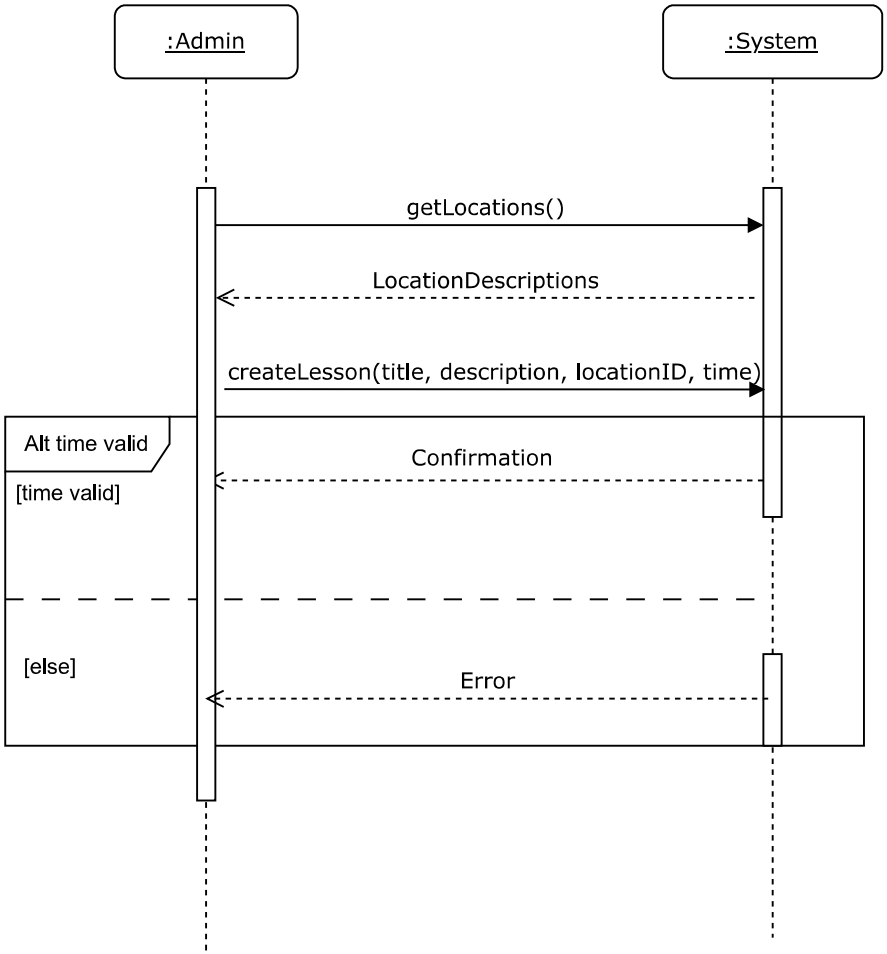
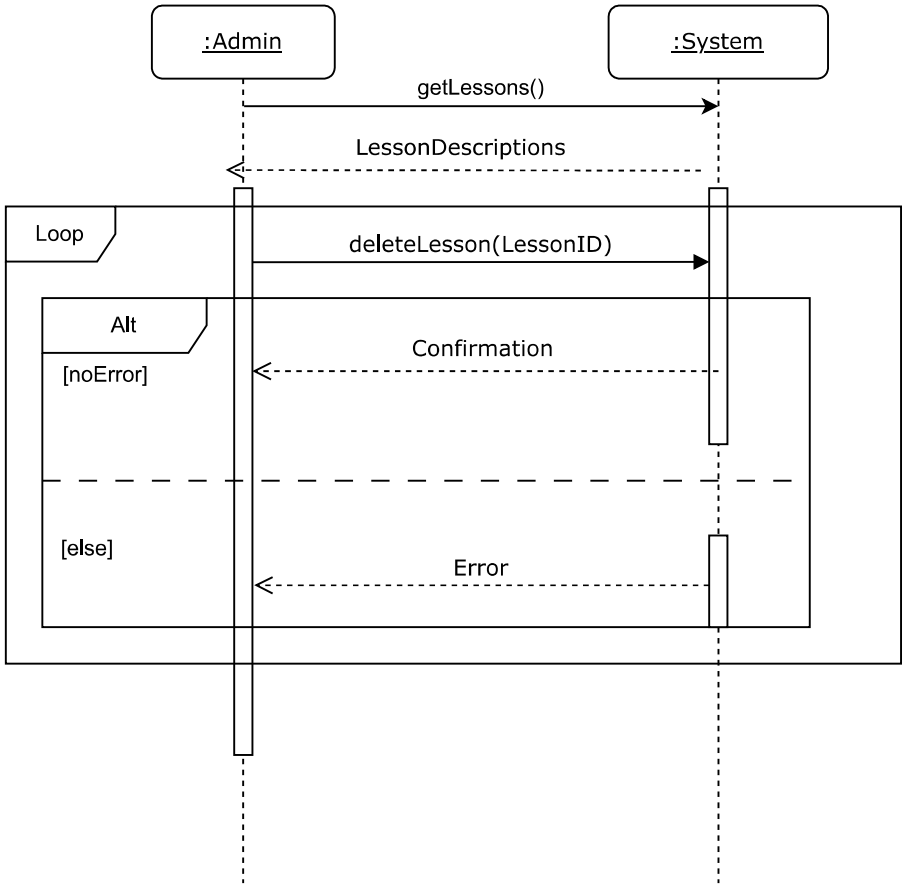
createLesson:



deleteLesson:



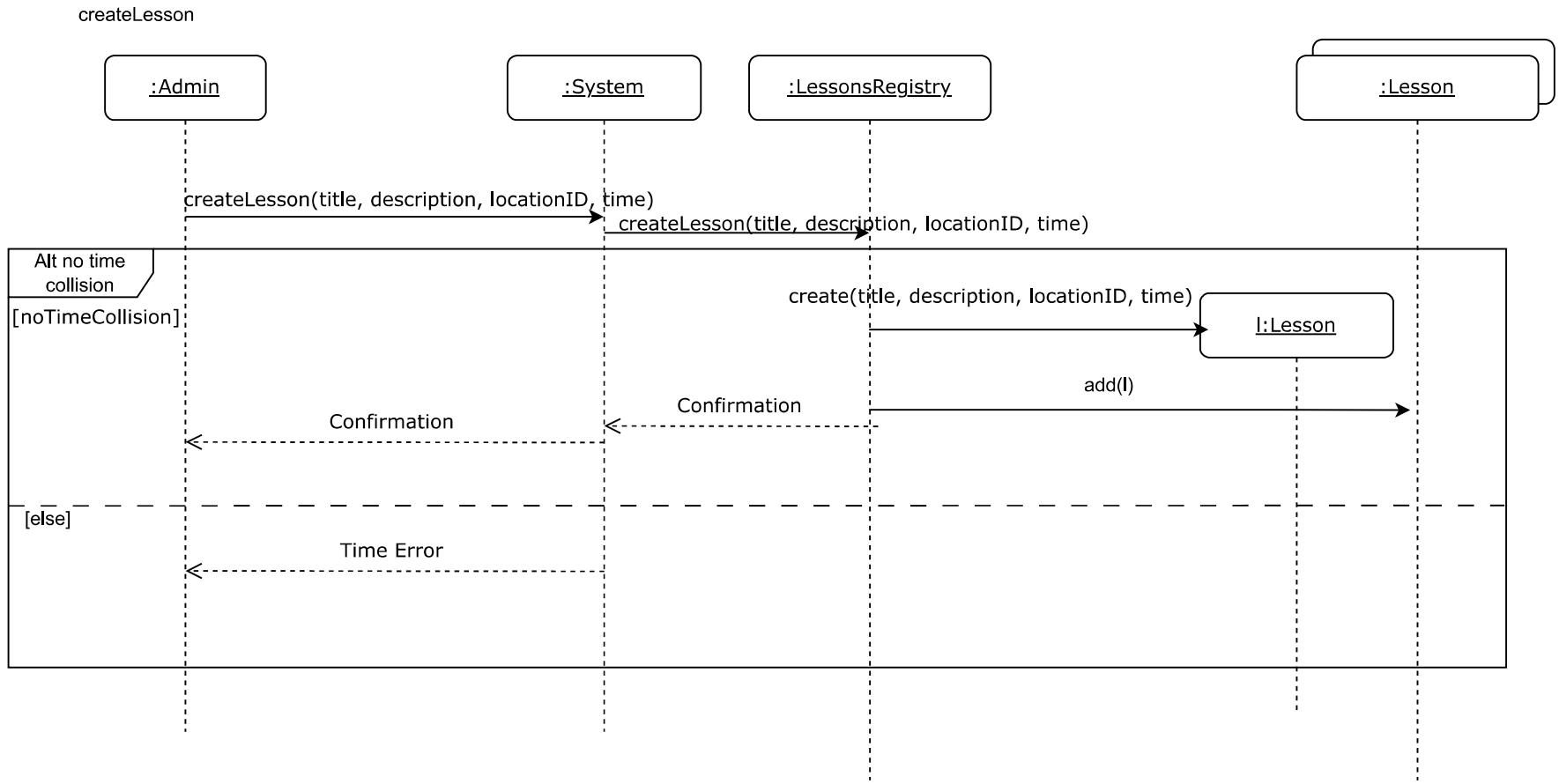
Contract: getLocations
Operation: getLocations()
Cross References: Use Case: Process Offerings
Preconditions: The user is an admin and is currently authenticated.
Postconditions: A list of all Location descriptions in the system
is returned to the admin.

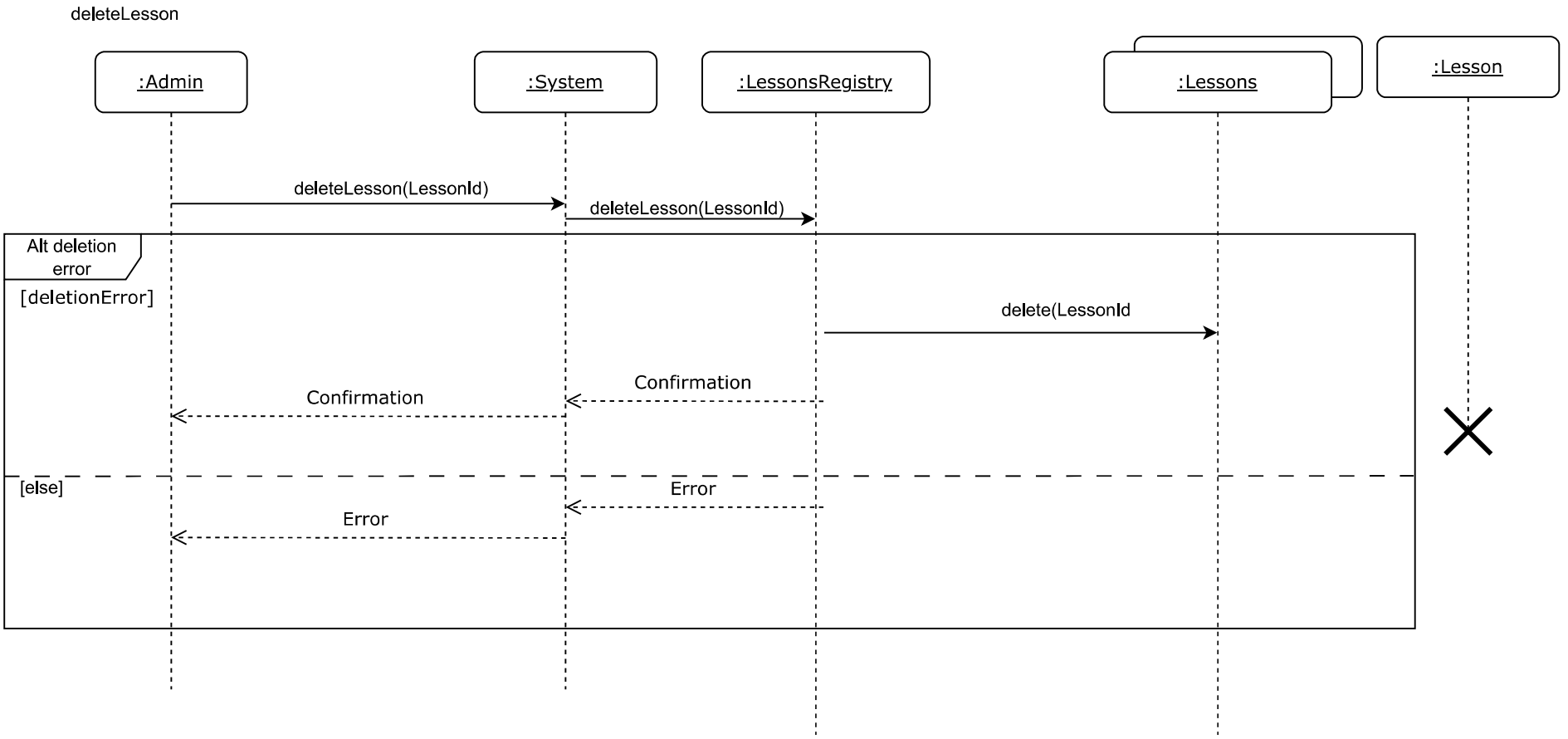
Contract: createLesson
Operation: createLesson(title, description, locationID, time)
Cross References: Use Case: Process Offerings
Preconditions:
The user is an admin and is currently authenticated. The time slot entered is not in conflict with any other pre-existing lesson at the same location.

Postconditions:
- A new Lesson instance *lesson* was created (instance creation)
- *lesson* was associated with the location (association formed)
- *lesson* was associated with a timespan (association formed)
- lesson was added to LessonRegistry.lessonsCollection (association formed)

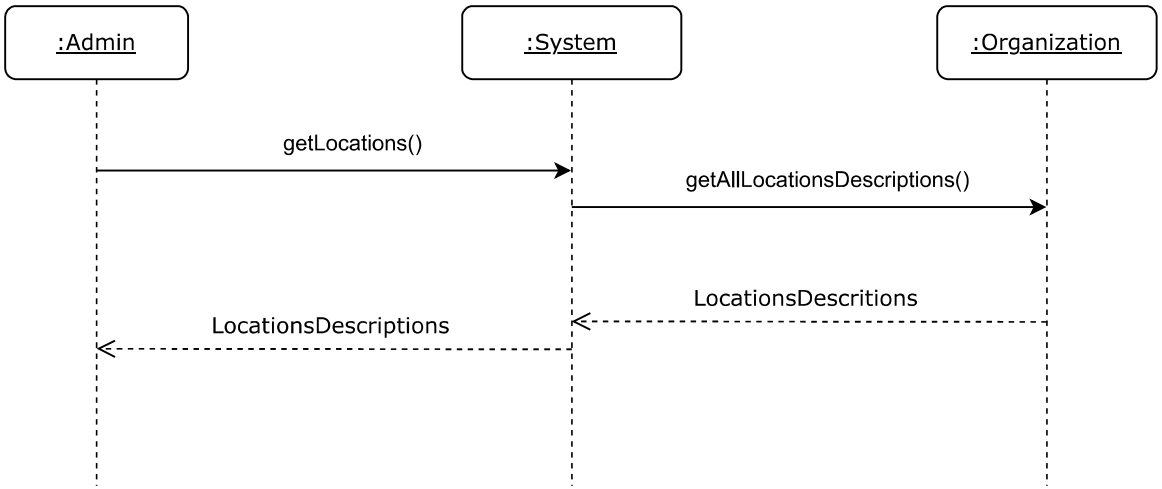
Contract: getLessons
Operation: getLessons()
Cross References: Use Case: Process Offerings
Preconditions: The user is an admin or an instructor
and is currently authenticated
Postconditions: A list of all Lesson descriptions in the system
is returned to the admin.

Contract: deleteLesson
Operation: deleteLesson(LessonId)
Cross References: Use Case: Process Offerings
Preconditions: The user is an admin and is currently authenticated, the
LessonId passed as a parameter references an existing Lesson.
Postconditions: The instance of Lesson with the corresponding LessonID
is deleted from the LessonsRegistry as well as any offering
or booking referencing it.

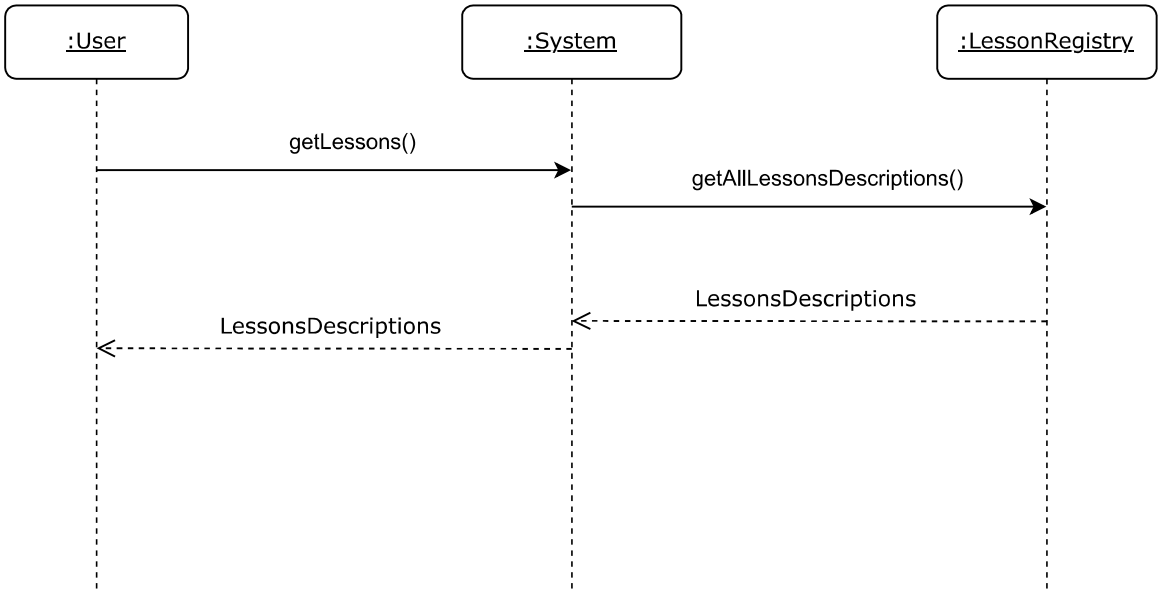

createLesson
:Admin
:System
:LessonsRegistry
:Lesson
createLesson(title, description, locationID, time)
createLesson(title, description, locationID, time)
Alt no time collision
[noTimeCollision]
create(title, description, locationID, time)
l:Lesson
add(l)
Confirmation
Confirmation
[else]
Time Error


deleteLesson
:Admin
:System
:LessonsRegistry
:Lessons
:Lesson
deleteLesson(LessonId)
deleteLesson(LessonId)
Alt deletion error
[deletionError]
delete(LessonId
Confirmation
Confirmation
[else]
Error
Error

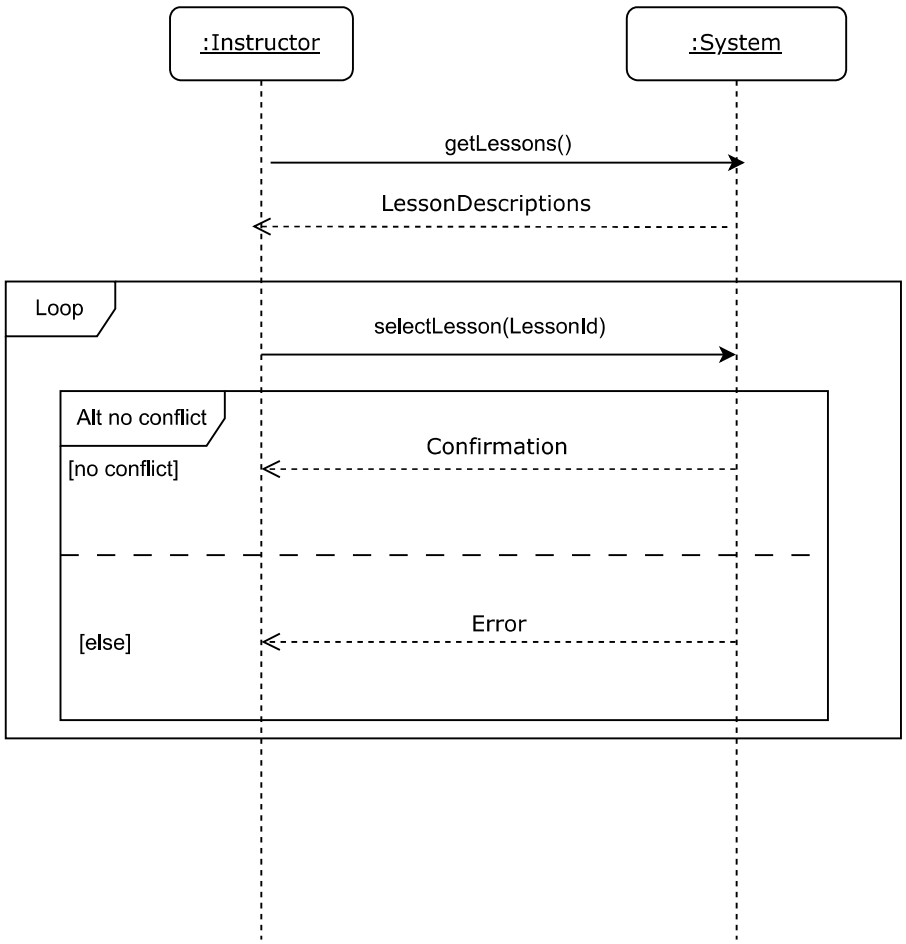
getLocations



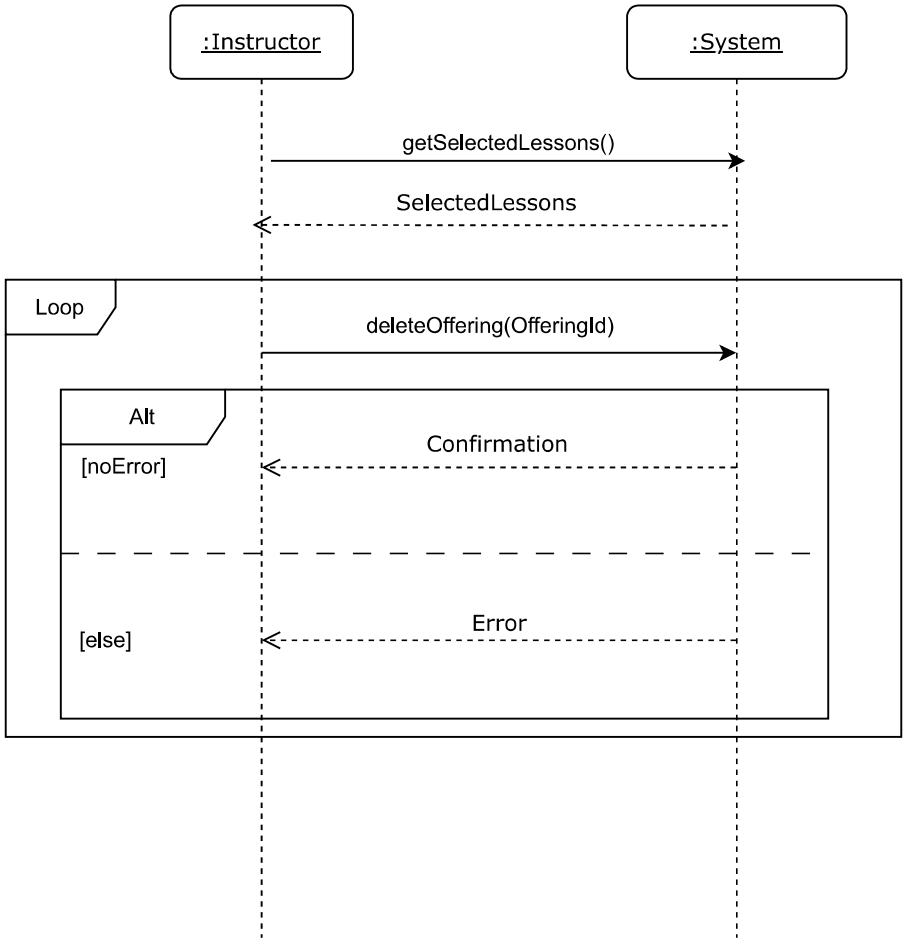
getLessons



SelectLesson



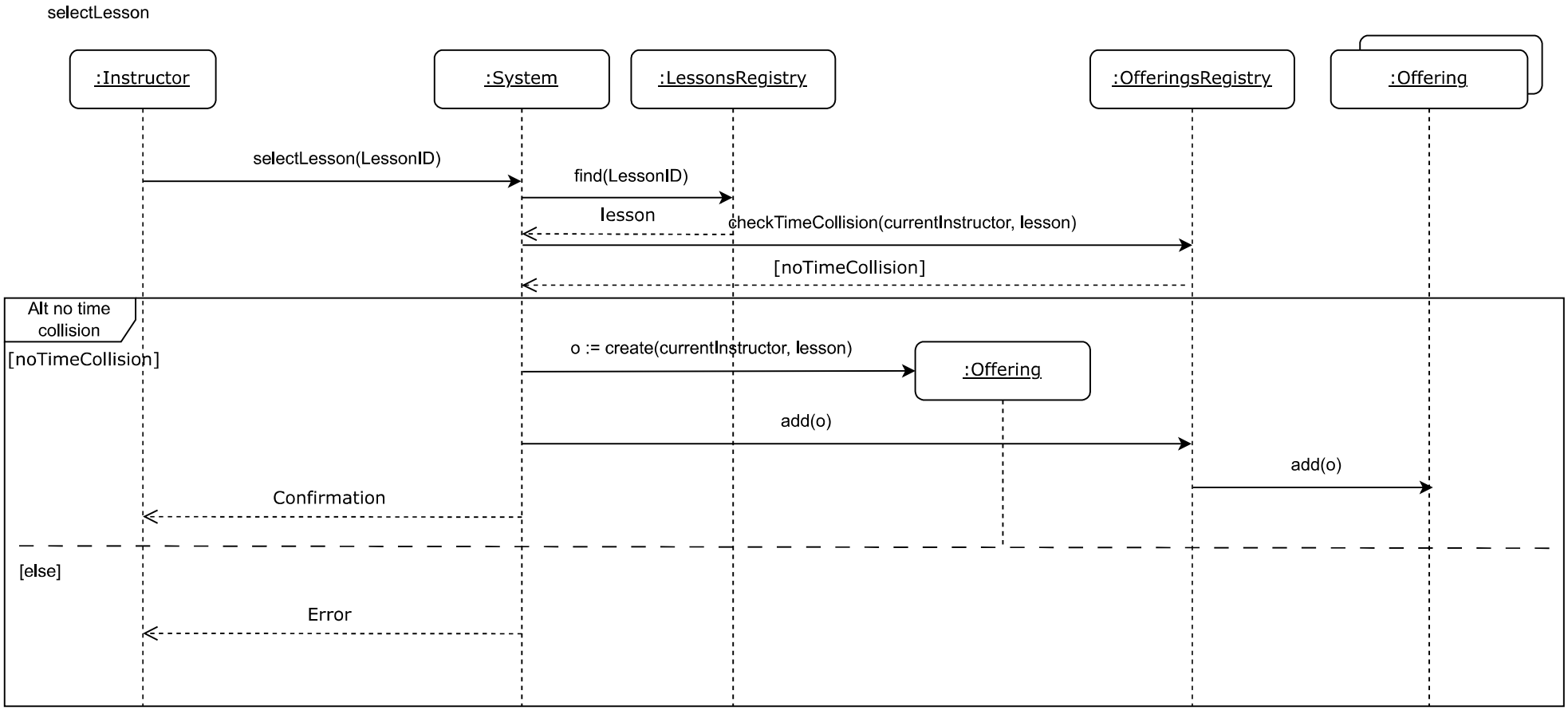
deleteSelectedLesson:

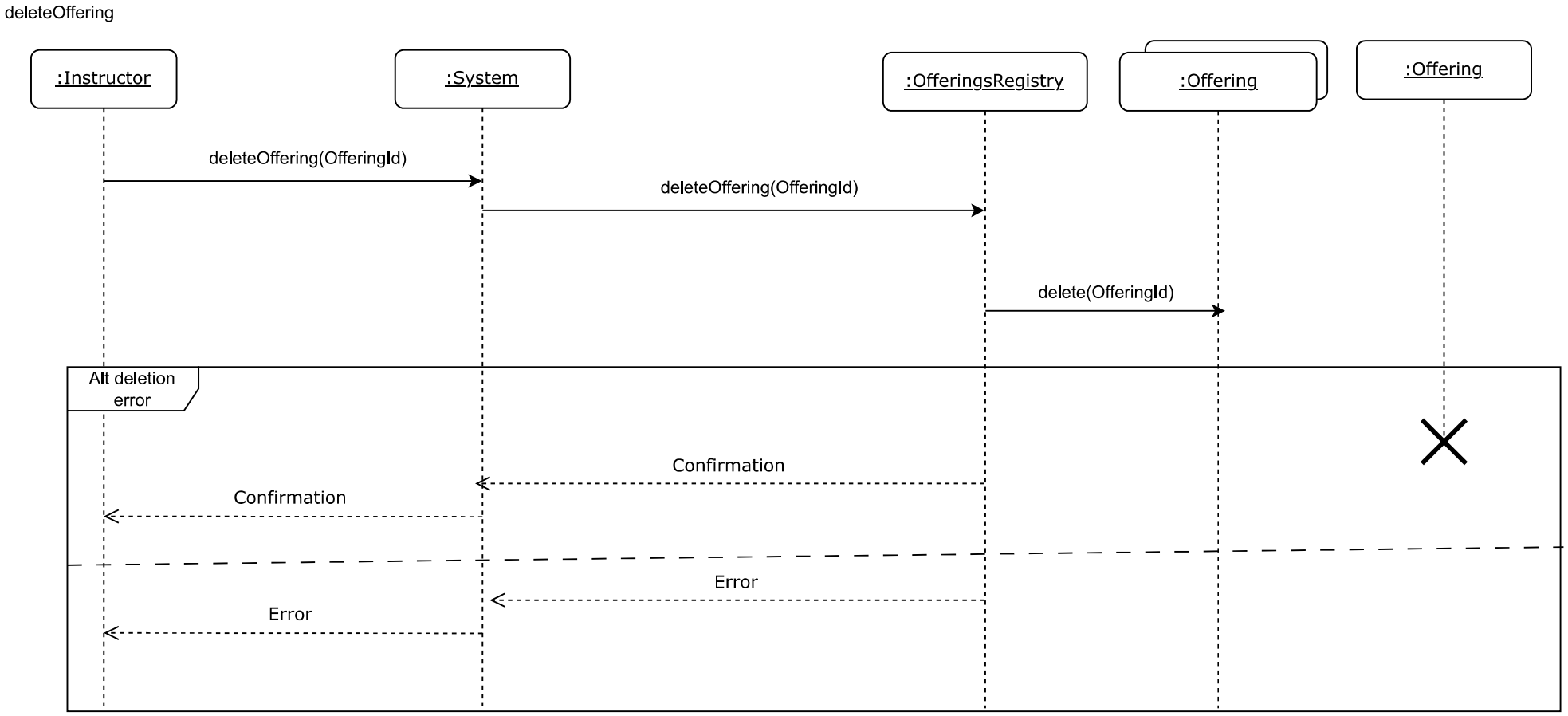


Contract: selectLesson
Operation: selectLesson(LessonId)
Cross References: Use Case: Process Offerings
Preconditions: The LessonId passed is valid and there is no time conflict with an lesson that was already selected.
Postconditions: a offering instance is created with its ids set to the instructors and the lesson id and is added to the list of offerings.

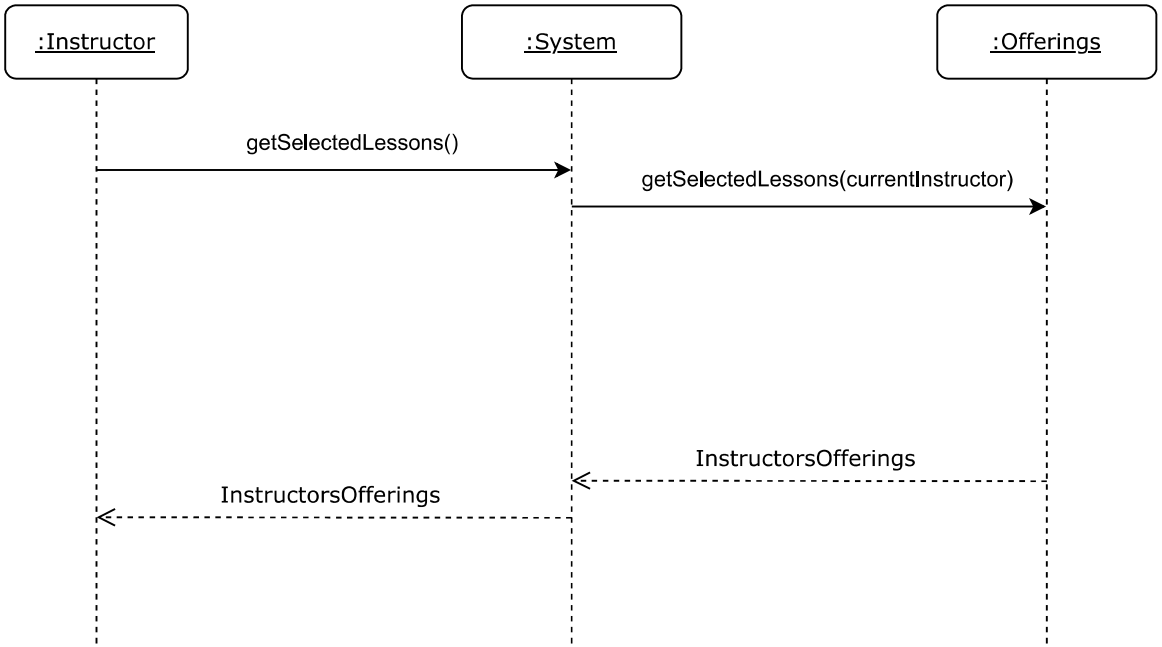
Contract: getSelectedLessons
Operation: getSelectedLessons()
Cross References: Use Case: Process Offerings
Preconditions: the instructor is authenticated.
Postconditions: a list of all Offering created by the instructor is returned.

Contract: deleteOffering
Operation: deleteOffering(OfferingId)
Cross References: Use Case: Process Offerings
Preconditions: the instructor is authenticated and LessonId is valid.
Postconditions: the instance of Offering that is referenced by the passed LessonId is deleted as well as any other objects referencing it.

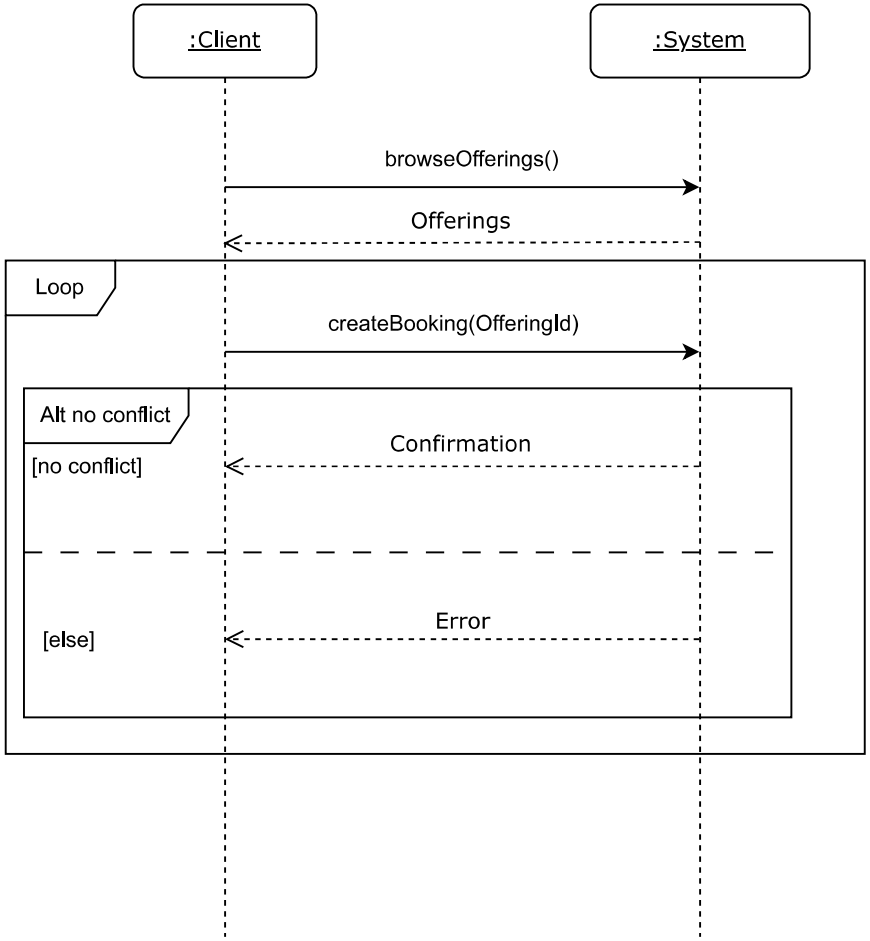

selectLesson
:Instructor
:System
:LessonsRegistry
:OfferingsRegistry
:Offering
selectLesson(LessonID)
find(LessonID)
lesson
checkTimeCollision(currentInstructor, lesson)
[noTimeCollision]
Alt no time collision
[noTimeCollision]
o := create(currentInstructor, lesson)
:Offering
add(o)
add(o)
Confirmation
[else]
Error


deleteOffering
:Instructor
:System
:OfferingsRegistry
:Offering
:Offering
deleteOffering(OfferingId)
deleteOffering(OfferingId)
delete(OfferingId)
Alt deletion
error
Confirmation
Confirmation
Error
Error

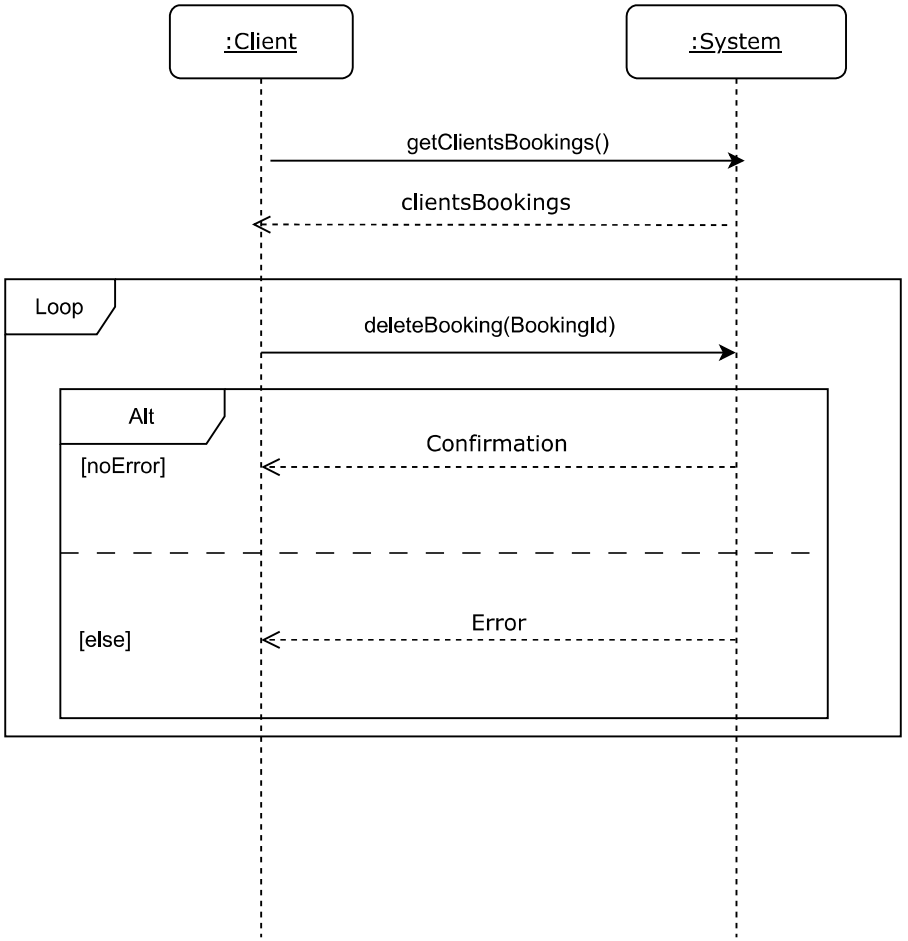
getSelectedLessons



createBooking:



deleteBooking:

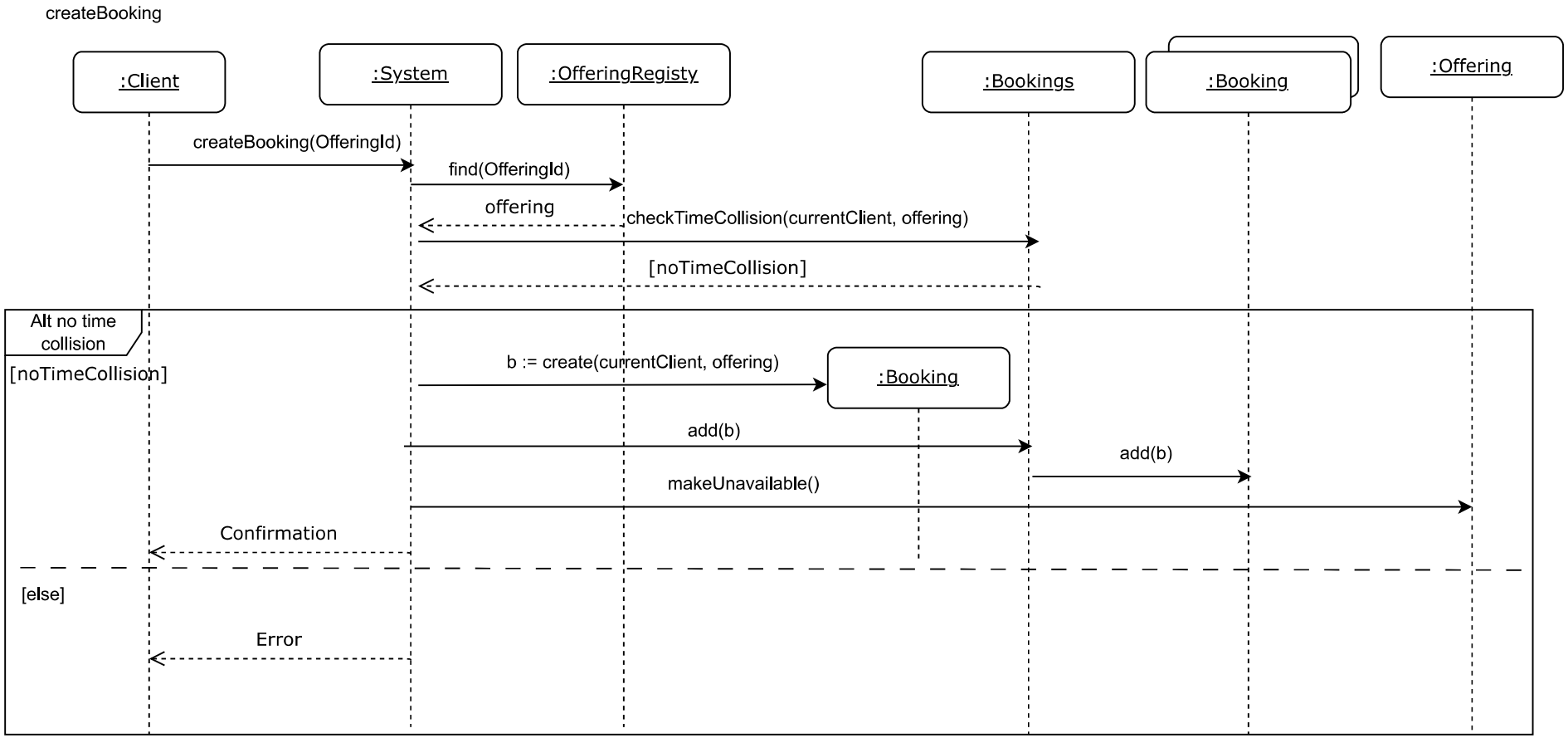


Contract: browseOfferings
Operation: browseOfferings()
Cross References: Use Case: Process Bookings
Preconditions: Offerings are available in the system.
Postconditions: A list of all available offerings is returned.

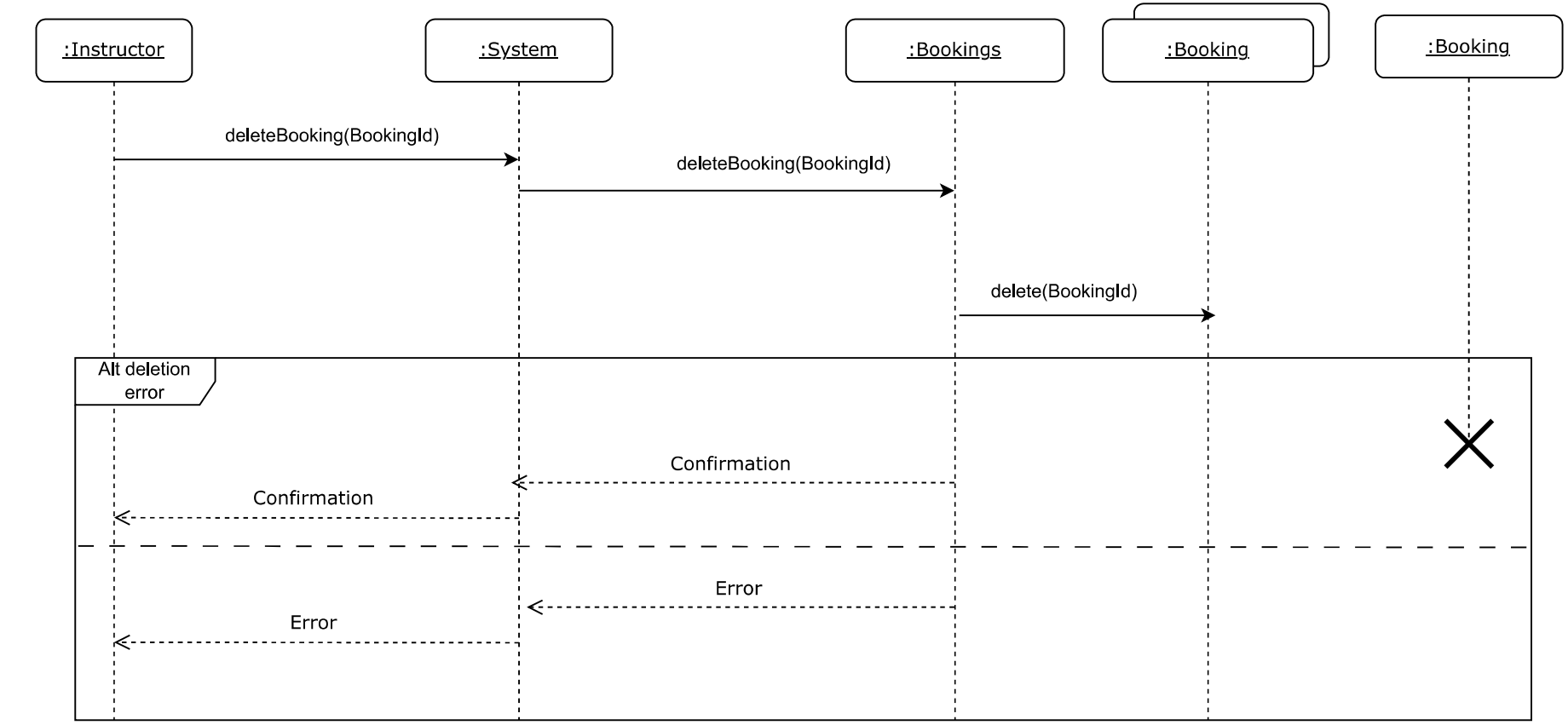
Contract: getClientsbookings
Operation: getClientsbookings()
Cross References: Use Case: Process Bookings
Preconditions: The user is a client and is authenticated.
Postconditions: A list of all the clients bookings is returned.

Contract: createBooking
Operation: createBooking(OfferingId)
Cross References: Use Case: Process Bookings
Preconditions: The user is a client and is authenticated and
there is no time conflict with the clients current active bookings
Postconditions: a booking instance is created with its ids set to the
client and the offerings id and is added to the list of bookings.
The offering is marked as unavailable.

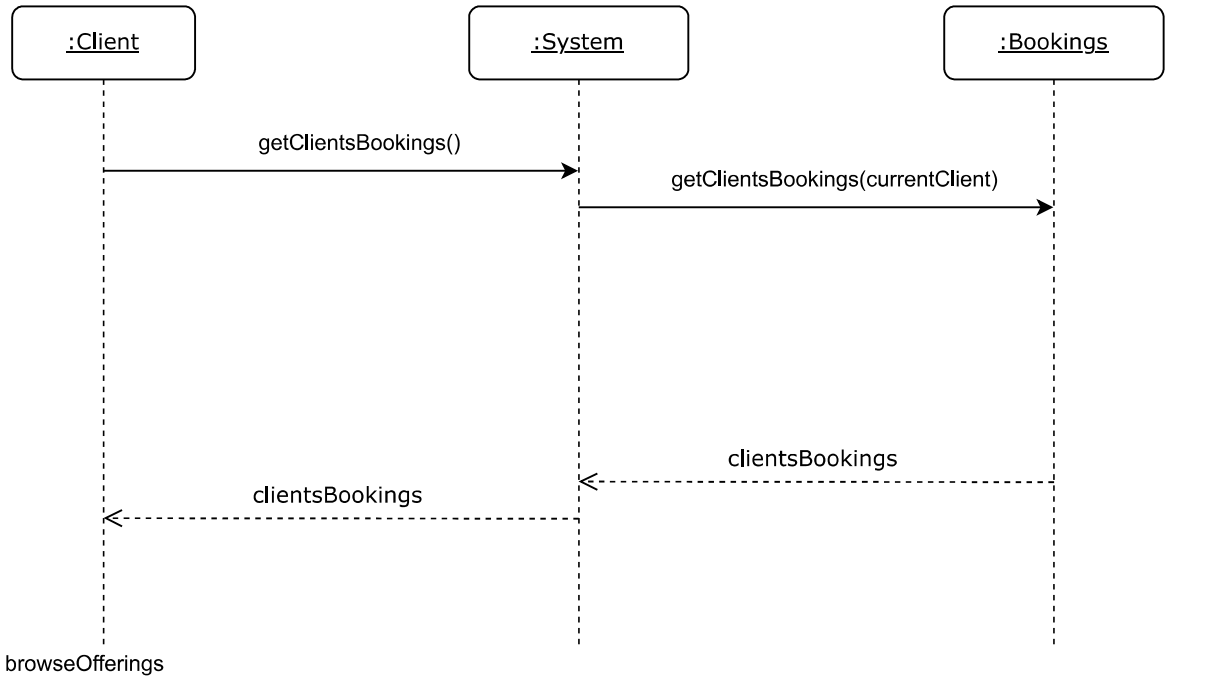
Contract: deleteBooking
Operation: deleteBooking(BookingId)
Cross References: Use Case: Process Bookings
Preconditions: The user is a client and is authenticated BookingId is valid.
Postconditions: The Booking instance is deleted from the bookings collection,
the offering it was associated with is now marked as available.


createBooking
:Client
:System
:OfferingRegisty
:Bookings
:Booking
:Offering
createBooking(OfferingId)
find(OfferingId)
offering
checkTimeCollision(currentClient, offering)
[noTimeCollision]
Alt no time collision
[noTimeCollision]
b := create(currentClient, offering)
:Booking
add(b)
add(b)
makeUnavailable()
Confirmation
[else]
Error

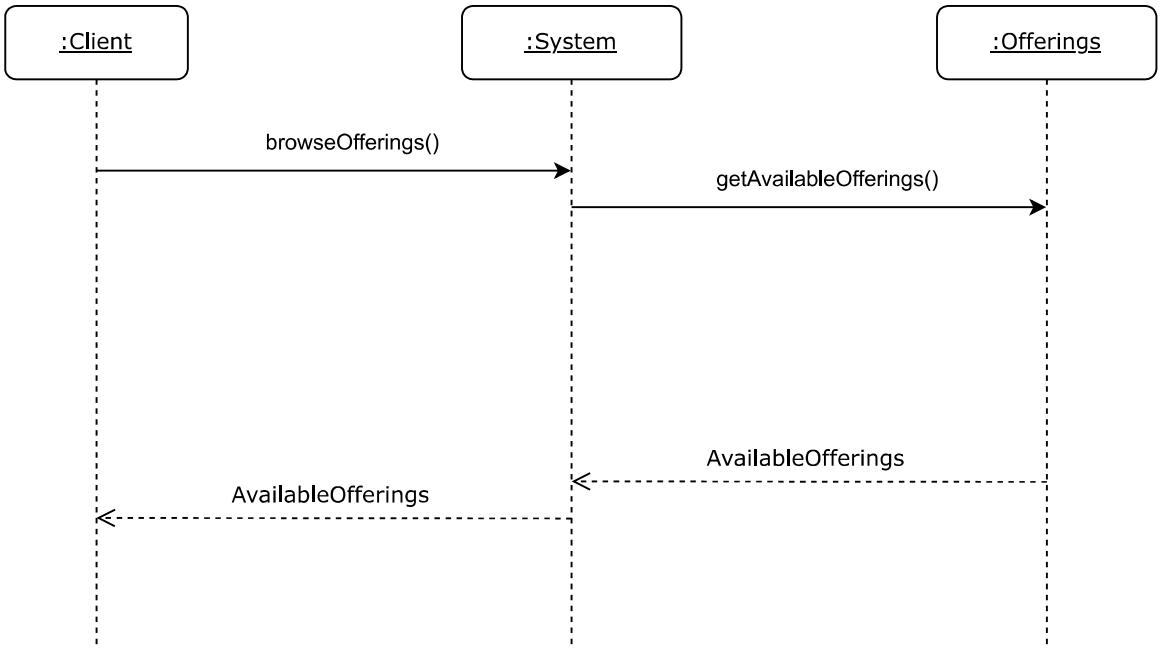
deleteBooking:

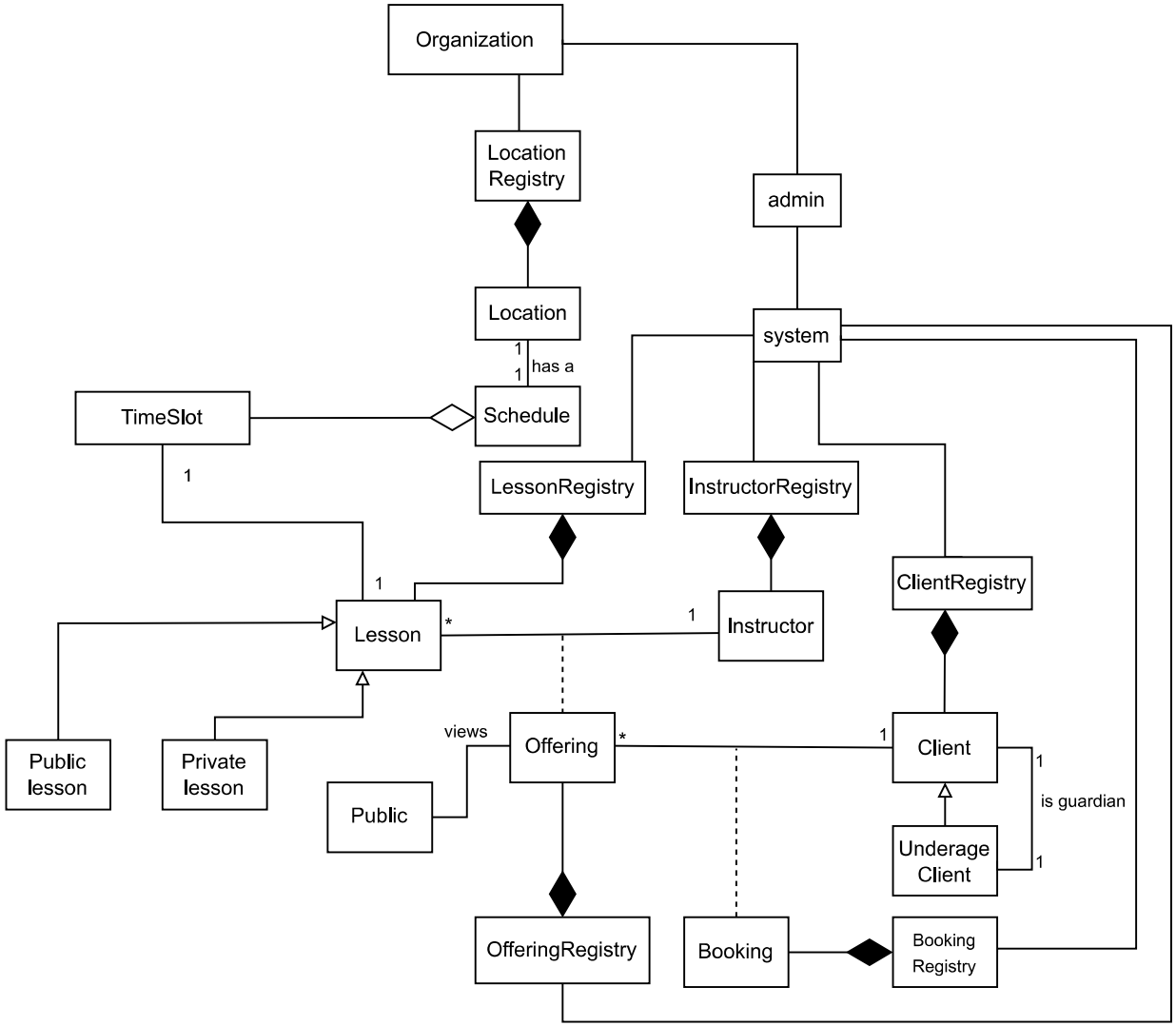


getClientsBookings:



browseOfferings




Organization
Location Registry
admin
Location
system
1
1
has a
TimeSlot
Schedule
1
LessonRegistry
InstructorRegistry
ClientRegistry
1
Lesson
*
1
Instructor
Public lesson
Private lesson
views
Offering
*
1
Client
1
is guardian
1
Public
Underage Client
OfferingRegistry
Booking
Booking Registry