10.2 Flight example

Description

In this chapter we illustrate nested classes by an example in the domain of flights and flight routes.

An airline company like SAS has a timetable with all their flight routes with flights, e.g. SK SK 1926 from Aarhus in Denmark to Oslo in Norway, and SK 1927 from Oslo to Aarhus.

A time table is used for two different purposes: as a basis for bookings, e.g. as in the chapter on travel bookings, and as a basis for a website that shows the status of flights at a specific airport, i.e. whether flights are on time, cancelled, delayed, have a new scheduled time of arrival, have arrived and at what time, etc.

Time tables, flight routes and flights are obviously represented by objects. A timetable has an entry for each of the different flight routes, and in the model each entry is represented by an object of class `FlightRoute`:

```
timeTable: obj
   entries: obj OrderedList(FlightRoute)
```

Each `FlightRoute` object has attributes that represent the flight number of the flight route, the source and destination airports, the scheduled departure and arrival time, and scheduled flying time. The flights of a route is represented by a list of `Flight` objects for each `FlightRoute` object.

```
class FlightRoute(flightNumber, origin, destination: var String):
   scheduledDepartureTime: var TimeOfDay
   scheduledArrivalTime: var TimeOfDay
   setTime(dt: var TimeOfDay, at: var TimeOfDay):
      scheduledDepartureTime := dt
      scheduledArrivalTime := at

   flights: obj OrderedList(Flight)
```

`scheduledDepartureTime` and `scheduledArrivalTime` are used for showing flight status, at airports or at the flight status website of the airline company. The type `TimeOfDay` is defined in section .

Setting up the flight routes of the time table is done by a sequence of computations that generate `FlightRoute` objects, set the values of their attributes and insert them into the `timeTable`:

```
   SK1926: obj FlightRoute("SK1926", "AAR", "OSL")
   Sk1926.setTimes(TimeOfDay(12.30 hours), TimeOfDay(13.50 hours))

   timeTable.entries.insert(SK1926)
   ...
   SK1927: obj FlightRoute("SK1927", "OSL", "AAR")
   SK1927.setTimes(TimeOfDay(14.20 hours), TimeOfDay(15.40 hours))

   timeTable.entries.insert(SK1927)
   ...
```

As mentioned above, flights are represented by objects. As the notion of flight is defined in the context of flight route, the

class `Flight` is defined in the context of the class `FlightRoute`. This is done by nesting the description of class `Flight` in the description of class `FlightRoute`:

```
class FlightRoute(flightNumber, origin, destination: ref String):
   scheduledDepartureTime: var TimeOfDay
   scheduledArrivalTime: var TimeOfDay
   -"-
   class Flight(departureDate: var Date):
      departureTime: var TimeOfDay
      arrivalTime: var TimeOfDay
      delayDeparture(newTime: var Time.Hours):
         delayed := True
         departureTime := newTime
      delay -> period: var Time.Hours:
         period := arrivalTime – scheduledArrivalTime
```

The type `Date` is defined in section and the type `Time.Hours` is defined in section :

Note that `FlightRoute` objects keep the flights for any date, while each `Flight` object has its departure date as an attribute.

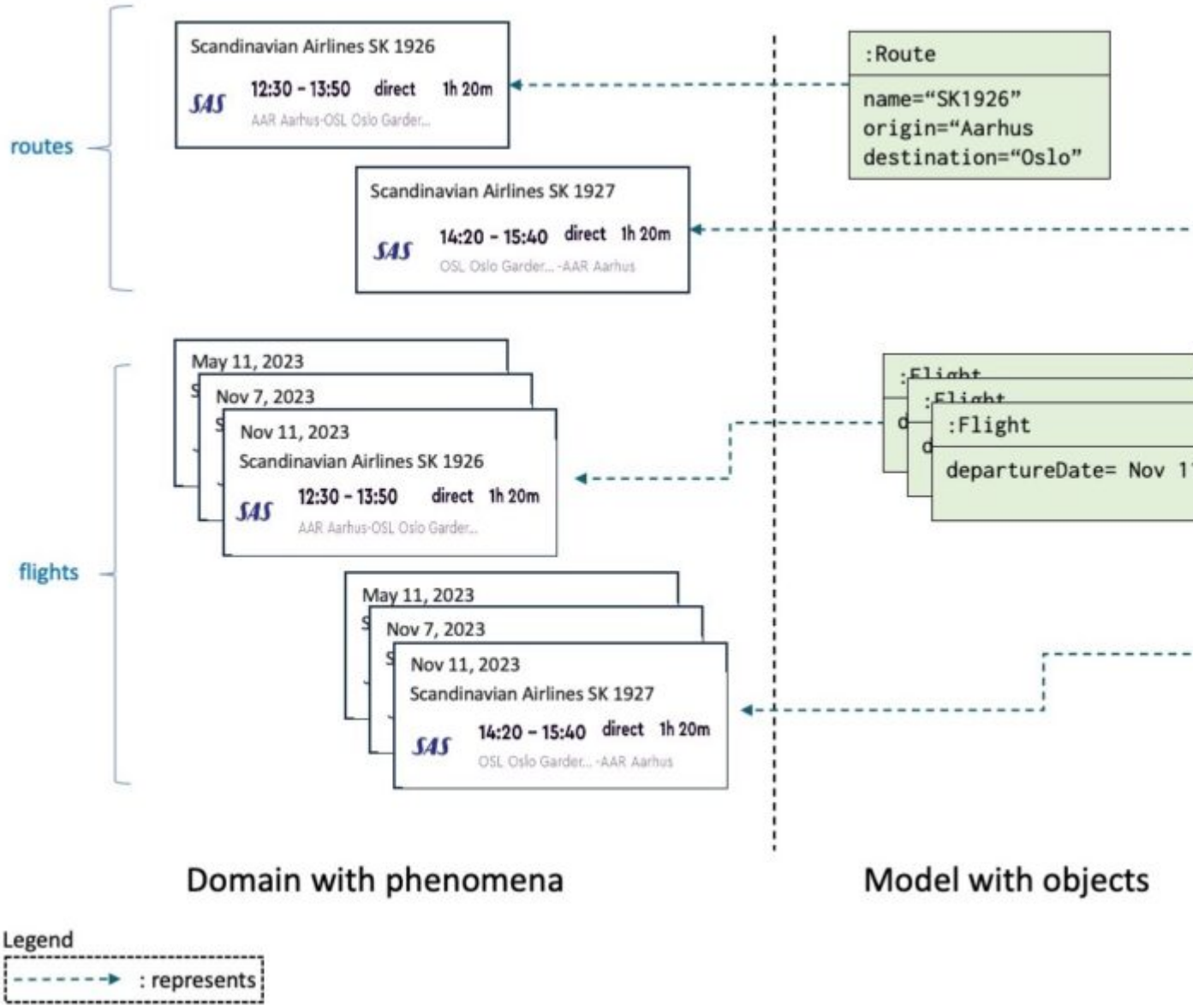
Departure time is represented by `departureTime`. Before take off it represents the scheduled/estimated departure time. After take off it represents the actual departure time.

The scheduled arrival time is common to all flights on a flight route, while the arrival time of a flight is represented by a variable. Before take off this is set to the `scheduledArrivalTime` and therefore represent the scheduled/estimated departure time. While flying it is set based upon flying conditions and landing conditions at the destination airport to represent the expected arrival time. After landing it represents the actual arrival time.

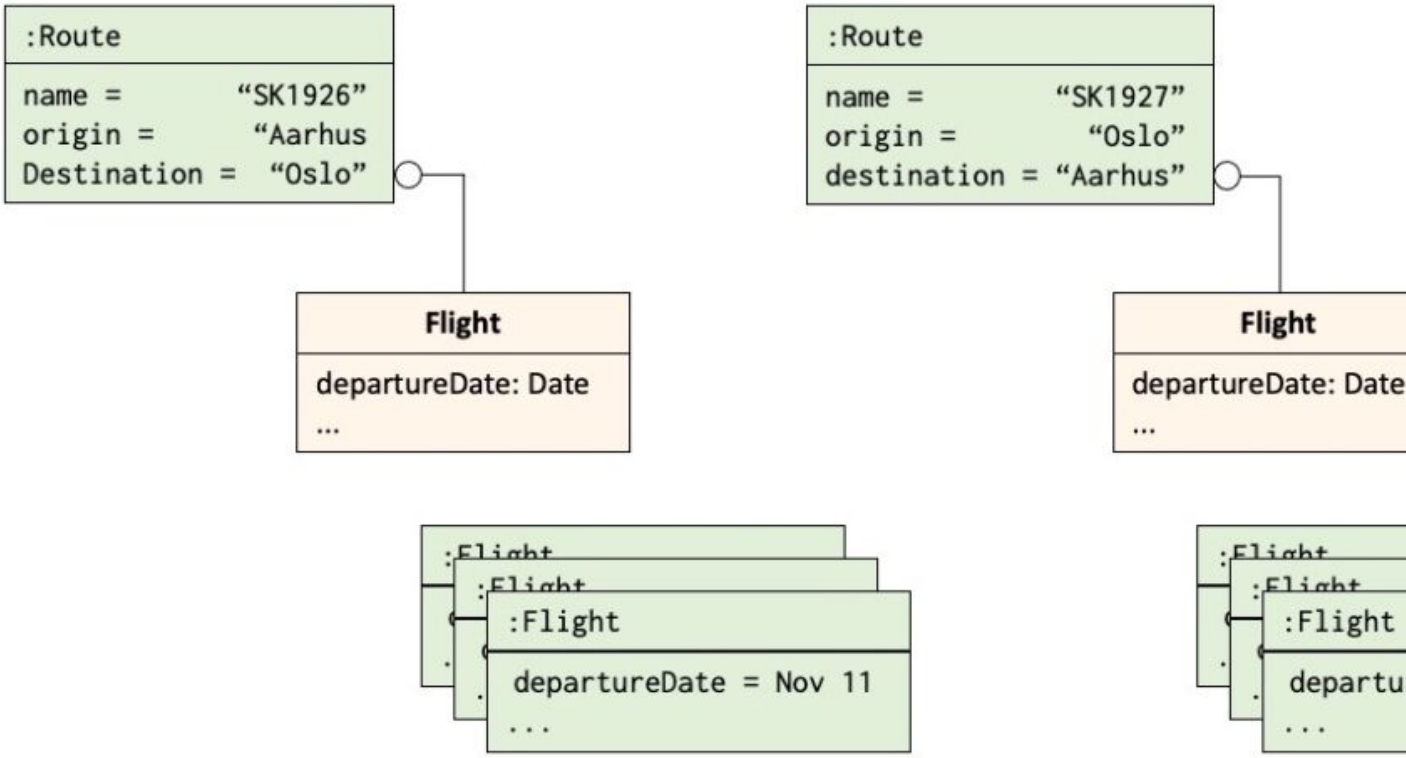
If the flight is delayed, the method `delayedDeparture` may be used to set the estimated departure time. In addition to assign the parameter `newTime` to `DepartureTime`, `delayed` is set to `True`.

As we have defined the class `FlightRoute`, so that each object of class `FlightRoute` represents a specific flight route, the class `Flight` is therefore defined in the context of class `FlightRoute`. A specific `FlightRoute` object will thereby have its own class `Flight` of objects representing flights on this flight route. Another `FlightRoute` object will have another class `Flight`.

The modeling of flight routes and flights is illustrated in the following figure. Each flight route is represented by a `FlightRoute` object, and the corresponding flights are represented by `Flight` objects held by the corresponding `FlightRoute` object.



The fact that class `Flight` is described nested in the description of class `FlightRoute` is illustrated in the next figure, using a circle-enclosed '+' annotated relation between objects of class `FlightRoute` and the class `Flight`. A specific `FlightRoute` object will thereby have its own class `Flight` of objects representing flights on that flight route. Another `FlightRoute` object will have another class `Flight`.



FlightRoute objects with nested Flight classes and corresponding objects

The following illustration shows how nesting is used to compute the delayas the difference between `scheduledArrivalTime` and `arrivalTime`. By nesting the `Flight` class in the `FlightRoute` class, the attributes of `FlightRoute` are directly visible in class `Flight`. The method `delay` (in class `Flight`) may therefore compute the delay of the flight by subtracting the `scheduledarrivalTime` (in the enclosing `FlightRoute`) from the local `Flight` property `ArrivalTime`:

```
class FlightRoute(flightNumber, origin, destination: ref String):
   scheduledDepartureTime: var TimeOfDay
   scheduledArrivalTime: var TimeOfDay
   _"_
      class Flight(departureDate: var Date):
         departureTime: var TimeOfDay
         arrivalTime: var TimeOfDay
         _"_
         delay -> period: var Time.Hours:
            period := arrivalTime - scheduledArrivalTime
```

As mentioned above, we shall use this model of as the difference between `scheduledArrivalTime` and `arrivalTime`

and flights for both booking of flights and for showing status of flights. In the next section, we look at what is required for booking, and then for the status of flights.