In my specification pdf, I forgot to include that

**tickets ≠ ∅**

So imagine that is included under the ticket declaration area.

I will be using this as a basis for my argument for invariant consistency.

issueTicket operation

tickets ≠ ∅ ∧   
∀ t1, t2 : tickets • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber ∧

¬ (∃ t: Ticket • t.serialNumber = issuedTicket?.serialNumber) ∧ issuedTicket?.status = unpaid ∧

tickets' = tickets ⋃ { issuedTicket? }

⇒

tickets’ ≠ ∅ ∧   
∀ t1, t2 : tickets’ • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber

To prove

tickets’ ≠ ∅ ∀ ∧   
t1, t2 : tickets’ • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber

Since they are conjoined statements, we can each separately.

Since tickets’ ≠ ∅ is a constraint that indicates the number of tickets shouldn’t be empty, if it is empty then this is false and there is nothing to prove. Based on the operation: tickets' = tickets ⋃ { issuedTicket? } this means that tickets’ will contain at least 1 ticket and hence is not empty.

To prove ∀ t1, t2 : tickets’ • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber

We will first change the above statement using tickets’ value from the operation

∀ t1, t2 : tickets ⋃ { issuedTicket? } • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber

From the invariant before, we know that all the tickets have a unique serial number to distinguish uniqueness among tickets. So we only need to prove that the new issued ticket ( issuedTicket? ) also has a unique serial number.

The operation contains ¬ (∃ t: Ticket • t.serialNumber = issuedTicket?.serialNumber)

which proves the issued ticket is unique.

Lastly, issuedTicket?.status = unpaid which is indicating the new issued ticket must have the status unpaid otherwise if it’s false then it’s an invalid ticket and would result in false meaning there is nothing to prove.

payTicket operation

tickets ≠ ∅ ∧

∀ t1, t2 : tickets • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber ∧

tickets ≠ ∅ ∧

payAmount? = which?.fineAmount ∧

∃ t : Ticket | t.serialNumber = which?.serialNumber •

tickets' = (tickets \ { which? } ) ⋃ { paidTicket : Ticket | paidTicket.serialNumber =

which?.serialNumber ∧ paidTicket.licensePlate = which?.licensePlate ∧ paidTicket.fineAmount = which?.fineAmount ∧ paidTicket.status = paid }

⇒

tickets’ ≠ ∅ ∧

payAmount? = which?.fineAmount ∧   
∀ t1, t2 : tickets’ • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber

To prove

tickets’ ≠ ∅ ∧

payAmount? = which?.fineAmount ∧   
t1, t2 : tickets’ • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber

Since they are conjoined statements, we can each separately.

Since we are changing only the status of a ticket, and from the previous invariant with the constraint of the ticket set not being empty and tickets’ ≠ ∅ thus if it were empty then there is nothing to prove.

As for the payAmount? = which?.fineAmount, merely a straightforward check on the selected ticket (which?) and the payAmount? to have the same value.

Lastly to prove ∀ t1, t2 : tickets’ • t1 = t2 ⇔ t1.serialNumber = t2.serialNumber we can modify tickets’ to it’s value.

tickets` = (tickets \ { which? } ) ⋃ { paidTicket : Ticket | paidTicket.serialNumber =

which?.serialNumber ∧ paidTicket.licensePlate = which?.licensePlate ∧ paidTicket.fineAmount = which?.fineAmount ∧ paidTicket.status = paid }

So substitute tickets` with it’s value.

In our operation, we have a statement

∃ t : Ticket | t.serialNumber = which?.serialNumber •

tickets' = (tickets \ { which? } ) ⋃ { paidTicket : Ticket | paidTicket.serialNumber =

which?.serialNumber ∧ paidTicket.licensePlate = which?.licensePlate ∧ paidTicket.fineAmount = which?.fineAmount ∧ paidTicket.status = paid }

which guarantees that the set is not empty and there is a ticket’s serial number that matches our selected ticket (which?) in which that ticket’s status is changed to paid thus not changing any other value and being added back into the set.

Thus only the selected ticket within our set changed it’s status value causing no change in the ticket set.

Since both of the operations maintained their invariant status before and after the operation, the ticket set is unique along with proving that the two operations preserving the invariant property of the TicketStore class.