OCL expressions for requirements

1. Offerings are unique. Multiple offerings on the same day and time slot must be offered at a different location.

```
context Offering
inv UniqueOffering:
  Offering.allInstances()->forAll(o1, o2 |
    o1 <> o2 implies
    (o1.date <> o2.date or o1.timeSlot <> o2.timeSlot or o1.location <> o2.location)
  )
```

2. Any client who is underage must necessarily be accompanied by an adult who acts as their guardian. For clients under the age of 18, there must be an assigned guardian.

```
context Client
inv UnderageGuardian:
  self.age < 18 implies self.guardian.oclIsKindOf(Adult)
```

3. The city associated with an offering must be one of the cities that the instructor has indicated in their availabilities.

```
context Offering
inv ValidInstructorCity:
  self.instructor.availabilities->includes(self.city)
```

4. A client does not have multiple bookings on the same day and time slot. This constraint ensures that a client cannot book more than one offering at the same time.

```
context Client
inv UniqueBooking:
  self.bookings->forAll(b1, b2 |
    b1 <> b2 implies (b1.date <> b2.date or b1.timeSlot <> b2.timeSlot)
  )
```

OCL expressions for operations

1. createOffering(location, city, time, type, duration, startDate, endDate)

```
context Admin::createOffering(location: String, city: String, time: LocalTime, type: String,
duration: Integer, startDate: LocalDate, endDate: LocalDate)
pre: not self.offering -> exists(o | o.location = location and
                                     o.city = city and
                                     o.time = time and
                                     o.startDate = startDate)
post: self.offering -> exists(o | o.location = location and
                                  o.city = city and
                                  o.time = time and
                                  o.type = type and
                                  o.duration = duration and
                                  o.startDate = startDate and
                                  o.endDate = endDate)
```

2. makeBooking(offering, client)

```
context Client::makeBooking(offering: Offering): Booking
pre:
        self.isRegistered()
        offering.isAvailable()
post:
        result.oclIsTypeOf(Booking)
        result.offering = offering
        offering.isAvailable() = false
        bookings -> includes(result)
        bookings-> size() = bookings@pre->size() + 1
```

3. takeOnOffering(offering: Offering)

```
context Instructor:: takeOnOffering (offering: Offering)
pre: not Offering.existsByCityAndLocationAndTimeAndStartDate(offering.location,
offering.city, offering.time, offering.startDate)
post:
        Offering.existsByCityAndLocationAndTimeAndStartDate(offering.location,
        offering.city, offering.time, offering.startDate)

        InstructorOfferings.includes(offering)
```