Interaction diagrams

- Sequence diagrams
- Collaboration diagrams

CE202 Software Engineering, Autumn term

Dr Michael Gardner, School of Computer Science and Electronic Engineering, University of Essex

Reminder (last lecture): from requirements to classes

- Requirements (use cases) are usually expressed in user language
- Use cases are units of development, but they are not structured like software
- The software we will implement consists of classes
- We need a way to translate requirements into classes

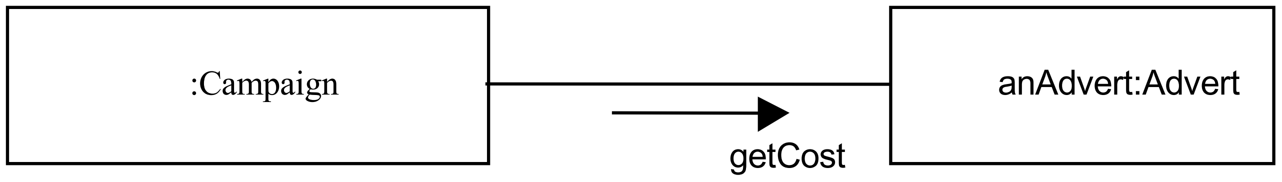
Reminder (last lecture): communication Diagram Approach

- Analyse one use case at a time
- Identify likely classes involved (the use case collaboration)
  - These may come from a domain model
- **Draw a communication diagram** that fulfils the needs of the use case (see next lecture)
- Translate this into a use case class diagram
- Repeat for other use cases
- Assemble the use case class diagrams into a single analysis class diagram

Object Messaging

Objects communicate by sending messages. Sending the message getCost() to an Advert object, might use the following syntax.

currentadvertCost = anAdvert.getCost()



Interaction & Collaboration

- A **collaboration** is a group of objects or classes that work together to provide an element of functionality or behaviour.
- An **interaction** defines the message passing between lifelines (e.g. objects) within the context of a collaboration to achieve a particular behaviour.

Modelling Interactions

- Interactions can be modelled using various notations
  - **Sequence** diagrams
  - **Collaboration** diagrams
  - Interaction overview diagrams – not covered
  - Timing diagrams – not covered

Sequence Diagrams

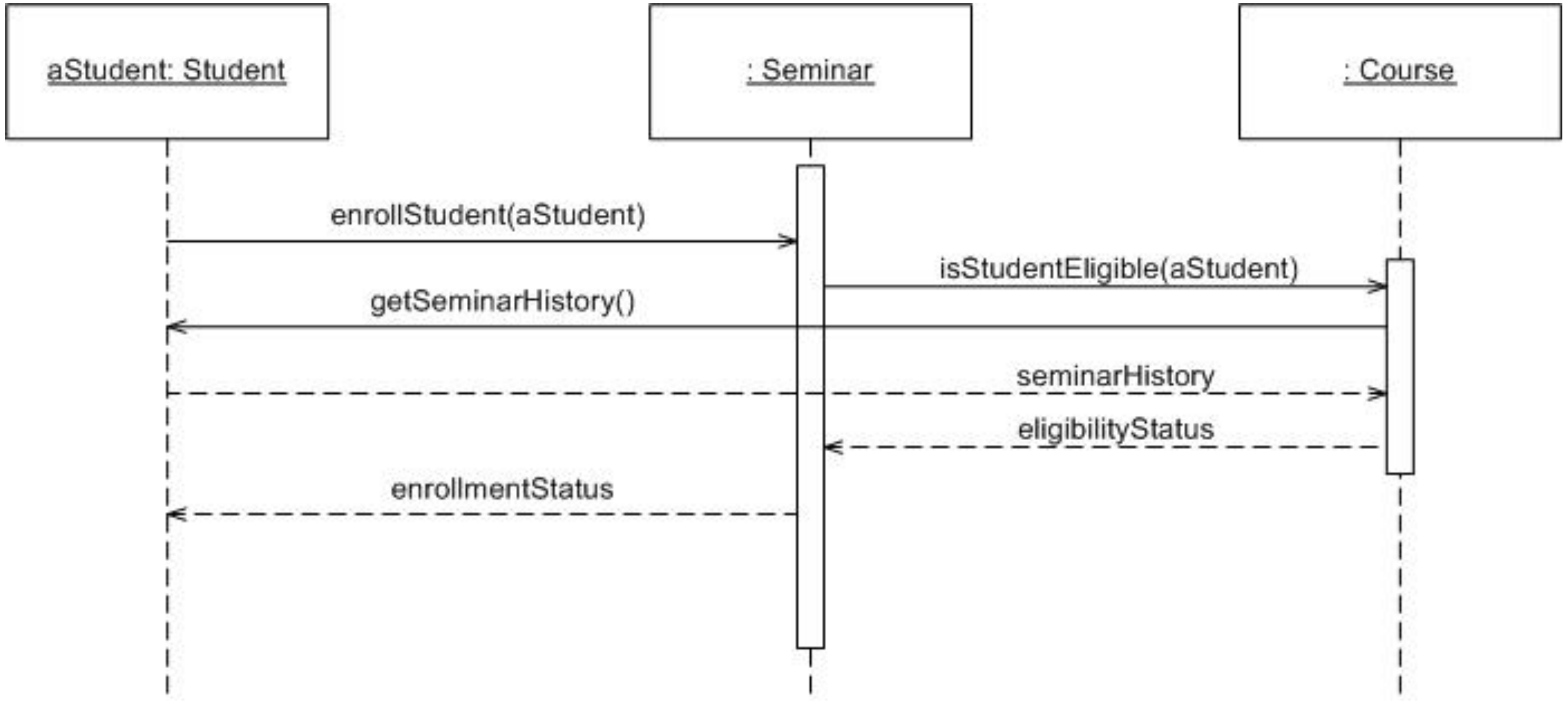
Sequence Diagrams

- Shows an interaction between lifelines (e.g. objects) arranged in a time sequence.
- Can be drawn at different levels of detail and to meet different purposes at several stages in the development life cycle.
- Typically used to represent the detailed object interaction that occurs **for one use case** or for one operation.

Sequence Diagrams

- Vertical dimension shows time.
- Objects (or subsystems or other connectable objects) involved in interaction appear horizontally across the page and are represented by lifelines.
- Messages are shown by a solid horizontal arrow.
- The execution or activation of an operation is shown by a rectangle on the relevant lifeline.

Example sequence diagram



Guidelines for Sequence Diagrams

1. Decide at what level you are modelling the interaction.

   Is it describing an operation, a use case, the messaging between components or the interaction of subsystems or systems?

2. Identify the main elements involved in the interaction.

   If the interaction is at use case level the collaborating objects may already have been identified through the use of CRC cards (discussed later)

3. Consider the alternative scenarios that may be needed.
4. Identify the main elements involved in the interaction.

Guidelines for Sequence Diagrams

5. Draw the outline structure of the diagram.
6. Add the detailed interaction.
7. Check for consistency with linked sequence diagrams and modify as necessary.
8. Check for consistency with other UML diagrams or models.

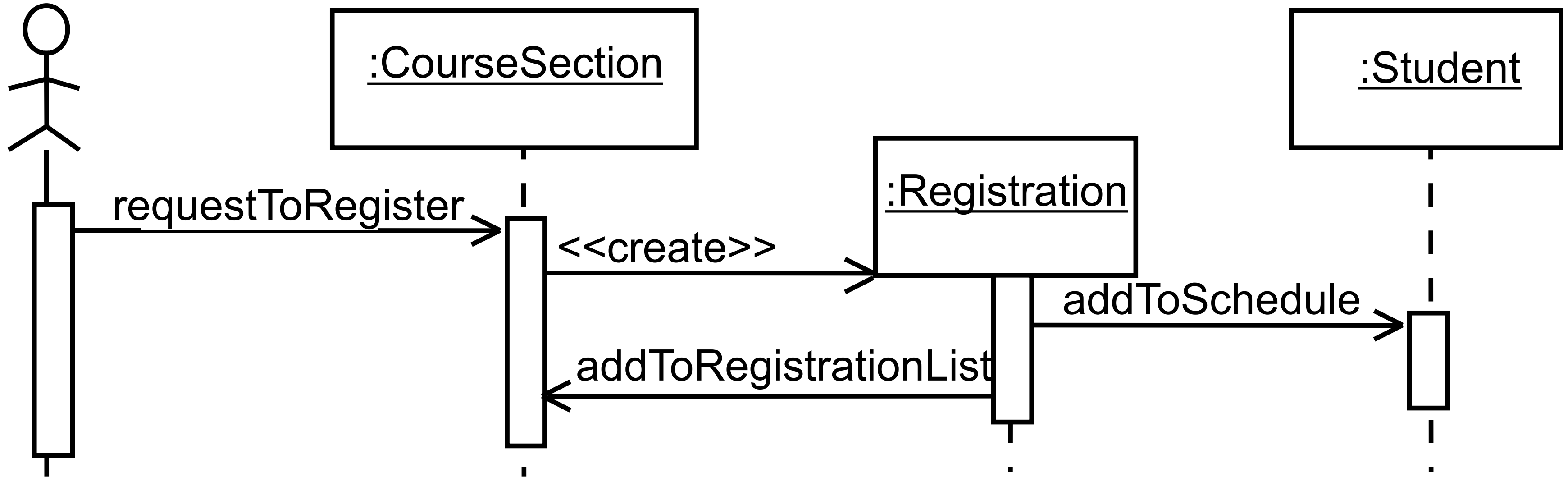
Model Consistency (with the class diagram)

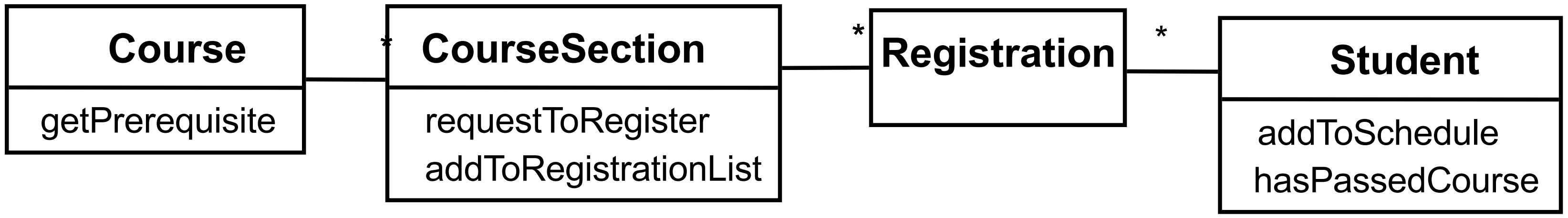
- The allocation of operations to objects must be consistent with the class diagram and the message signature must match that of the operation.
  - Can be enforced through CASE tools.
- Every sending object must have the object reference (**attribute**) for the destination object.
  - Either an association exists between the classes or another object passes the reference to the sender.
  - This issue is key in determining association message pathways and should be carefully analysed.
- Every receiving object must have the message listed as an **operation** in the class diagram

Model Consistency

- All forms of interaction diagrams used should be consistent.
- Messages on interaction diagrams must be consistent with the state machine for the participating objects. (see lecture next week)
- Implicit state changes in interactions diagrams must be consistent with those explicitly modelled in the state machine. (see lecture next week)

Sequence vs. class diagrams (consistency)

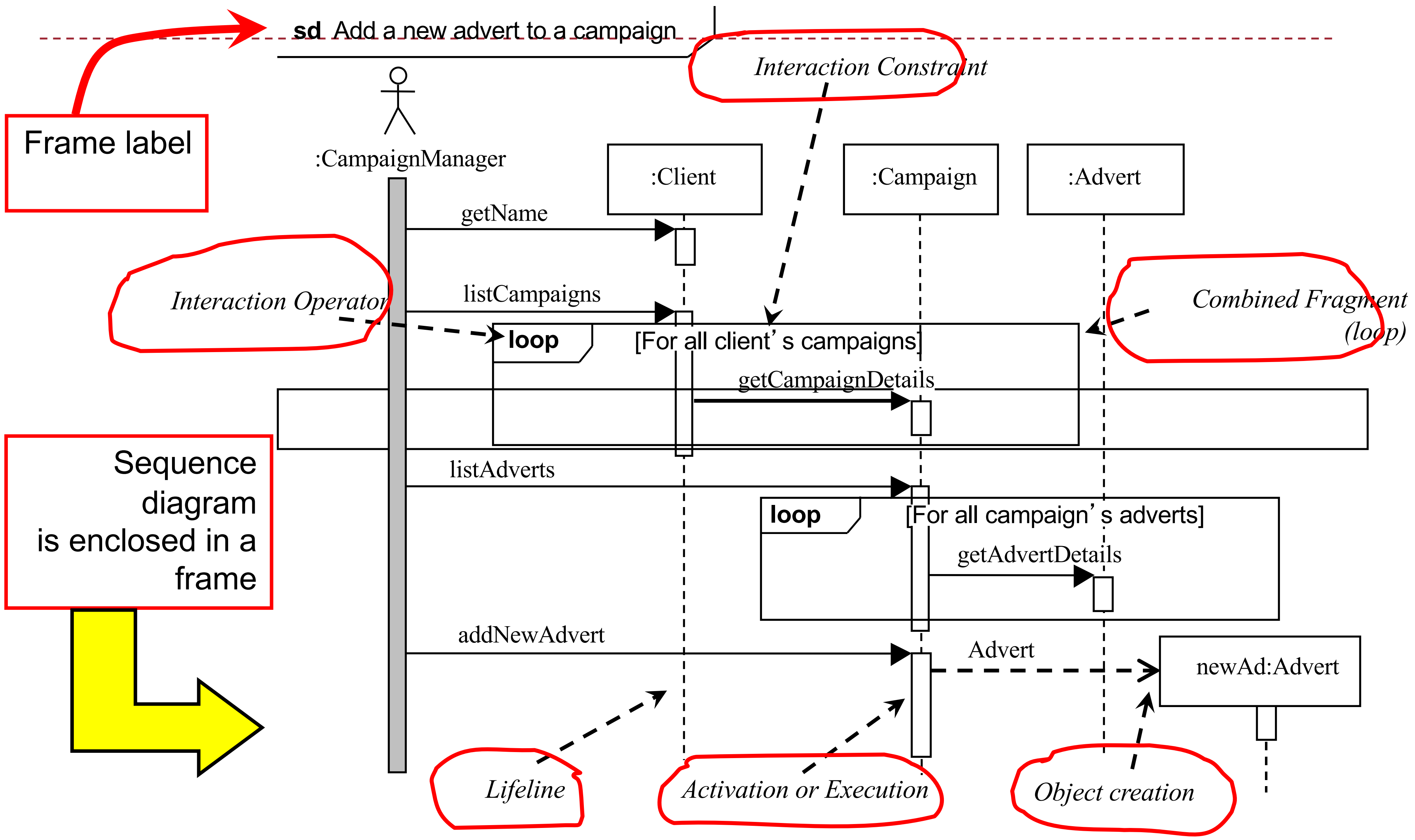




Sequence diagrams

- In the main we will be involved with fairly **simple** sequence diagrams which model the message passing between objects in a use case (see previous example)
- The following slides provide additional detail on the **full** use of sequence diagrams

Sequence diagram (full syntax)



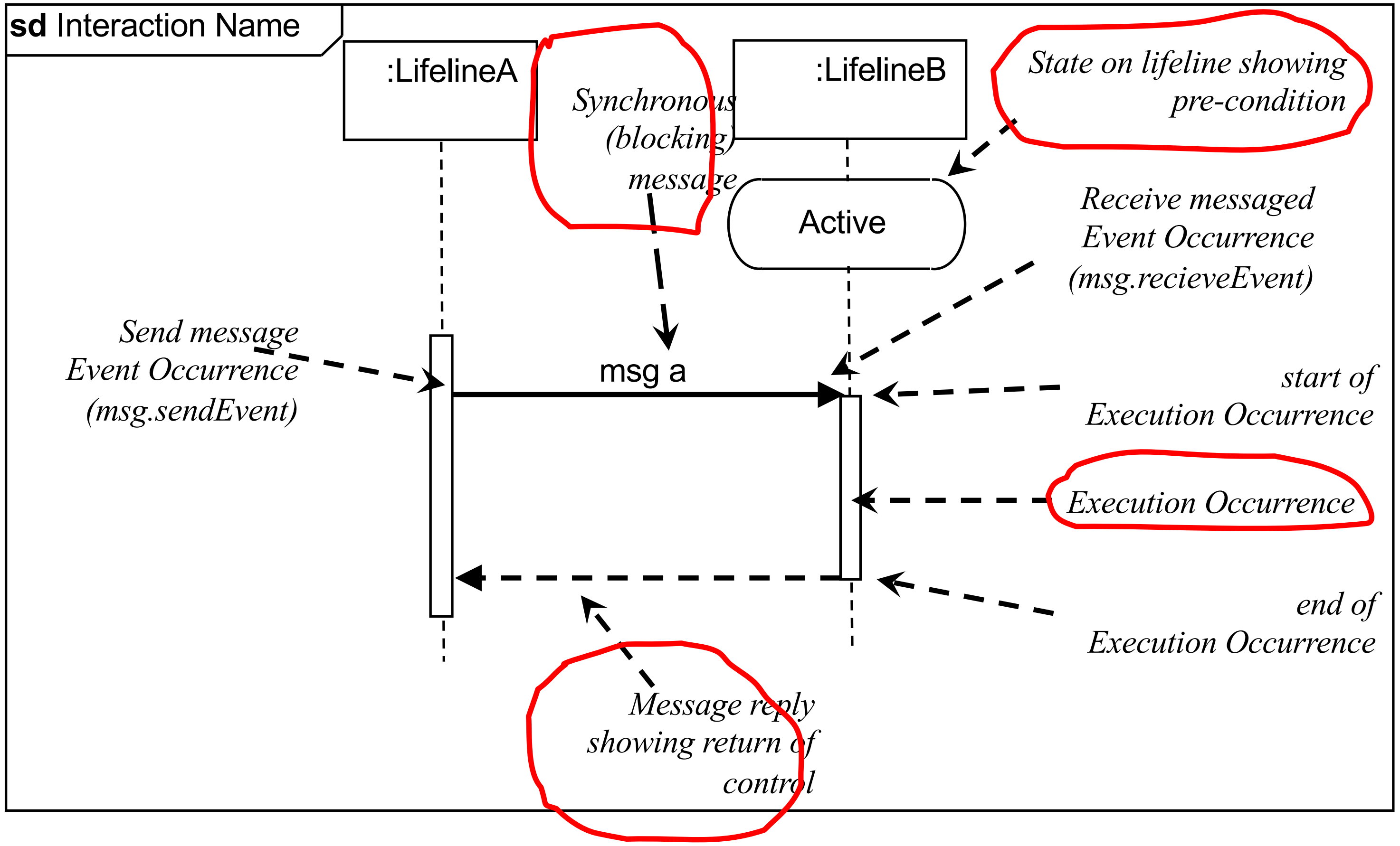
Sequence Diagram

- Iteration is represented by *combined fragment* rectangle with the *interaction operator* ‘loop’ .
- The loop combined fragment only executes if the guard condition in the interaction constraint evaluates as true.
- Object creation is shown with the construction arrow (dashed) going to the object symbol for the Advert lifeline.

Synchronous Message

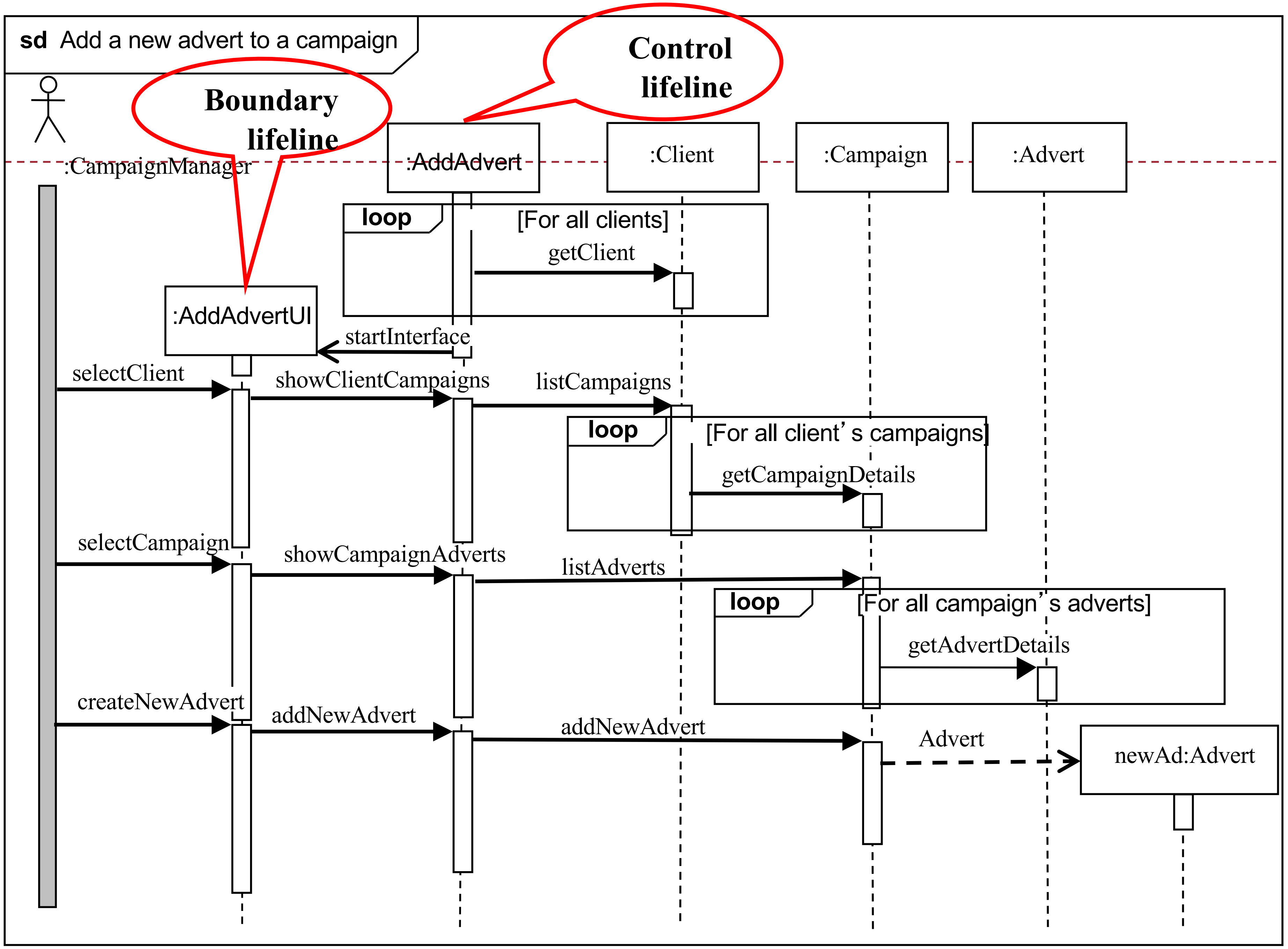
- A *synchronous message* or *procedural call* is shown with a full arrowhead, causes the invoking operation to suspend execution until the focus of control has been returned to it.

Further Notation

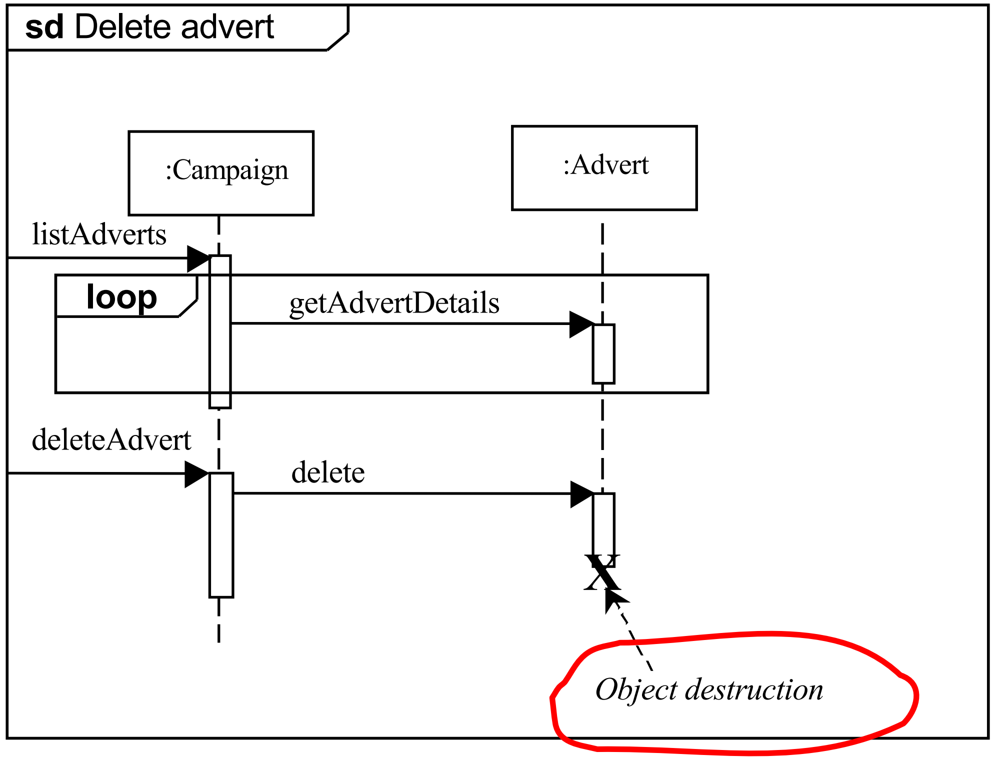

sd Interaction Name
:LifelineA
:LifelineB
Synchronous (blocking) message
State on lifeline showing pre-condition
Active
Receive messaged Event Occurrence (msg.recieveEvent)
Send message Event Occurrence (msg.sendEvent)
msg a
start of Execution Occurrence
Execution Occurrence
end of Execution Occurrence
Message reply showing return of control

Boundary & Control Classes

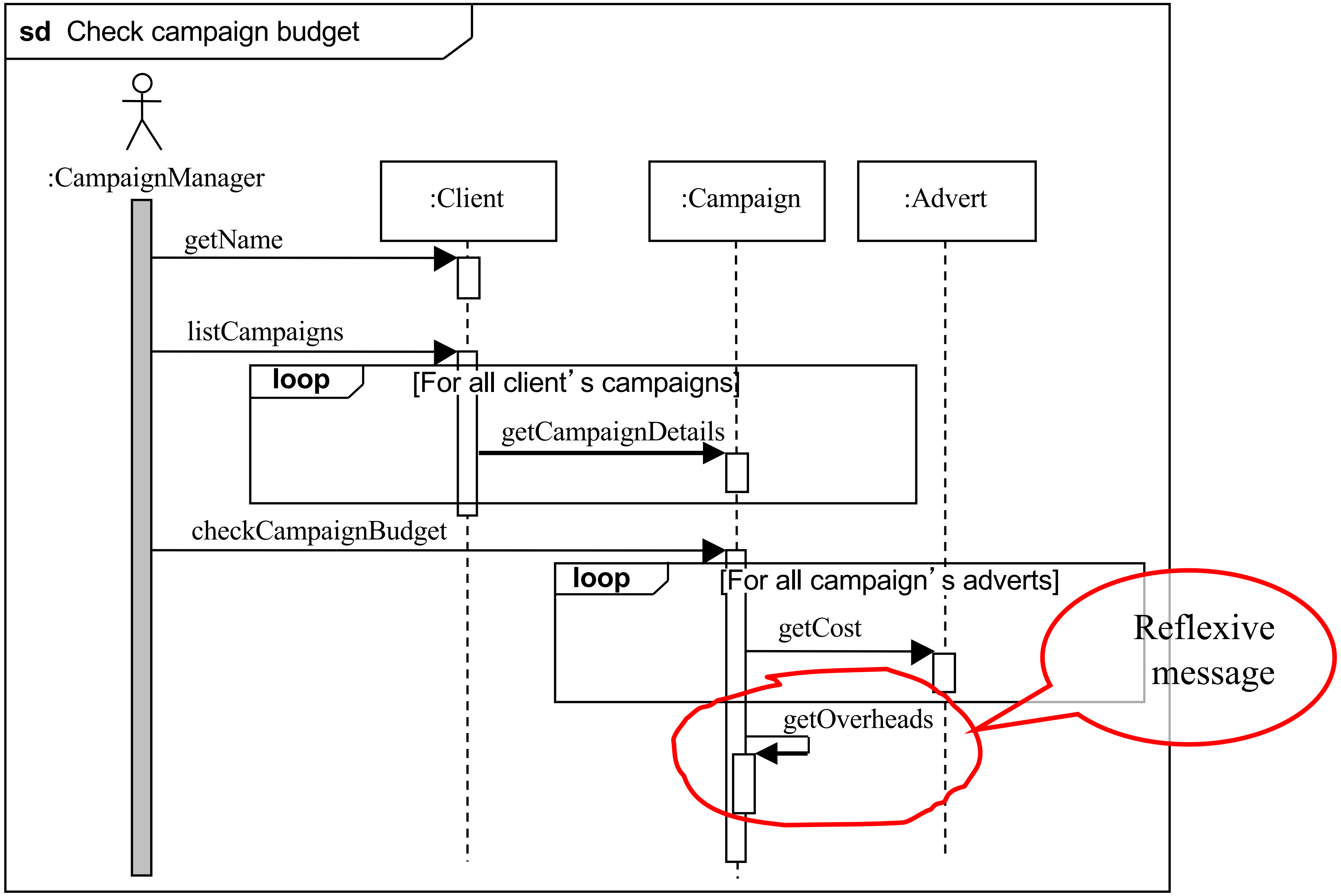
- Most use cases imply at least one boundary object that manages the dialogue between the actor and the system – in the next sequence diagram it is represented by the lifeline `:AddAdvertUI`
- The control object is represented by the lifeline `:AddAdvert` and this manages the overall object communication.


sd Add a new advert to a campaign
Boundary lifeline
Control lifeline
:CampaignManager
:AddAdvertUI
:AddAdvert
:Client
:Campaign
:Advert
loop [For all clients]
getClient
startInterface
selectClient
showClientCampaigns
listCampaigns
loop [For all client's campaigns]
getCampaignDetails
selectCampaign
showCampaignAdverts
listAdverts
loop [For all campaign's adverts]
getAdvertDetails
createNewAdvert
addNewAdvert
addNewAdvert
Advert
newAd:Advert

Object Destruction



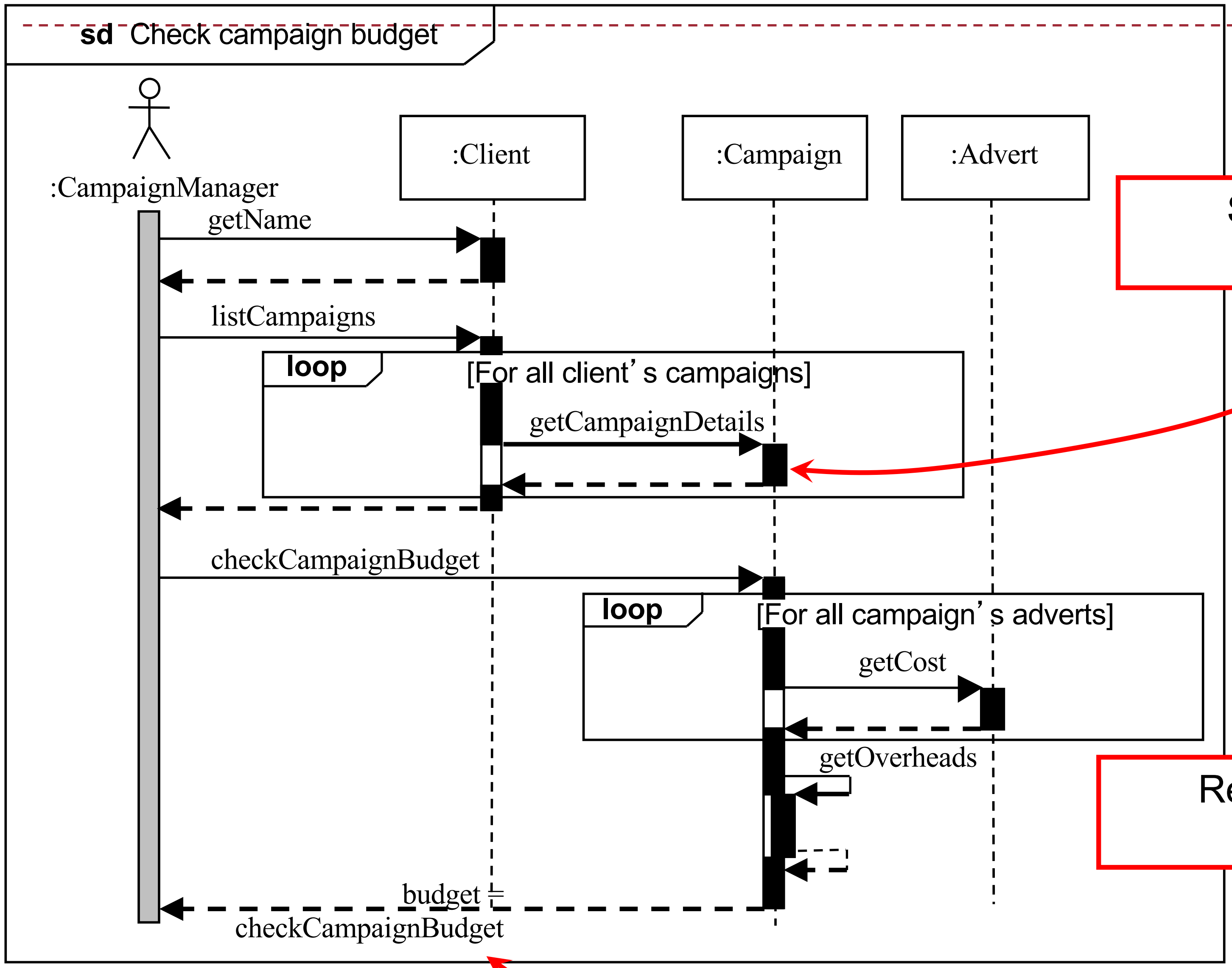
Reflexive Messages



Focus of Control

- Indicates times during an activation when processing is taking place within that object.
- Parts of an activation that are not within the focus of control represent periods when, for example, an operation is waiting for a return from another object.
- May be shown by shading those parts of the activation rectangle that correspond to active processing by an operation.

Focus of Control



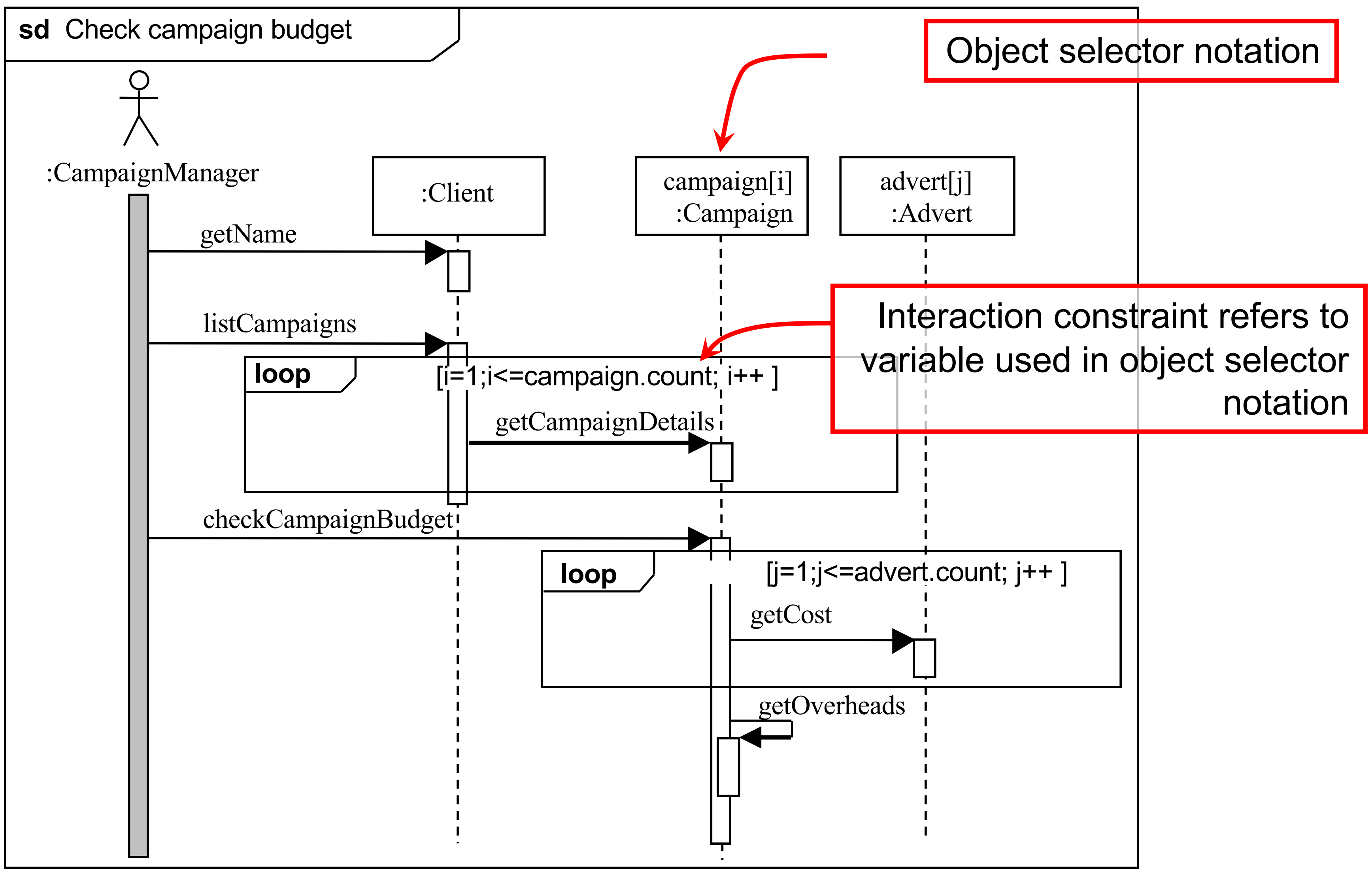
Shading showing the focus of control

Reply with the return-value shown

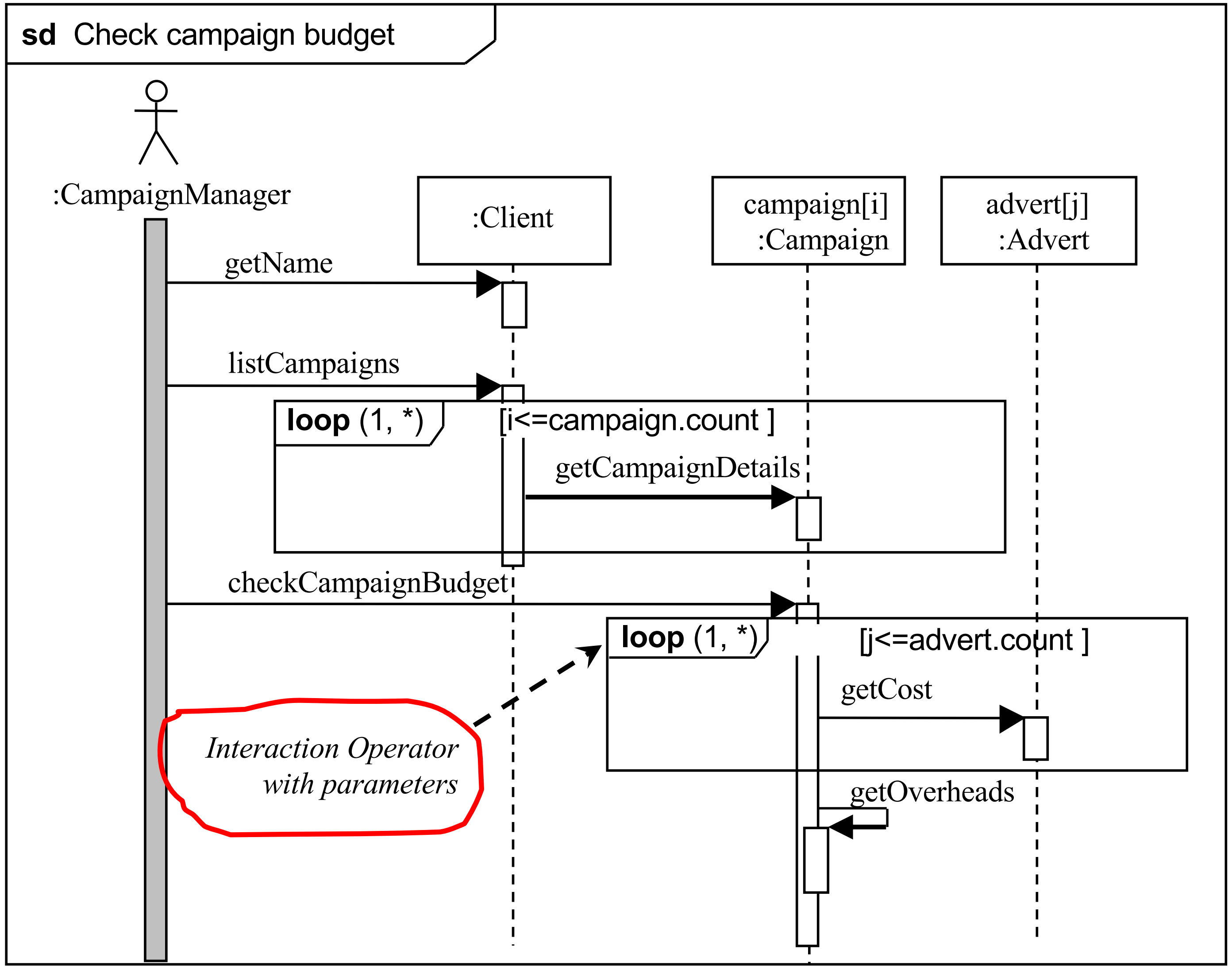
Reply Message

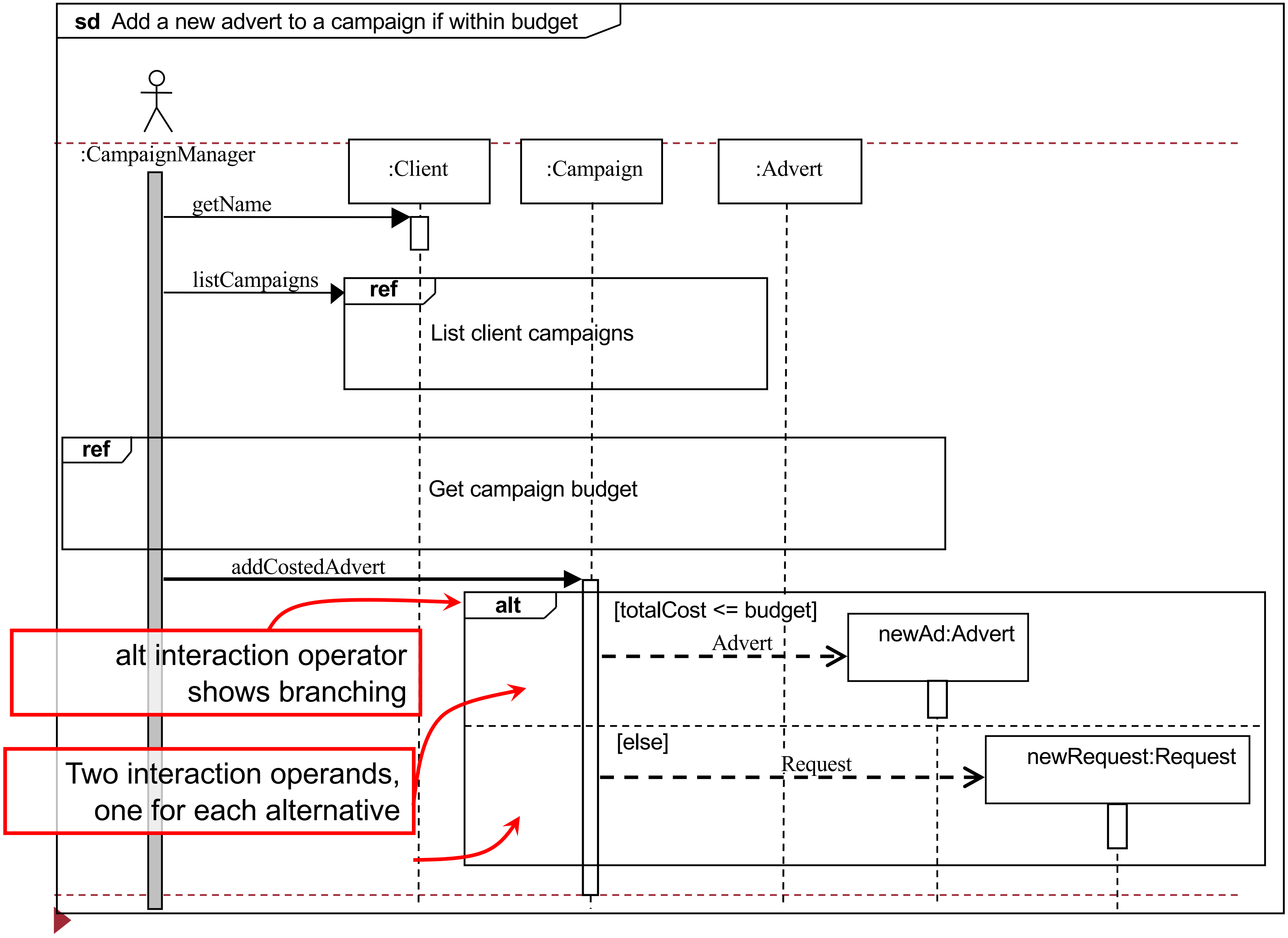
- A *reply message* returns the control to the object that originated the message that began the activation.
- Reply messages are shown with a dashed arrow, but it is optional to show them at all since it can be assumed that control is returned to the originating object at the end of the sequence

Object Selector Notation



Interaction Operators



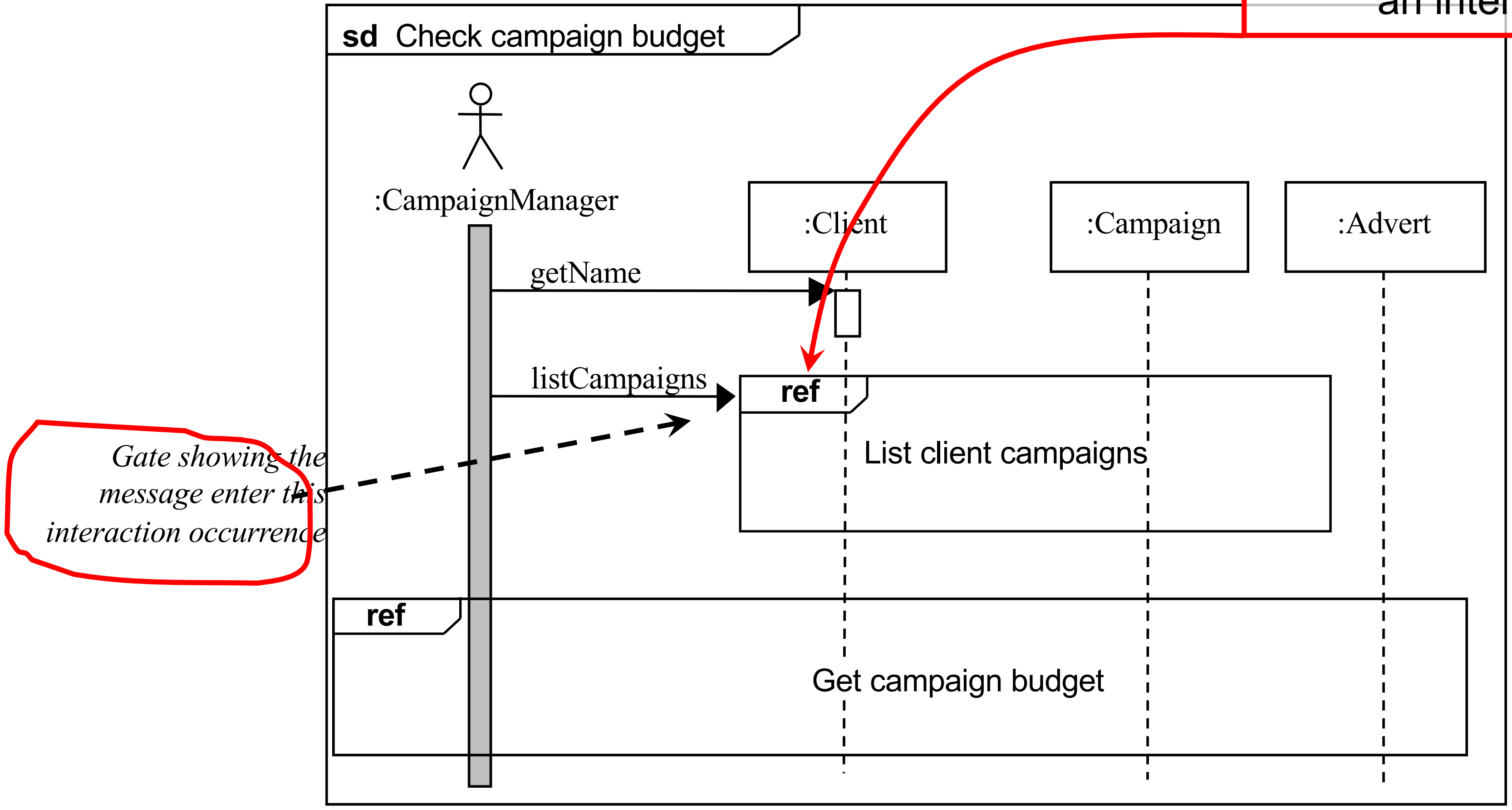

sd Add a new advert to a campaign if within budget
:CampaignManager
:Client
:Campaign
:Advert
getName
listCampaigns
ref
List client campaigns
ref
Get campaign budget
addCostedAdvert
alt
[totalCost <= budget]
newAd:Advert
Advert
[else]
Request
newRequest:Request
alt interaction operator shows branching
Two interaction operands, one for each alternative

Handling Complexity

- Complex interactions can be modelled using various different techniques
  - Interaction fragments
  - Lifelines for subsystems or groups of objects
  - Continuations

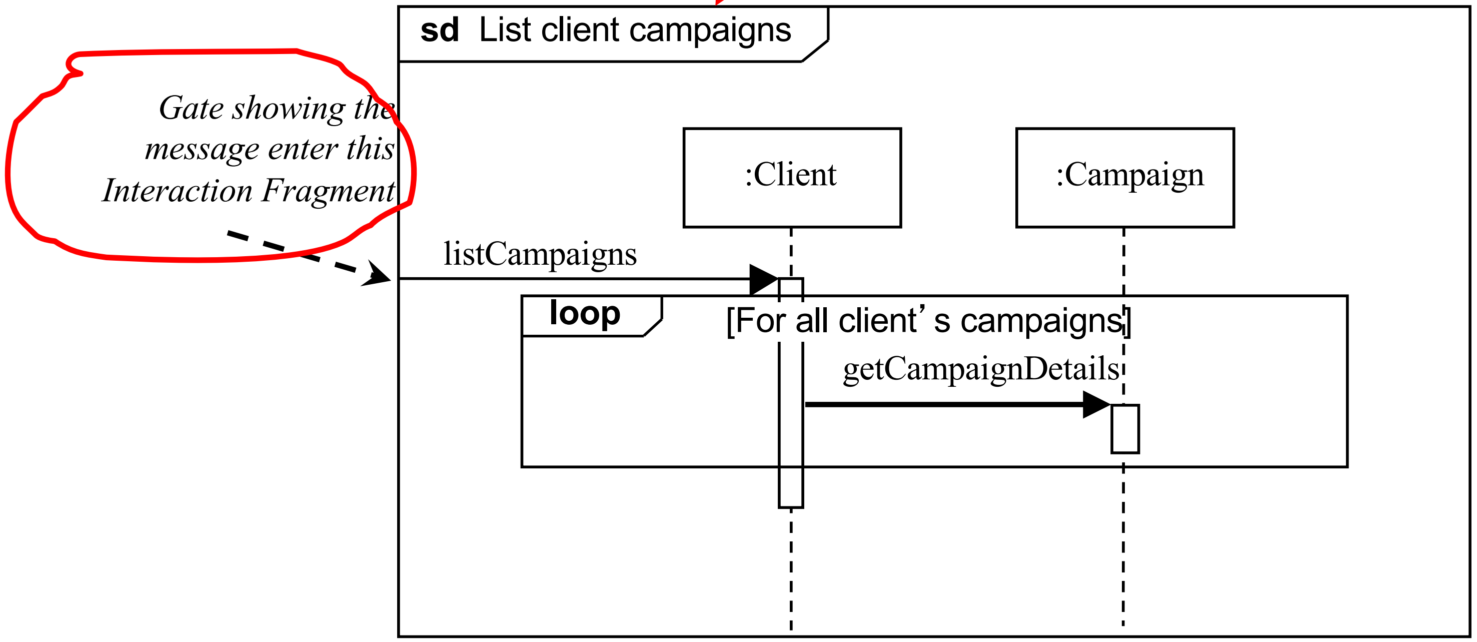
Using Interaction Fragments

ref interaction operator indicates interaction occurrence that references an interaction fragment



Interaction Fragment

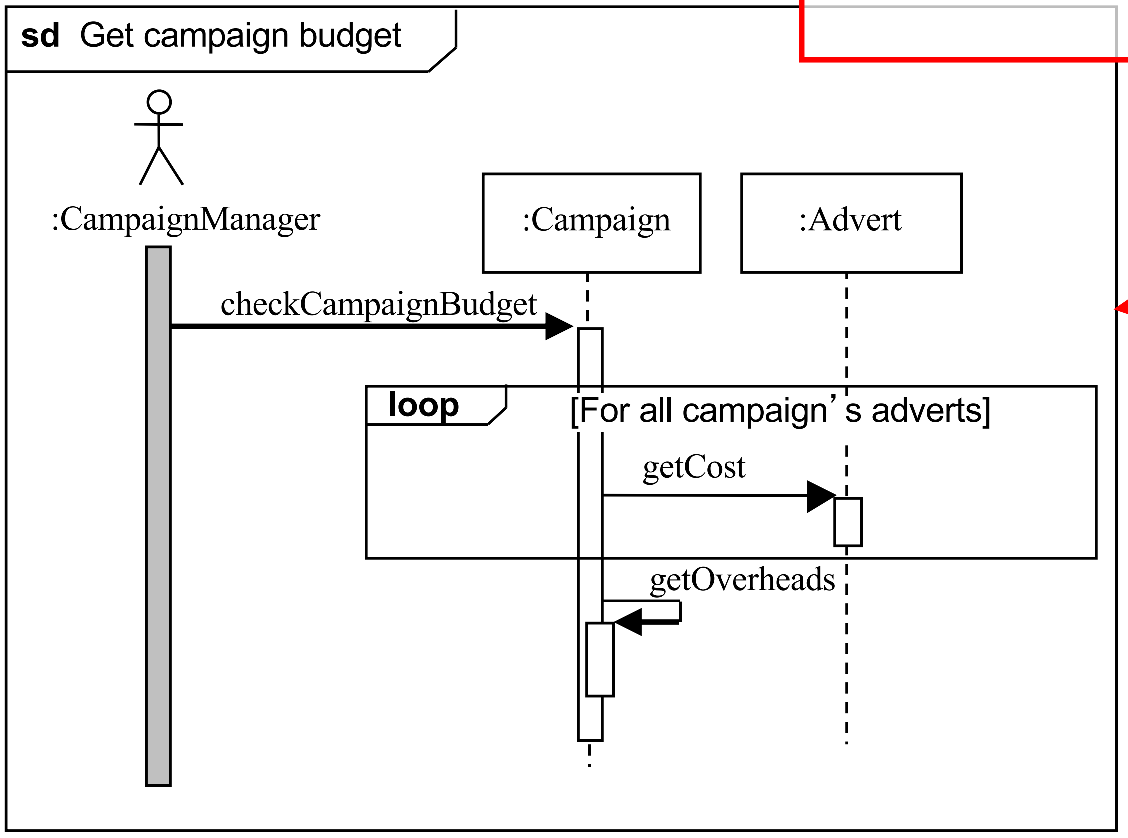
Interaction fragment that is referenced in `Check campaign budget` sequence diagram

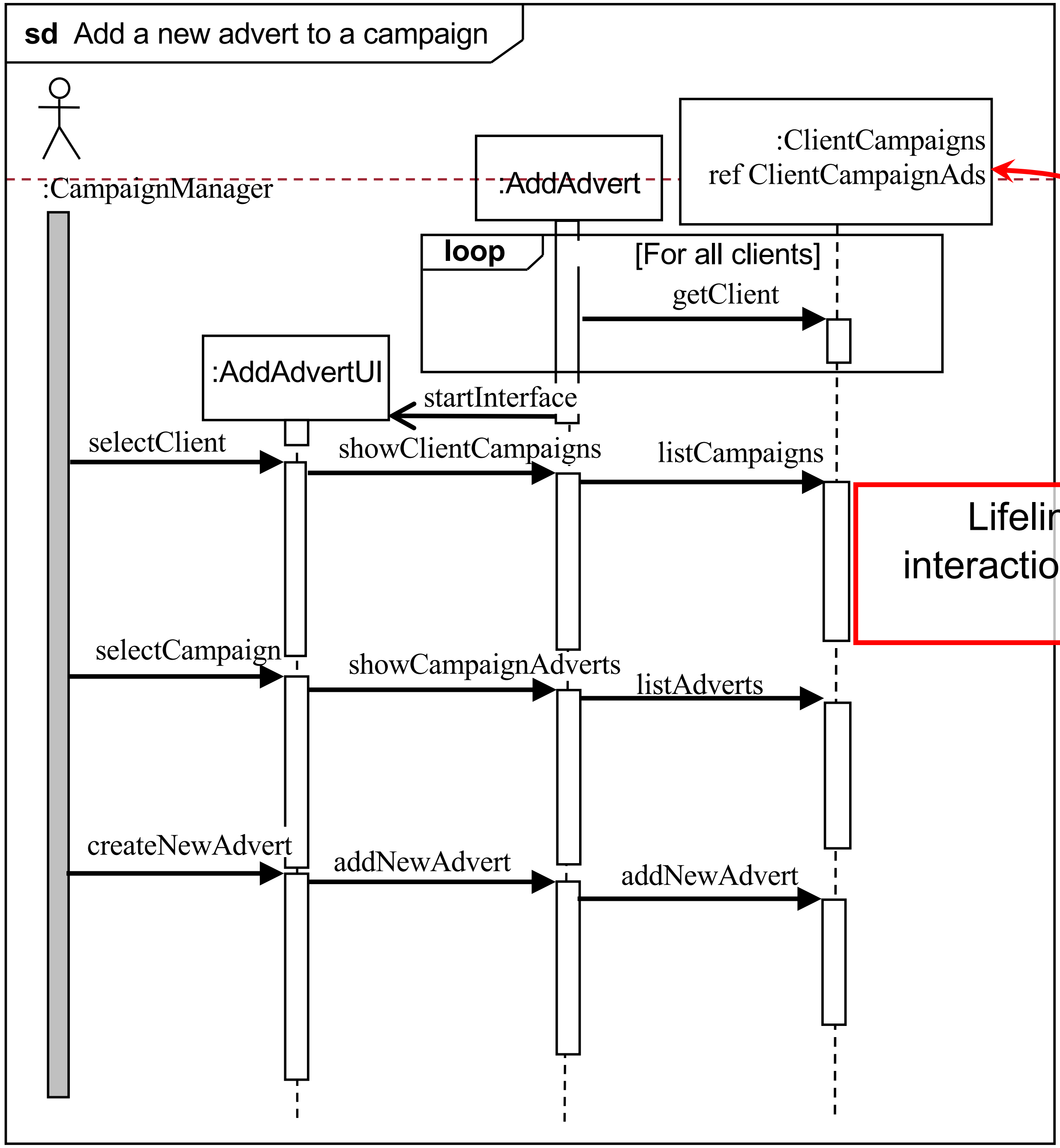


Gate showing the message enter this Interaction Fragment
sd List client campaigns
:Client
:Campaign
listCampaigns
loop
[For all client’s campaigns]
getCampaignDetails

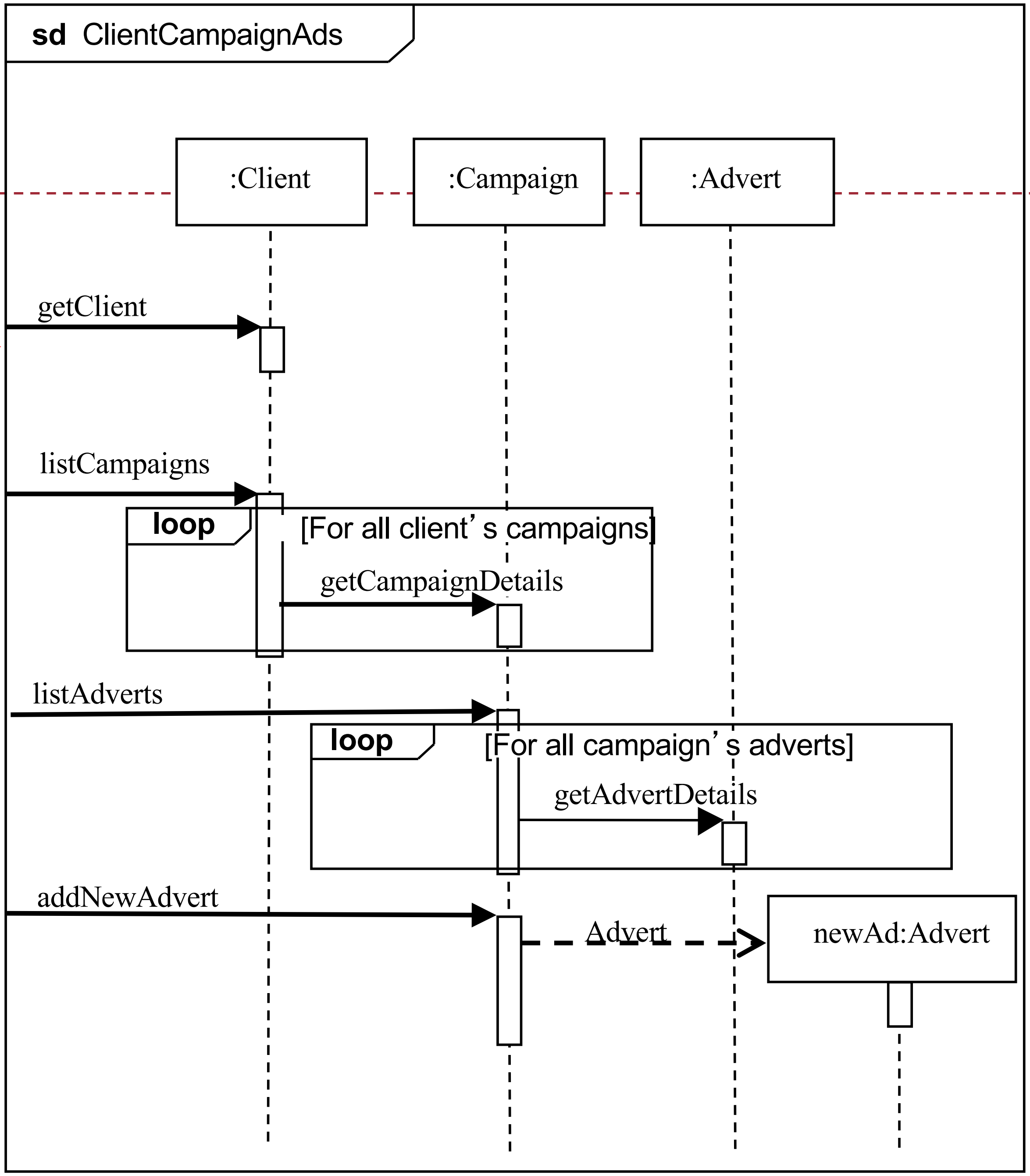
Interaction Fragment

Interaction fragment that is also referenced in `Check campaign budget` sequence diagram





Lifeline representing the interaction between a group of objects



Sequence diagram referenced in the `Add a new advert to a campaign` sequence diagram

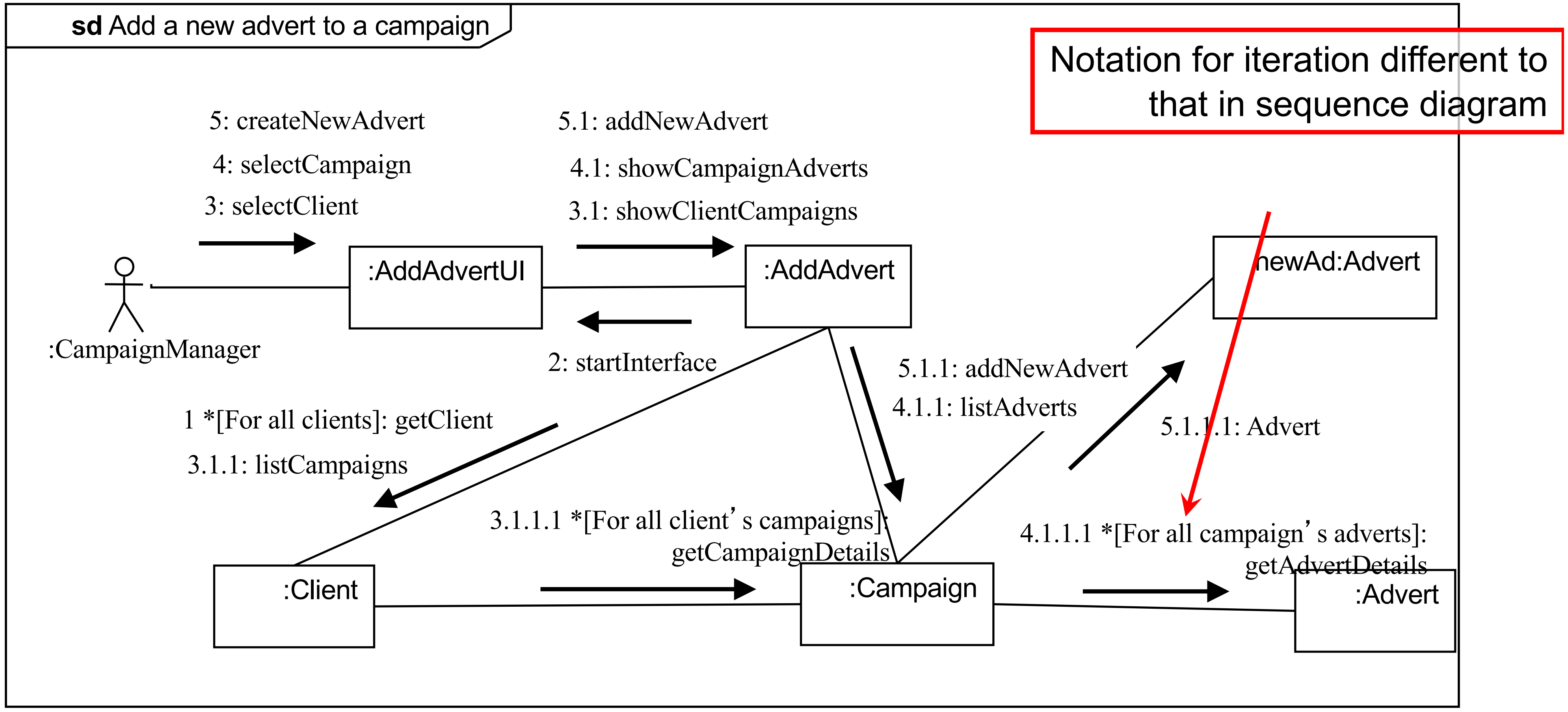
Communication diagrams

Communication (or Collaboration) Diagrams

- Hold the same information as sequence diagrams.
- Show links between objects that participate in the collaboration.
- No time dimension, sequence is captured with sequence numbers.
- Sequence numbers are written in a nested style (for example, 3.1 and 3.1.1) to indicate the nesting of control within the interaction that is being modelled.

Communication Diagrams

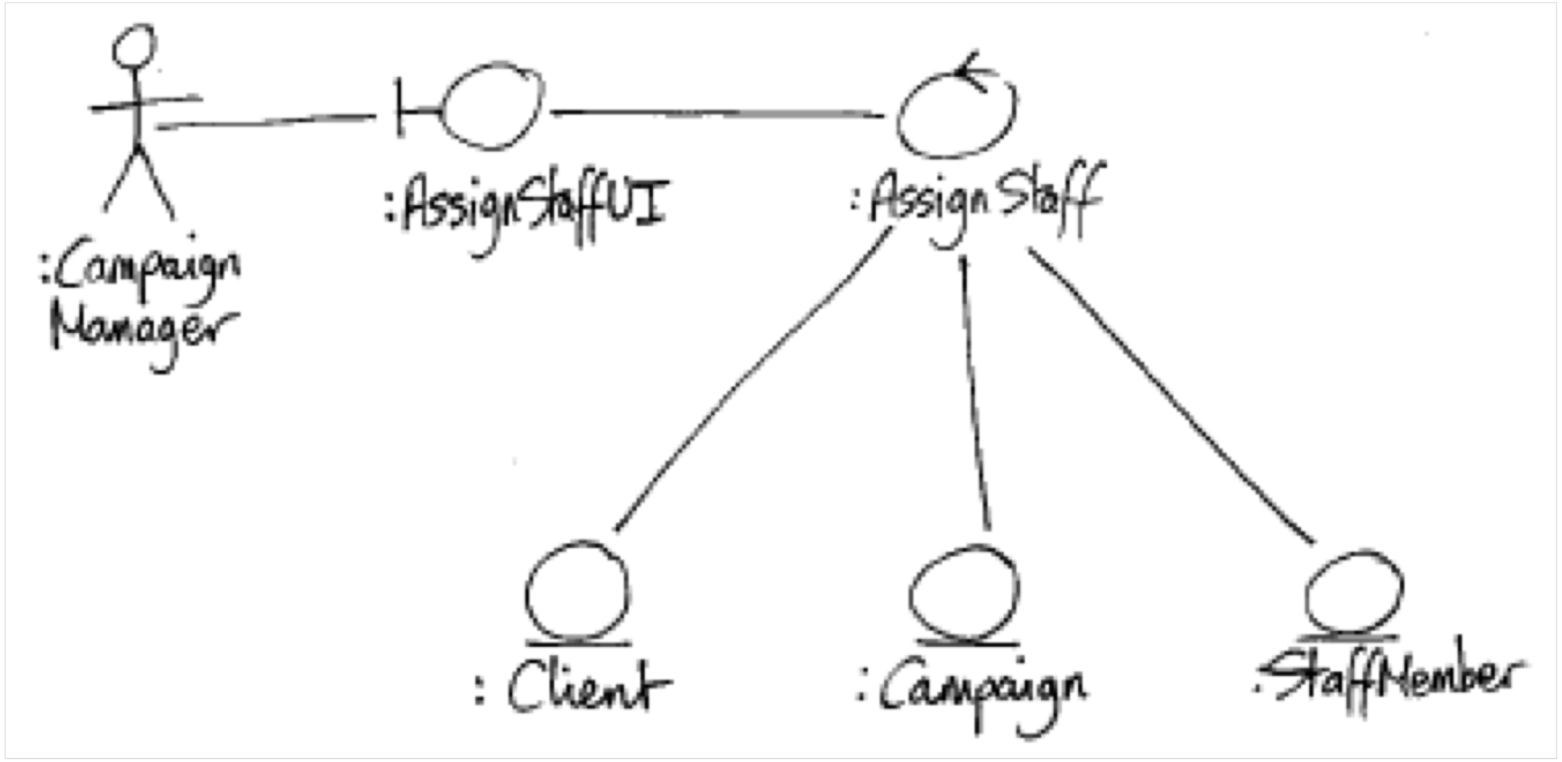
Compare with Sequence diagram on slide 17



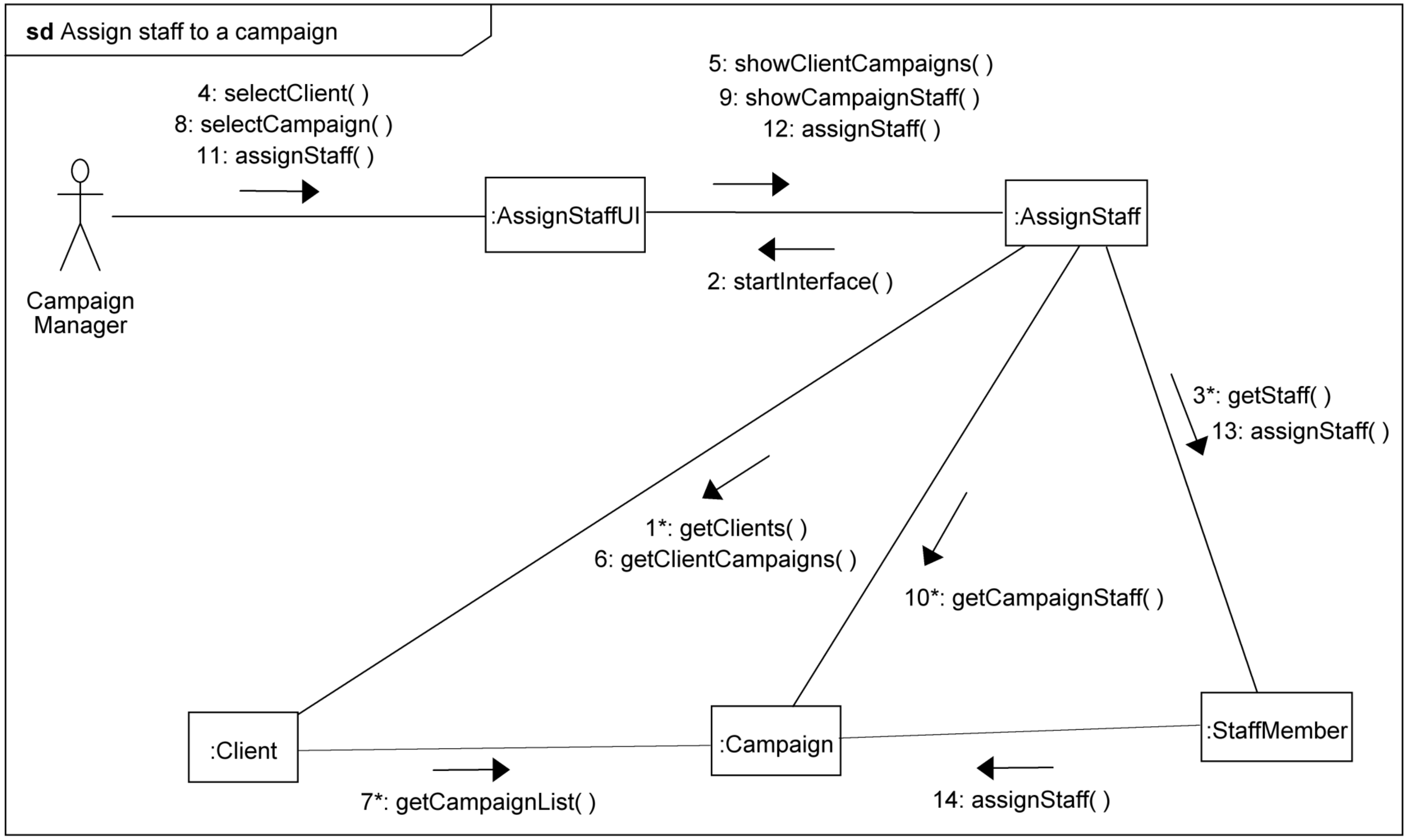
Communication diagram: Message Labels

Type of message	**Syntax example**
Simple message.	`4: addNewAdvert`
Nested call with return value. *The return value is placed in the variable* `name`.	`3.1.2: name = getName`
Conditional message. *This message is only sent if the condition* `[balance > 0]` *is true.*	`5 [balance > 0]: debit(amount)`
Iteration	`4.1 *[For all adverts]: getCost`

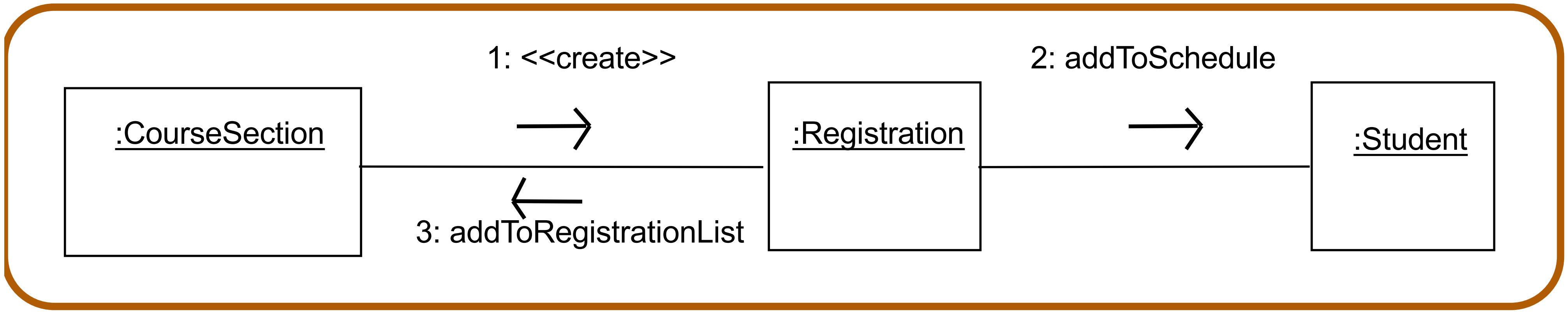
Early Draft Communication Diagram



More Developed Communication Diagram



Collaboration diagrams: possible implementations

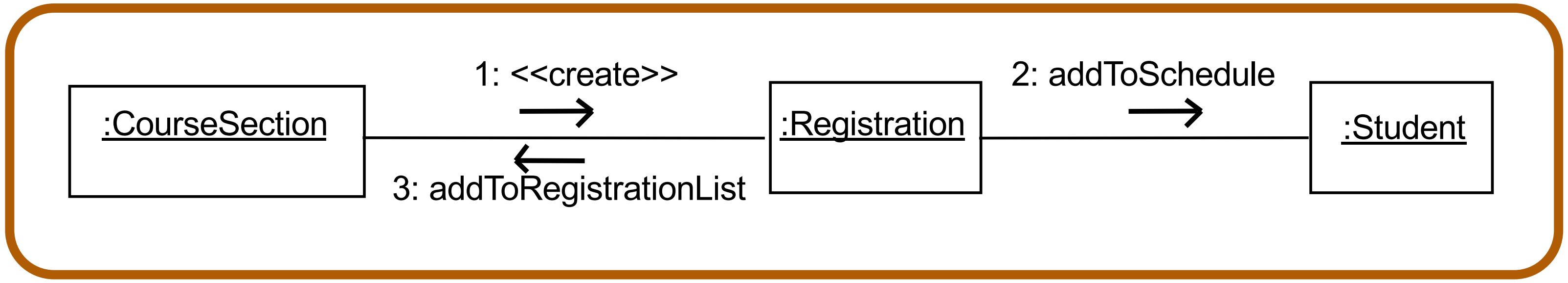


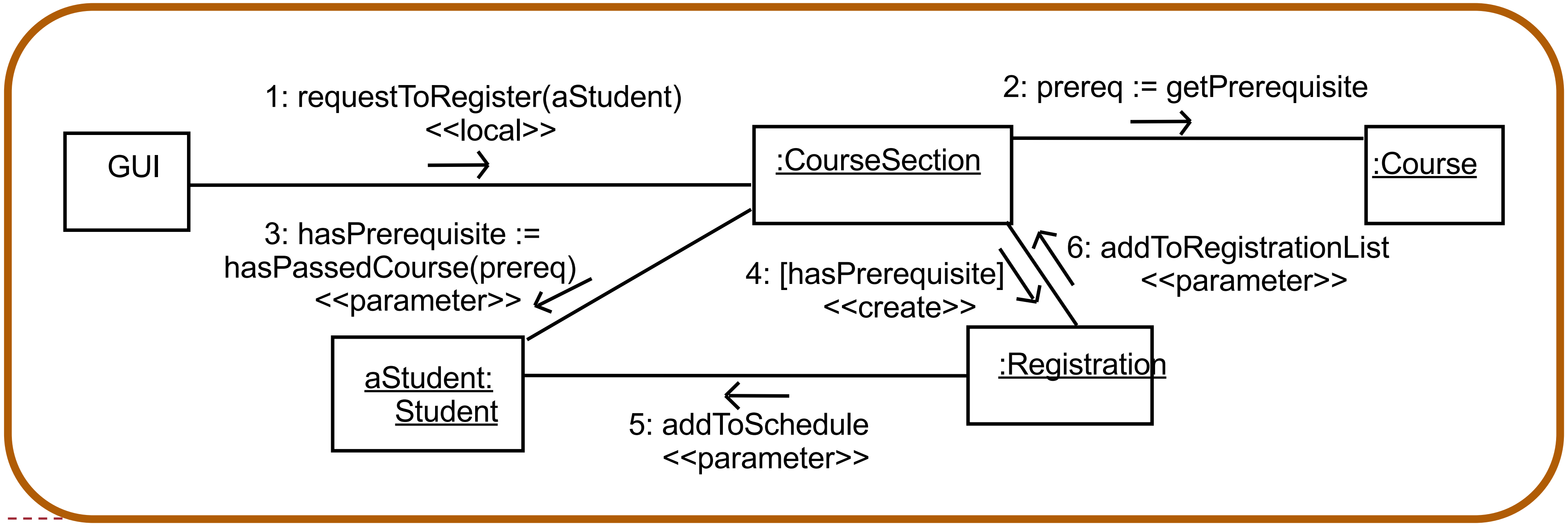
```
public class courseSection {
  void someMethod(Student s)
    Registration r = new Registration(this, s);
    ...
```

```
public class Registration {
  Registration(c CourseSection, Student s) {
    student.addtoSchedule();
    c.addtoRegistrationlist(this);
    ...
```

See slides 13-14 on Model Consistency

Collaboration diagrams: abstraction and refinement





[Lethbridge & Laganière 2001]

Summary

- Objects collaborate to provide system functionality
- Two approaches which model the same thing:
  - Sequence Diagrams
  - Collaboration/Communication Diagrams
- Guidelines for producing Sequence diagrams (will be the same for Collaboration diagrams
- Need to check the consistency between diagrams
- Can be used to create the class diagram
- Can generate the initial code for each Class based on these diagrams

Exercises

- Write a set of Java classes and their methods that implement some of the diagrams in this presentation

Further reading

- Object-Oriented Software Engineering: Practical Software Development using UML and Java, Second Edition, Timothy C. Lethbridge and Robert Laganière, McGraw Hill, 2001
- See Chapter 9 in Bennett