Collaboration Diagrams

- Class exercise
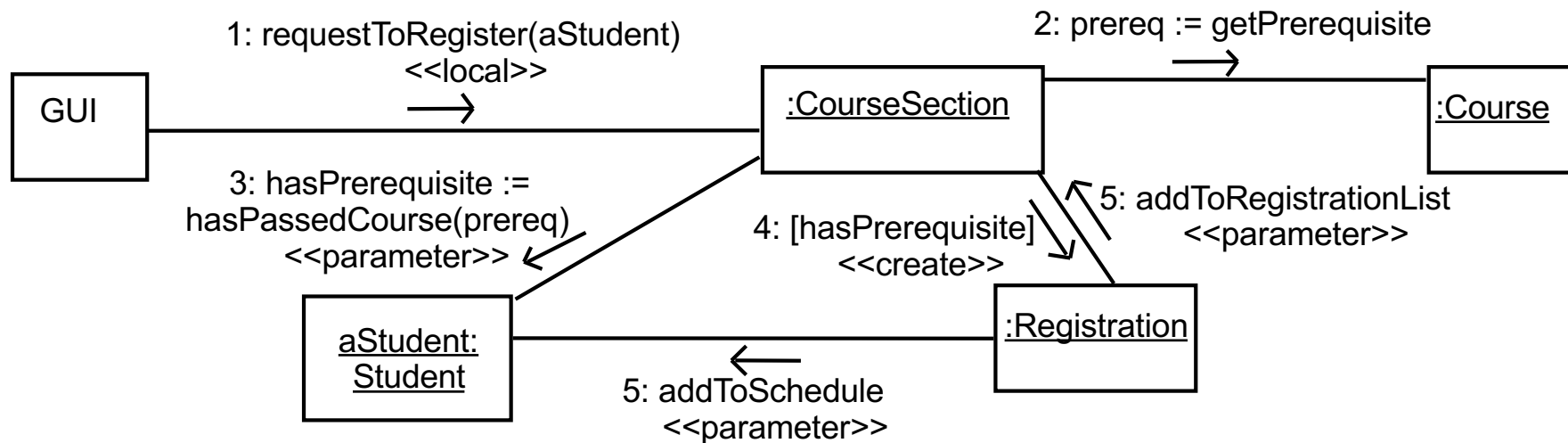
Exercise 1A: sequence diagram from a use-case scenario

▸ *Draw a sequence diagram for the following warehouseOnFire scenario. Include the objects bob, alice, john, FRIEND, and instances of other classes you may need. Draw only the first five message sends.*

▸ Scenario name: warehouseOnFire

▸ Participating actor instances: bob, alice:FieldOfficer, john:Dispatcher

▸ Flow of events:

  ▸ 1. Bob, driving down the main street in his police car, notices smoke coming out of a warehouse. His partner, Alice, activates the 'Report Emergency' function from her FRIEND laptop.

  ▸ 2. Alice enters the address of the building, a brief description of its location (ie. northwest corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene given that area appears to be relatively busy. She confirms her input and waits for an acknowledgement.

  ▸ 3. John, the Dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the Incident site and send their estimated arrival time (ETA) to Alice.

▸ 2 ▸ 4. Alice receives the acknowledgement and the ETA

Exercise 1B: Convert your sequence diagram into a collaboration diagram

Exercise 2: code vs. interaction diagram

▸ Write a Java implementation that is modelled by the following diagram:

[Lethbridge & Laganière 2001]

Exercise 3: sequence vs. interaction diagram

▶ Draw a sequence diagram that models as much as possible (but not more than that!) of the contents that is being modelled by the following diagram:

```
        1: requestToRegister(aStudent)                    2: prereq := getPrerequisite
                  <<local>>                                          ⟶
  ┌──────┐            ⟶              ┌──────────────┐                        ┌──────────┐
  │ GUI  │───────────────────────────│ :CourseSection │──────────────────────│ :Course  │
  └──────┘                           └──────────────┘                        └──────────┘
        3: hasPrerequisite :=                         5: addToRegistrationList
     hasPassedCourse(prereq)      4: [hasPrerequisite]      <<parameter>>
          <<parameter>>               <<create>>
         ┌──────────┐                           ┌──────────────┐
         │ aStudent: │──────────────────────────│ :Registration │
         │  Student  │      5: addToSchedule     └──────────────┘
         └──────────┘        <<parameter>>
```