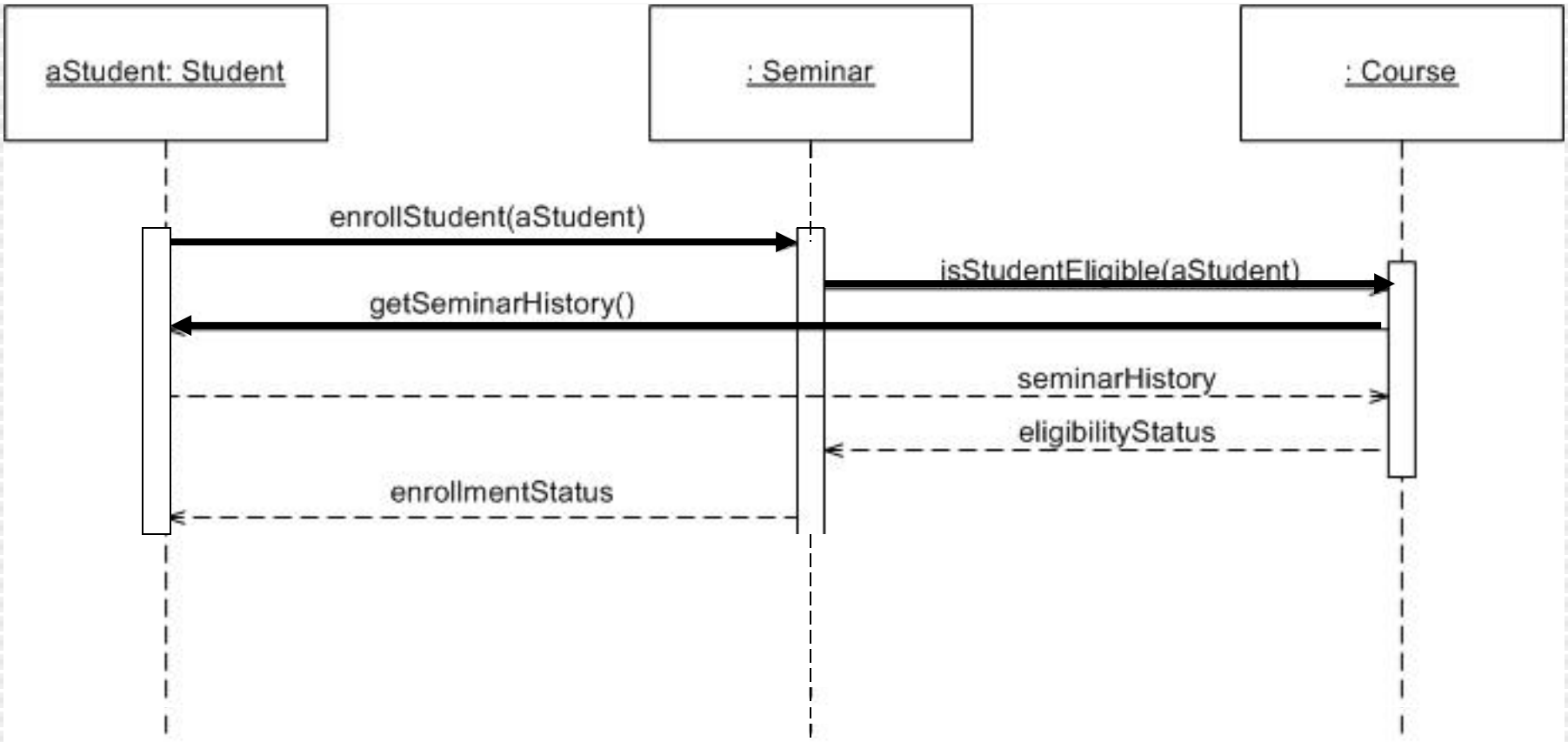
# Sequence Diagrams

# Sequence Diagrams

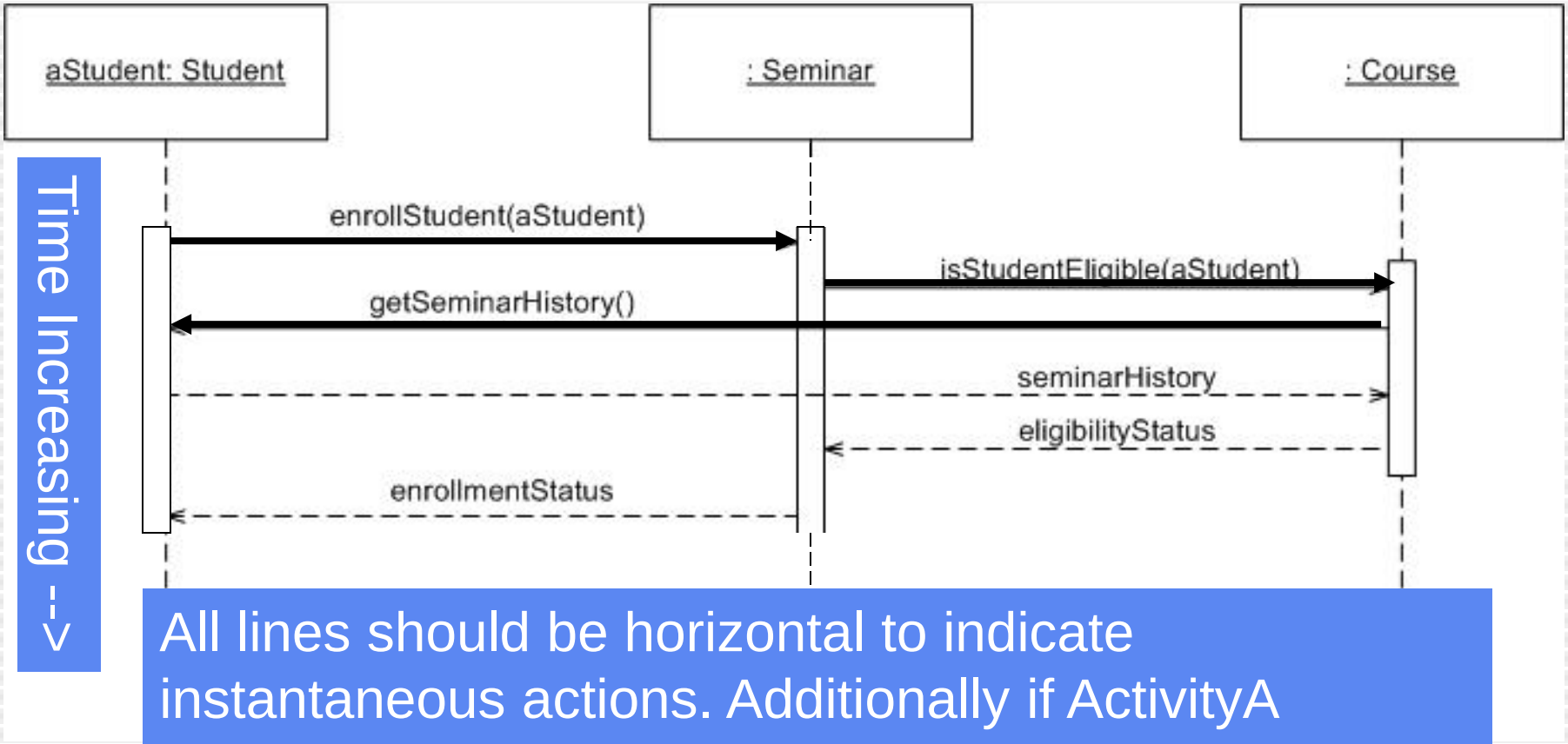
- Describe the flow of messages, events, actions between objects
- Show concurrent processes and activations
- Show time sequences that are not easily depicted in other diagrams
- Typically used during analysis and design to document and understand the logical flow of your system

Emphasis on time ordering!

# Sequence Diagram



# Sequence Diagram

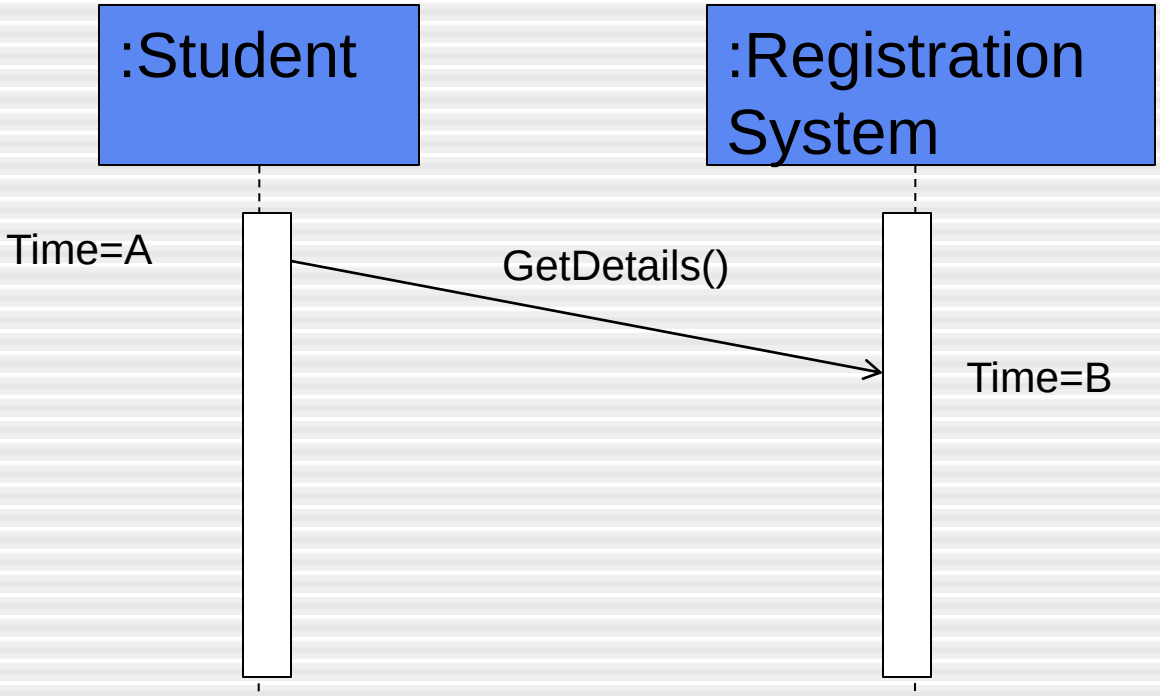



All lines should be horizontal to indicate instantaneous actions. Additionally if ActivityA happens before ActivityB, ActivityA must be above activity A

**Lower = Later!**

Coming up: Components

# Diagonal Lines

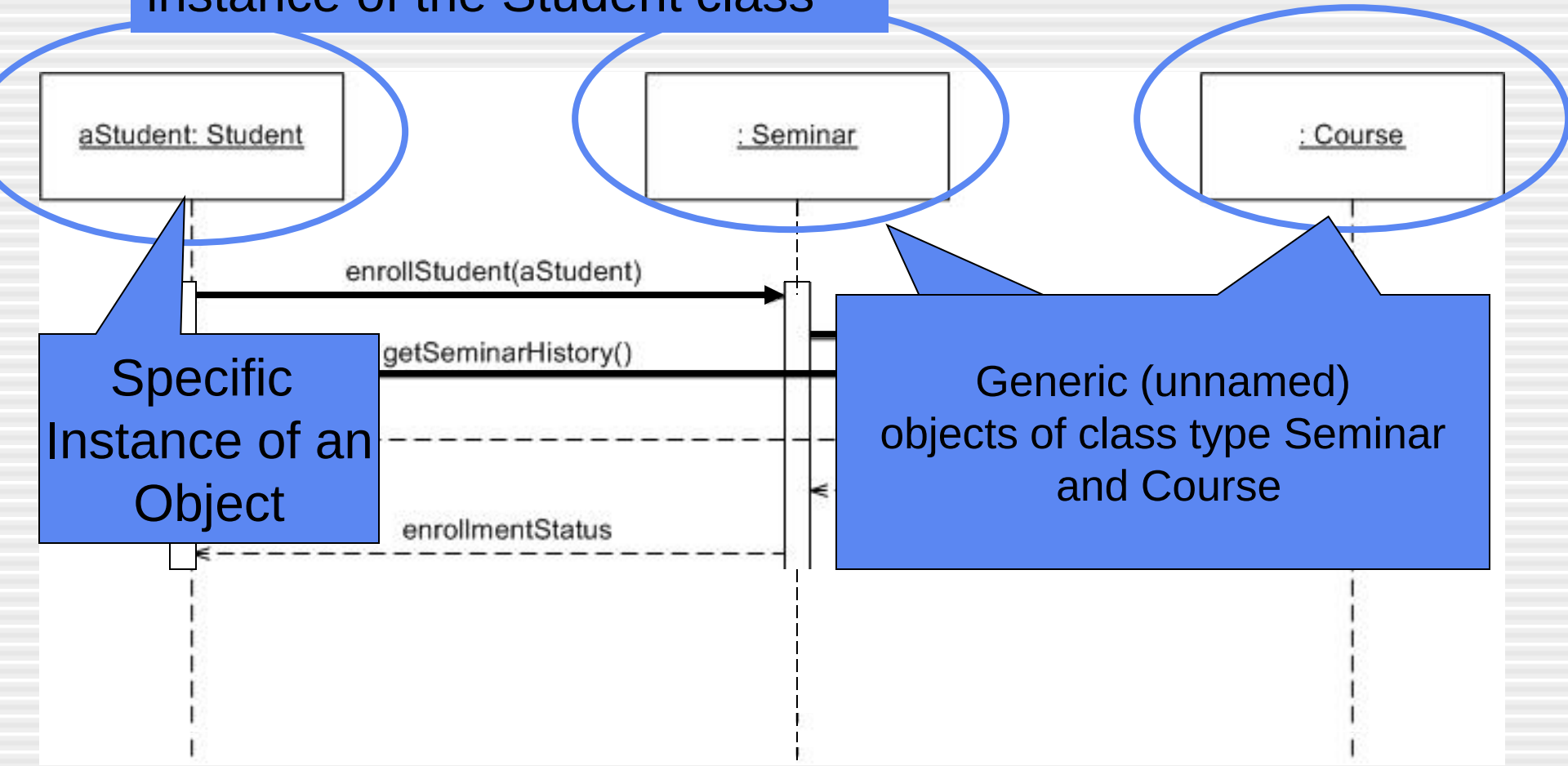



- What does this mean?

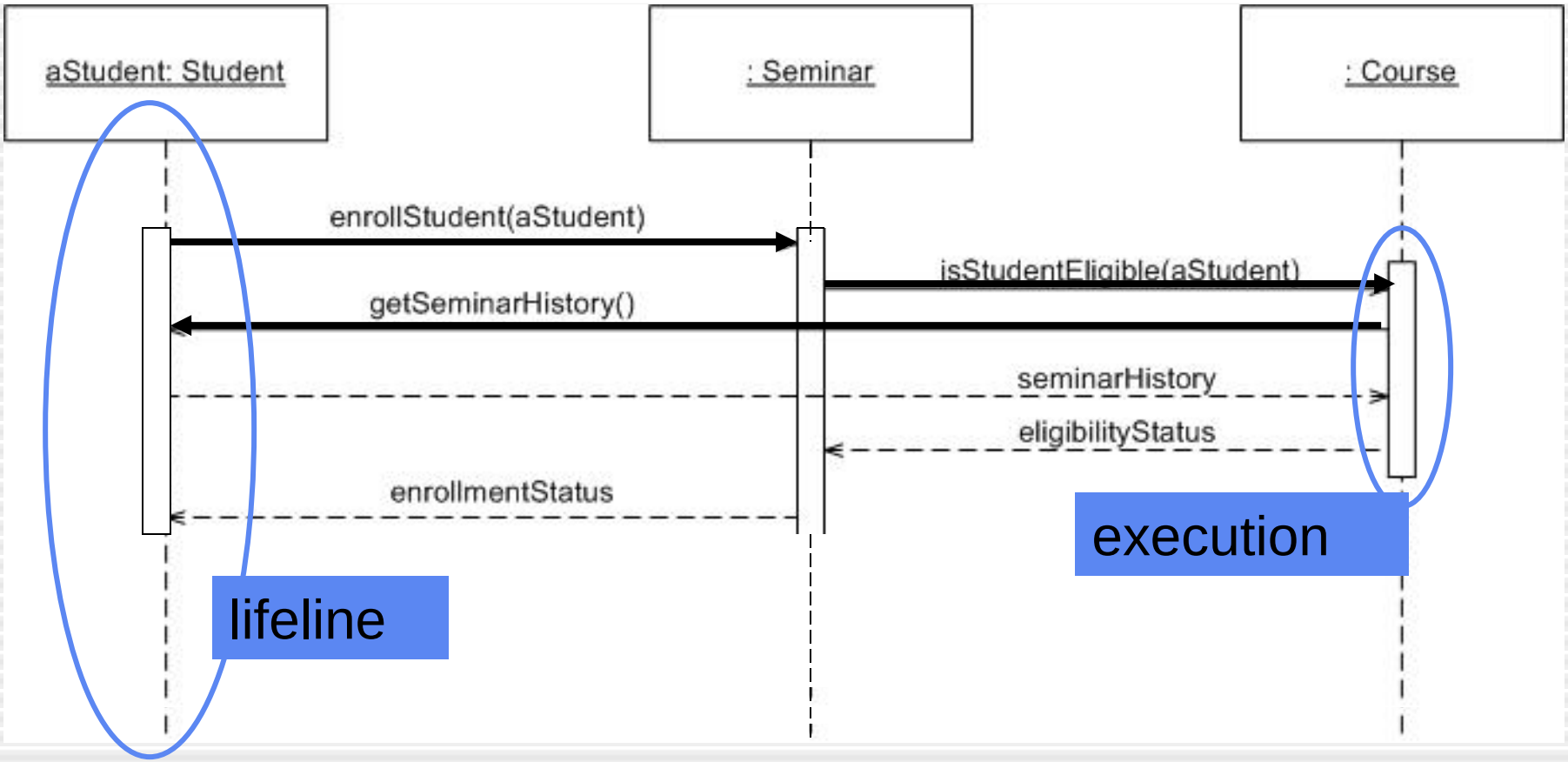
Do you typically care?

# Components

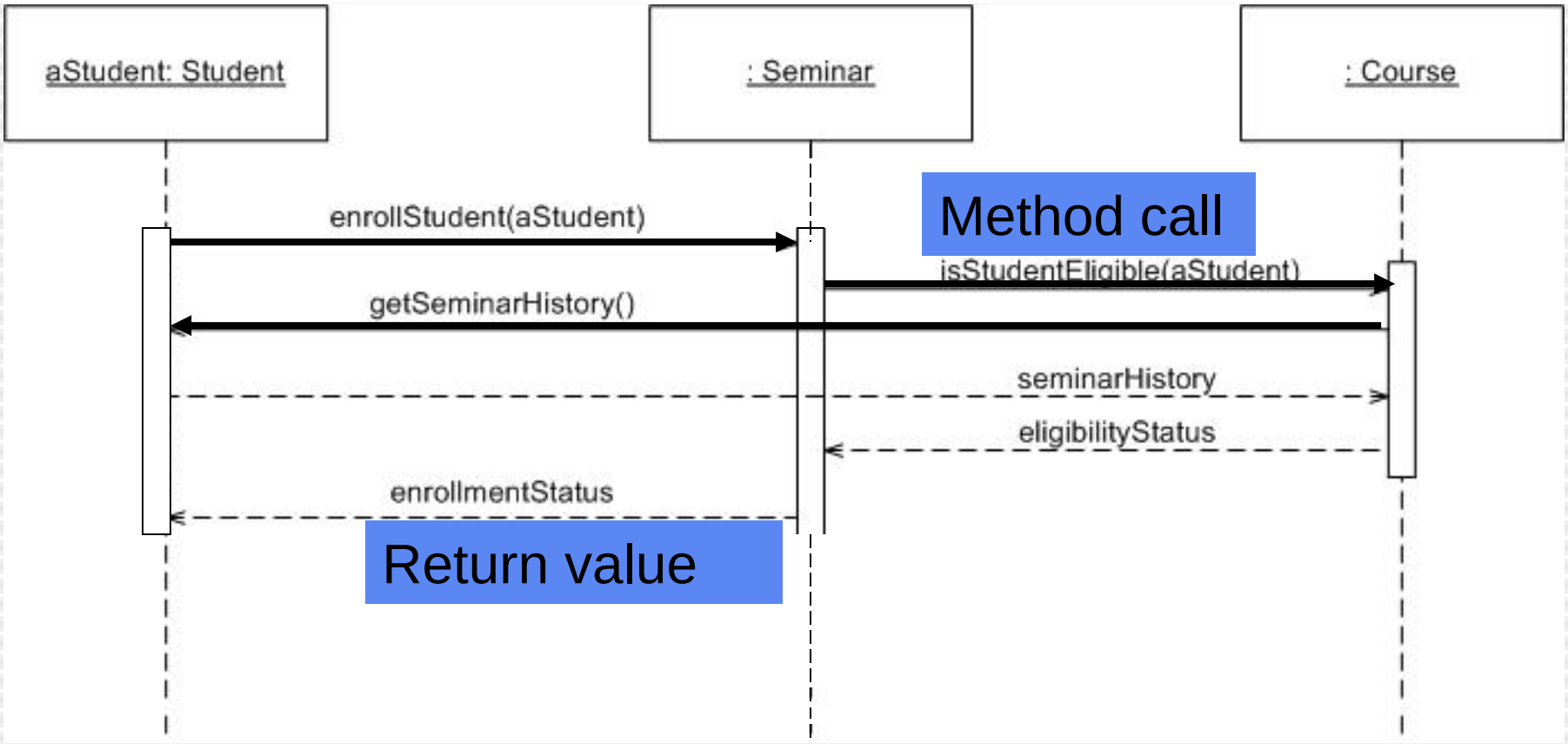
Objects: aStudent is a specific instance of the Student class



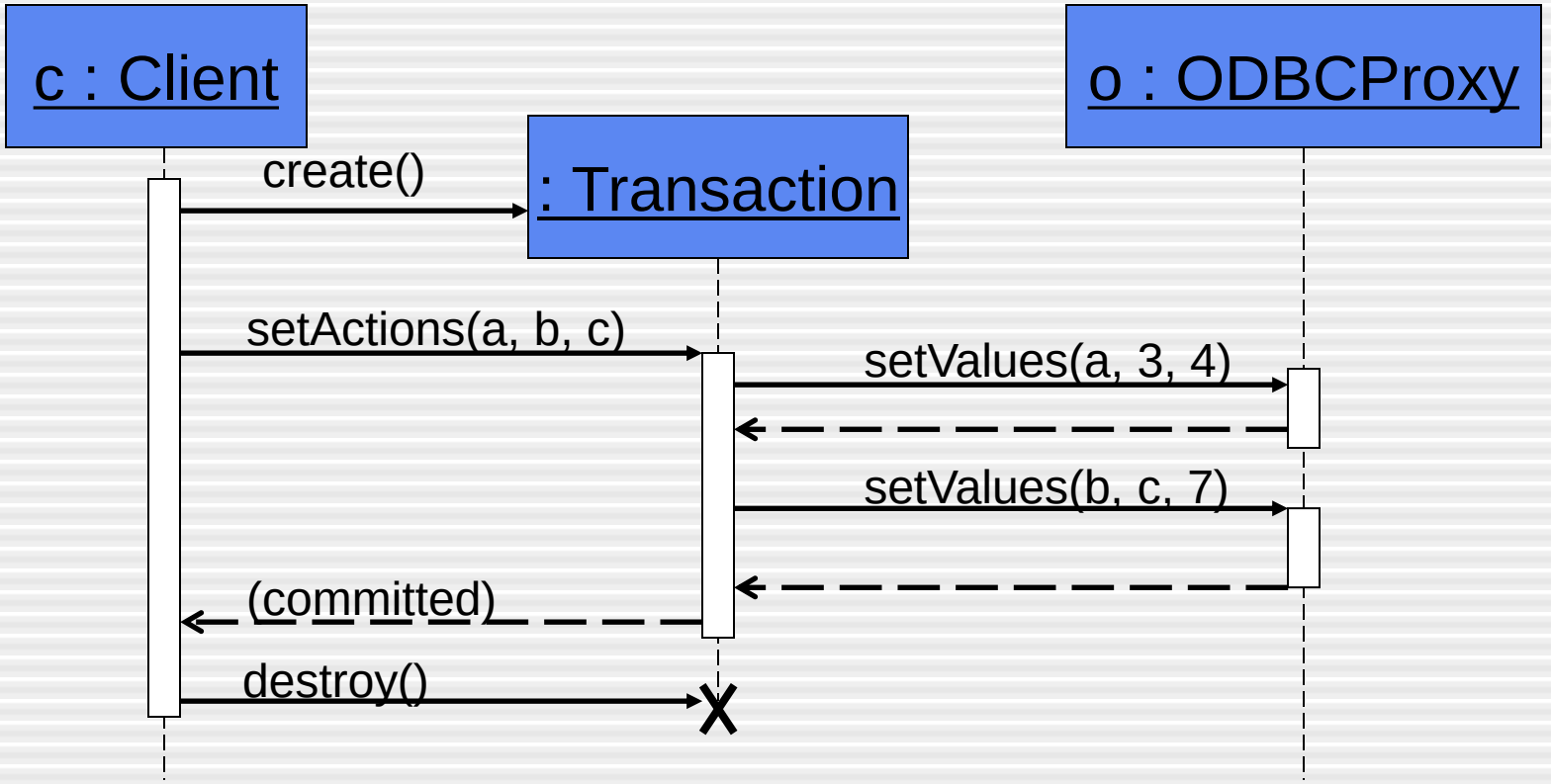
# Components


aStudent: Student
: Seminar
: Course
enrollStudent(aStudent)
isStudentEligible(aStudent)
getSeminarHistory()
seminarHistory
eligibilityStatus
enrollmentStatus
lifeline
execution

# Components


aStudent: Student
: Seminar
: Course
enrollStudent(aStudent)
isStudentEligible(aStudent)
getSeminarHistory()
seminarHistory
eligibilityStatus
enrollmentStatus
Method call
Return value

# Components



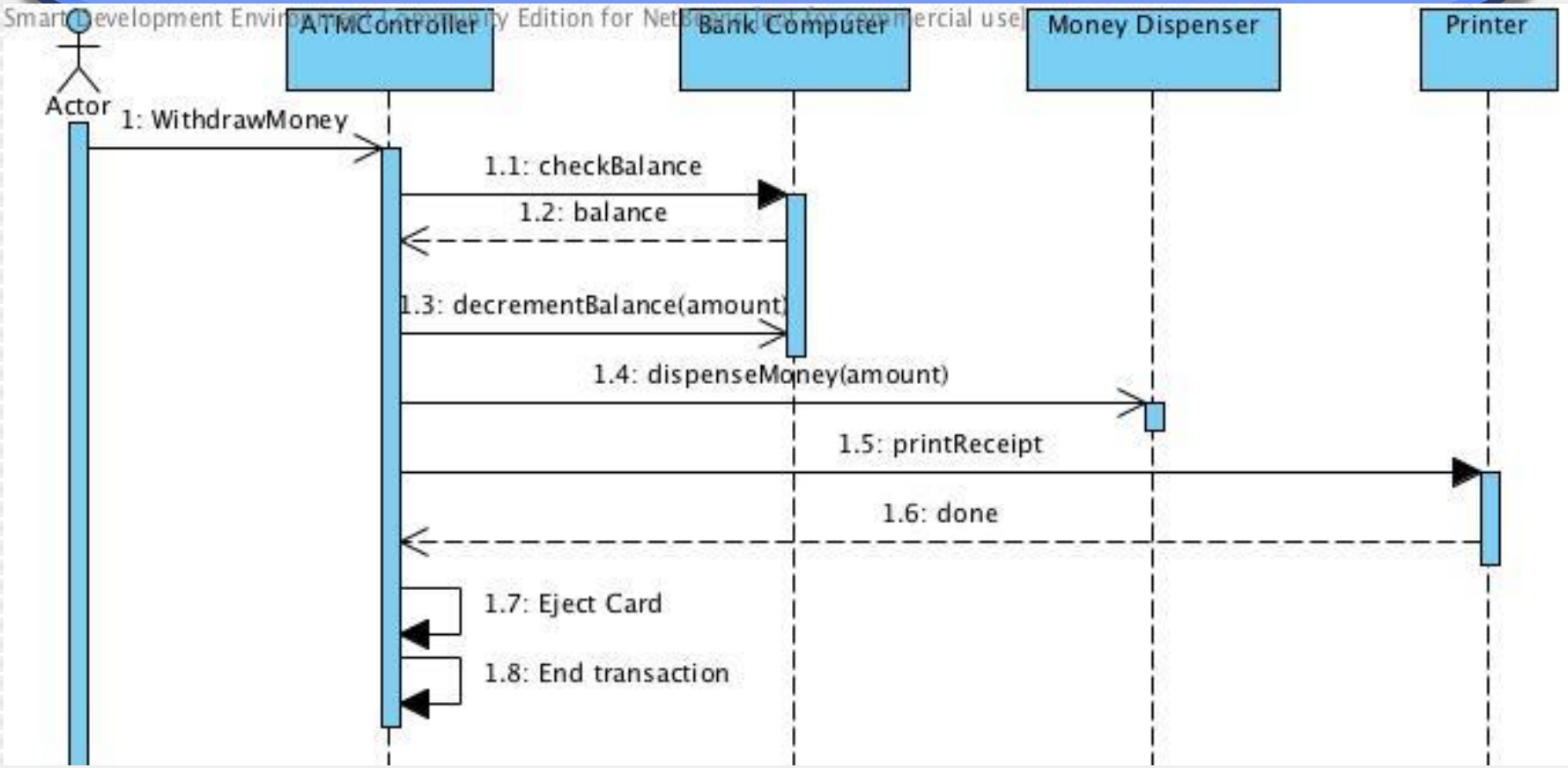

create()
destroy()

Synchronous message

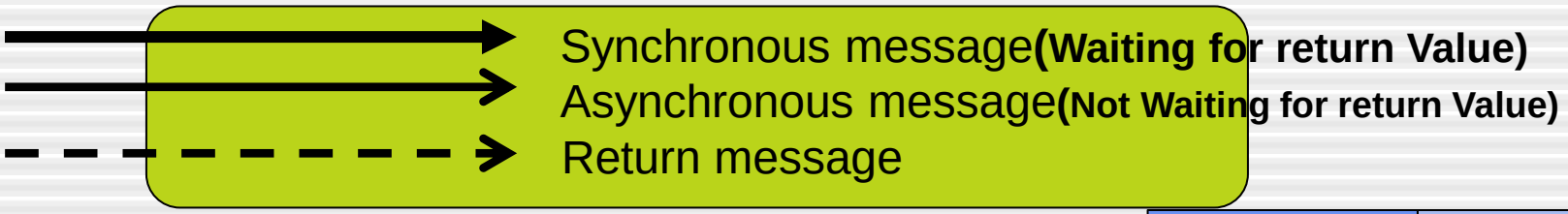
Asynchronous message

Return message

# Async Message Example

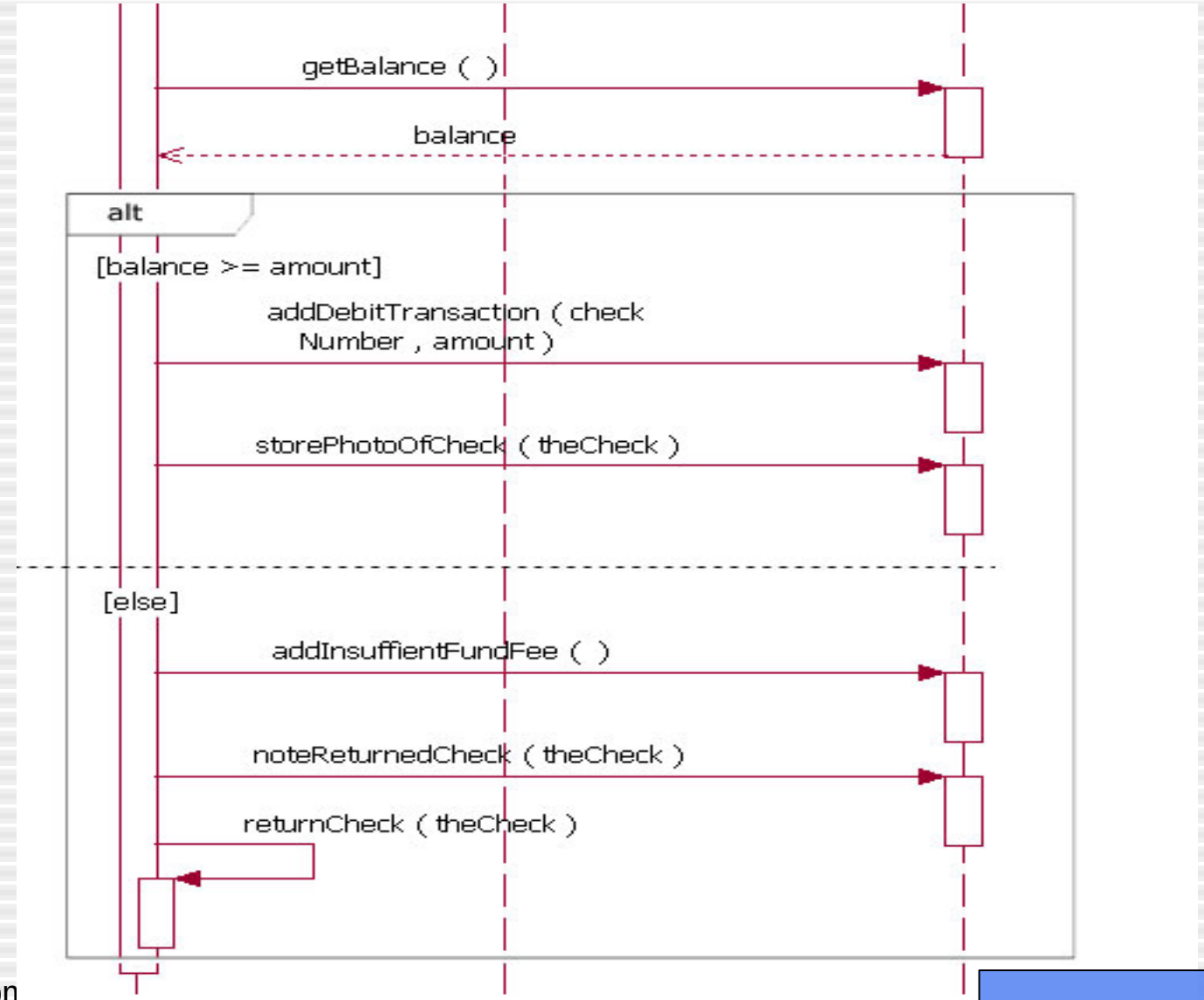



There are problems here… what are they?

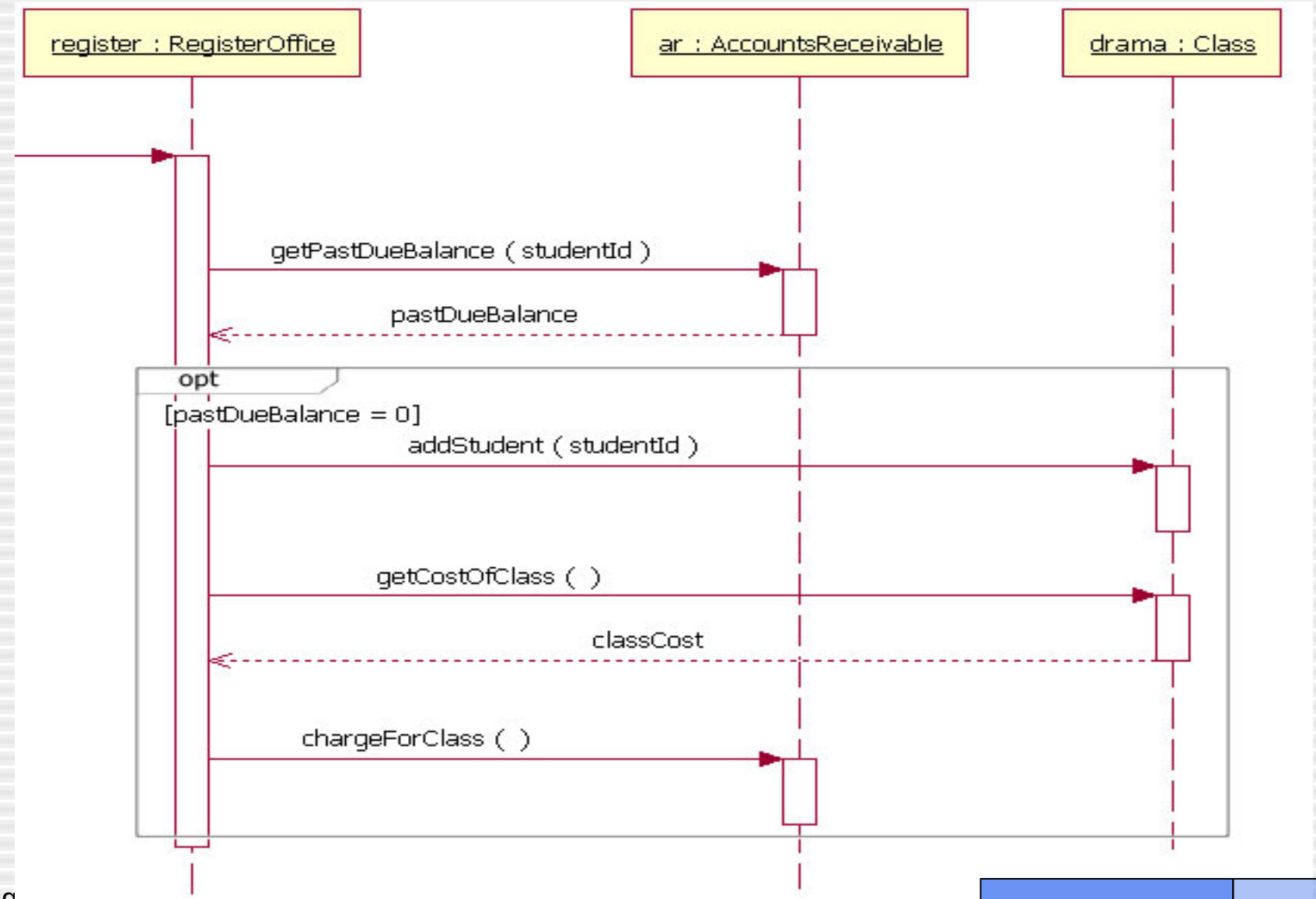



Coming up: Components: Guards

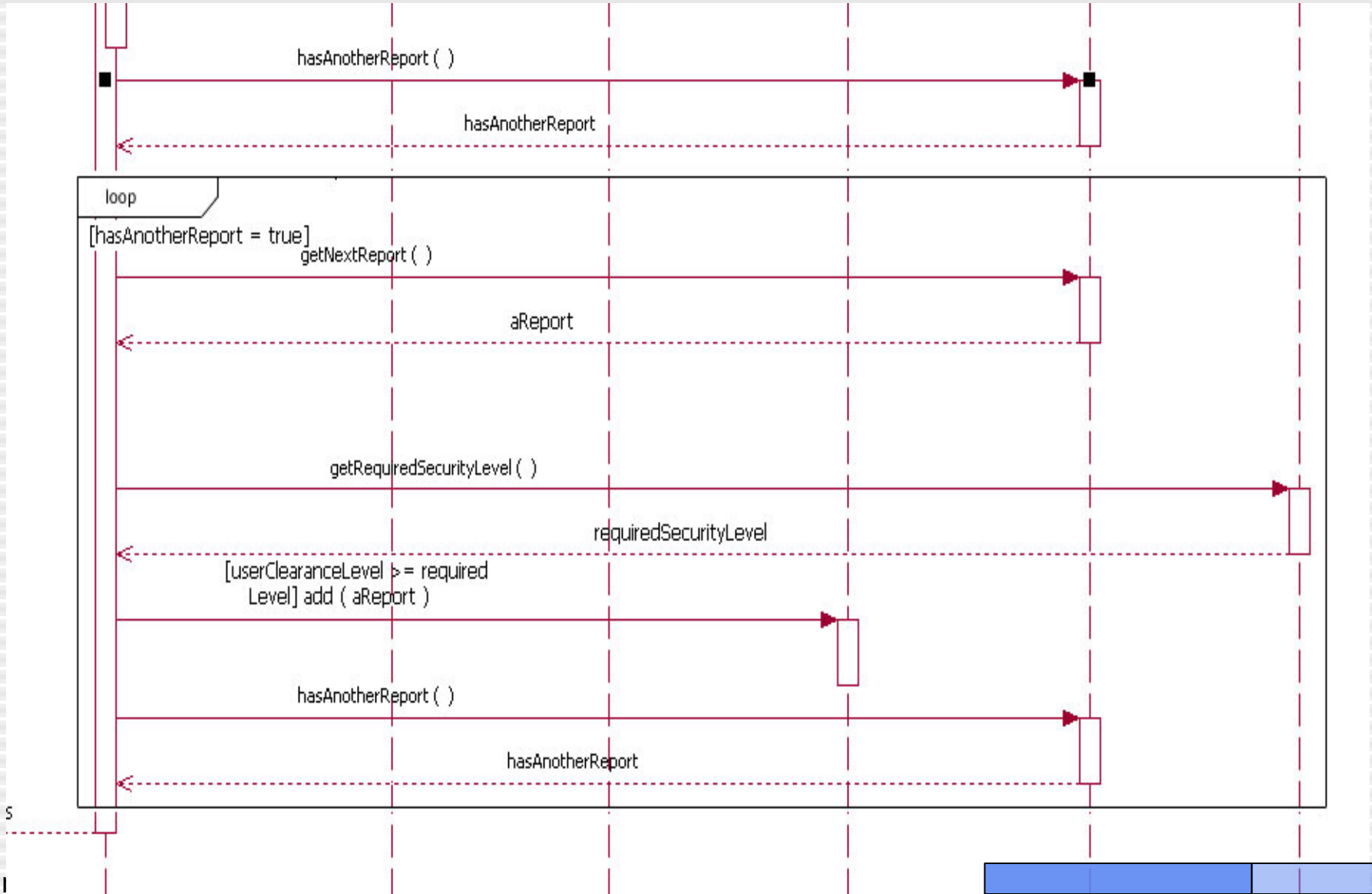
# Components: alt/else



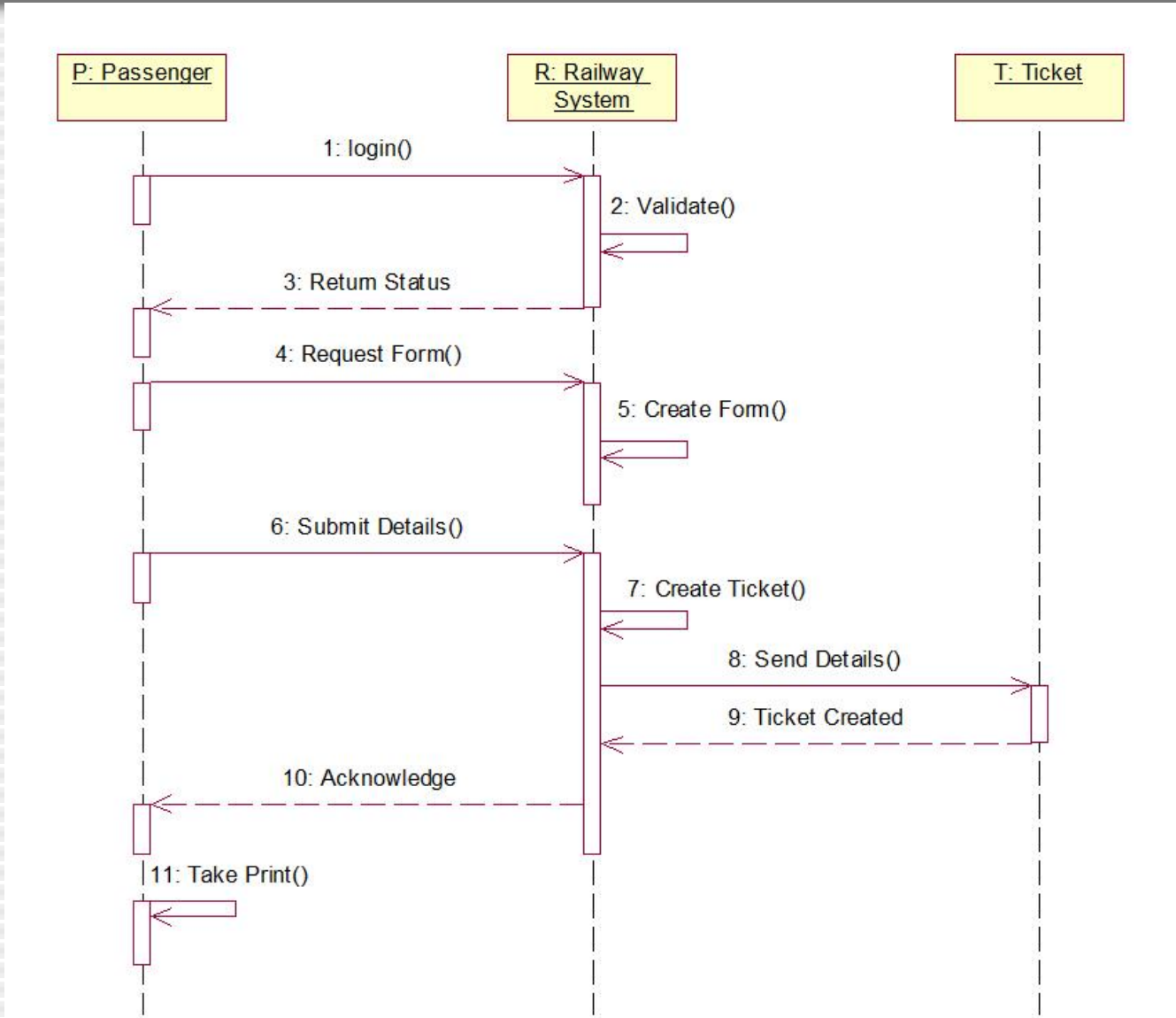
# Components: option



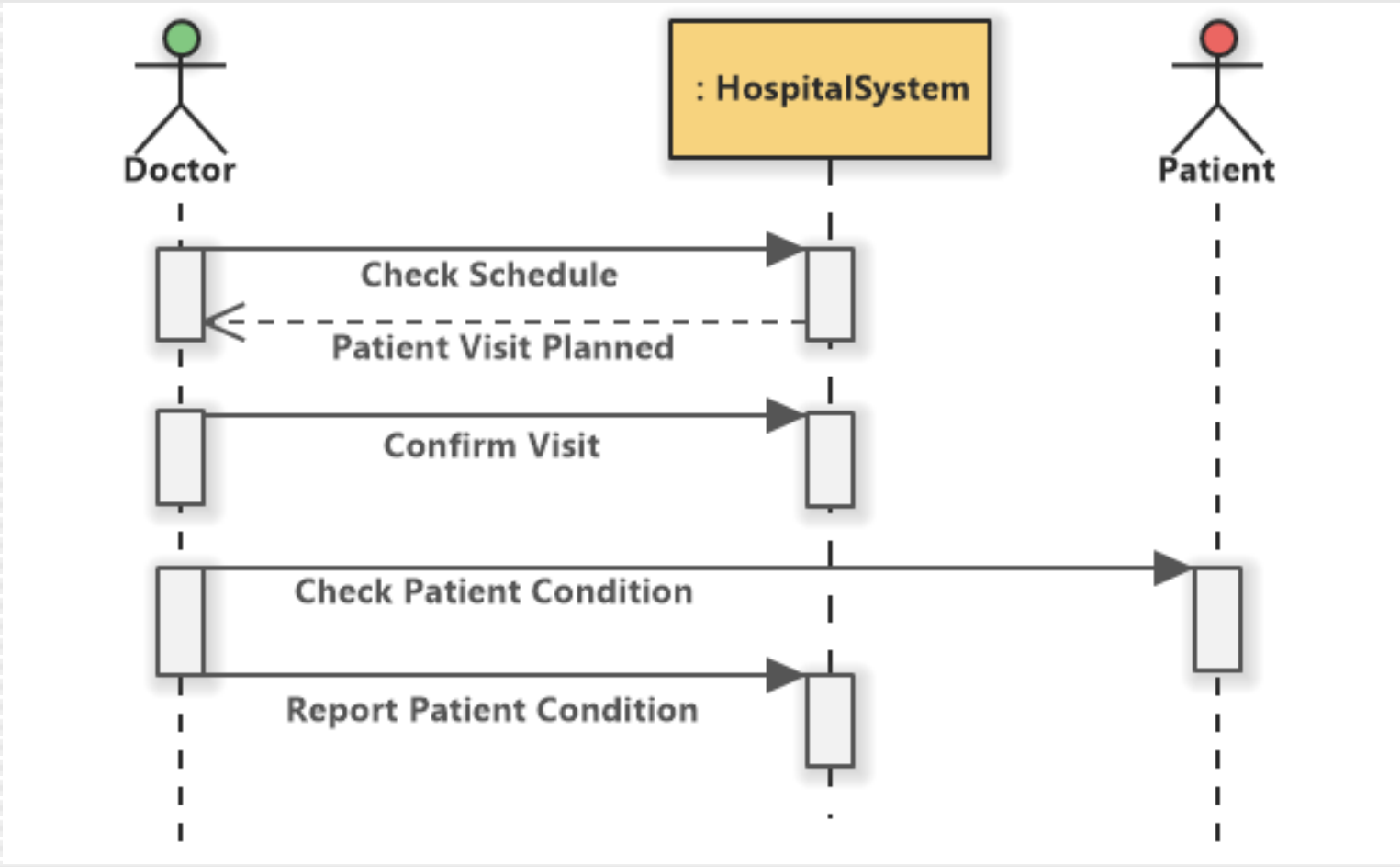
# Components: loop



# Railway Reservation S/M



# Hospital Management S/M



# Rules of thumb

- Rarely use options,loops,alt/else
  - These constructs complicate a diagram and make them hard to read/interpret.
  - Frequently it is better to create multiple simple diagrams
- Create sequence diagrams for use cases when it helps clarify and visualize a complex flow
- Remember: the goal of UML is communication and understanding

# Summary

- Sequence diagrams model object interactions with an emphasis on time ordering
- Method call lines
  - Must be horizontal!
  - Vertical height matters! “Lower equals Later”
  - Label the lines
- Lifeline – dotted vertical line
- Execution bar – bar around lifeline when code is running
- Arrows
  - Synchronous call (you’re waiting for a return value) – triangle arrow-head
  - Asynchronous call (not waiting for a return) – open arrow-head
  - Return call – dashed line

# In class exercise

- Draw a sequence diagram for:
  - Returning a movie to Netflix. When you return it a person scan's it in, the s/w detects that and then goes through a sequence of steps to get the new movie to you. Think of the different classes that would be involved: Queue, Shipping, Inventory, UserAccount, other classes??

# In class exercise

- Draw a sequence diagram for:
  - Adding a picture to Flickr (or any online image database). Login, pick an album, upload a picture, etc… Think about the software classes that would be involved – WebGUI (think of this as reporting what the user does), UserAccount, Album, AlbumList, etc…
  - Don’t forget to check and update their current disk usage. For this diagram show the check coming back as acceptable.. you would do a second diagram for them running over quota.

# In class exercise

- Draw a sequence diagram for:
  - Getting on a flight. Start at home, check in at the counter, go through security, and end up at the gate. (If you have time during the exercise, get yourself to your seat.)
    - You may get searched in security

# In class exercise

- Draw a sequence diagram for:
  - Getting money from an ATM machine
    - Treat each part of the ATM as a class
      - Money dispenser
      - Screen
      - Keypad
      - Bank computer
      - Etc…