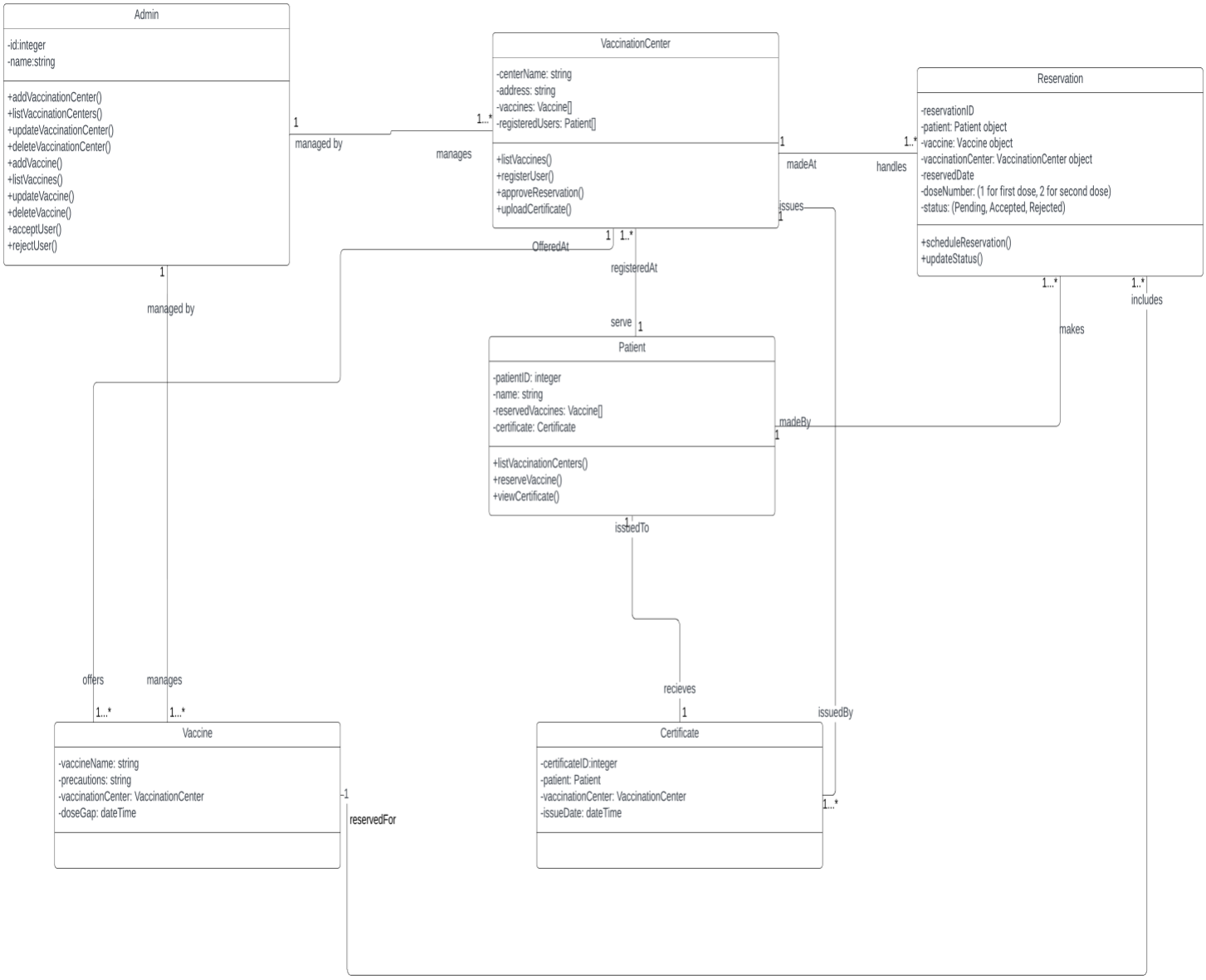
Class Diagram



OCL

Admin Class:

Invariants:

context Admin

inv: self.manages->notEmpty()

inv: self.manages->forAll(vc | vc.managedBy = self

inv: self.managesVaccines->notEmpty()

inv: self.managesVaccines->forAll(v | v.managedByAdmin = self)

Pre &Post Conditions:

context Admin::deleteVaccinationCenter(vc: VaccinationCenter)

pre: self.manages->includes(vc)

post: not self.manages->includes(vc)

context Admin::addVaccine(v: Vaccine)

pre: v.name ->notEmpty()

post: self.managesVaccines->includes(v)

context Admin::addVaccinationCenter(newCenterName: String, newLocation: String)

pre: newCenterName ->notEmpty()and newLocation ->notEmpty()

post: self.manages->exists(vc | vc.centerName = newCenterName)

context Admin::listVaccinationCenters(): Set(VaccinationCenter)

pre: self.manages->notEmpty()

context Admin::updateVaccinationCenter(existingCenter: VaccinationCenter, newCenterName: String, newLocation: String)

pre: self.manages->includes(existingCenter) and newCenterName ->notEmpty() and newLocation ->notEmpty()

post: existingCenter.centerName = newCenterName and existingCenter.location = newLocation

context Admin::deleteVaccinationCenter(vc: VaccinationCenter)

pre: self.manages->includes(vc)

post: not self.manages->includes(vc)

context Admin::addVaccine(newVaccineName: String, newPrecautions: String, newTimeGap: Integer)

pre: newVaccineName ->notEmpty() and newPrecautions ->notEmpty() and newTimeGap > 0

post: self.managesVaccines->exists(v | v.name = newVaccineName

context Admin::listVaccines(): Set(Vaccine)

pre: self.managesVaccines->notEmpty()

context Admin::updateVaccine(existingVaccine: Vaccine, newVaccineName: String, newPrecautions: String, newTimeGap: Integer)

pre: self.managesVaccines->includes(existingVaccine) and newVaccineName ->notEmpty() and newPrecautions ->notEmpty() and newTimeGap > 0

post: existingVaccine.name = newVaccineName and existingVaccine.precautions = newPrecautions and existingVaccine.timeGap = newTimeGap

context Admin::deleteVaccine(v: Vaccine)

pre: self.managesVaccines->includes(v)

post: not self.managesVaccines->includes(v)

context Admin::acceptUser(patient: Patient)

pre: patient.name ->notEmpty()

post: No specific postcondition

context Admin::rejectUser(patient: Patient)

pre: patient.name ->notEmpty()

post: No specific postcondition

Collection Operations:

context Admin

def: listVaccinationCenters(): Set(VaccinationCenter) =self.manages

context Admin

def: listVaccines(): Set(Vaccine) =self.managesVaccines.vaccine

VaccinationCenter Class:

Invariants:

context VaccinationCenter

inv: self.centerName ->notEmpty()

inv: self.location ->notEmpty()

inv: self.offers->notEmpty()

inv: self.offers->forAll(v | v.offeredAt = self)

Pre &Post Conditions:

context VaccinationCenter::registerUser(patient: Patient)

pre: patient.name ->notEmpty()

post: self.registeredAt->includes(patient)

context VaccinationCenter::approveReservation(reservation: Reservation)

pre: reservation.status = 'pending'

post: reservation.status = 'approved'

context VaccinationCenter::uploadCertificate(patient: Patient, certificateID: String, issueDate: Date)

pre: patient.name -> notEmpty() and certificateID ->notEmpty() and issueDate <> null

post: No specific postcondition

context VaccinationCenter::listVaccines(): Set(Vaccine)

pre: self.offers->notEmpty()

Collection Operations:

context VaccinationCenter

def: listVaccines(): Set(Vaccine) =self.offers

Vaccine Class

Invariants:

context Vaccine

inv: self.name ->notEmpty()

inv: self.precautions ->notEmpty()

inv: self.timeGap > 0

Patient Class

Invariants:

context Patient

inv: self.patientID->notEmpty()

inv: self.name->notEmpty()

Pre &Post Conditions:

context Patient::reserveVaccine(v: Vaccine)

pre: v.offeredAt->notEmpty()

post: self.madeBy->exists(r | r.reservedFor = v)

context Patient::viewCertificate(certificate: Certificate)

pre: self.receives->includes(certificate)

post: No postconditions for viewing certificate

Collection Operations:

context Patient

def: listRegisteredVaccinationCenters(): Set(VaccinationCenter) =self.registeredAt

Reservation Class:

Invariants:

context Reservation

inv: self.reservationID->notEmpty()

self.reservedDate->notEmpty()

self.doseNumber->notEmpty()

self.status->notEmpty()

inv: self.madeAt->size() = 1

inv: self.includes->size() >= 1

inv: self.includes->forAll(vaccine | vaccine.reservedFor = self)

Pre &Post Conditions:

context Reservation::scheduleReservation()

pre: self.madeAt->notEmpty()

post: No postconditions for scheduling reservation

context Reservation::updateStatus(newStatus: String)

pre: newStatus = 'approved' or newStatus = 'rejected'

post: self.status = newStatus

Certificate Class :

Invariants:

context Certificate

inv: self.certificateID->notEmpty()

self.issueDate->notEmpty()

inv: self.issuedBy->size() = 1

inv: self.issuedTo->size() = 1