OCL for class representing reservation

```
context TripGateway::createTrip(
        clientLastName : String,
        clientId : String,
        selectedConnections : Sequence(Connection),
        travelerNames : Sequence(String),
        ages : Sequence(Integer),
        identificationNumbers : Sequence(String),
        isFirstClass : Boolean
)     : Trip


pre ClientLastNameValid:
        clientLastName <> '' and clientLastName.size() > 0

pre ClientIdValid:
        clientId <> '' and clientId.size() > 0

pre NonEmptySelections:
        selectedConnections->notEmpty()

pre NonEmptyTravelerData:
        travelerNames->notEmpty() and ages->notEmpty() and identificationNumbers ->notEmpty()

pre SameCardinality:
        travelerNames->size() = ages->size() and
        travelerNames->size() = identificationNumbers->size()

pre EachConnectionHasRoutes:
        selectedConnections->forAll(c : Connection | c.routes->notEmpty())

pre TravelerNamesValid:
        travelerNames->forAll(name : String | name <> '' and name.size() > 0)

pre AgesValid:
        ages->forAll(a : Integer | a > 0 and a <= 150)

pre IdentificationNumbersValid:
        identificationNumbers->forAll(id : String | id <> '' and id.size() > 0)

post ResultNotNull:
        result <> null

post TripIdFormat:
        result.tripId.size() = 8

post TripClientCopied:
        result.clientId = clientId and result.clientLastName = clientLastName

post ReservationsPerTraveler:
        result.reservations->size() = travelerNames->size()

post ReservationsWellFormed:
        result.reservations->forAll(r : Reservation |
                r.travelerName <> '' and r.age > 0 and r.identificationNumber <> '' and r.ticket <> null and r.routes->notEmpty())

post TravelersMatchInput:
        result.reservations->forAll(r : Reservation |
                travelerNames->includes(r.travelerName) and ages->includes(r.age) and identificationNumbers
                        >includes(r.identificationNumber))

post ClassMatchesParameter:
        result.reservations->forAll(r : Reservation | r.ticket.isFirstClass = isFirstClass)

post ReservationIdsUniqueWithinTrip:
        result.reservations->forAll(r1, r2 : Reservation | r1 <> r2 implies r1.reservationId <> r2.reservationId)

post RoutesCopied:
        let totalRoutes : Integer = selectedConnections->collect(c : Connection | c.routes->size()) >sum() in result.reservations->forAll(r : Reservation |
           r.routes->size() = totalRoutes
        )

post TicketPricePerReservation:
    result.reservations->forAll(r : Reservation |
        r.ticket.price =
           r.routes->collect(route : Route |
              if isFirstClass
              then route.firstClassTicket
              else route.secondClassTicket
```

```
            endif
        )->sum()
)
```