

6. Car Rental Case Study in UML

6.1 Getting Started

- To follow: A medium sized case study for a car rental administration system
- Explanation of central UML diagrams and language features therein
- Used here: Use case, class, object, statechart, sequence, collaboration, and activity diagrams
- Development of the case study also demonstrates a typical development process
- Diagram order in this document does not reflect the order occurring in the development

Overview on the Used Development Process

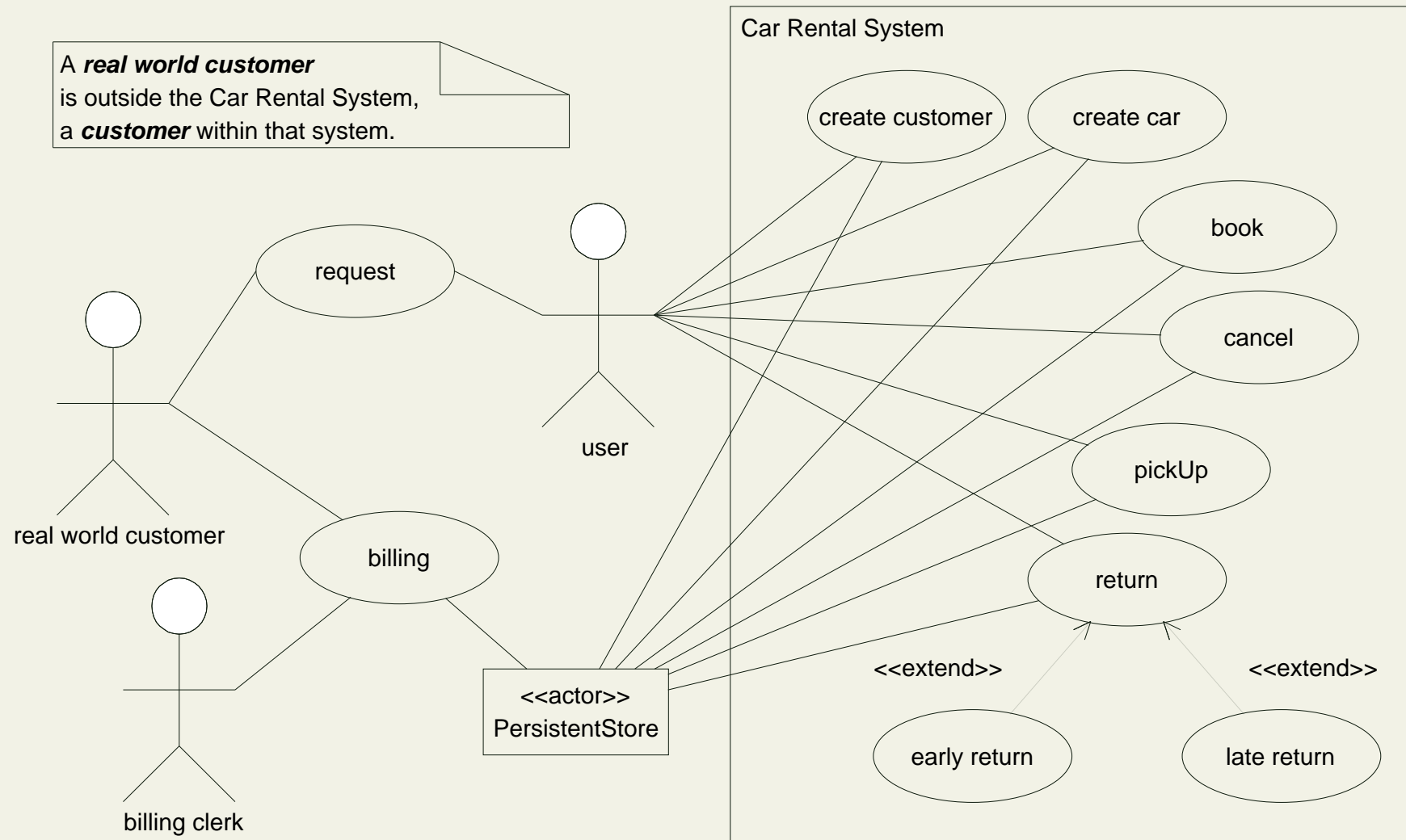
- (1) Start with a use case diagram giving an overview on the system, its boundaries and its main functionality
- (2) Develop object diagrams and sequence diagrams as desired scenarios for the system structure and behavior
- (3) Develop a class diagram and a statechart diagram for each class; start with a textual description of the statechart diagrams; continue with a more formal description for the statecharts

Overview on the Used Development Process

- (4) Check whether the existing object diagrams and sequence diagrams are consistent with the class diagram and the statechart diagrams; if necessary, modify them
- (5) Develop more object and sequence diagrams
- (6) If an operation seems to become stable (no changes in the diagrams for the parts with that operation), start to develop activity diagrams for that operation; begin with a textual description and advance to a more formal one
- (7) Repeat steps (4)-(6) until an acceptable model is found

6.2 Use Cases

Use Case Diagram



Details for Use Case create customer

Use case name: create customer

Goal: to create a new customer

Precondition: the real world customer to be recorded is currently not represented

Postcondition: a new customer exists

Actors: user

Triggering event: a real world customer must be recorded

Description: -

Extensions: -

Alternatives: -

Details for Use Case create car

Use case name: create car

Goal: to create a new car

Precondition: the real world car to be recorded is currently not represented

Postcondition: a new car exists

Actors: user

Triggering event: a real world car must be recorded

Description: -

Extensions: -

Alternatives: -

Details for Use Case book

Use case name: book

Goal: to enter a car rental booking

Precondition: the booking details are plausible

Postcondition: a new booking exists; the booking is now an open booking

Actors: user

Triggering event: a real world customer requests a booking

Description: the real world customer wants to rent a real world car of a certain category; start day of the rental is the current day or a day after the current day; end day of the rental lies after the start day

Extensions: -

Alternatives: -

Details for Use Case cancel

Use case name: cancel

Goal: to prevent that a car must be picked up for a booking

Precondition: the booking to be canceled is present

Postcondition: the booking is marked as closed; no car will be picked up for this booking

Actors: user

Triggering event: a real world customer requests a cancelation of a booking

Description: -

Extensions: -

Alternatives: triggering event - the start day of a booking is passed and no car has been picked up by the customer for that booking

Details for Use Case pickUp

Use case name: pickUp

Goal: to deliver a car for a car rental

Precondition: a booking is present

Postcondition: a suitable car is marked as unavailable (a real world car is given to a real world customer); the booking becomes a current booking

Actors: user

Triggering event: a real world customer requests a pick up

Description: a suitable car must be found among the currently available cars; if none is present, a new car may be added (a new real world car is purchased)

Extensions: -

Alternatives: -

Details for Use Case return

Use case name: return

Goal: to return a car for a car rental

Precondition: a current booking exists and a car has been delivered

Postcondition: the booking becomes closed; the car becomes available (real world customer has returned real world car)

Actors: user

Triggering event: a real world customer requests a return on the end day of the booking

Description: -

Extensions:

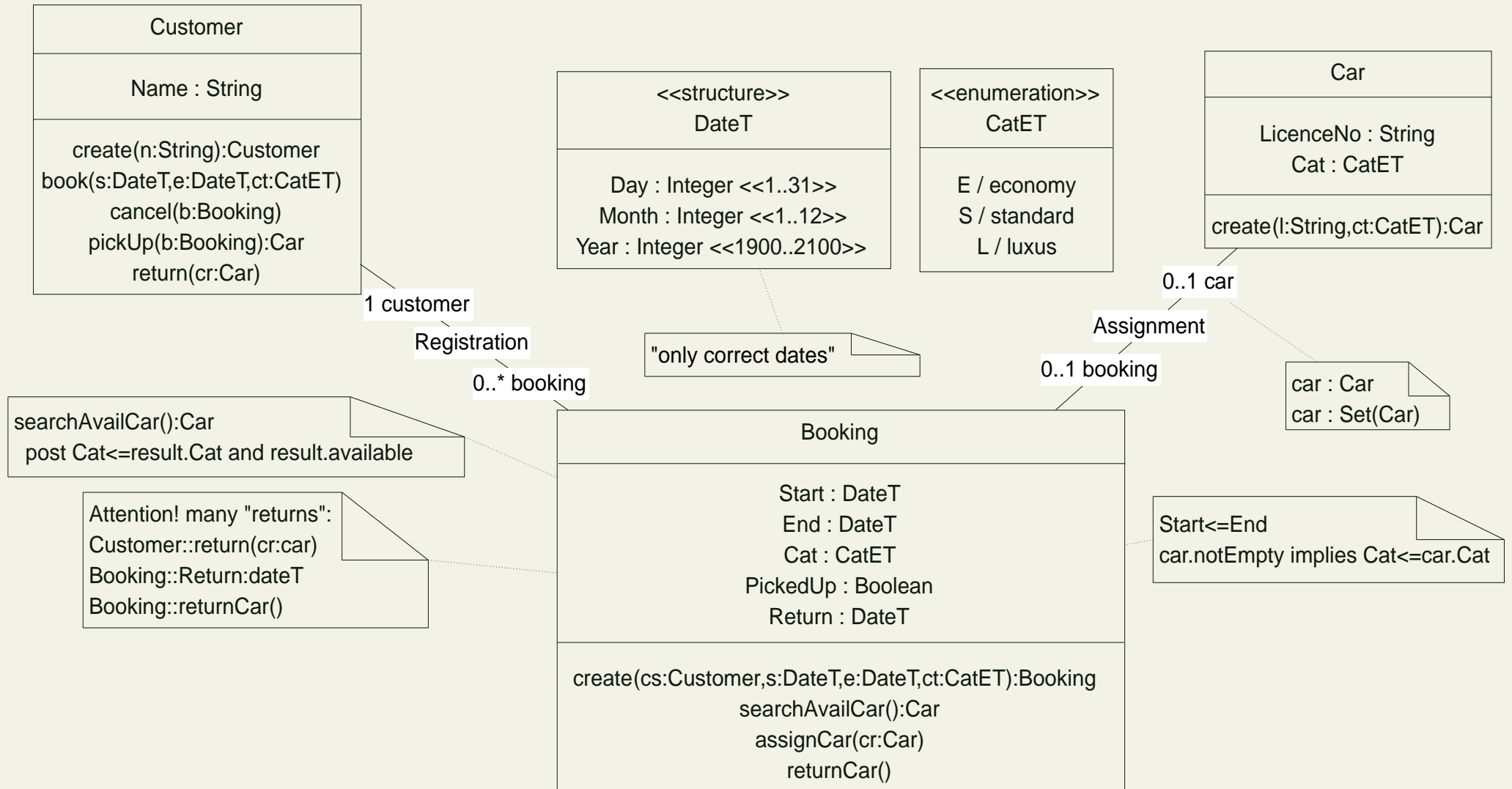
early return: a rented car is returned before the end date of the booking

late return: a rented car is returned after the end date of the booking

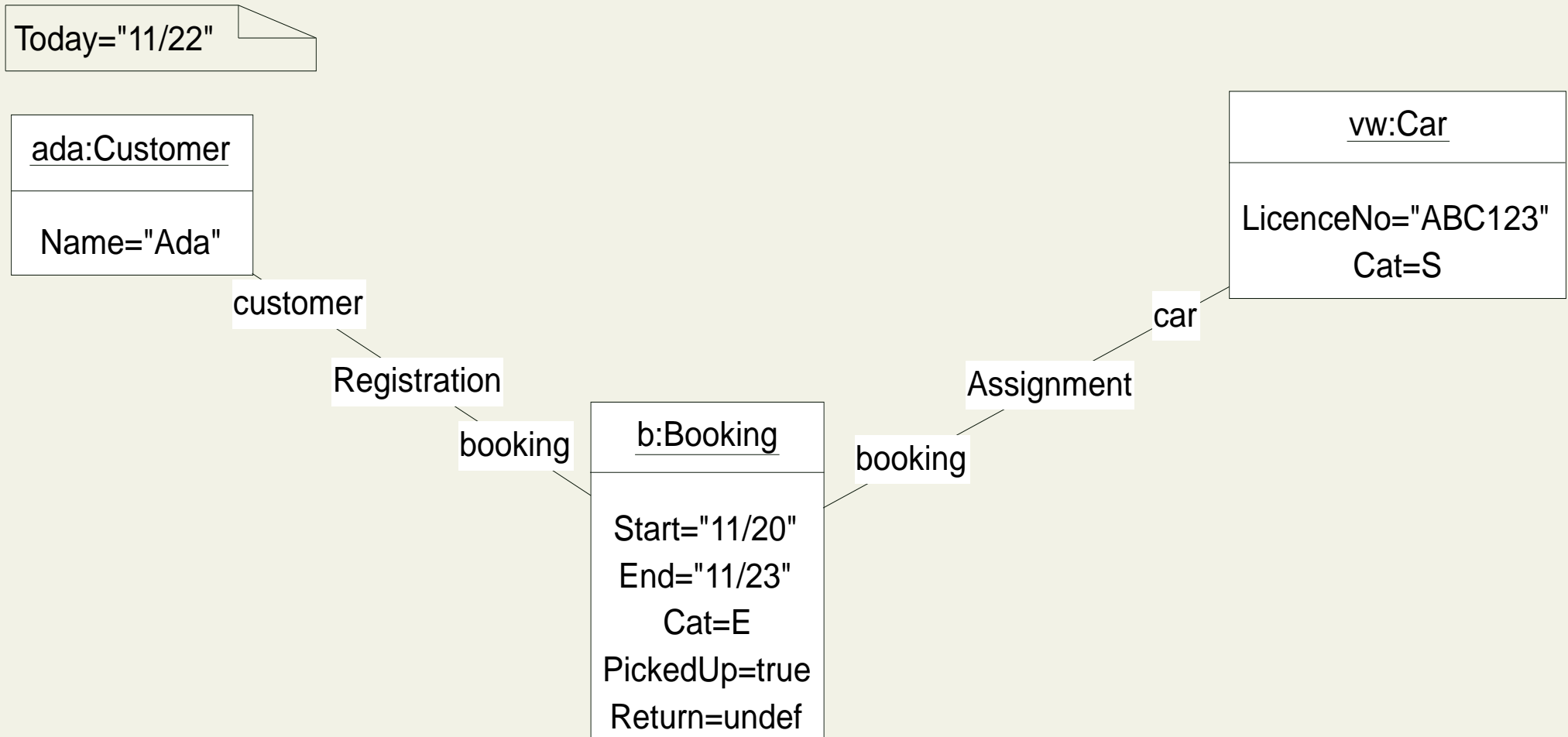
Alternatives: -

6.3 Static Structure Diagrams

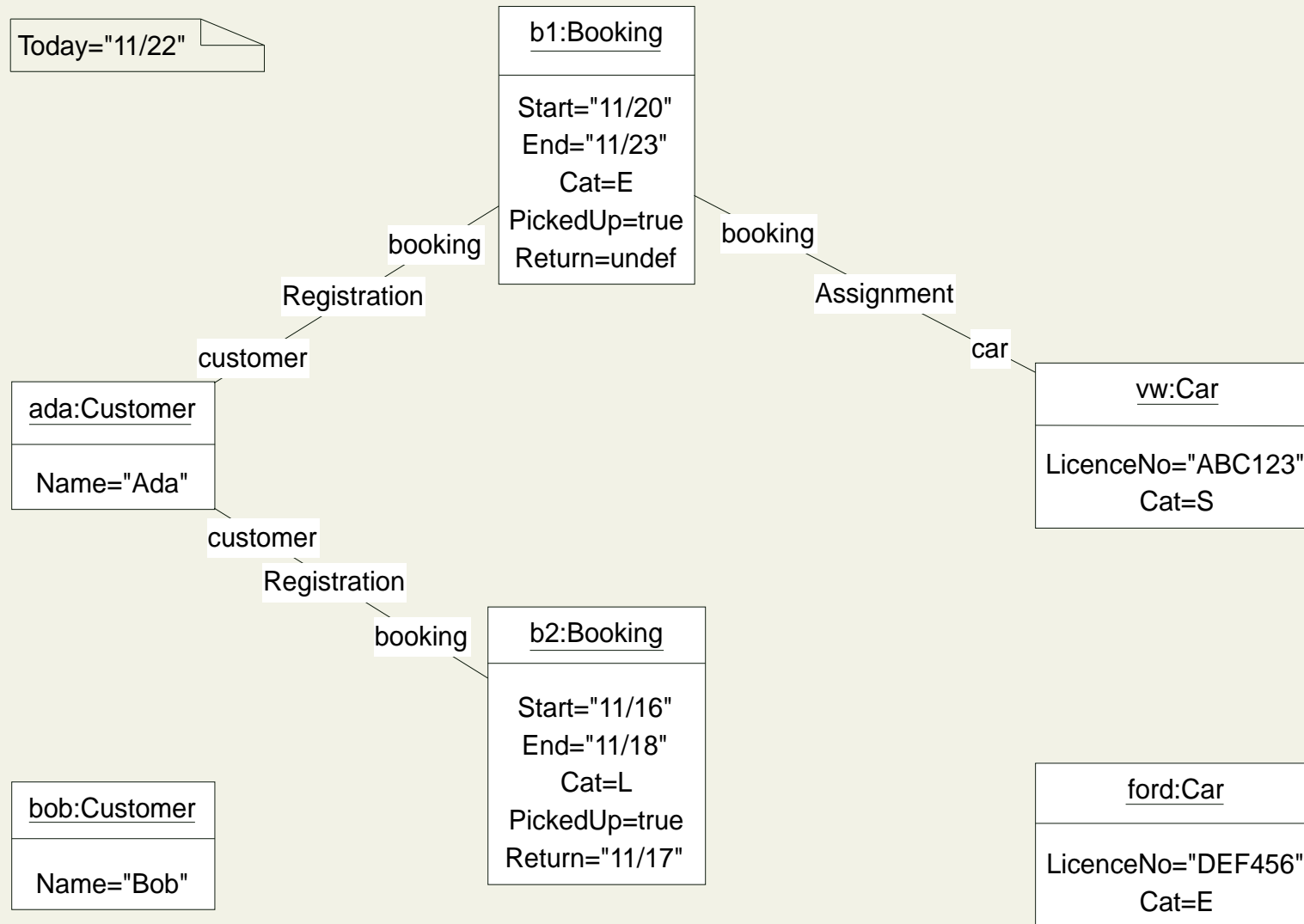
Class Diagram



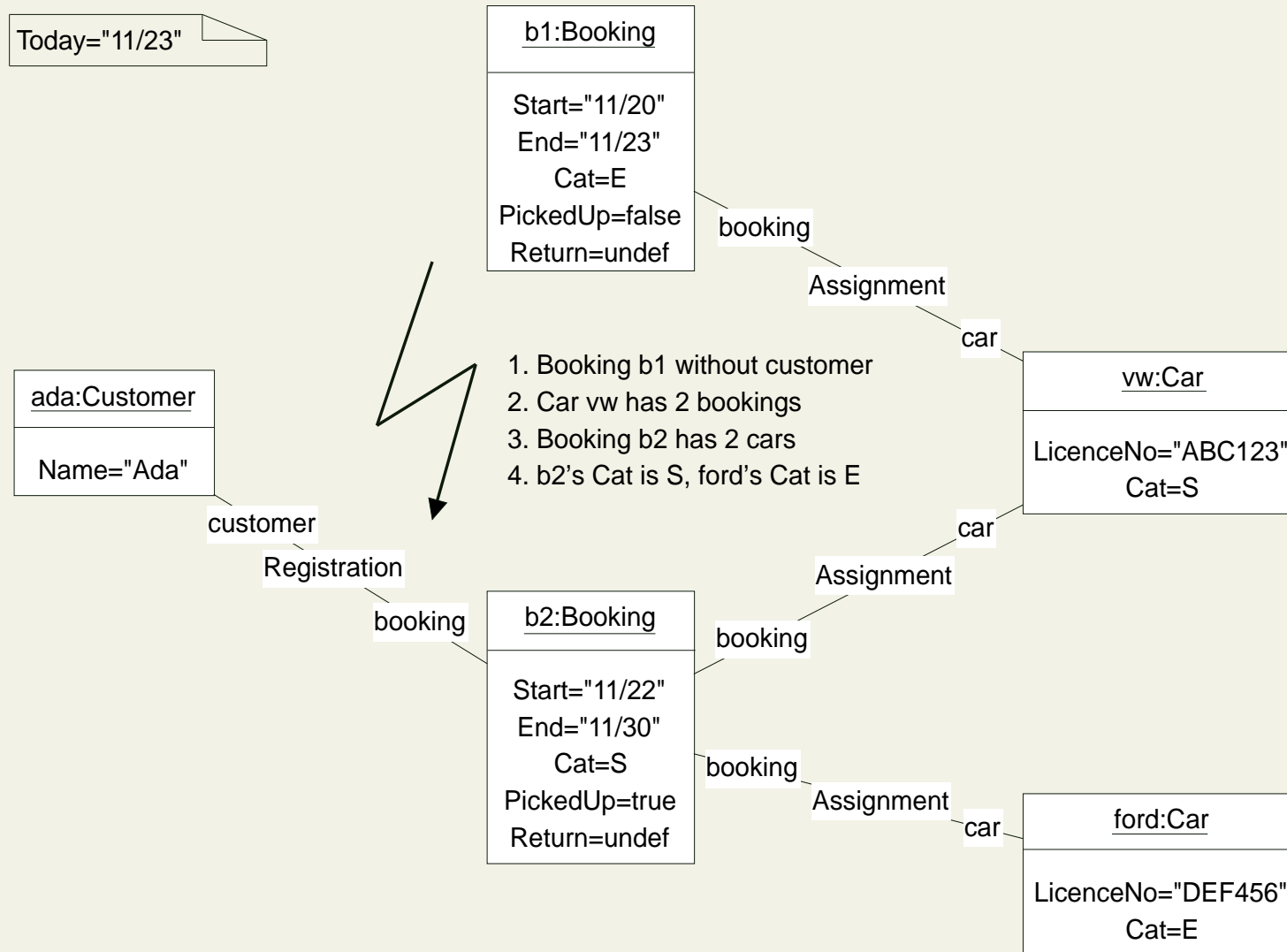
Allowed Object Diagram: 3 Classes, 3 Objects



Allowed Object Diagram: 3 Classes, 6 Objects

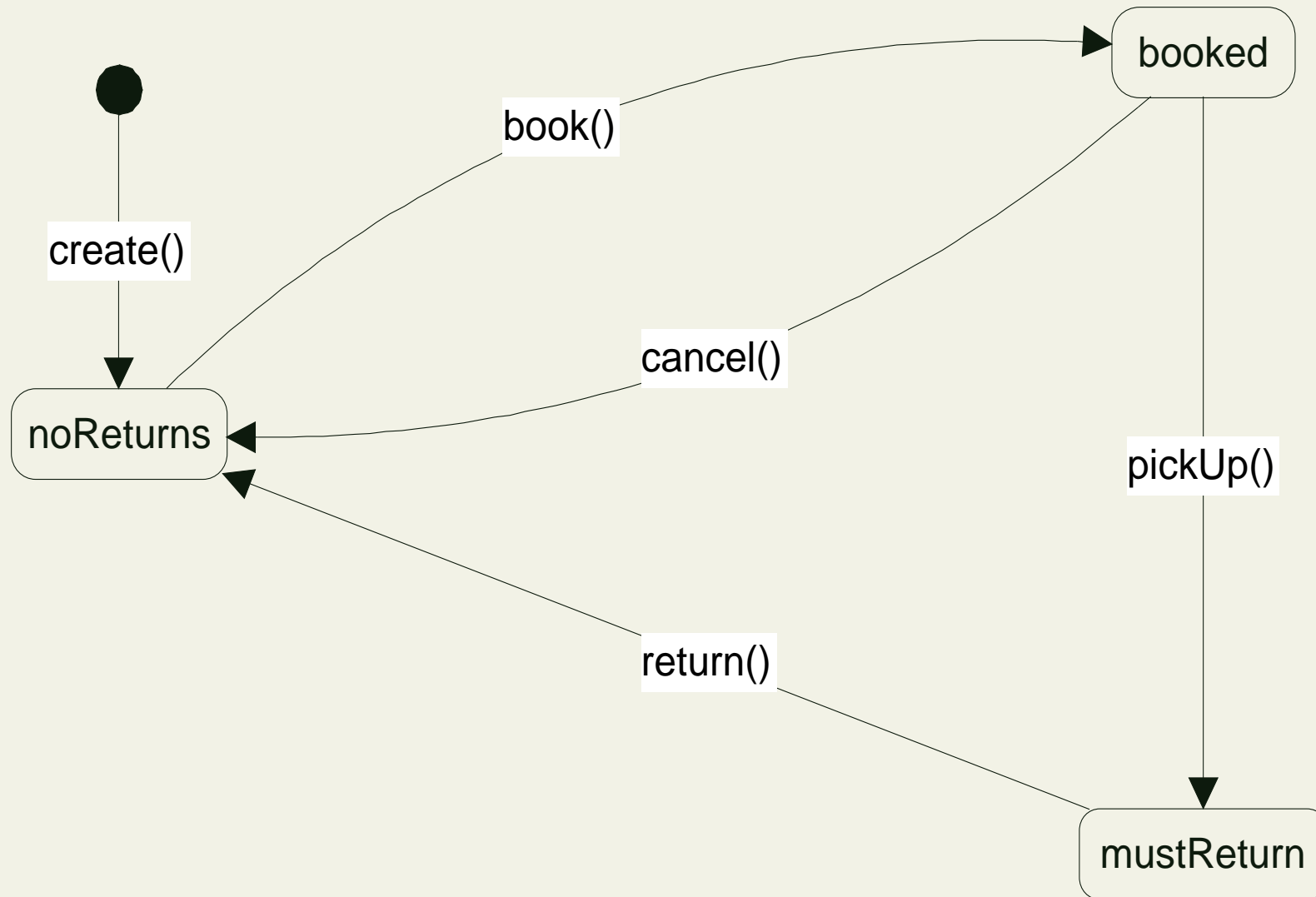


Disallowed Object Diagram

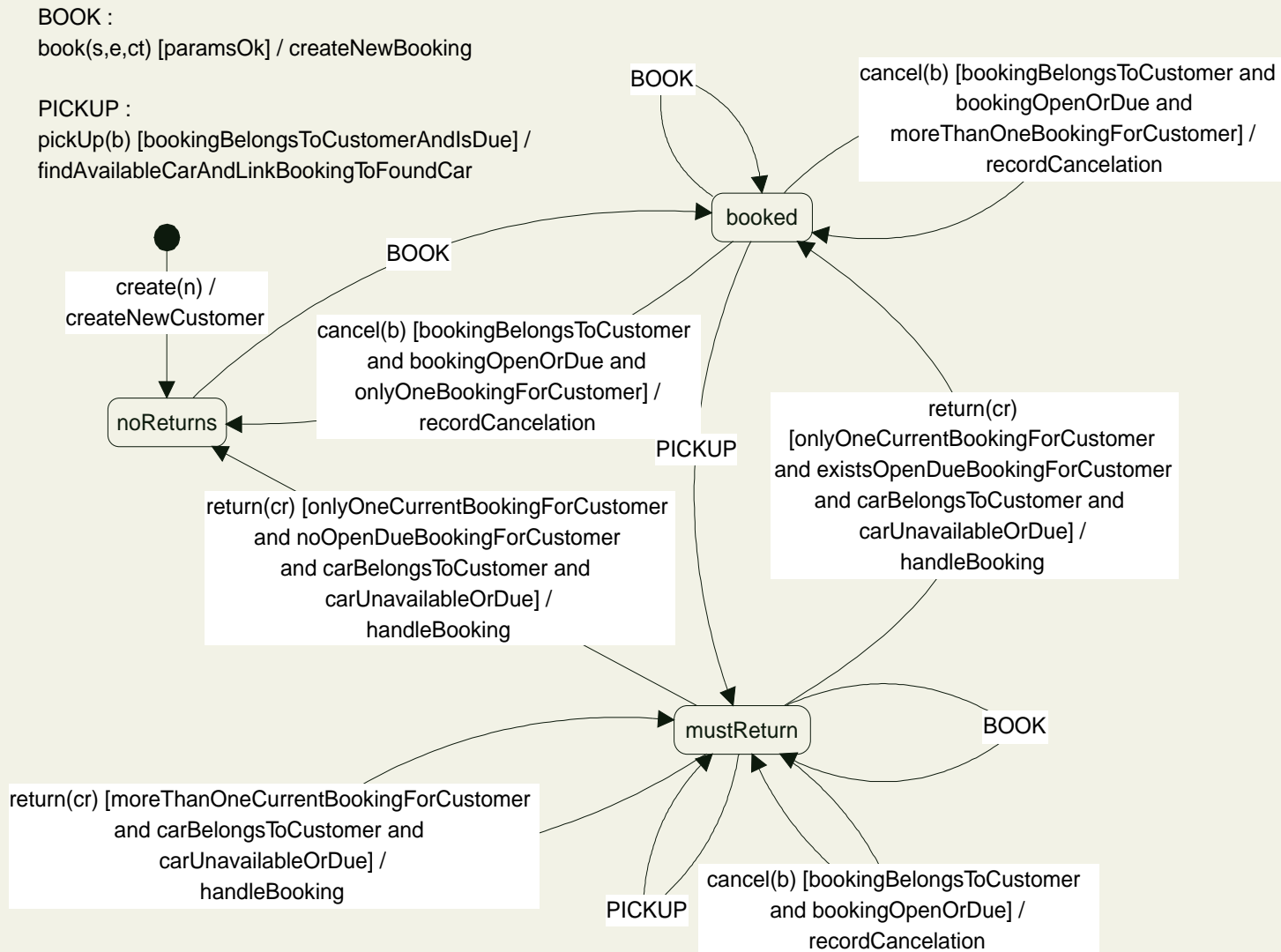


6.4 Statechart Diagrams

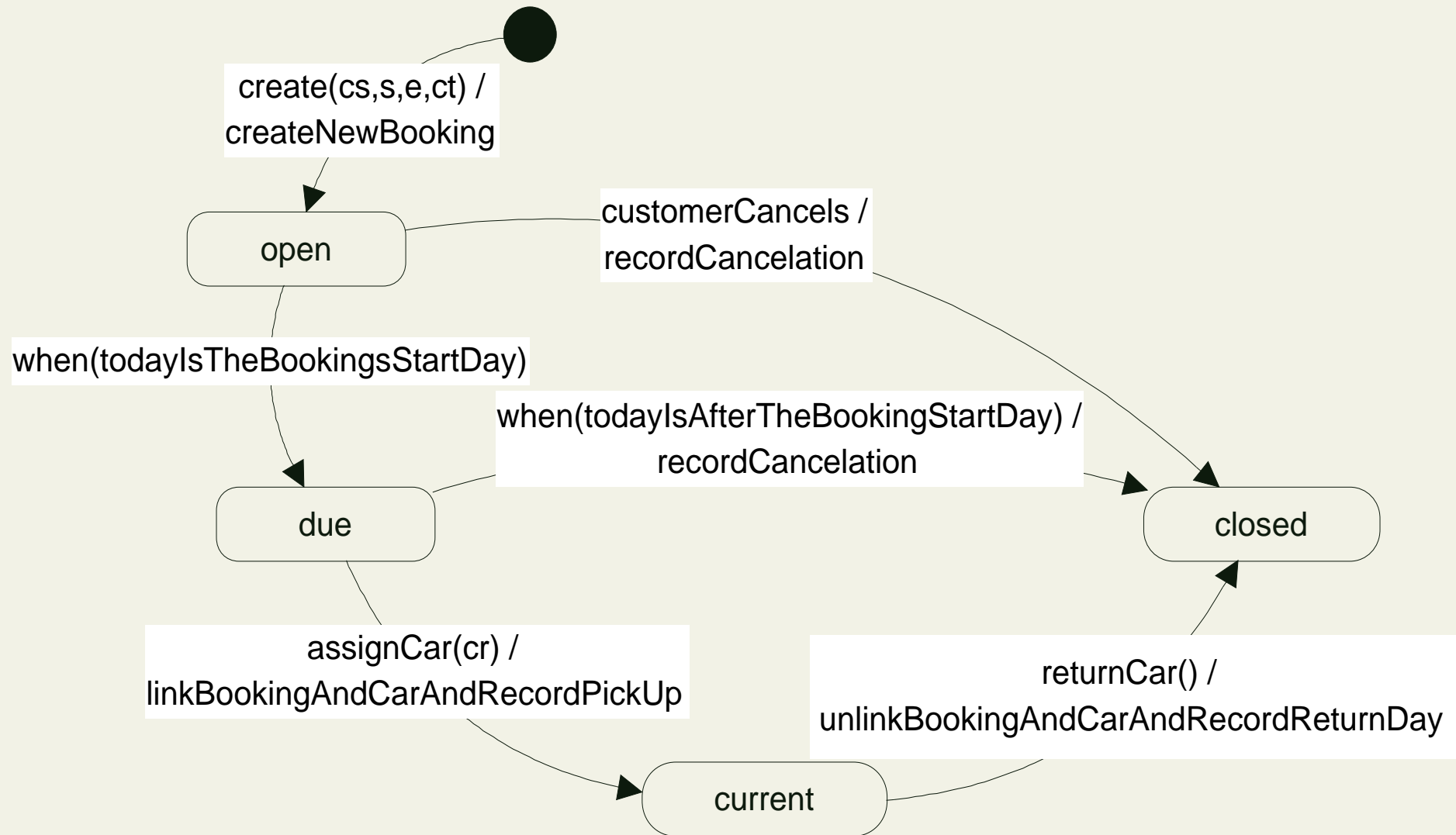
Main Idea of Customer Statechart



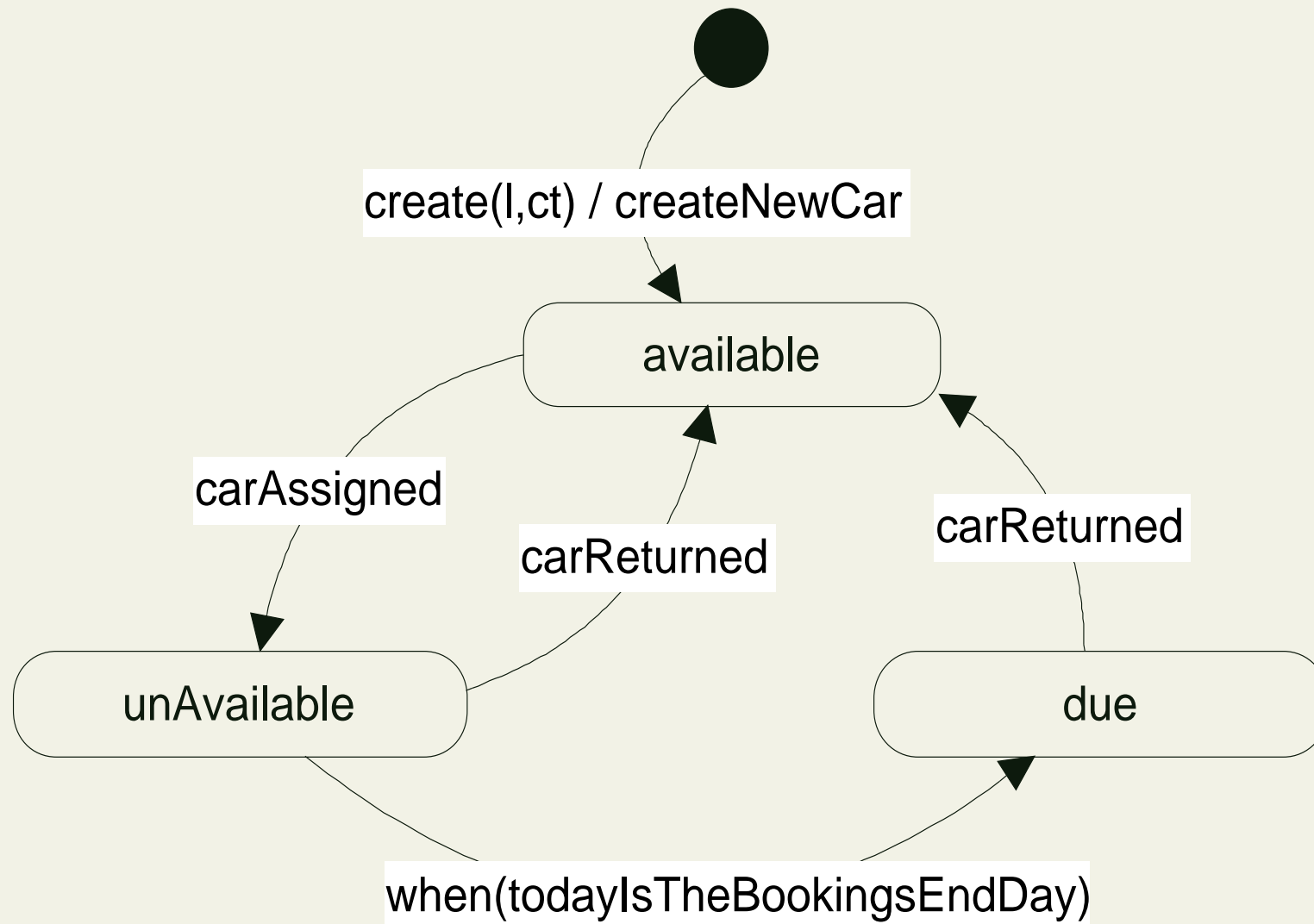
Customer Statechart with Textual Details



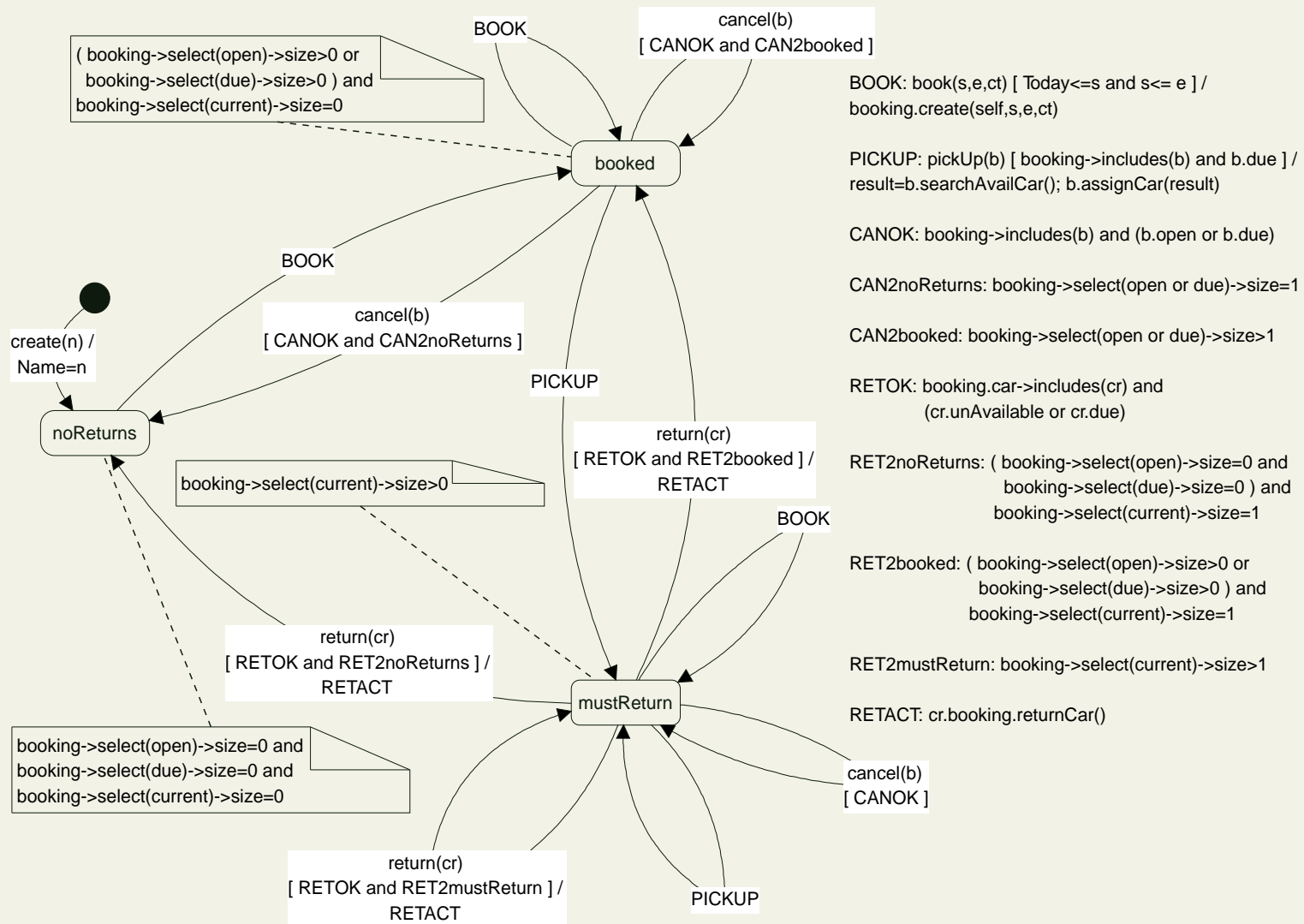
Booking Statechart with Textual Details



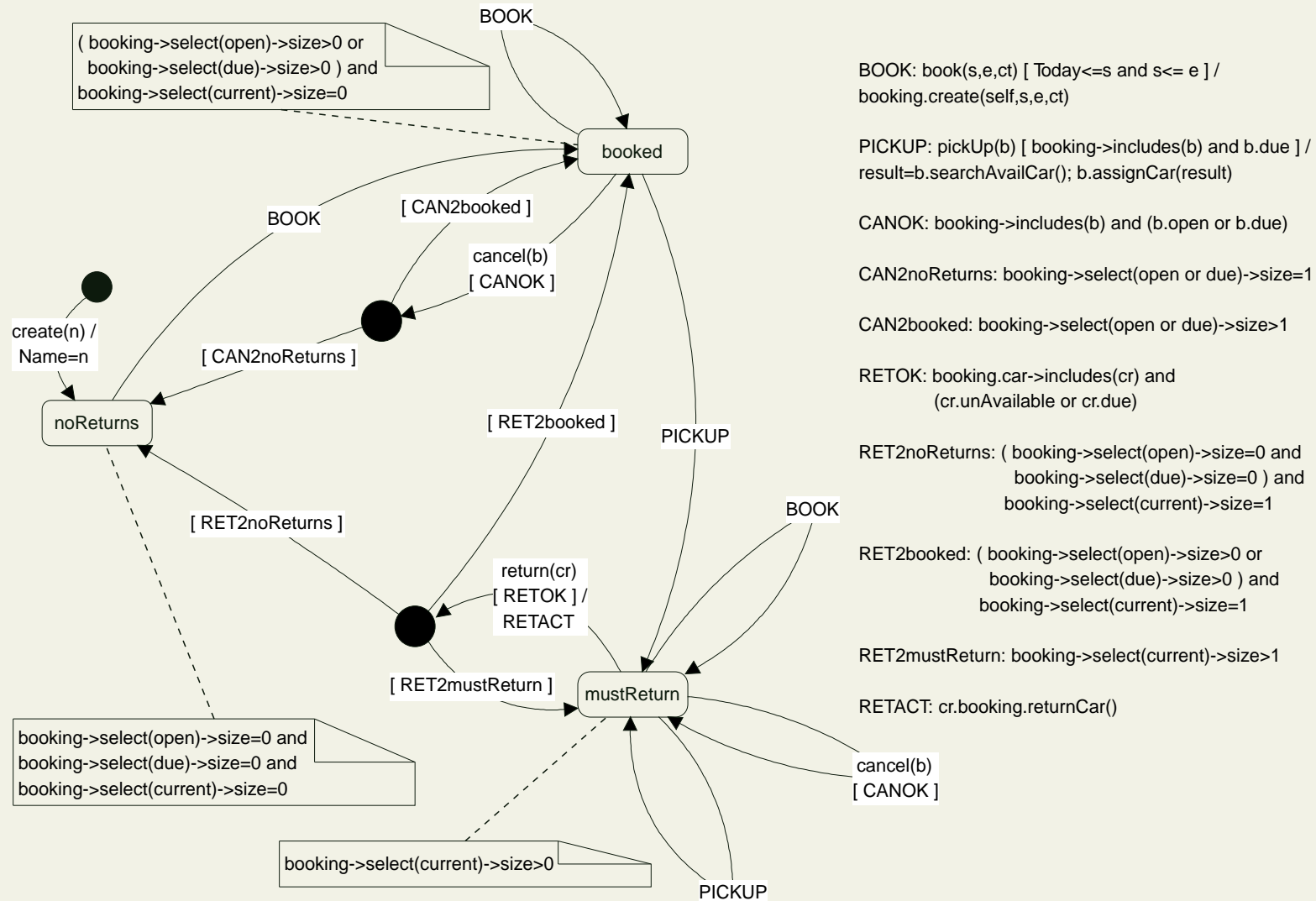
Car Statechart with Textual Details



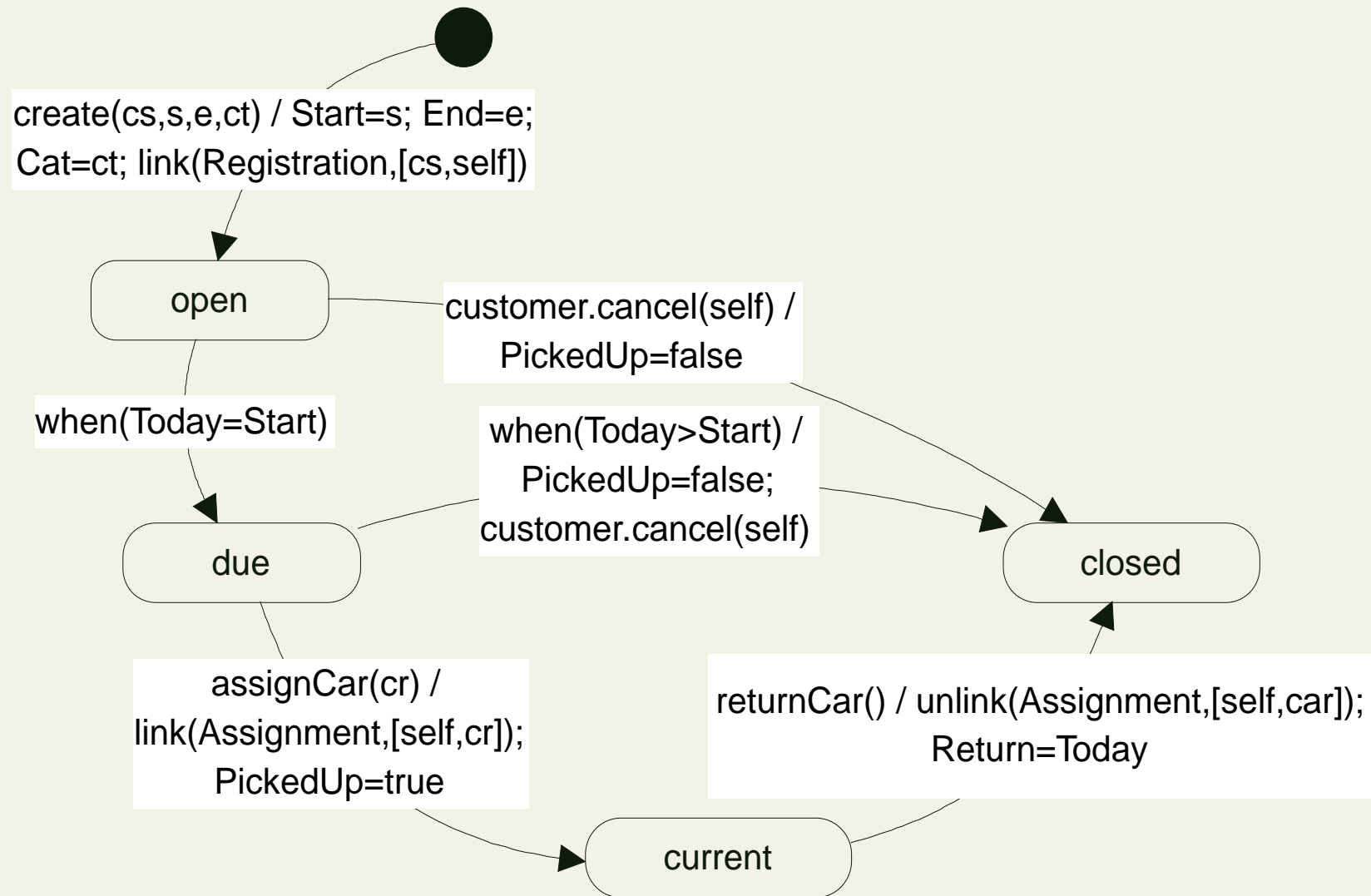
Customer Statechart with Formal Details



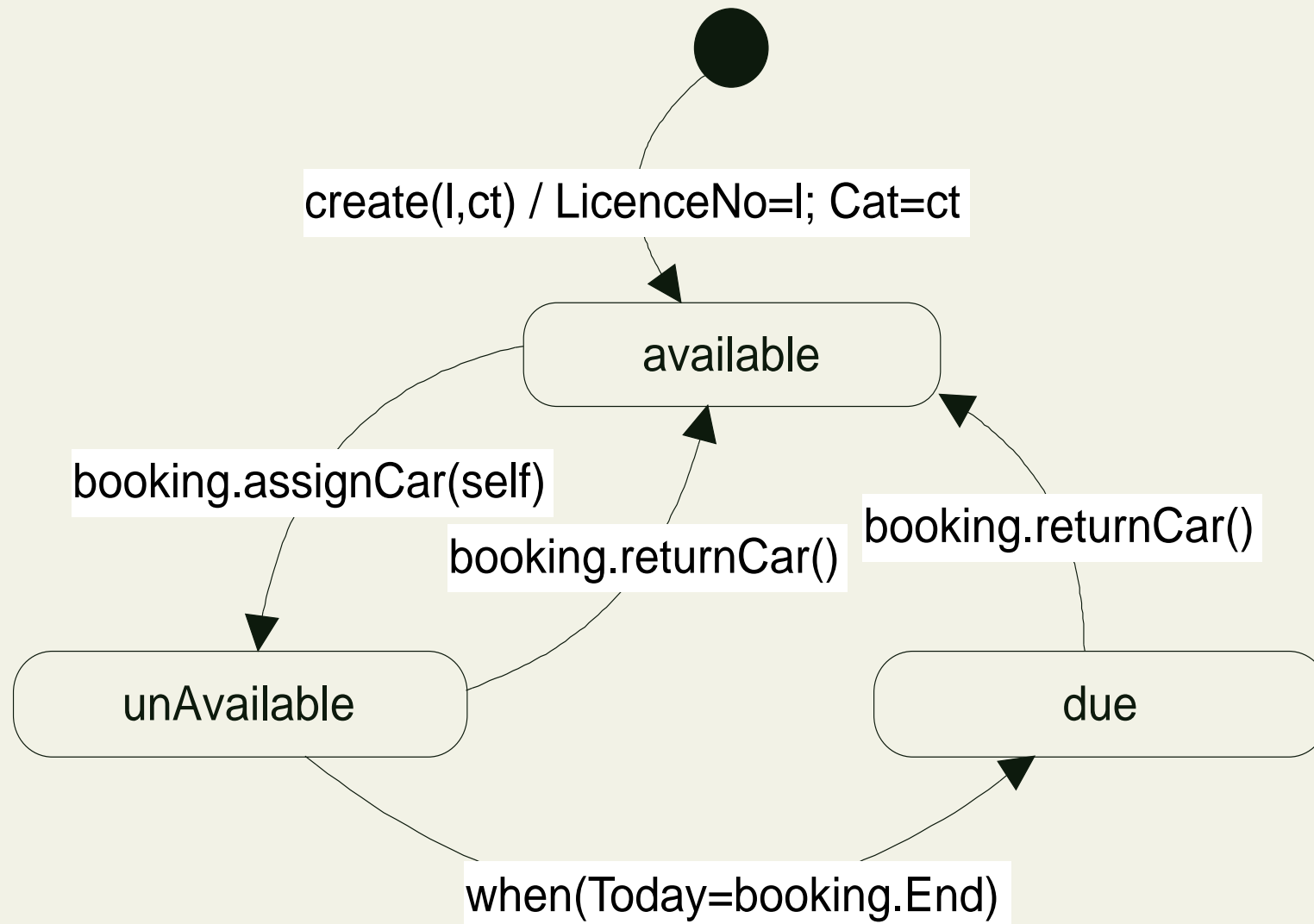
Customer Statechart with Formal Details Using Junction Points



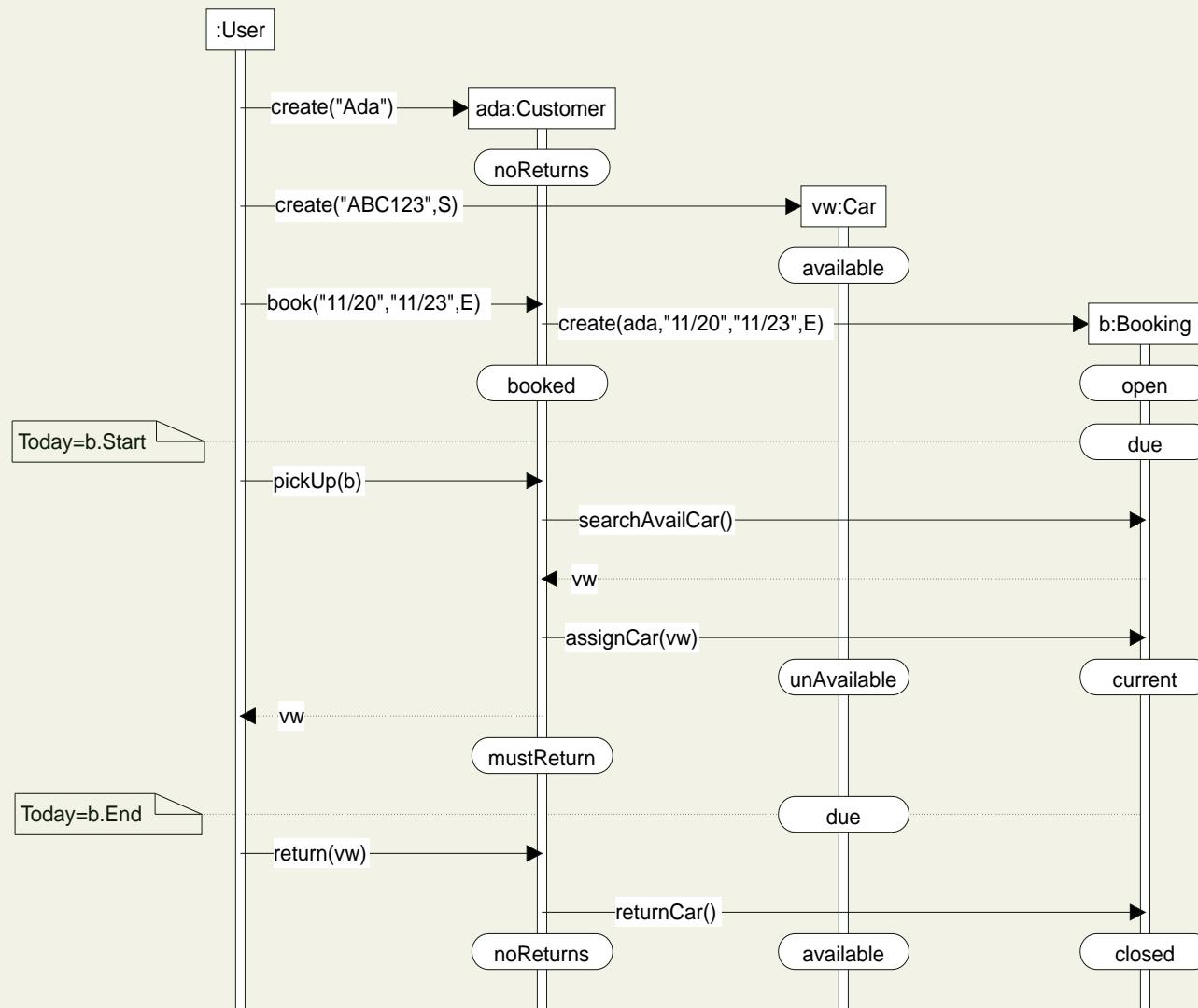
Booking Statechart With Formal Details



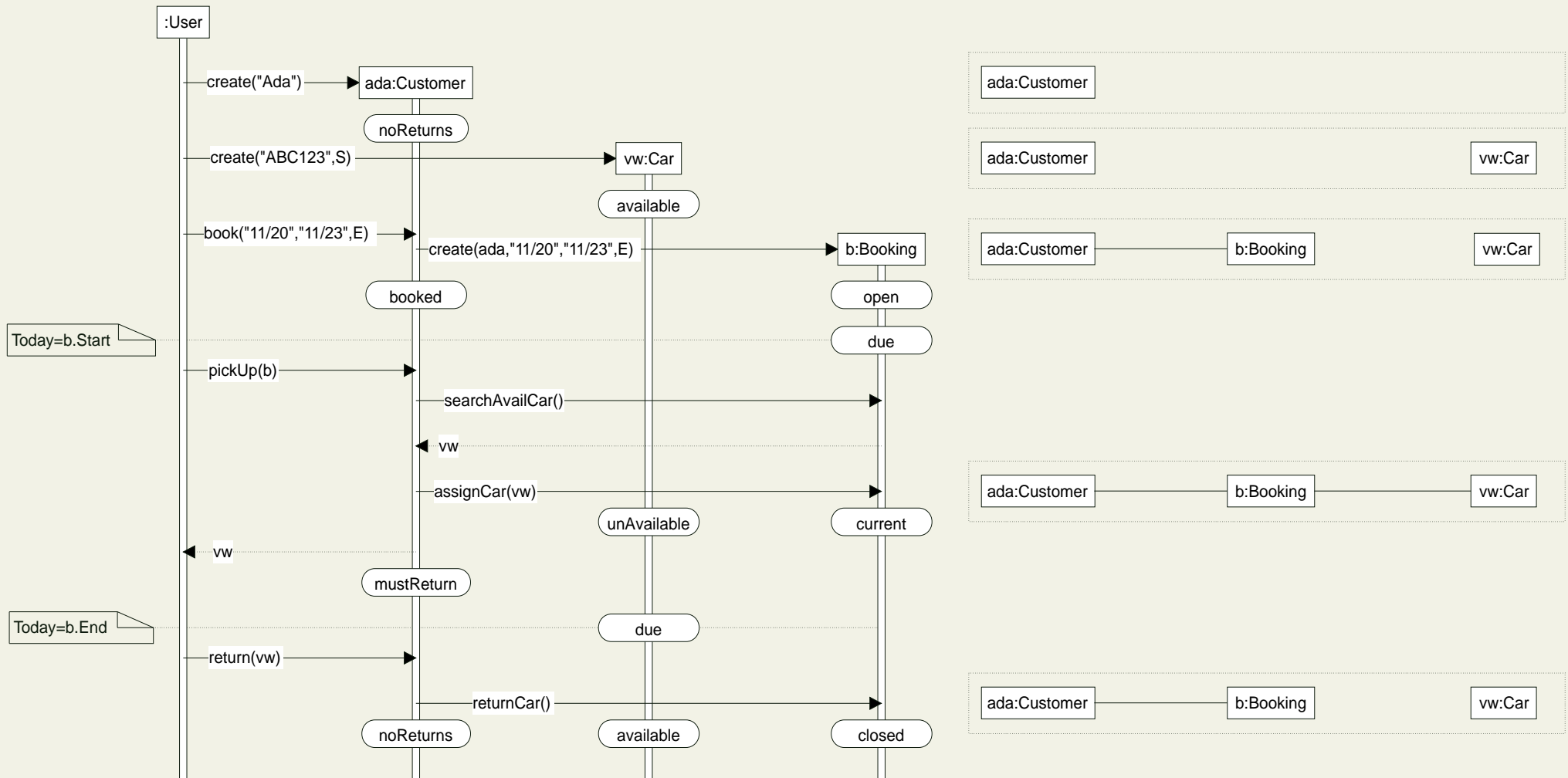
Car Statechart with Formal Details



Sequence Diagram for Booking: `Car.create`; `Booking.create`

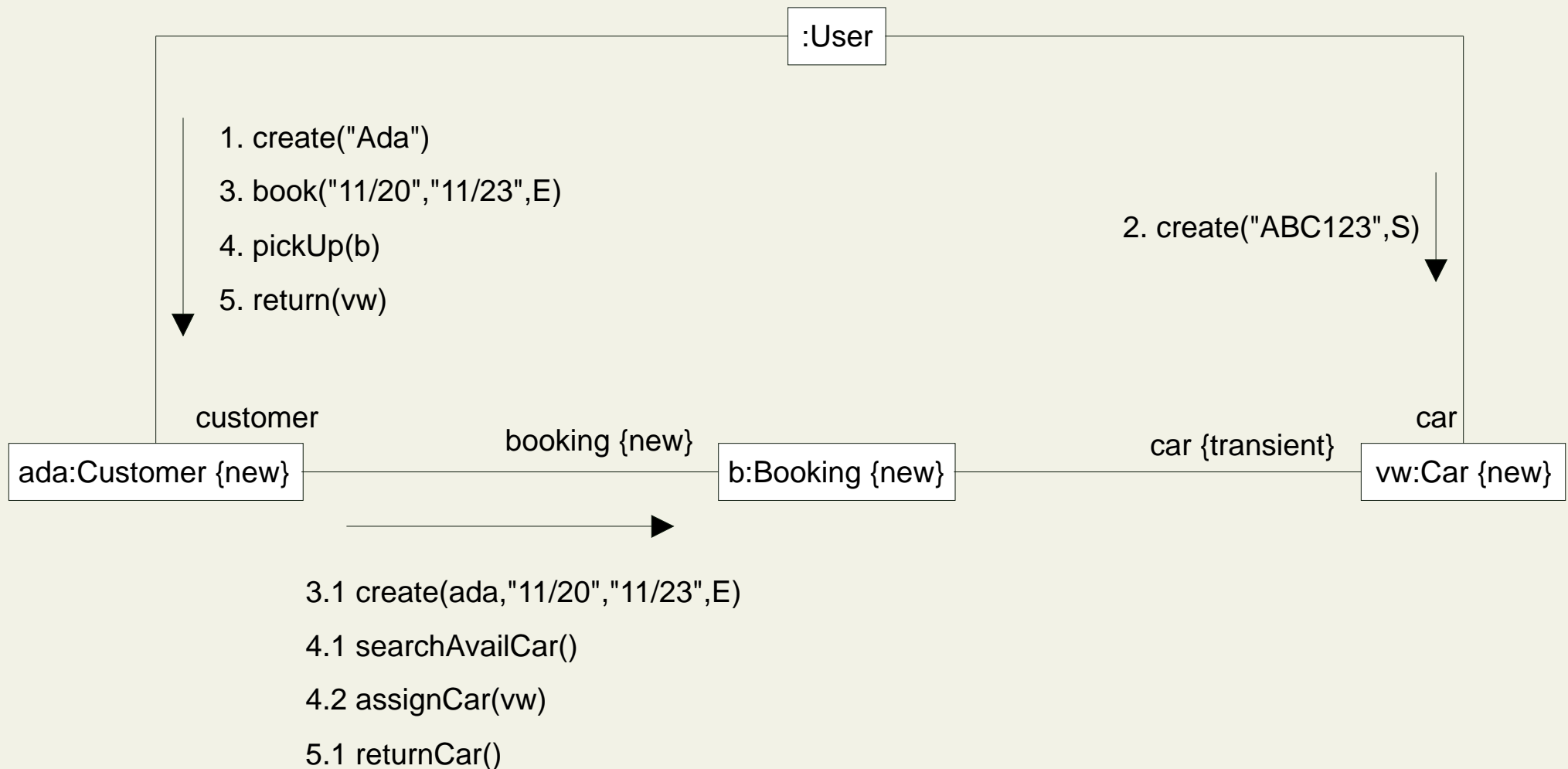


Sequence Diagram for Booking: Car.create; Booking.create (with objects)

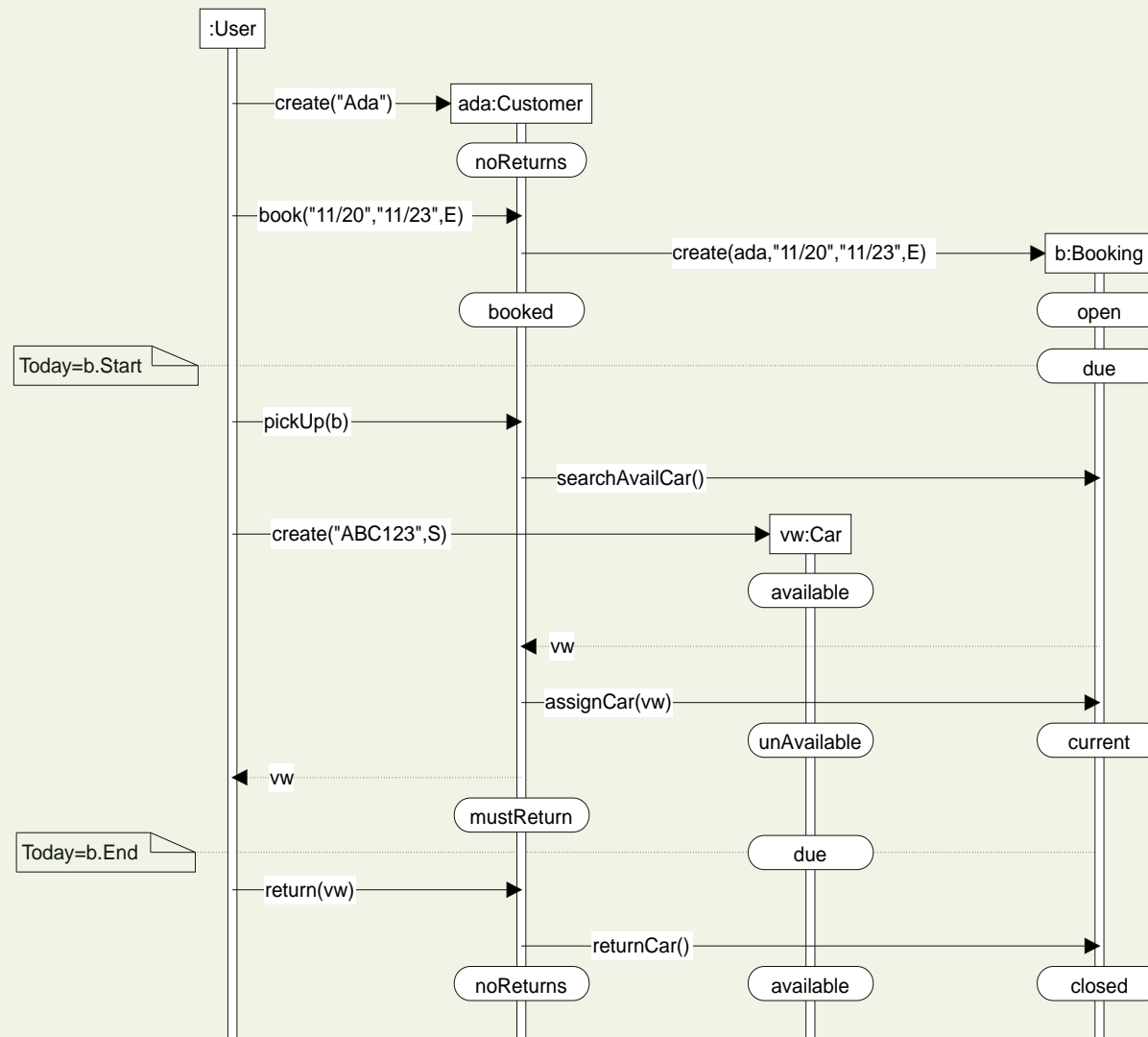


6.5 Collaboration Diagrams

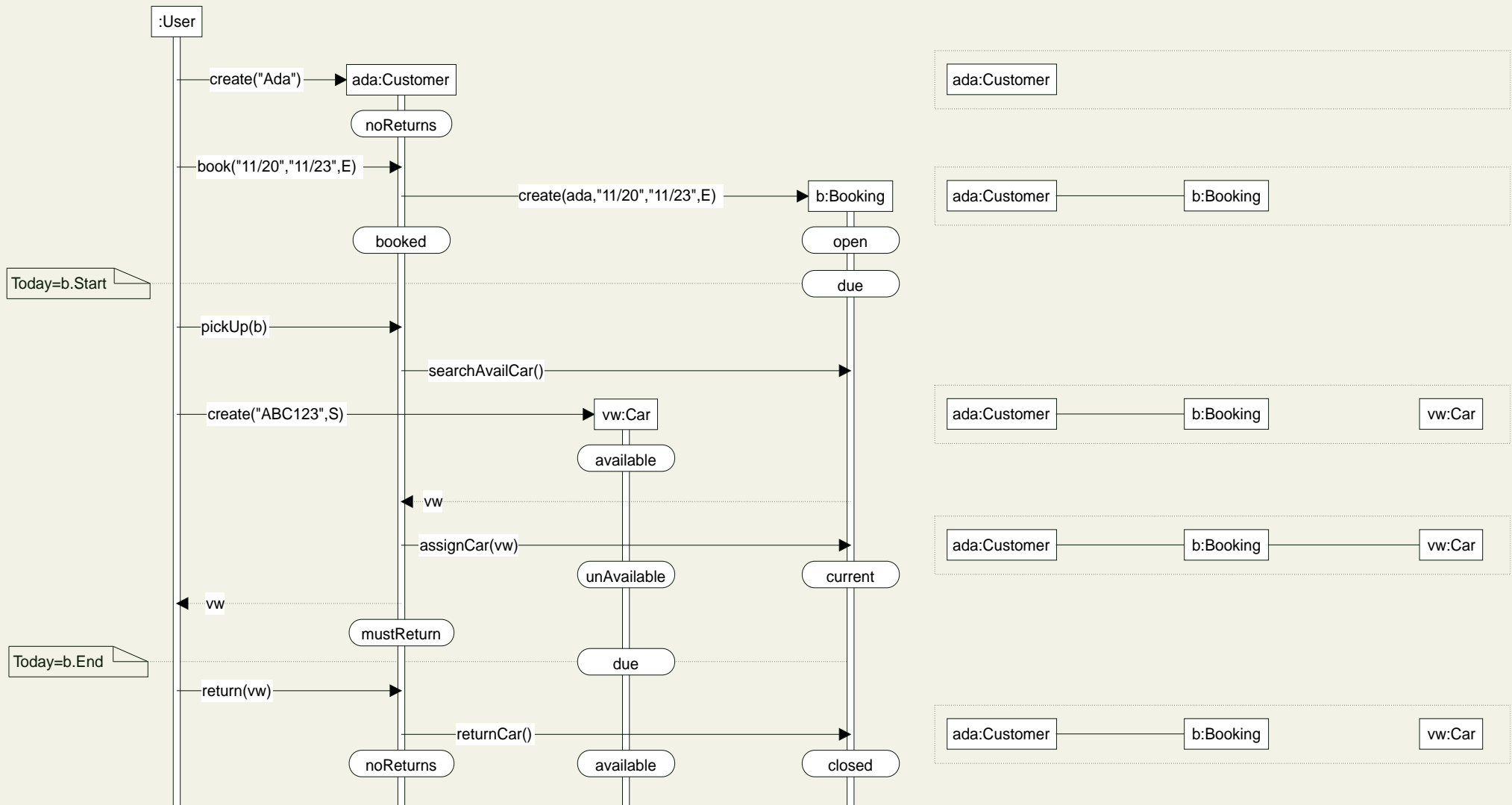
Collaboration Diagram for Booking: `Car.create`; `Booking.create`



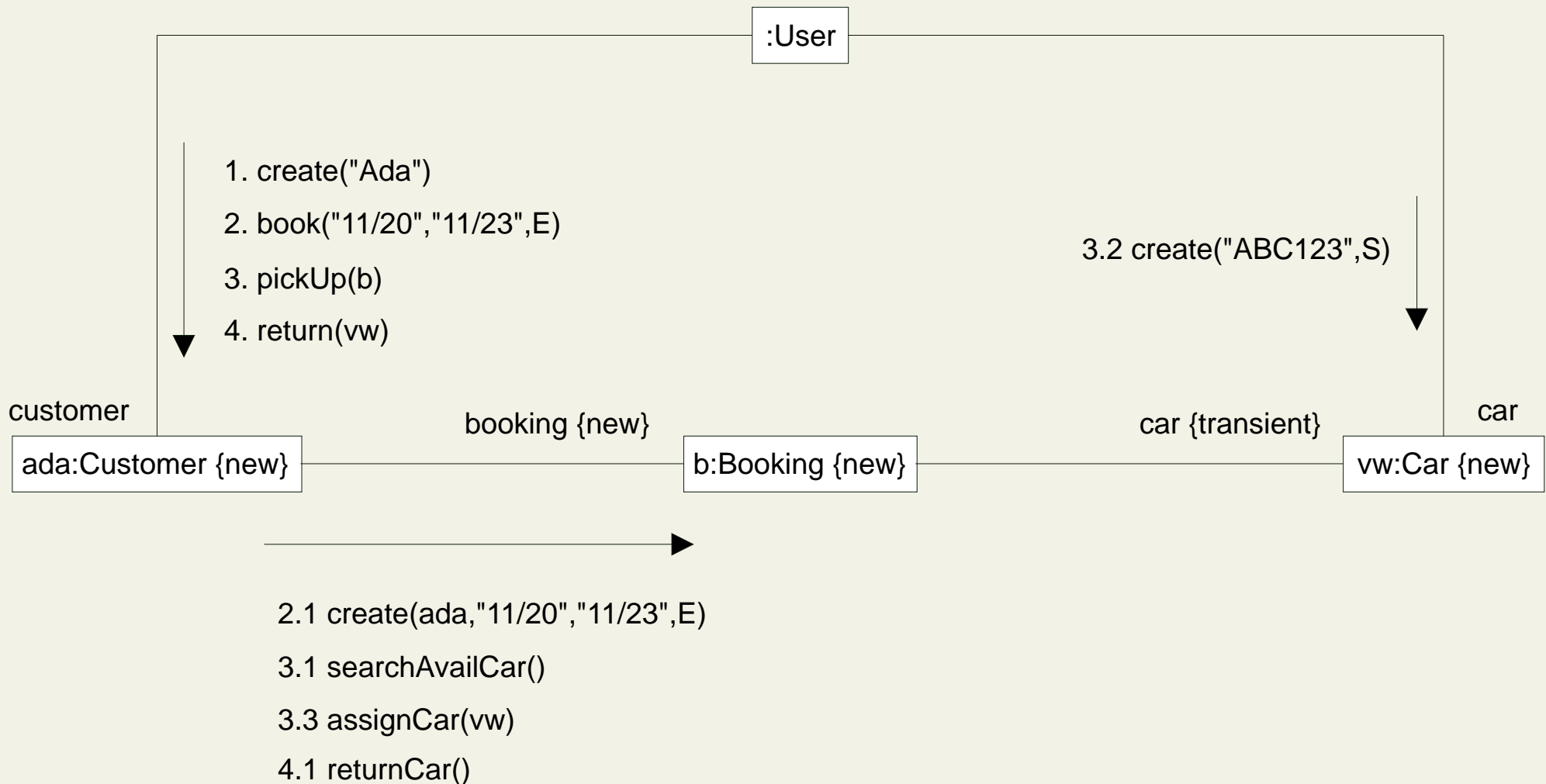
Sequence Diagram for Booking: Booking.create; Car.create



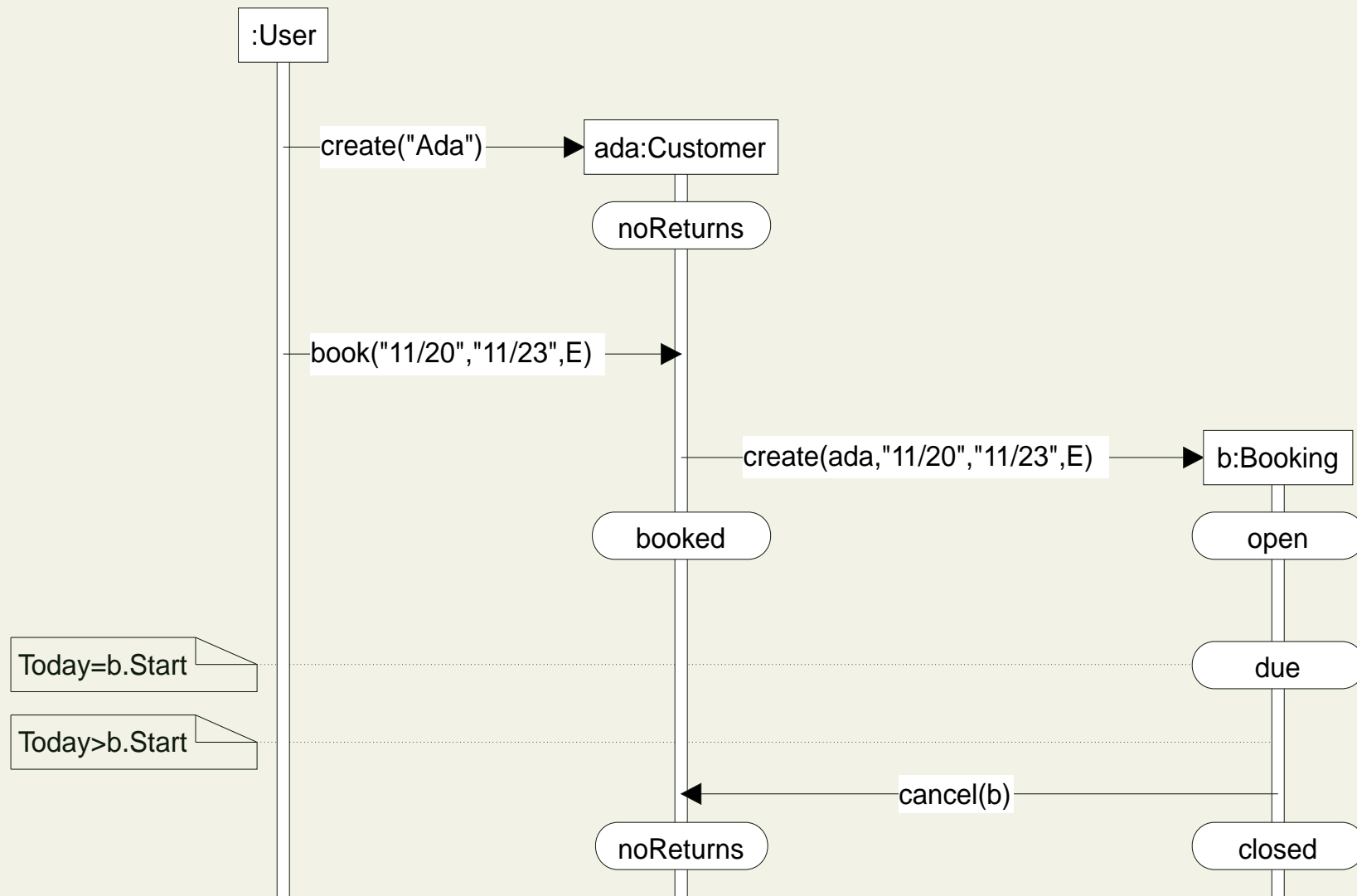
Sequence Diagram for Booking: Booking.create; Car.create (with objects)



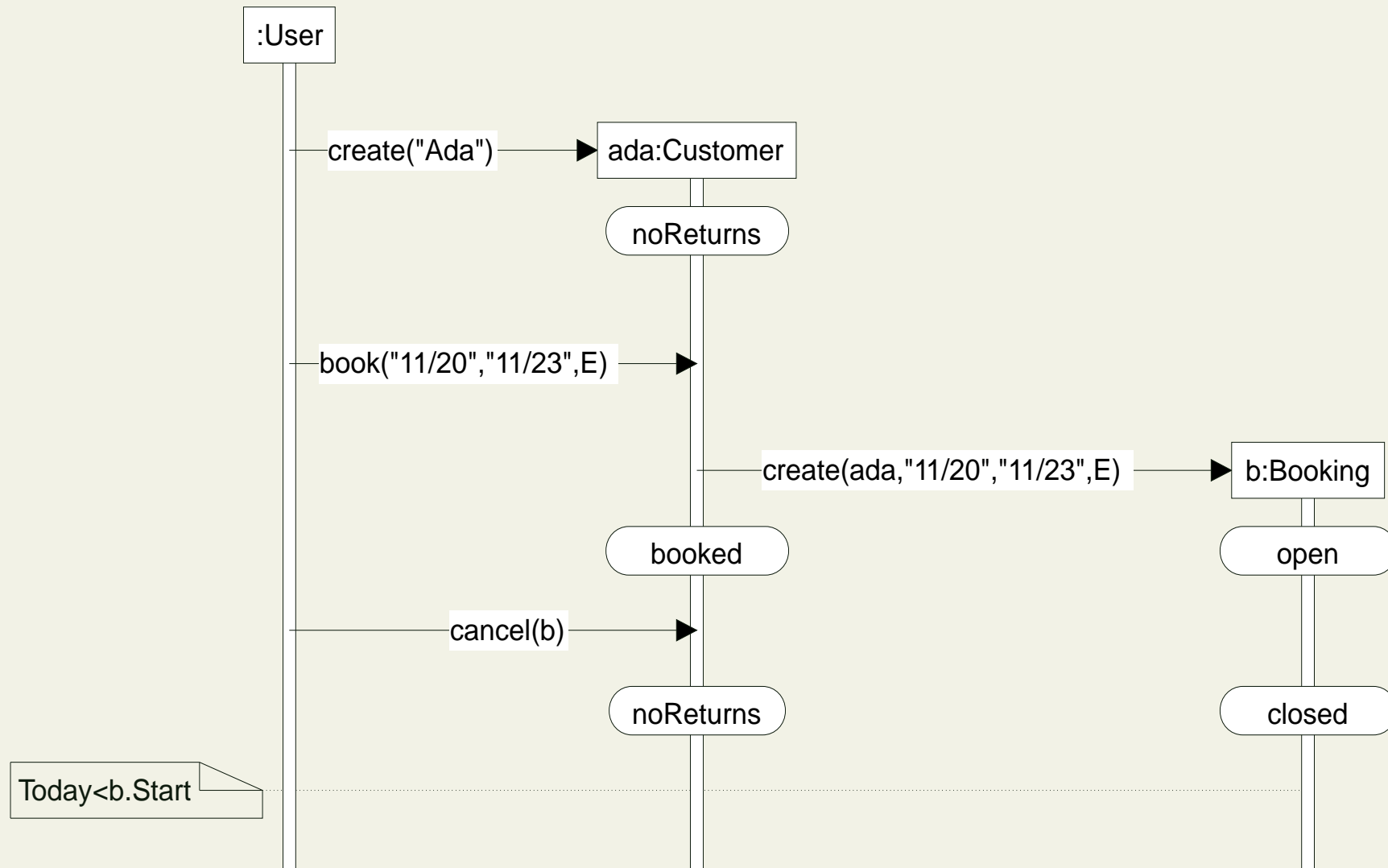
Collaboration Diagram for Booking: Booking.create; Car.create



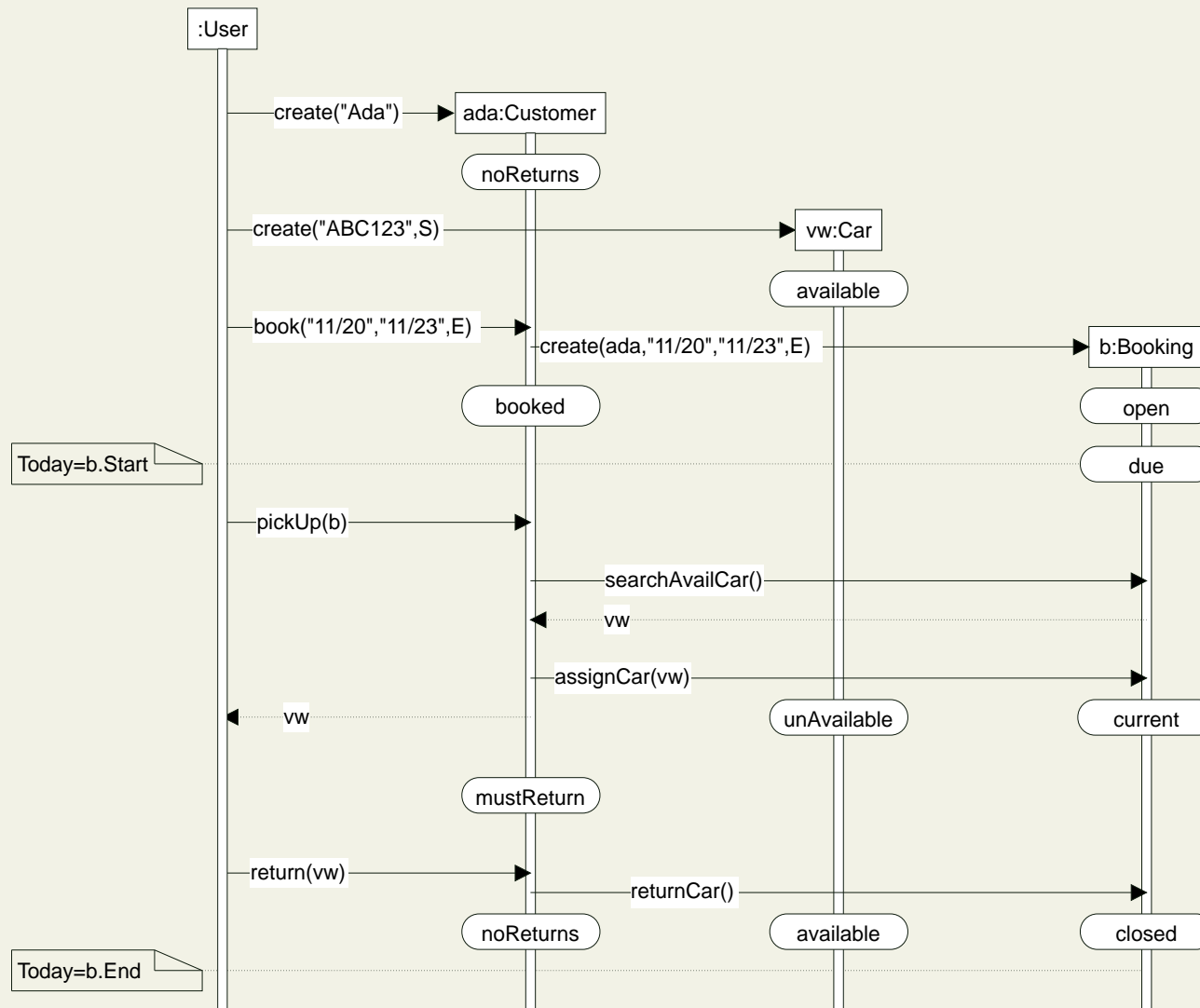
Sequence Diagram for Booking with Implicit Cancel



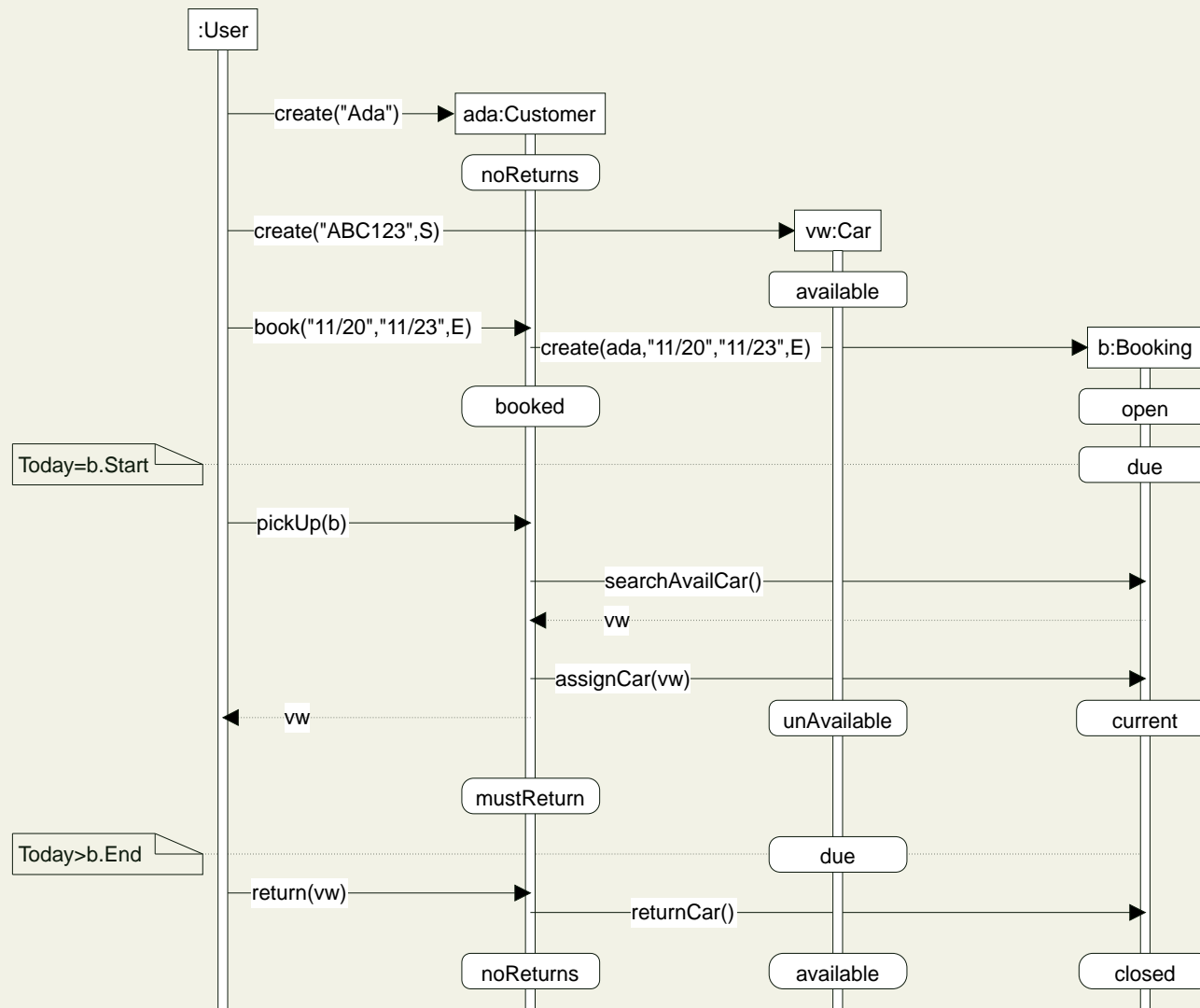
Sequence Diagram for Booking with Explicit Cancel



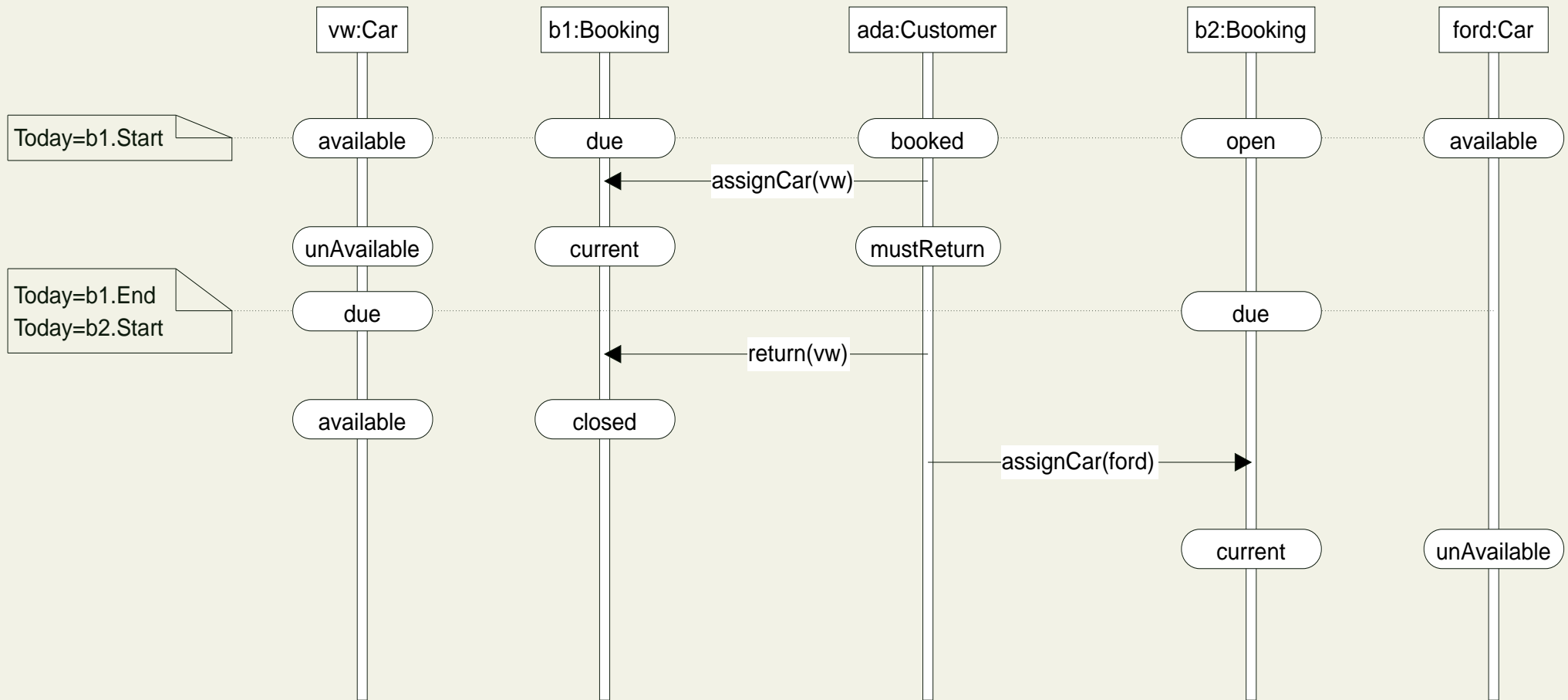
Sequence Diagram for Booking with Early Return



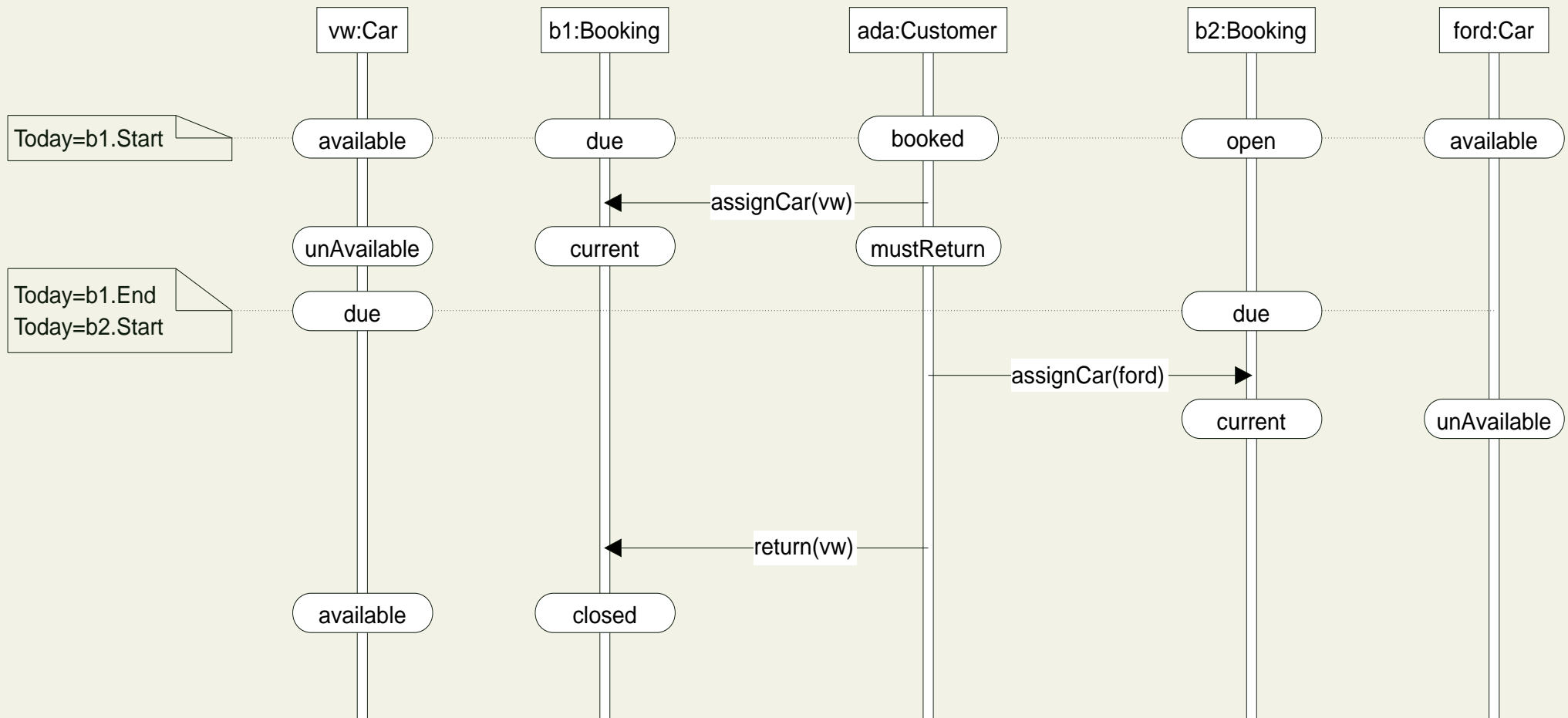
Sequence Diagram for Booking with Late Return



Sequence Diagram for Two Bookings: `b1.return;` `b2.assignCar`

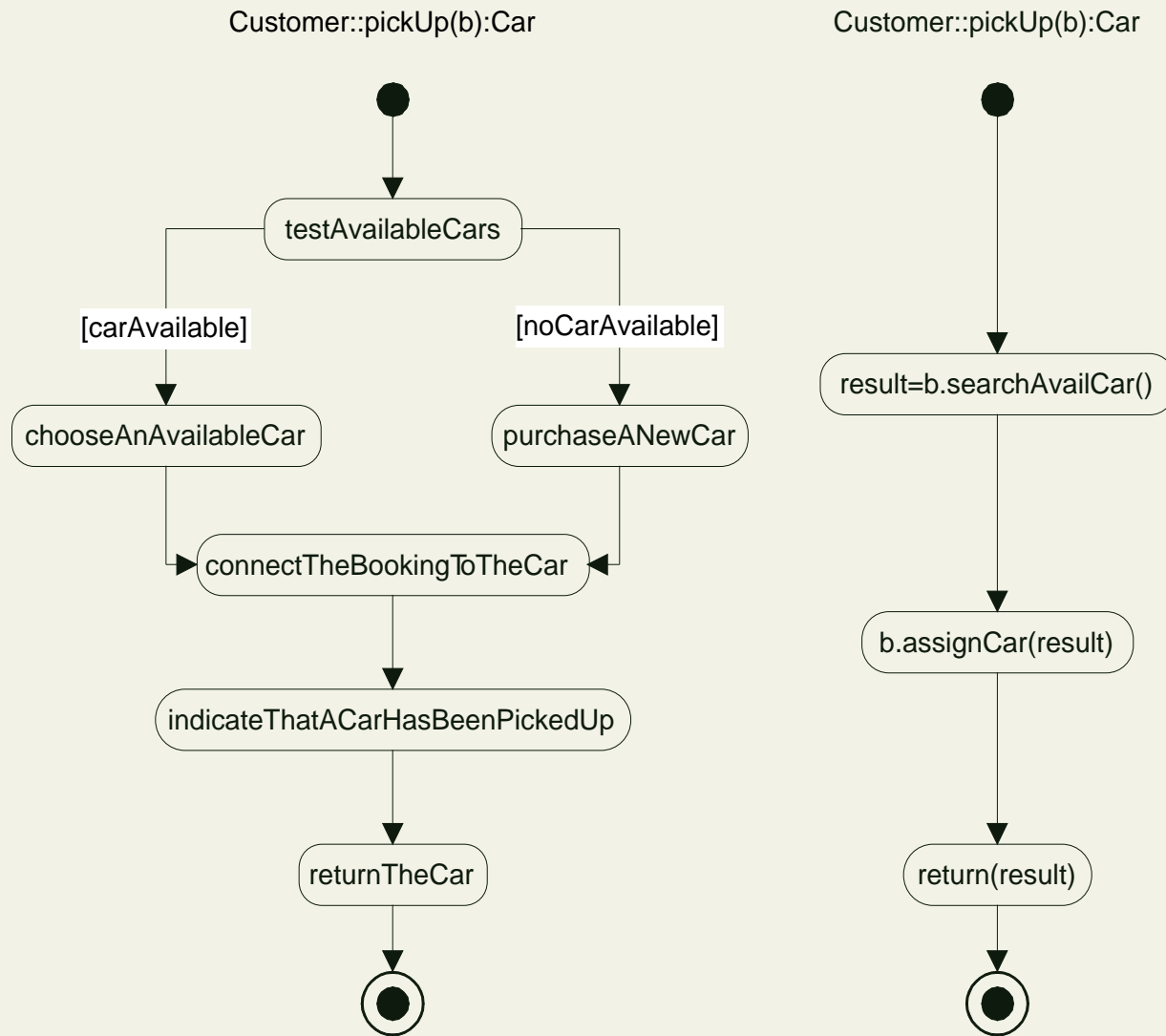


Sequence Diagram for Two Bookings: `b2.assignCar; b1.return`



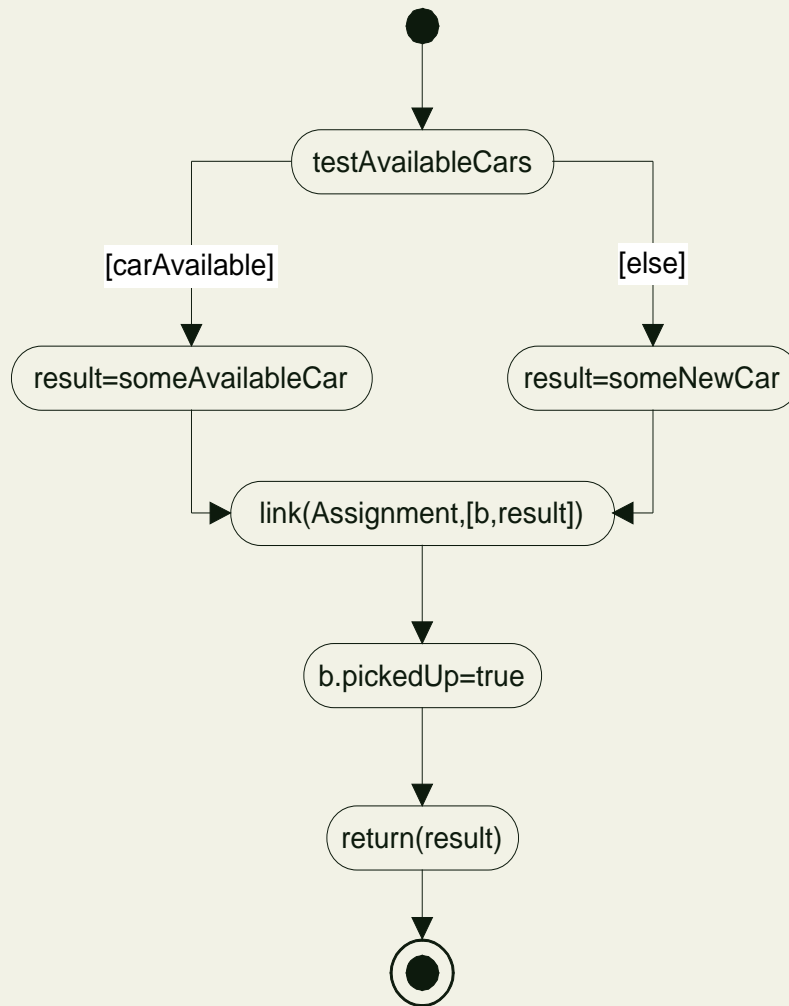
6.6 Closing

Activity Diagrams for `pickUp`: Textual Level and Operation Level

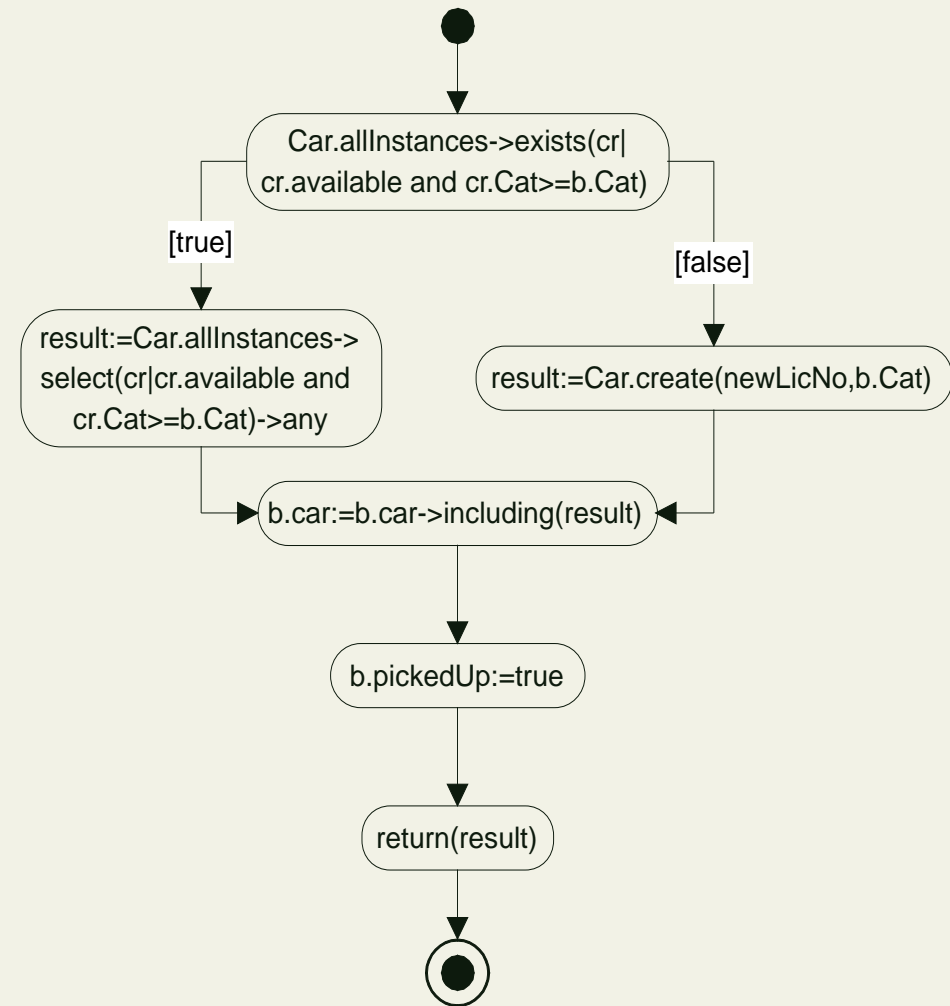


Activity Diagrams for `pickUp`: Statement Level and Formal Level

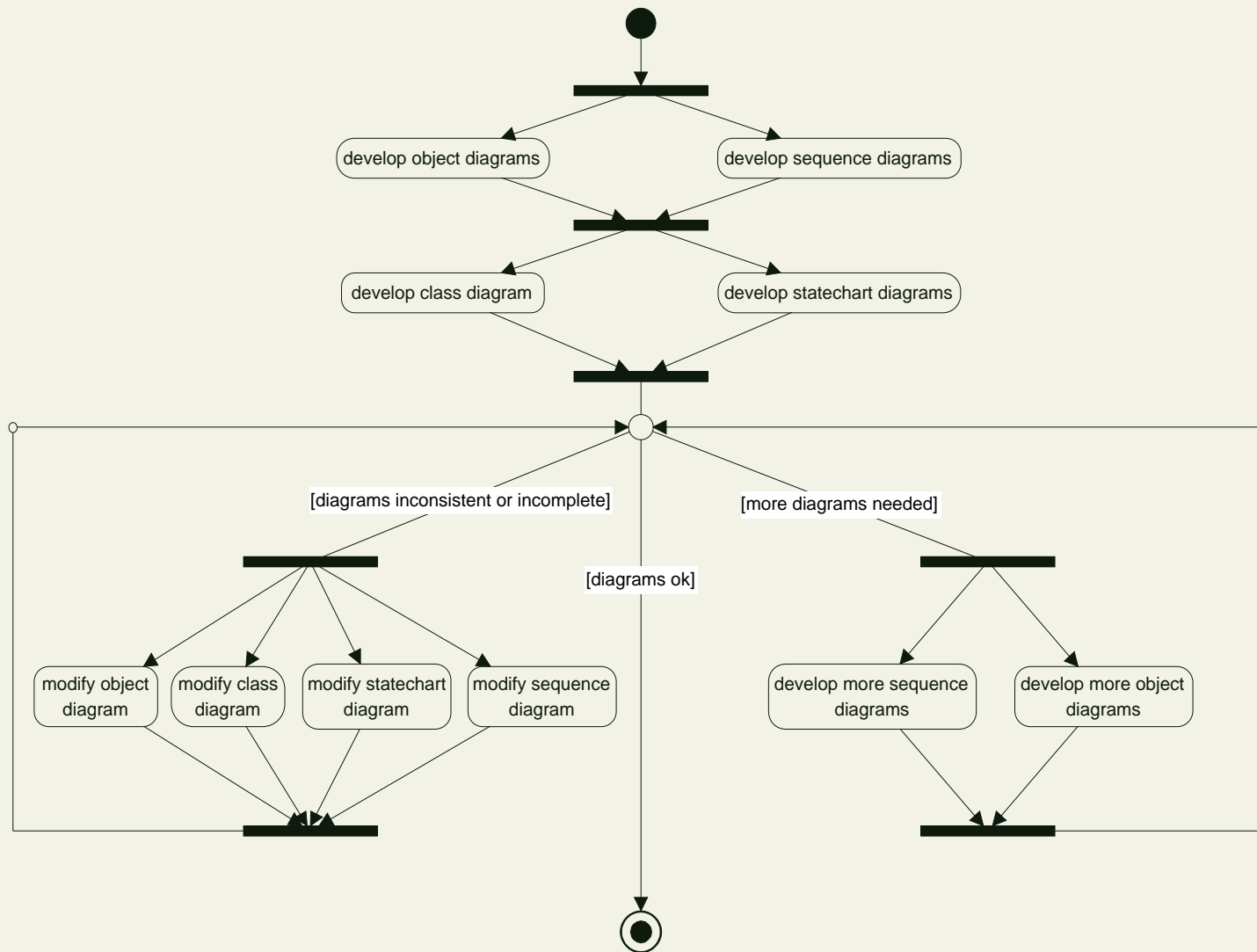
Customer::`pickUp`(b):Car



Customer::`pickUp`(b):Car



Developing a Consistent Model



Details for Diagram Consistency

From class diagram to object diagrams

- for each class there is at least one object diagram with an object of that class
- for each attribute and role name there is at least one object diagram with an object with that attribute and role name

From class diagram to sequence diagrams

- for each class there is at least one sequence diagram with an object of that class
- for each operation there is at least one sequence diagram with that operation as a message

From class diagram to statechart diagrams

- for each attribute there is at least one operation that modifies that attribute in some statechart
- for each operation there is at least one statechart where that operation occurs as a call event or as a call action

From statechart diagrams to class diagram

- each call event refers to an operation in a class
- each attribute and role name occurring in a guard refers to an attribute in a class and a role name of a class
- each call action, assignment action, and link/unlink action refers to an operation, attribute, and role name in a class

From object diagrams to class diagram

- each class of an object occurs as a class
- each attribute and role name refers to an attribute in a class and a role name of a class

From sequence diagram to class diagram

- each object refers to a class
- each message refers to an operation in a class

From sequence diagram to statechart diagram

- each state refers to a state in the statechart diagram
- each message sequence is allowed by at least one statechart event and action order
- each state sequence is allowed by at least one state sequence order

From statechart diagram to the sequence diagram

- each call action completely occurs in the sequence diagram (all sub-actions from the call action occur)

