Date
+GetCost():Money

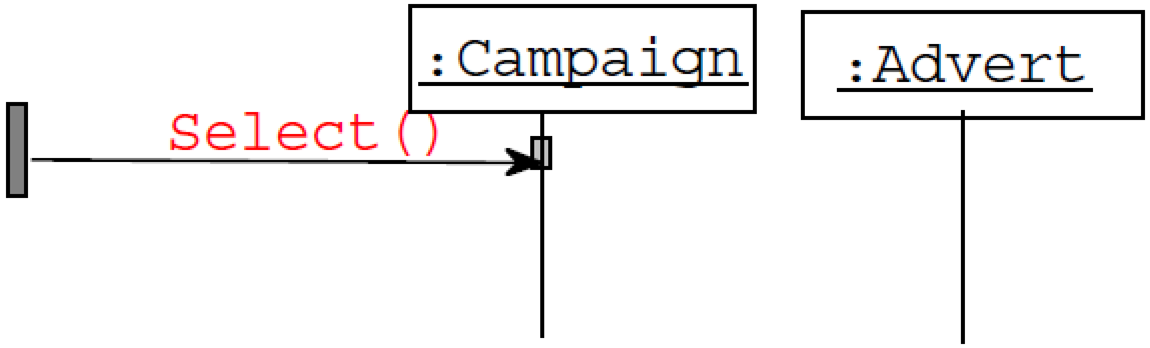
# *Drawing a Sequence Diagram*

- *Where do we start?*
- *Select the relevant* `Campaign`*, probably using its name.*
- *How we select it is something we leave for the design phase:*
  - ✓ *it could be from a list box;*
  - ✓ *it could involve a separate window on the screen;*
  - ✓ *it could involve some kind of index.*
- *These are design issues, which we shall leave for now, although we should document them if the customer expressed a preference at this stage.*

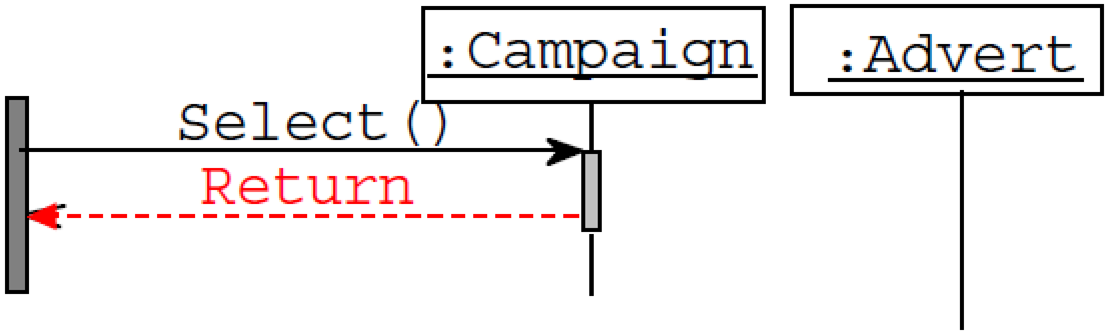
# Creating a Sequence Diagram

Check Campaign Budget

| Description |
|---|
| Select Campaign |



*We can also add in a* `Return`

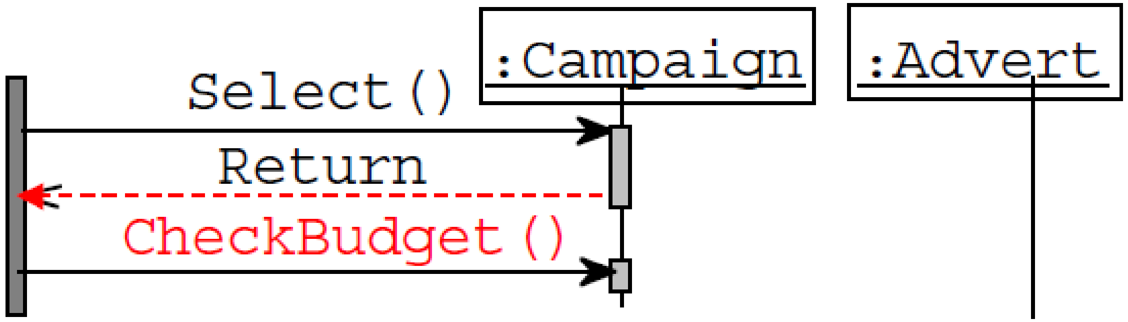


# *Creating a Sequence Diagram*

- *We then need to send a message to the* `Campaign` *to check its budget.*

Check Campaign Budget

| Description |
|---|
| Select Campaign |
| - |
| Check Budget |



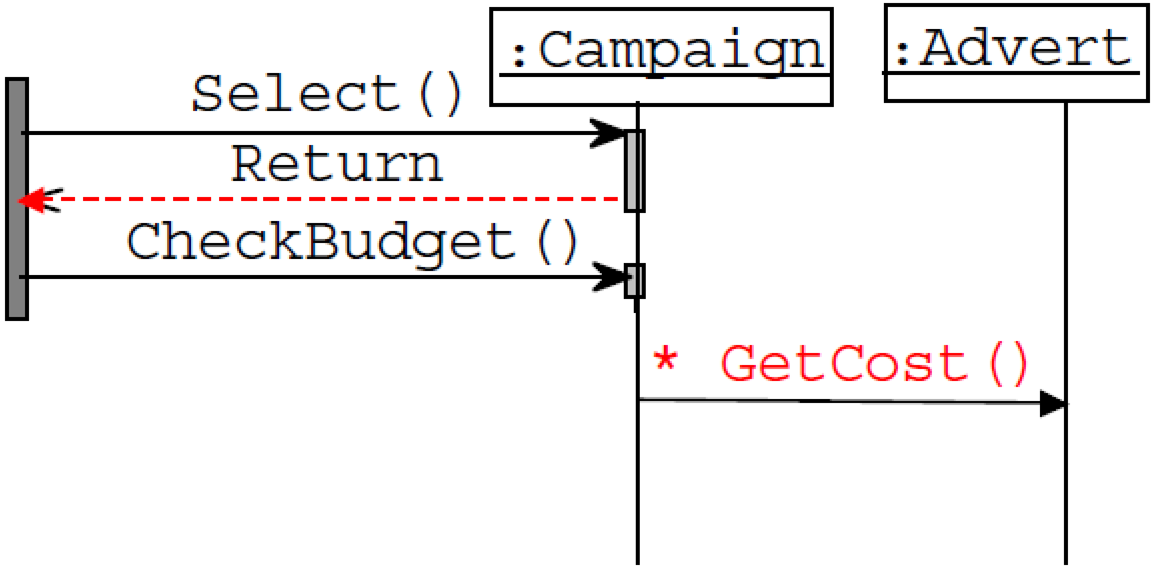
- *Note there is no* `Return` *here. Where does control go?*

# Creating a Sequence Diagram

Check Campaign Budget

Description

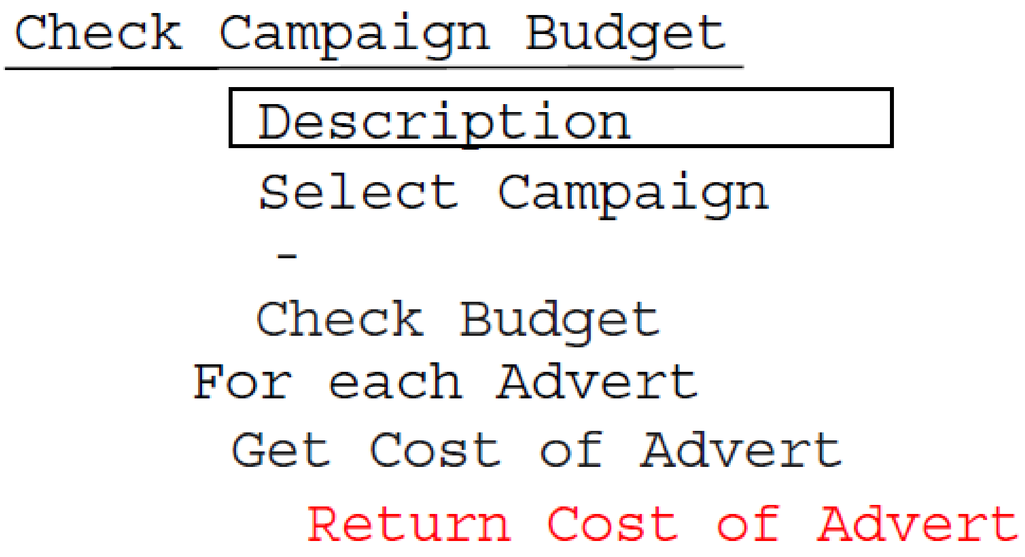
Select Campaign
-
Check Budget
For each Advert
Get Cost of Advert



- *Note the * for iteration.*
- *We are assuming here that:* `Campaign` *knows about all the* `Adverts` *that are associated with it because of the aggregation association shown earlier.*

# Creating a Sequence Diagram

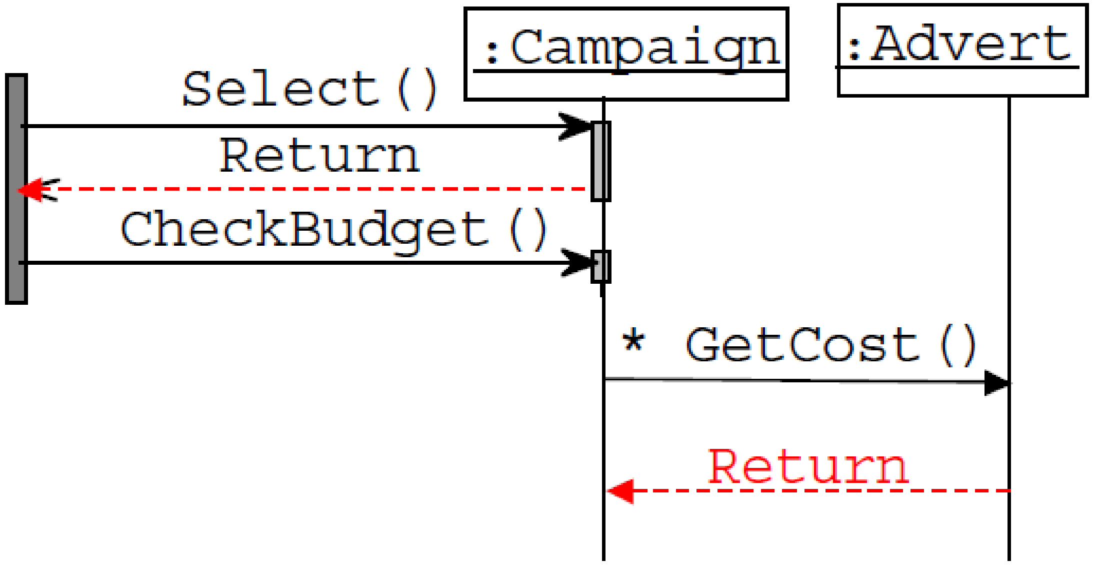
- *What happens next?*



Check Campaign Budget

Description

Select Campaign
-
Check Budget
For each Advert
Get Cost of Advert
Return Cost of Advert



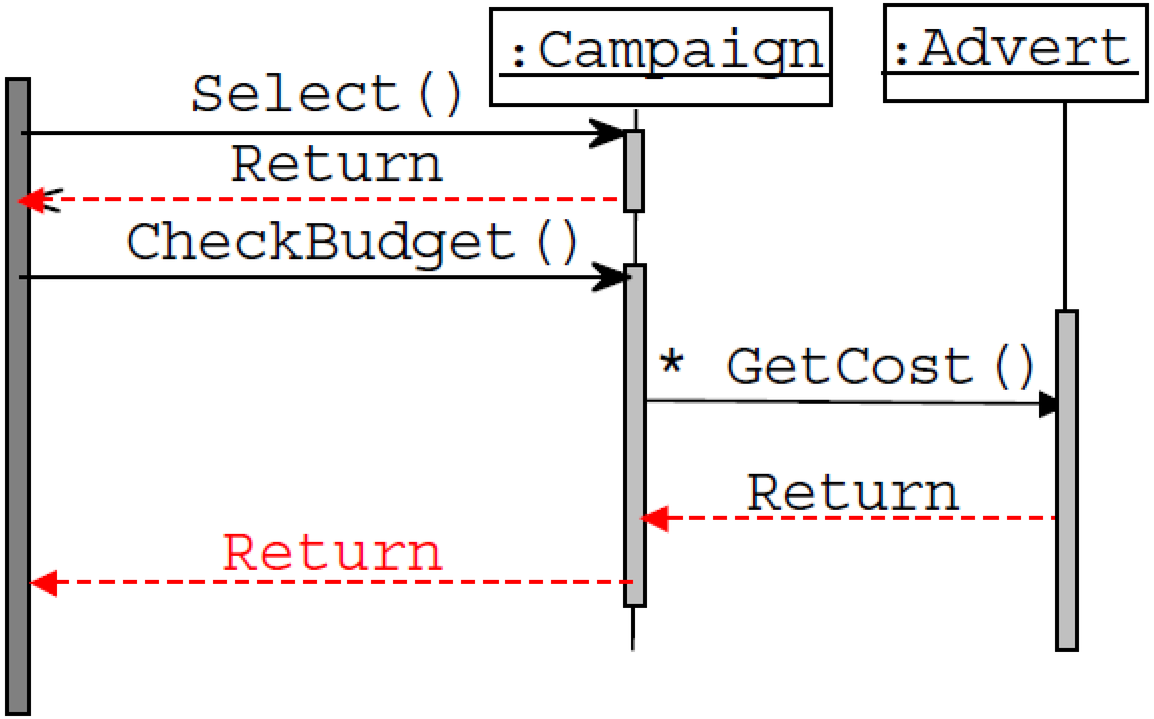
- Advert *returns its cost, in this case the* EstimatedCost *of the* Advert.
- *Once all the* Advert*'s costs have been fetched and summed up, the total can be taken away from the* EstimatedCost *of the* Campaign.

# Creating a Sequence Diagram

Check Campaign Budget

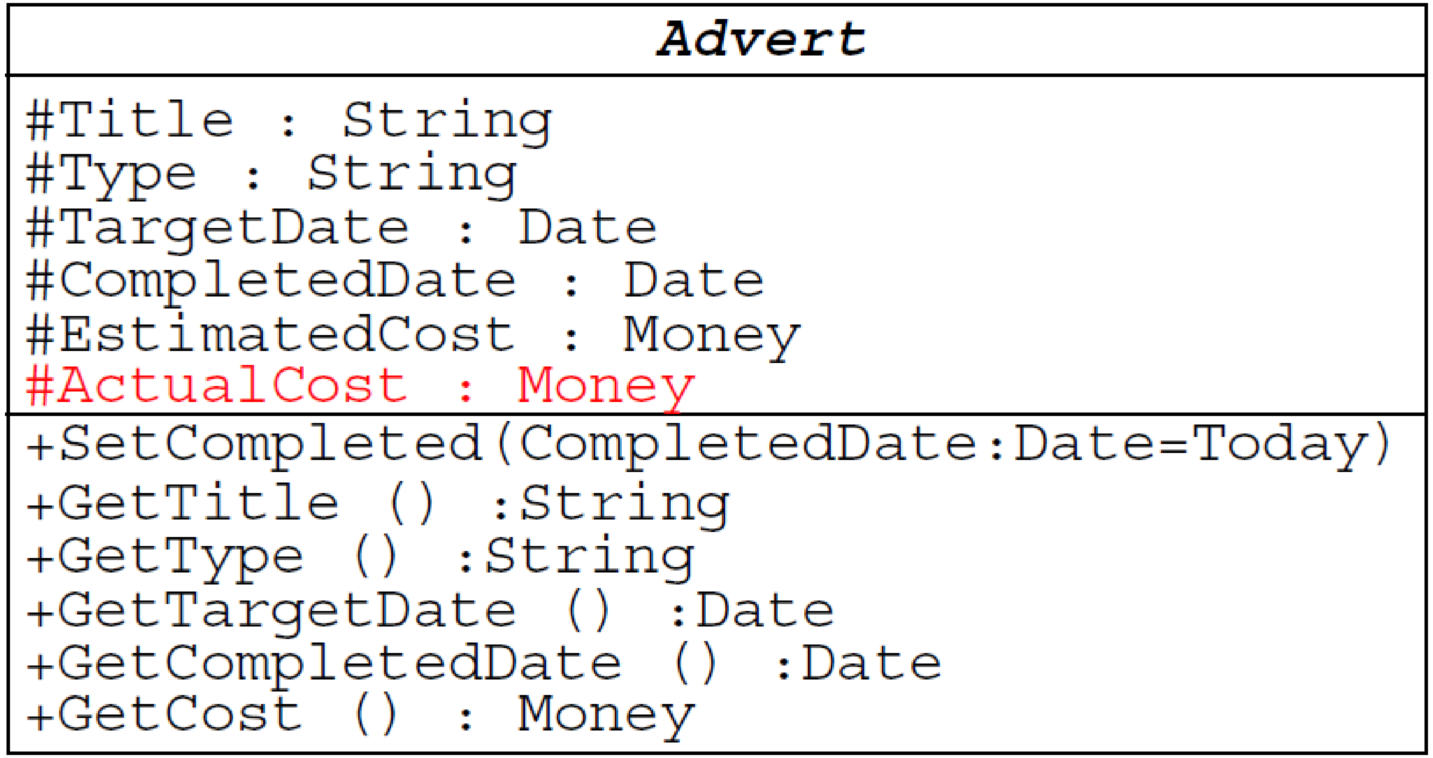
Description

Select Campaign
-
Check Budget
For each Advert
Get Cost of Advert
Return Cost of Advert
Return (Estimated Cost
- Cost of Adverts)



- *Now* Campaign *can return the difference between estimated cost and actual cost.*

# …Back to Class Diagrams…



```
Advert
#Title : String
#Type : String
#TargetDate : Date
#CompletedDate : Date
#EstimatedCost : Money
#ActualCost : Money
+SetCompleted(CompletedDate:Date=Today)
+GetTitle () :String
+GetType () :String
+GetTargetDate () :Date
+GetCompletedDate () :Date
+GetCost () : Money
```

- *We could add a new attribute to* `Advert` *called* `ActualCost`*, set when* `Advert` *is completed.*
- *Now* `GetCost()` *can return the* `ActualCost` *if it exists, otherwise it uses* `EstimatedCost().`

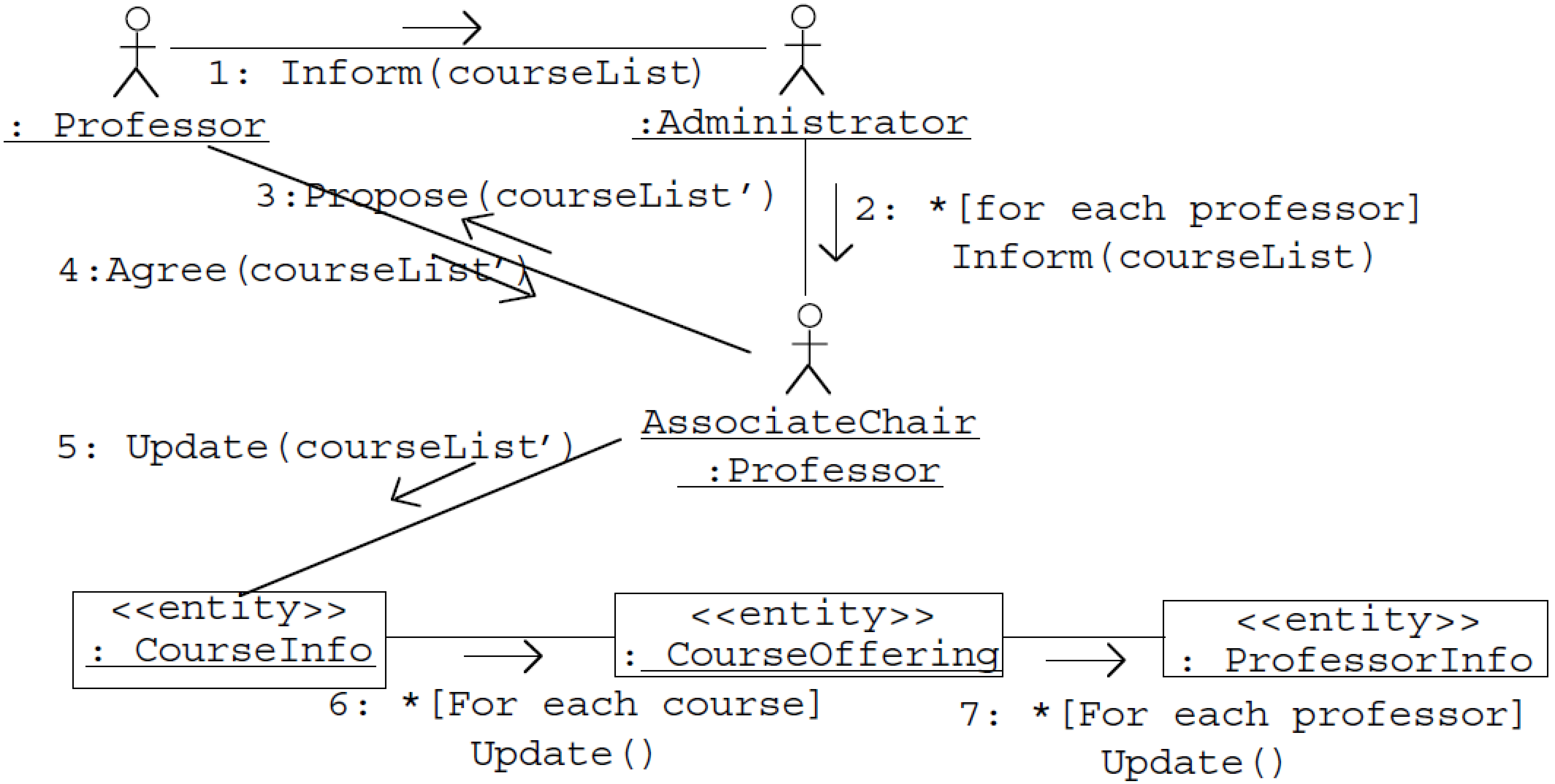
# *How to Use Sequence Diagrams*

- *In general, you may need several sequence diagrams to describe a single use case.*
- *A use case may involve complex control logic; sequence diagrams on the other hand should remain easy to read and understand.*
- *For a complex use case, use several sequence diagrams.*

# Collaboration Diagrams

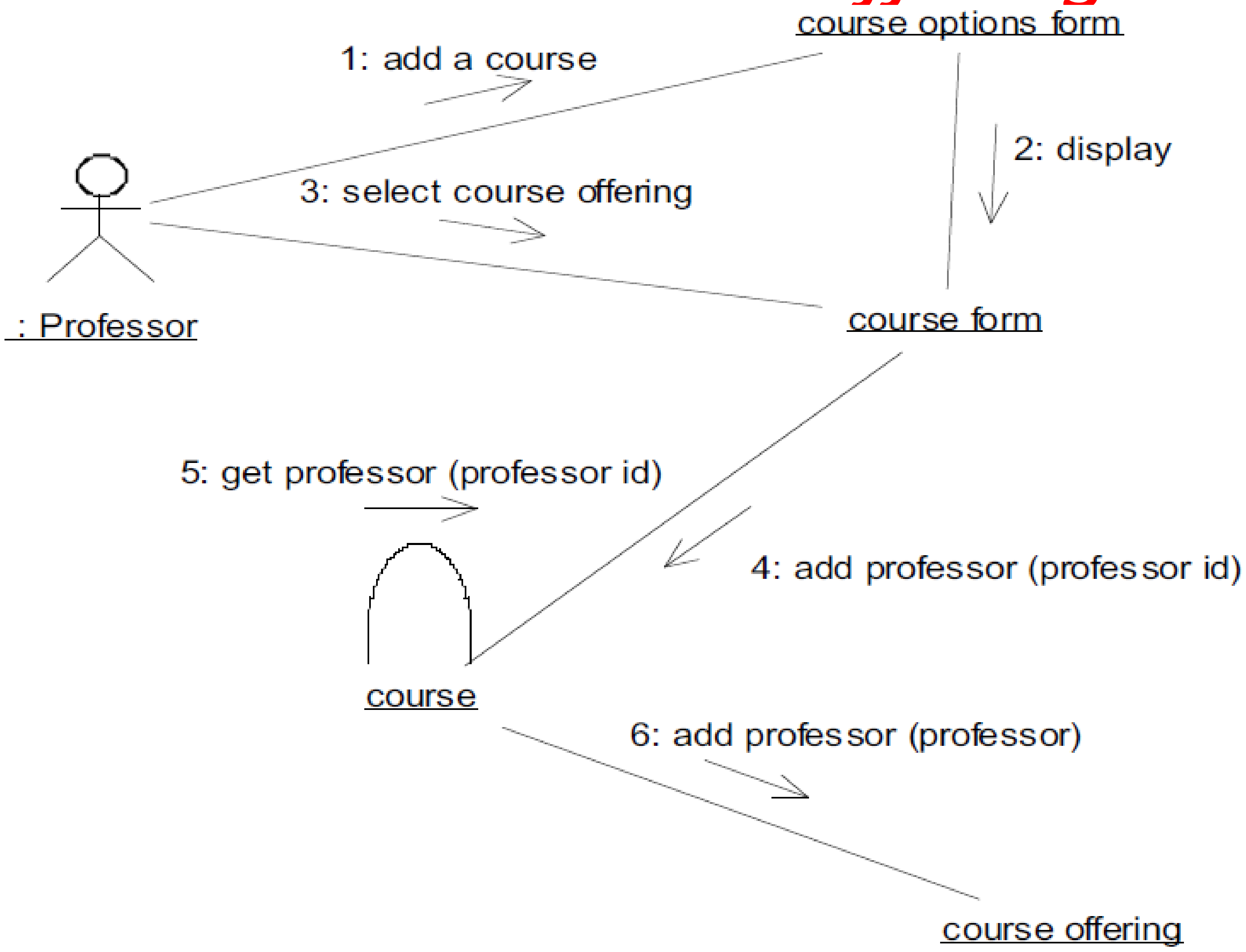
- *These diagrams are comparable to sequence diagrams. In fact, you can map every sequence diagram to an equivalent collaboration diagram and vice versa.*
- *Collaboration diagrams show interactions without the time dimension.*
- *Like sequence diagrams, collaboration diagrams are intended to model scenarios; each scenario describes a possible sequence of events and actions.*
- *Collaboration diagrams capture more directly the interactions between actors and objects.*

# Select Courses to Teach



: Professor
1: Inform(courseList)
:Administrator
2: *[for each professor] Inform(courseList)
3:Propose(courseList')
4:Agree(courseList')
AssociateChair :Professor
5: Update(courseList')
<<entity>> : CourseInfo
<<entity>> : CourseOffering
<<entity>> : ProfessorInfo
6: *[For each course] Update()
7: *[For each professor] Update()

# Add a Course Offering


course options form
1: add a course
2: display
3: select course offering
: Professor
course form
5: get professor (professor id)
4: add professor (profes sor id)
course
6: add profes sor (professor)
course offering

# Additional Readings

- *[Booch99] Booch, G. et al. The Unified Modeling Language User Guide. Chapters 15, 18, 27. Addison-Wesley.*
- *[Fowler00] Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modelling Language. Chapter 5. Addison-Wesley.*