

```

1  open util/time
2  open util/integer
3
4  sig Duration{
5      start: one Time,
6      end: one Time
7  }{
8      gt[end,start]
9  }
10
11 sig Position{}
12
13 sig User{
14     username: one String,
15     position: Position lone -> lone Time
16 }
17
18 sig Visitor{}
19
20 sig QRcode{}
21
22 sig QRreader{
23     scanned: QRcode lone-> lone Time,
24     acceptableQRcodes: QRcode -> Time
25 }
26
27 sig QueueBoard{
28     show: Int lone -> lone Time
29 }
30
31 sig TicketPrinter{
32     toPrint: PhysicalTicket lone -> lone Time
33 }
34
35 abstract sig Ticket{
36     store: one Store,
37     qrcode: one QRcode,
38     number: one Int,
39     permanenceTime: one Duration,
40     checkInTime: lone Time,
41     lineUpTime: one Time,
42     ticketStatus: TicketStatus lone -> lone Time
43 }{
44     number > 0
45     gt[checkInTime, lineUpTime]
46     #ticketStatus > 0 and #ticketStatus < 5
47 }
48
49 abstract sig TicketStatus{}
50     one sig Released extends TicketStatus{}
51     one sig Current extends TicketStatus{}
52     one sig CheckedIn extends TicketStatus{}
53     one sig Expired extends TicketStatus{}
54
55 sig InstantTicket extends Ticket{
56     user: one User,
57     waitingTime: one Duration,
58     travelTime: one Duration
59 } {

```

```

60     gte[travelTime.start,waitingTime.start]
61     gte[travelTime.end,waitingTime.end] <=> travelTime.start = waitingTime.start
62     waitingTime.start = lineUpTime
63 }
64
65 sig BookedTicket extends Ticket{
66     reservation: one Time,
67     user: one User
68 }{
69     gt[reservation, lineUpTime]
70 }
71
72 sig PhysicalTicket extends Ticket{
73     visitor: one Visitor,
74     waitingTime: one Duration
75 }{
76     waitingTime.start = lineUpTime
77     gt[permanenceTime.start,waitingTime.end]
78 }
79
80 sig Store{
81     name: one String,
82     address: one String,
83     defaultPermanenceTime: set Duration,
84     maximumPeopleCapacity: one Int,
85     qrReader: some QRreader,
86     queueBoard: some QueueBoard,
87     ticketPrinter: some TicketPrinter
88 }{
89     maximumPeopleCapacity > 0
90     #defaultPermanenceTime>2
91 }
92
93
94
95
96 fact defaultPermanenceTimeSameLenghtInAStore{
97     all s:Store |
98         (all pt: s.defaultPermanenceTime | pt.end = pt.start.next)
99     or
100     (all pt: s.defaultPermanenceTime | pt.end = pt.start.next.next)
101     or
102     (all pt: s.defaultPermanenceTime | pt.end = pt.start.next.next.next)
103 }
104
105 fact usernameUnique{
106     no disjoint u1,u2: User | u1.username=u2.username
107 }
108
109 fact storeNameAddressUnique{
110     no disjoint s1,s2: Store | s1.name=s2.name and s1.address=s2.address
111 }
112
113 fact QRcodeIsUnique{
114     no disjoint t1,t2: Ticket | t1.qrcode=t2.qrcode
115 }
116
117 fact oneInstantTicketPerUser{
118     no disjoint t1,t2: InstantTicket | t1.checkInTime=none and t2.checkInTime=none

```

```

    and t1.user=t2.user
119 }
120
121 fact oneBookedTicketPerUser{
122     no disjoint t1,t2: BookedTicket | t1.checkInTime=none and t2.checkInTime=none
123     and t1.user=t2.user
124 }
125
126 fact noQRcodeWithoutATicket{
127     all qc: QRcode | one tck: Ticket | qc=tck.qrcode
128 }
129
130 fact noVisitorWithoutTicket{
131     all v: Visitor | some pt:PhysicalTicket | pt.visitor=v
132 }
133
134 fact noPositionWithoutUser{
135     all p: Position | some u: User, t:Time | u.position.t=p
136 }
137
138 fact ticketStateChart{
139     --These first 3 lines are only to generate a world in which is possible to see
140     all the possible states
141     some tck: Ticket | one t': Time | tck.ticketStatus.t' = Current
142     some tck: Ticket | one t': Time | tck.ticketStatus.t' = CheckedIn
143     some tck: Ticket | one t': Time | tck.ticketStatus.t' = Expired
144     all tck: Ticket | all t: TicketStatus.(tck.ticketStatus) |
145         --Once a Ticket is "Expired"...
146         (tck.ticketStatus.t = Expired =>
147             ((all t':TicketStatus.(tck.ticketStatus) | t!=t' and gte[t',t] =>
148                 tck.ticketStatus.t' != CheckedIn and tck.ticketStatus.t' != Current)
149             and
150             (one t'': TicketStatus.(tck.ticketStatus) | tck.ticketStatus.t'' =
151                 Current and t''=t.prev.prev.prev)))
152         and
153         --Once a Ticket is "Current"...
154         (tck.ticketStatus.t = Current =>
155             ((all t': TicketStatus.(tck.ticketStatus) | t!=t' and gte[t',t] =>
156                 tck.ticketStatus.t' != Released)
157             and
158             (one t'': TicketStatus.(tck.ticketStatus) | tck.ticketStatus.t'' =
159                 Released)))
160         and
161         --Once a Ticket is "CheckedIn"...
162         (tck.ticketStatus.t = CheckedIn =>
163             ((all t': TicketStatus.(tck.ticketStatus) | t!=t' and gte[t',t] =>
164                 tck.ticketStatus.t' != Current and tck.ticketStatus.t' != Released)
165             and
166             (one t'': TicketStatus.(tck.ticketStatus) | tck.ticketStatus.t'' =
167                 Current)))
168     }
169
170 fact lineUpTimeIsReleasedStatusTime{
171     all tck: Ticket |
172         tck.lineUpTime = Released.(tck.ticketStatus)
173 }
174
175 fact checkInTimeIsCheckedInStatusTime{
176     all tck: Ticket |

```

```

169         tck.checkInTime = CheckedIn.(tck.ticketStatus)
170     }
171
172     fact noDevicewithoutStore{
173         (all qr:QRreader | one s:Store | qr in s.qrReader)
174         and
175         (all qb:QueueBoard | one s:Store | qb in s.queueBoard)
176         and
177         (all tp:TicketPrinter | one s:Store | tp in s.ticketPrinter)
178     }
179
180     fact permanceTimeIsDefaultPermanenceTimeForVisitor{
181         all pt: PhysicalTicket | one d: Duration |
182             pt.permanenceTime = d and (d in pt.store.defaultPermanenceTime)
183     }
184
185     fact oneCurrentTicketForStoreAtTime{
186         no disjoint tck1,tck2: Ticket | one t:Time | tck1.ticketStatus.t=Current and
187             tck2.ticketStatus.t=Current and tck1.store=tck2.store
188     }
189
190     fact oneUserPositionAtLiningUpInstantTicketTime{
191         all tck:InstantTicket | one p:Position |
192             tck.lineUpTime in p.(tck.user.position)
193     }
194
195     fact ticketPermanenceTimeCorrespondencesIfCheckedIn{
196         all tck: Ticket |
197             CheckedIn.(tck.ticketStatus)!=none =>
198                 tck.checkInTime = tck.permanenceTime.start
199     }
200
201     fact ticketScannedBeforeCheckedIn{
202         all tck: Ticket |
203             CheckedIn.(tck.ticketStatus) != none =>
204                 (one qr: QRreader | (qr in tck.store.qrReader) and
205                     (tck.qrcode)->(CheckedIn.(tck.ticketStatus).prev) in qr.scanned)
206     }
207
208     fact ticketWaitingTimeTravelTimeCorrsepondences{
209         all tck: PhysicalTicket |
210             Current.(tck.ticketStatus)!=none =>
211                 (tck.waitingTime.end=Current.(tck.ticketStatus))
212             else
213                 (gt[tck.waitingTime.end, Released.(tck.ticketStatus)])
214         all tck: InstantTicket |
215             Current.(tck.ticketStatus)!=none =>
216                 (gte[tck.waitingTime.end, tck.travelTime.end] =>
217                     (tck.waitingTime.end=Current.(tck.ticketStatus))
218                     else
219                         (tck.travelTime.end=Current.(tck.ticketStatus)))
220             else
221                 (gt[tck.waitingTime.end, Released.(tck.ticketStatus)] and
222                     gt[tck.travelTime.end, Released.(tck.ticketStatus)])
223     }
224
225     fact bookedTicketBecomeCurrent{
226         all tck: BookedTicket |
227             tck.reservation = Current.(tck.ticketStatus)

```

```

225 }
226
227 fact physicalTicketStoreEqualTicketPrinterStore{
228     all tp: TicketPrinter, tck: tp.toPrint.Time | tp in tck.store.ticketPrinter
229 }
230
231 fact queueBoardShowTheCurrentTicketNumberOfStore{
232     (all tck: Ticket, t:Time |
233         tck.ticketStatus.t = Current => (all qb: tck.store.queueBoard| qb.show.t =
234             tck.number))
235
236     and
237     (all t:Time, s: Store | (all tck: store.s| tck.ticketStatus.t != Current) <=>
238         s.queueBoard.show.t = none)
239 }
240
241 fact oneTicketPrinterPrintPhysicalTicketAtReleasingTime{
242     (all tck: PhysicalTicket|
243         one tp: TicketPrinter|
244             (tp in tck.store.ticketPrinter) and (tck->(Released.(tck.ticketStatus))
245                 in tp.toPrint) and (all tp':TicketPrinter| tp!=tp' => tck.(tp'.toPrint)
246                     = none))
247     (all tp: TicketPrinter |
248         all t: PhysicalTicket.(tp.toPrint) | tp.toPrint.t.ticketStatus.t = Released)
249 }
250
251 fact whenQRcodeAreAcceptableQRcodes{
252     all tck: Ticket, t:Time | all qr: QRreader|
253         (gte[t,Current.(tck.ticketStatus)] and (qr in tck.store.qrReader) and
254             (CheckedIn.(tck.ticketStatus)!= none => gt[CheckedIn.(tck.ticketStatus),t])
255             and (Expired.(tck.ticketStatus)!=none => gt[Expired.(tck.ticketStatus),t]))
256         <=> (tck.qrcode->t in qr.acceptableQRcodes)
257 }
258
259 -----Pred
260 pred hasUserAValidInstantTicket[u:User,t:Time]{
261     one tck: InstantTicket | tck.user=u and tck.ticketStatus.t != Expired
262 }
263
264 pred hasUserAValidBookedTicket[u:User,t:Time]{
265     one tck: BookedTicket | tck.user=u and tck.ticketStatus.t != Expired
266 }
267
268 pred hasVisitorAValidPhysicalTicket[v:Visitor,t:Time]{
269     one tck: PhysicalTicket | tck.visitor=v and tck.ticketStatus.t != Expired
270 }
271
272 pred hasUserAValidInstantTicketForTheStore[u:User,t:Time,s:Store]{
273     one tck: InstantTicket| tck.user=u and tck.ticketStatus.t != Expired and
274         tck.store=s
275 }
276
277 pred hasUserAValidBookedTicketForTheStore[u:User,t:Time,s:Store]{
278     one tck: BookedTicket| tck.user=u and tck.ticketStatus.t != Expired and
279         tck.store=s
280 }
281
282 pred userMakeAInstantTicketReservation[u:User,s:Store,t:Time,tck: InstantTicket]{
283     //preconditions
284     not hasUserAValidInstantTicket[u,t]
285     not hasUserAValidBookedTicketForTheStore[u,t,s]

```

```

275     //postconditions
276     tck.user=u
277     tck.store=s
278     hasUserAValidInstantTicket[u,t.next]
279     not hasUserAValidBookedTicketForTheStore[u,t.next,s]
280     tck.ticketStatus.(t.next) = Released
281 }
282
283 pred userMakeABookedTicketReservation[u:User,s:Store,t:Time,tck: BookedTicket] {
284     //preconditions
285     not hasUserAValidBookedTicket[u,t]
286     not hasUserAValidInstantTicketForTheStore[u,t,s]
287     //postconditions
288     tck.user=u
289     tck.store=s
290     hasUserAValidBookedTicket[u,t.next]
291     not hasUserAValidInstantTicketForTheStore[u,t.next,s]
292     tck.ticketStatus.(t.next) = Released
293 }
294
295 pred
296 visitorMakeAPhysicalTicketReservation[v:Visitor,s:Store,t:Time,tck:PhysicalTicket] {
297     //postconditions
298     tck.visitor=v
299     tck.store=s
300     tck.ticketStatus.(t.next) = Released
301 }
302
303 pred userChekInWithInstantTicket[u:User,s:Store,t:Time,tck:InstantTicket] {
304     //preconditions
305     hasUserAValidInstantTicketForTheStore[u,t,s]
306     not hasUserAValidBookedTicketForTheStore[u,t,s]
307     tck.store=s
308     tck.user=u
309     gt[t,Current.(tck.ticketStatus)]
310     //postconditions
311     CheckedIn.(tck.ticketStatus)=t.next
312 }
313
314 pred userChekInWithBookedTicket[u:User,s:Store,t:Time,tck:BookedTicket] {
315     //preconditions
316     hasUserAValidBookedTicketForTheStore[u,t,s]
317     not hasUserAValidInstantTicketForTheStore[u,t,s]
318     tck.store=s
319     tck.user=u
320     gt[t,Current.(tck.ticketStatus)]
321     //postconditions
322     CheckedIn.(tck.ticketStatus)=t.next
323 }
324
325 pred visitorChekInWithPhysicalTicket[v:Visitor,s:Store,t:Time,tck:PhysicalTicket] {
326     //preconditions
327     hasVisitorAValidPhysicalTicket[v,t]
328     tck.store=s
329     tck.visitor=v
330     gt[t,Current.(tck.ticketStatus)]
331     //postconditions
332     CheckedIn.(tck.ticketStatus)=t.next
333 }

```

```
333
334  pred show{
335      #BookedTicket>0
336      #InstantTicket>0
337      #PhysicalTicket>0
338  }
339
340  run show for 6 but 10 int, exactly 2 String
341  run userMakeAInstantTicketReservation for 6 but 10 int, exactly 2 String
342  run userMakeABookedTicketReservation for 6 but 10 int, exactly 2 String
343  run visitorMakeAPhysicalTicketReservation for 6 but 10 int, exactly 2 String
344  run userChekInWithInstantTicket for 6 but 10 int, exactly 2 String
345  run userChekInWithBookedTicket for 6 but 10 int, exactly 2 String
346  run visitorChekInWithPhysicalTicket for 6 but 10 int, exactly 2 String
347
```