VOLET CLIENT

Vol

1. Un aéroport est identifié par trois lettres uniques à chaque aéroport *[OCL]*.

context Aeroport
inv: self.id_aeroport.size() = 3 and Aeroport.AllInstances->isUnique(id_aeroport)

2. La partie alphabétique de l'ID d'un vol est unique à chaque compagnie et la partie numérique est unique à chaque vol au sein de la même compagnie *[OCL]*

Context CompagnieAerienne

inv: self.vols->forAll(v1, v2 | v1 <> v2 implies

(v1.vol_id.substring(1, 2) = v2.vol_id.substring(1, 2) implies

v1.vol_id.substring(3, v1.vol_id.size()) <> v2.vol_id.substring(3, v2.vol_id.size())))

3. *L'aéroport de départ et d'arrivée d'un vol doit être différent [OCL].*

context VolAerien
inv: self.voyageDe.id_aeroport <> self.voyageVers.id_aeroport

4. *Tous les sièges d'une même section ont le même prix [OCL].*

context SectionAvion
inv: self.sieges->forAll(s1, s2 | s1.prix = s2.prix)

Bateau

5. *Un port est identifié par trois lettres uniques à chaque port [OCL].*

Context Port
inv: self.id_port.size() = 3 and self.id_port -> isUnique(id_port)

6. *Un itinéraire ne peut pas durer plus de 21 jours [OCL].*

context itineraire_naval
inv: self.duree <= 21

7. *Le port de départ et d'arrivée doit être le même [OCL].*

context Port
*inv: self.*voyageDe.*id_port* = self.voyageVers.*port*

8. *Un paquebot peut être assigné à plusieurs itinéraires tant qu'ils ne se chevauchent pas [OCL].*

Context Paquebot
inv: self.itineraires->forAll(i1, i2 | i1 <> i2 implies
(i1.dateFin < i2.dateDebut or i2.dateFin < i1.dateDebut))

9. *Toutes les cabines d'une même section ont le même prix [OCL].*

context SectionPaquebot

inv: self.cabines -> forAll(c1, c2 | c1.prix = c2.prix)

Train

10. *Une gare est identifiée par trois lettres uniques à chaque gare [OCL].*

context Gare

inv: self.id_gare.size() = 3 and Gare.allInstances() -> isUnique(id_gare)

VOLET ADMINISTRATIF

AVION et TRAIN

11. Le client peut réserver un siège disponible dans un vol (trajet) donné [OCL].

context VolAerien
inv: self.section.sieges -> exist(s | s.isReserved() = false)

context TrajetTrain
inv: self.sections.sieges -> exist(s | s.isReserved() = false)

12. Un siège réservé devient assigné à un passager une fois payé: le siège est donc confirmé [OCL].

context Paiement
inv: if self.estNecessairePour.paymentDone = true
then self.estNecessairePour.isPlaceAvailable = false

BATEAU

13. Le client peut réserver une cabine disponible pour un itinéraire donné [OCL].

context ItineraireNaval
inv: self.sections->select(cabines)->exist(c | c.isReserved() = false)