10.2.2 Status of flights

Description

In order to provide status on flights, we extend the first definition of class `Flight` with a couple of data-items, and later by some methods that provide the status:

```
   class FlightRoute(FlightNumber, origin, destination: var String):
      scheduledDepartureTime: var TimeOfDay
      scheduledArrivalTime: var TimeOfDay
      -"-
      class Flight(departureDate: var Date):
         departureTime: var TimeOfDay
         arrivalTime: var TimeOfDay:
         flightTime: -> ft: var Time.Hours
            ft := arrivalTime - departureTime
         delayed: var Boolean
         delayDeparture(newTime: var
 TimeOfDay):
            -- this is called in case the departure is delayed
            delayed := True
            departureTime := newTime
         delay -> period: var Time.Hours:
            period := arrivalTime – scheduledArrivalTime

         cancelled: var Boolean
         cancel:
            cancelled := True
         hasArrived: var Boolean
         hasTakenOff: var Boolean
```

In the event of cancellation, the method `cancel` is called.

At take off, `departureTime` is set to the time of take off, `hasTakenOff` is set to `True`, and `hasArrived` is set to `False`. While flying the attribute `arrivalTime` is set based on weather condition and the landing condition of the destination airport. It is therefore assumed that this is set based on real time information from the plane. At arrival, `hasArrived` is set to `True` and `hasTakenOff` to `False`.

Status of flights are provided in two different ways, either given the origin/destination airport at a given date, or given the name of the flight route, e.g. SK1926.

The method `departureStatus` defined below is called with some interval before take off, to provide the text that may be displayed:

```
class Flight(departureDate: var Date):
   -"-
   departureStatus -> info: var String:
      info := ("Flight " + FlightNumber + " at: " + departureDate.asString)
      if cancelled :then
         info := info + "is cancelled"
      :else
         if delayed :then
            info := info + "Estimated departure time: "+
                    departureTime +
                    " expected arrival time: " +
                    (departureTime + flightTime)
         :else
            info := info + " On schedule: " +
                    scheduledDepartureTime.t.magnitude
```

The next method is called after take off:

```
class Flight(departureDate: var Date):
   -"-
   arrivalStatus -> info: var
 String:
      info := ("Flight " + name + " at: " + departureDate.asString)
      info := info + "Departed at: "+ departureTime
      if (not hasArrived) :then         info := info + " expected at: " + arrivalTime
      :else
         info := info + " arrived at: " + ArrivalTime +
                 " delayed: " + delay
```

Given the above status methods in `Flight`, we have three ways of selection which flights we want the status: for all flights in all flight routes of a time table, for flights departing or arriving from a given airport at a given date, or flights of a given route at a given airport and date.

Status of flights in the time table

Based upon the entries in the time table, flight status for all flights is provided by the following method:

```
showFlightStatus:
   timeTable.scanTimeTable
      fr: ref FlightRoute
      fr := current
      fr.scan
         if (not hasTakenOff) :then
            currentFlight.departureStatus.print
         :else
            currentFlight.arrivalStatus.print
         newline
```

The method `showFlightStatus` is a submethod of the method `scanTimeTable` in `timeTable`. The method `scanTimeTable` scans the `entries` list of `FlightRoute` objects; for each `FlightRoute`, held by `fr,` a singular method being a submethod of `fr.scan` scans the `Flight` objects of the `flights` list in the `FlightRoute` held by `fr`.

As described above, the method `showFlightStatus` is based upon a `scanTimeTable` method in `timeTable`:

```
timeTable: obj
   entries: obj OrderList(FlightRoute)
   scanTimeTable:
      current: ref FlightRoute
      entries.scan
          this(scanTimeTable).current := current
          inner(scanTimeTable)
```

As explained before, the `inner(scanTimeTable)` is executed for each element in `entries`, and what is executed is the statements of the `showFlightStatus` submethod of `timeTable.scanTimeTable`

The above is in turn based upon a `scan` method of `FlightRoute`, scaning all the `Flight` objects of the list `flights`:

```
class FlightRoute(flightNumber, origin, destination: var String):
   -"-
   scan:
       currentFlight: ref Flight
       flights.scan
          currentFlight := current
          inner(scan)
```

From/to a given airport, at a given date

The following method delivers the list of flights from a given airport at a given date:

```
fromAirport(ap: var String, d: var Date)
   -> flights: ref OrderedList(Flight):

   timeTable.routesFrom(ap).scan
      current.flights.scan
         if (current.date = d) :then
            flights.insert(current)
```

This is based upon the method `routesFrom` in `timeTable`, delivering the list of routes departing from a given airport:

```
timeTable: obj
   -"-
   routesFrom(ap: var String) -> routes: OrderedList(FlightRoute):
      entries.scan
      if (current.origin = ap :then
         routes.insert(current)
```

It is left as a simple exercise to make the method that delivers the list of flights to a given destination airport at a given date.

Answer:

```
toAirport(ap: var String, d: var Date)
   -> flights: ref OrderedList(Flight):
   timeTable.routesTo(ap).scan
      current.flights.scan
         if (current.date = d) :then
            flights.insert(current)
```

based upon:

```
timeTable: obj
   -"-
   routesTo(ap: var String) -> routes: OrderedList(FlightRoute):
      entries.scan
      if (current.destination = ap :then
         routes.insert(current)
```

Before the list of `Flight` objects delivered by these two methods are used for producing the status website, the list delivered by `fromAirport` should be sorted according to departure time, while the list of `Flight` objects delivered by `toAirport` should be sorted according to arrival time.

Given these two lists of `Flight` objects, the status website can produce the two strings delivered by the methods `departureStatus` and `arrivalStatus`.

```
fromAirport("OSL", Date(6, 6, 2024)).scan
   current.departureStatus.print
fromAirport("OSL", Date(6, 6, 2024)).scan
   current.arrivalStatus.print
```

```
fromAirport("ARR", Date(6, 6, 2024)).scan
   current.departureStatus.print
fromAirport("ARR", Date(6, 6, 2024)).scan
   current.arrivalStatus.print
```

Given the flight number, airport, and a given date

The following method produces the list of flights given a certain route, from a given airport at a certain date:

```
onFlightNumberFrom(fn: var String, from: var String d: var Date)
   -> flights: ref OrderedList(Flight):
   theRoute: ref FlightRoute
   theRoute := timeTable.lookupRoute(fn)
   if theRoute.origin = from :then
      theRoute.flights.scan
         if (current.date = d) :then flights.insert(current)
```

This is based on a simple `lookupRoute` in `TimeTable`:

```
timeTable: obj
   entries: obj OrderList(FlightRoute)
   scanTimeTable:
      current: ref FlightRoute
      entries.scan
          this(scanTimeTable).current := current
          inner(scanTimeTable)

   lookupRoute(fn: var String) -> theRoute: ref FlightRoute: scanTimeTable
      if current.flightNumber = fn :then
         theRoute := current
         leave(lookupRoute)
```

It is left as a simple exercise to make the method that produces the list of flights given a certain route, to a given airport at a certain date:

Answer:

```
onFlightNumberTo(fn: var String, to: var String d: var Date)
   -> flights: ref OrderedList(Flight):
   theRoute: ref FlightRoute
   theRoute := timeTable.lookupRoute(fn)
   if theRoute.destination = to :then
      theRoute.flights.scan
         if (current.date = d) :then flights.insert(current)
```

Given these two lists of `Flight` objects, the status website can produce the two strings delivered by the methods `departureStatus` and `arrivalStatus`.

```
onFlightNumberFrom("SK1926", "ARR" Date(6, 6, 2024)).scan
   current.departureStatus.print
onFlightNumberTo("SK1926", "OSL" Date(6, 6, 2024)).scan
   current.arrivalStatus.print
```