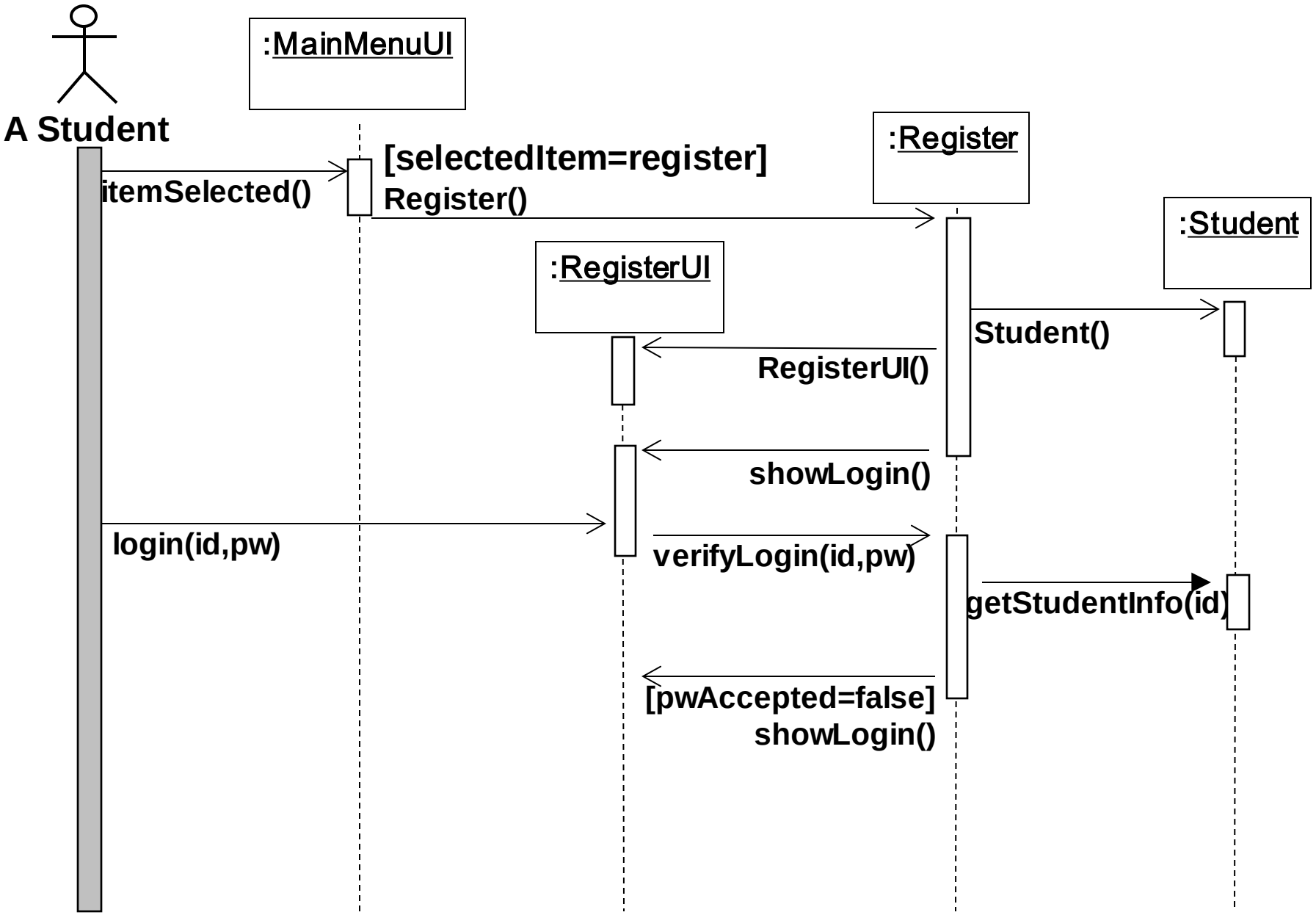
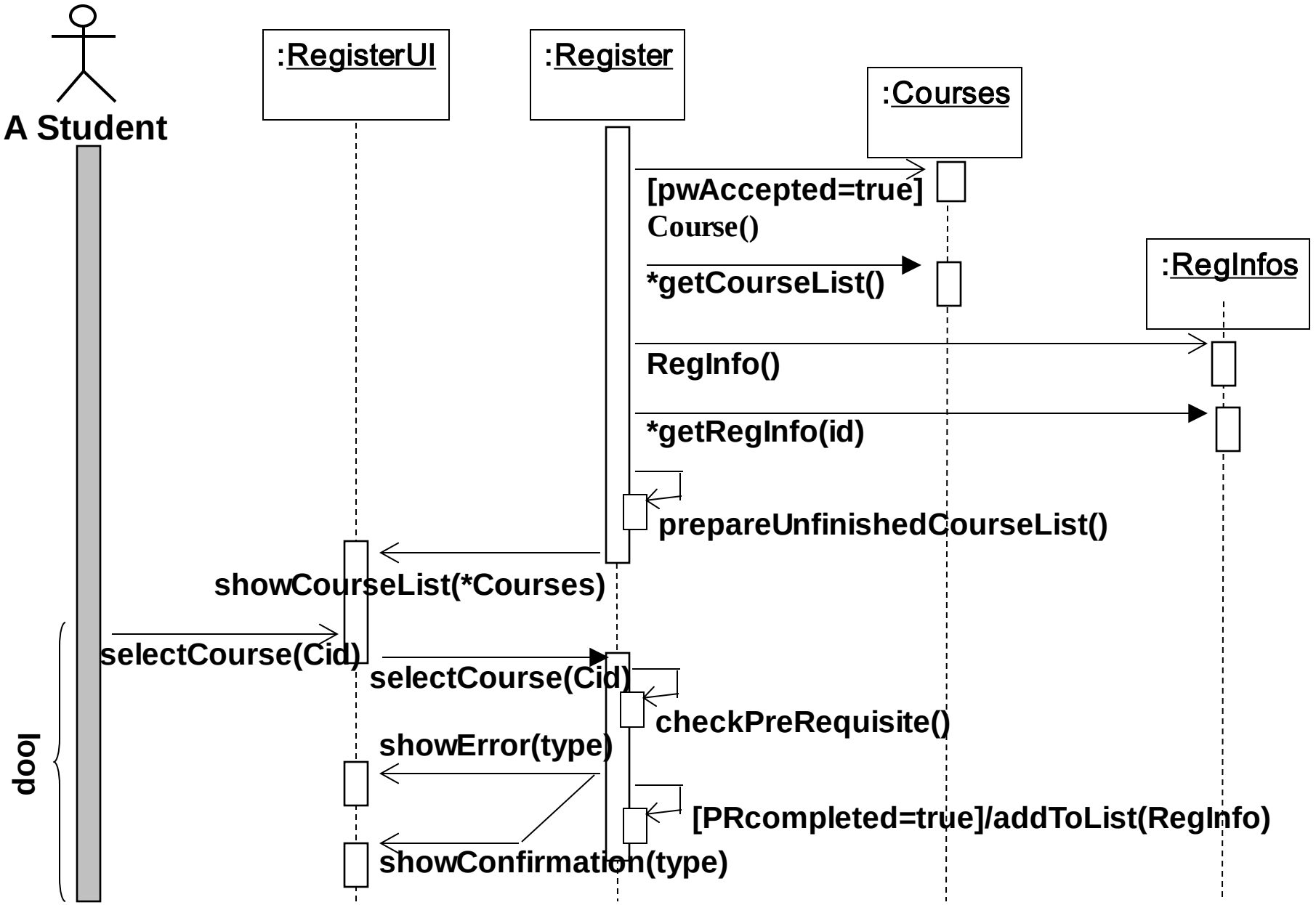
Register: typical course of events

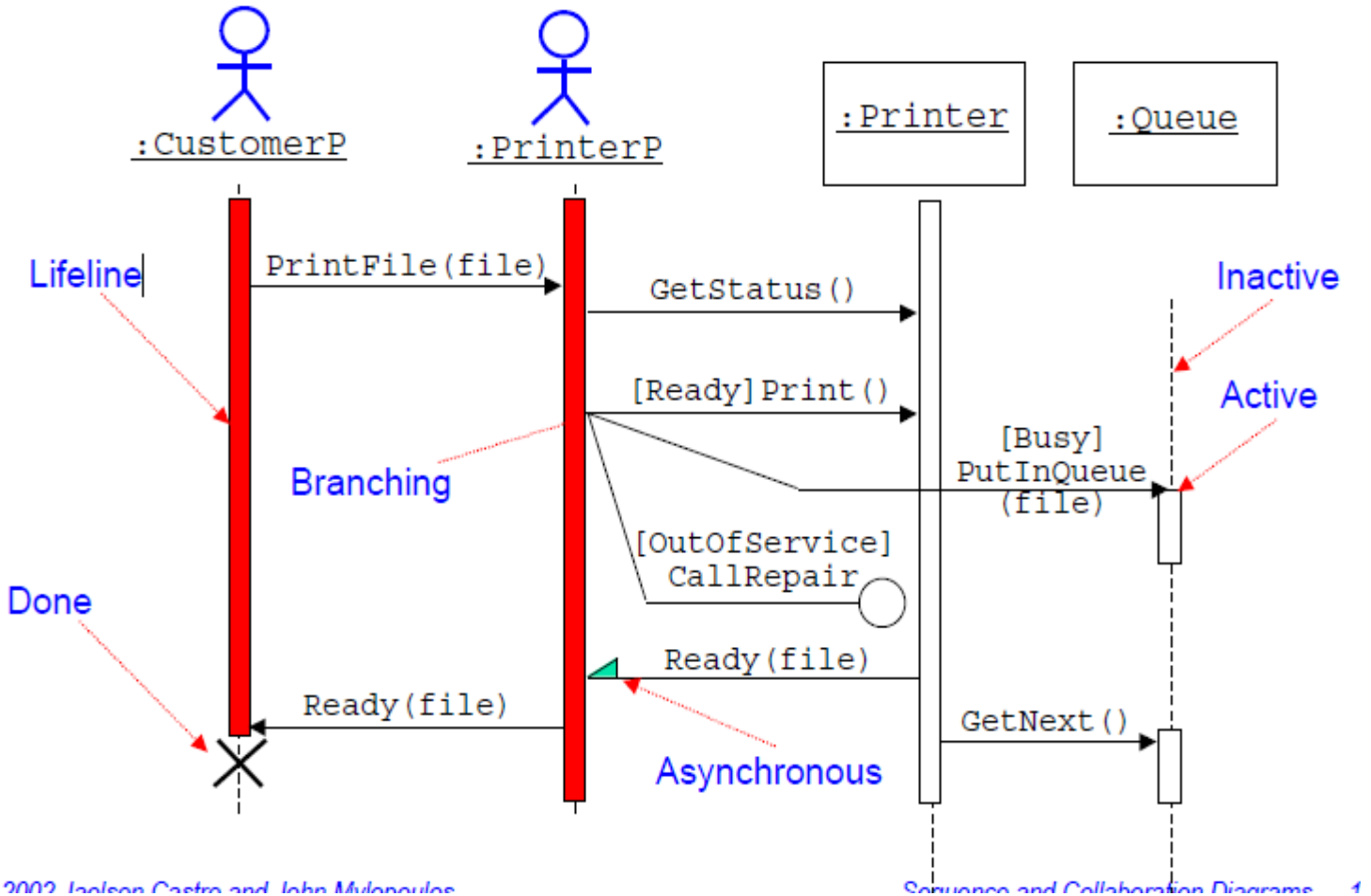
Actor Action (Student)	Actor Action (Adviser)	System Response
1. Open registration Interface		2. Ask for Student Id, Password
3. Provide Id, Password		4. Show List of unfinished courses
5. Select a course		6. Get Pre-requisite list for that course
		7. Check results of those courses
		8. If not passed, then give message to student and deselect the course Otherwise, do nothing
9. Finalize selection		10. Check total credit hour
		11. If ok, then proceed to next step otherwise, give a message requesting the student to revise his selection
		12. Notify Adviser
	13. Ask for details	14. Show details
	15. Authorize	16. Update state
		17. update registration information for selected courses and show confirmation message, Print course card

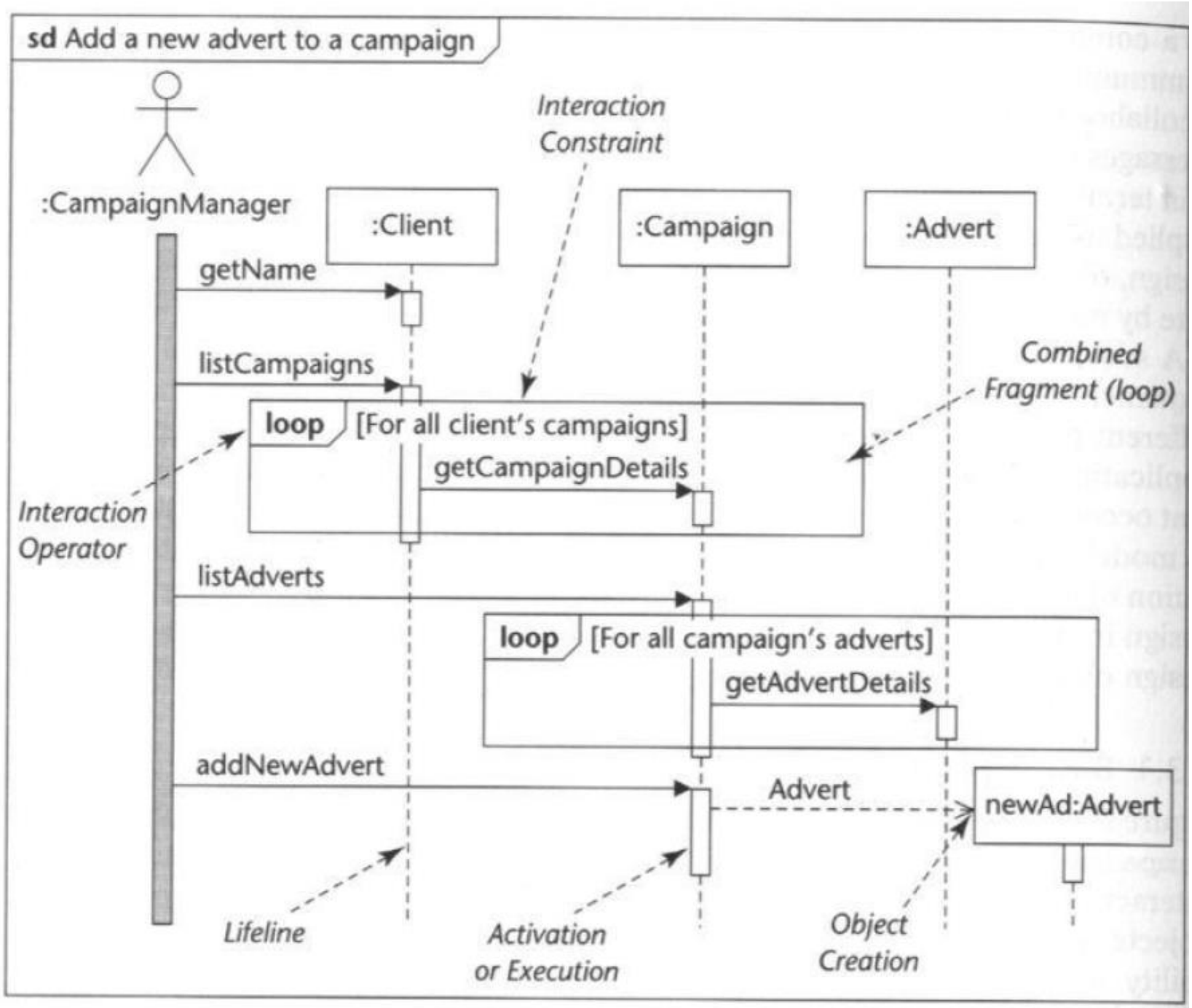
Register: Sequence Diagram 1



Register: Sequence Diagram 2






sd Add a new advert to a campaign
:CampaignManager
Interaction Constraint
:Client
:Campaign
:Advert
getName
listCampaigns
Combined Fragment (loop)
loop
[For all client's campaigns]
getCampaignDetails
Interaction Operator
listAdverts
loop
[For all campaign's adverts]
getAdvertDetails
addNewAdvert
Advert
newAd:Advert
Lifeline
Activation or Execution
Object Creation

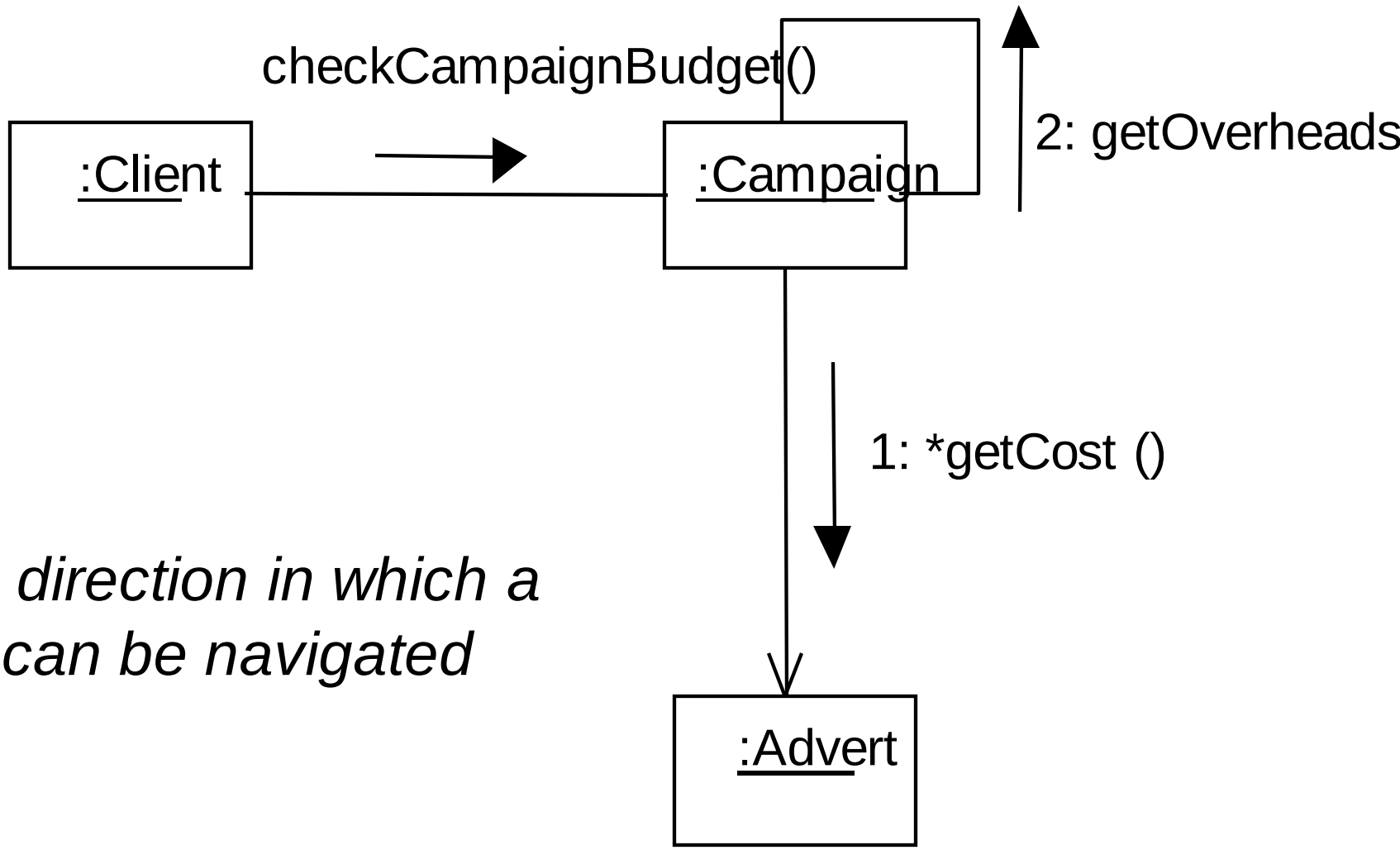
Collaboration Diagrams

- 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

Message Labels

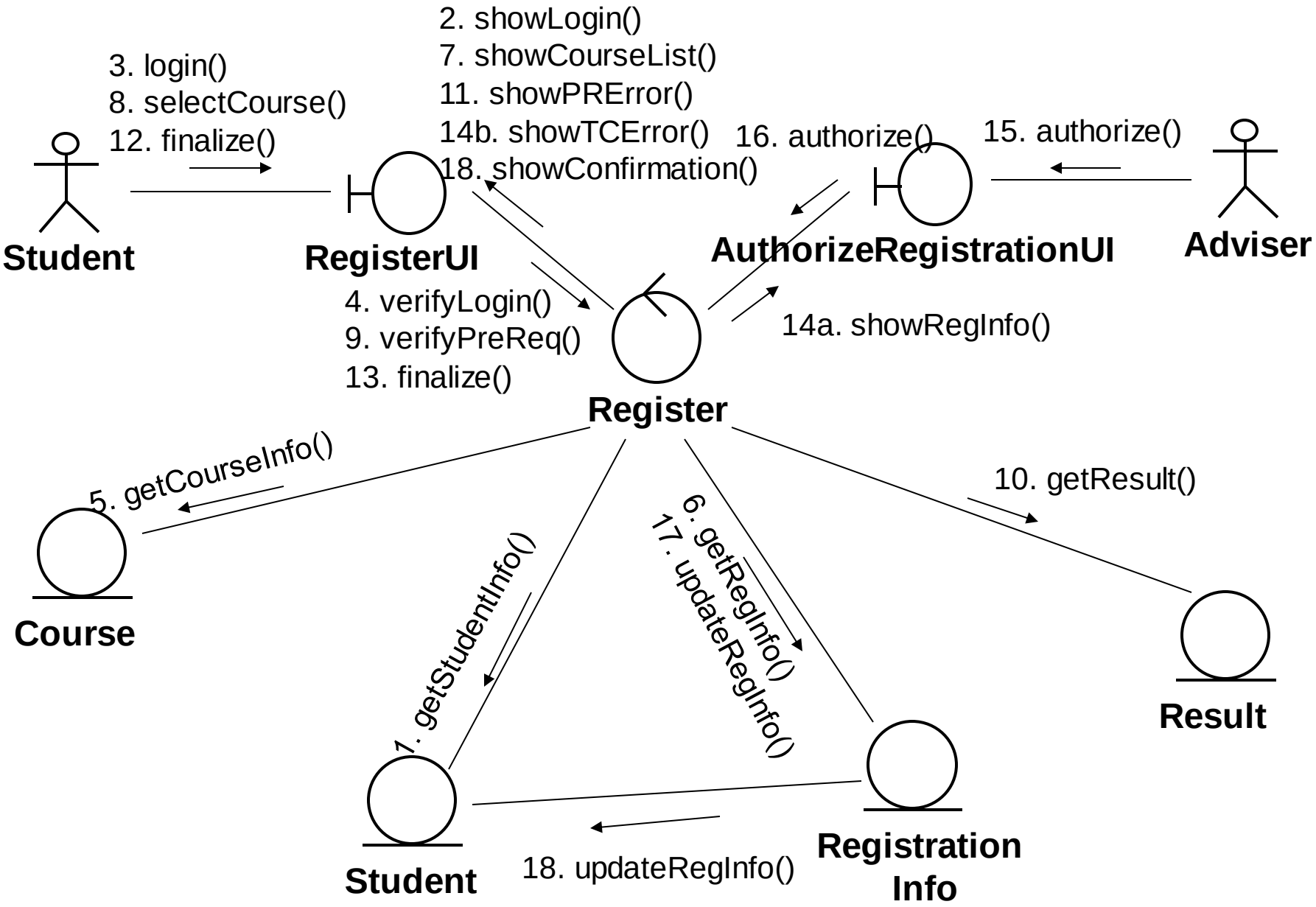
Type	Example
• **Simple message.**	4: addNewItem()
•**Nested call with return value.** –The return value is placed in the variable name.	3.1.2: name:= getName()
•**Conditional message.** –This message is only sent if the condition [balance > 0] is true.	[balance > 0] 5: debit(amount)
•**Synchronization with other threads**. –*Message* 4: playVideo() *is invoked only once the two concurrent messages* 3.1a and 3.1b *are completed.*	3.1a, 3. 1b / 4:playVideo()

Navigating Links



The direction in which a link can be navigated

Register: Collaboration Diagram



Model Consistency

- The allocation of operations to objects must be consistent with the class diagram and the message signature must match that of the operation

- Every sending object must have the object reference for the destination object
  - Either an association exists between the classes or another object passes the reference to the sender
  - Message pathways should be carefully analysed