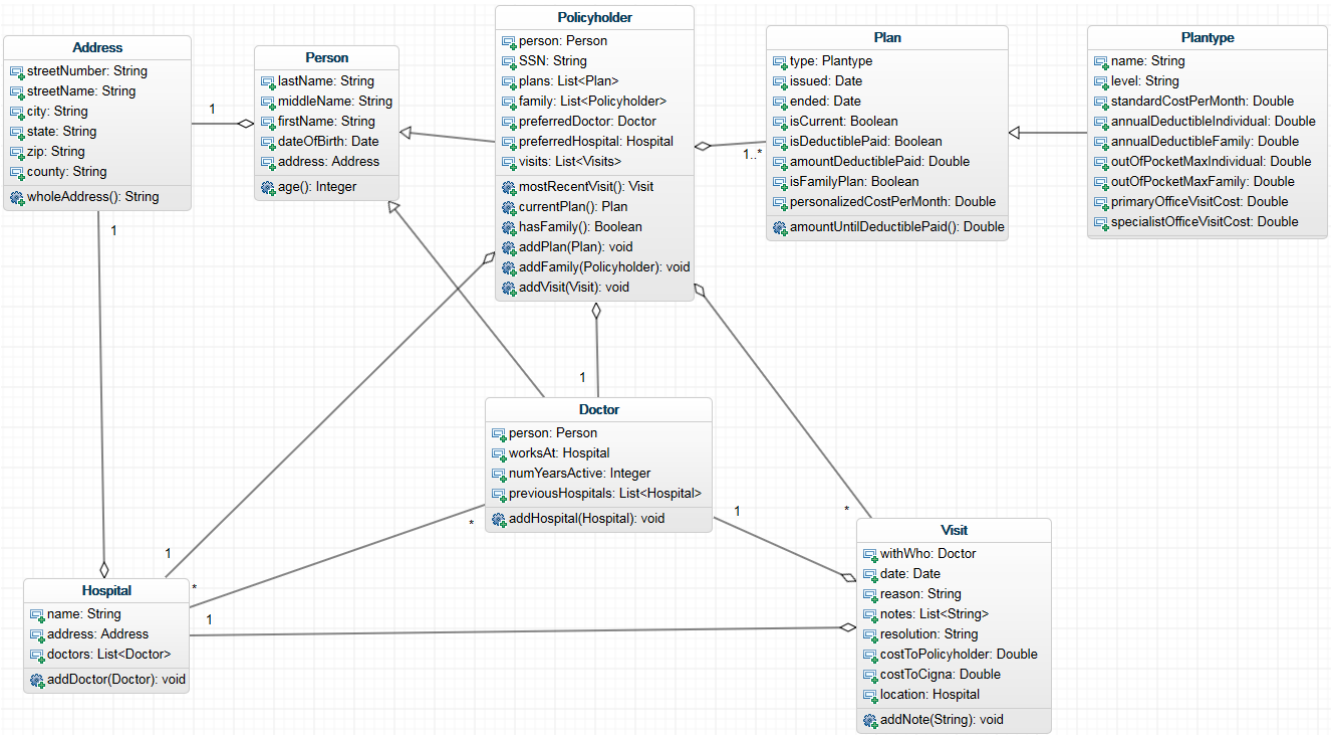
Cigna Database Architecture Redesign

Tim Mendez

tmendez@calpoly.edu

LinkedIn

October 16, 2014



Disclaimers

I used my best judgment as to what attributes each class should contain. In reality, I would consult those using any software relying on this design to get a better understanding of what data needs to be readily available.

I would also create a class for Resolution (extended by Visit) containing any other Doctors referred to and for what, any prescriptions issued, etc. However, for this project I felt that would be overkill as it would open a whole new can of worms – creating a Referral class with Doctor referral hierarchy, a Prescription class with a whole mess of attributes, any other related scheduled visits linked, advice and whether it was followed, the list goes on.

Explanation

All classes have accessor methods for setting and getting attributes as well as a constructor that requires most, if not all, attributes to be passed as parameters.

Any listed methods are to get potentially useful data, or to add to Lists.

Person contains an Address

Attributes

- Last name
- Middle name (may be null)
- First name
- Date of birth
- Address

Methods

- int age(); No parameters, returns the current age of the Person

Doctor is a Person, contains Hospitals

Attributes

- Person
- Hospital where the doctor is currently employed
- Number of years working as a doctor
- List of all hospitals where the doctor has ever been employed

Methods

- addHospital(Hospital); Hospital to add where the doctor worked, no return

Policyholder is a Person, contains Plans, contains Visits

Attributes

- Person
- Social Security Number
- List of Plans
- List of Policyholders in this Policyholder's family (may be null)
- Preferred Doctor (may be null, initially)
- Preferred Hospital (may be null, initially)
- List of Visits (may be null, initially)

Methods

- Visit mostRecentVisit(); No parameters, returns this Policyholder's most recent Visit
- Plan currentPlan(); No parameters, returns this Policyholder's insurance plan
- Boolean hasFamily(); No parameters, returns true if this Policyholder has any other family under the same Plan, false otherwise
- addPlan(Plan); Policyholder's current Plan, no return
- AddFamily(Policyholder); Policyholder's family member, no return
- addVisit(Visit); Polichholder's most recent Visit, no return

Hospital contains an Address

Attributes

- Name
- Address
- List of Doctors who work at this Hospital

Methods

- addDoctor(Doctor); Doctor who now works at this Hospital, no return

Address
Attributes
- Street number
- Street name
- City
- State
- Zip code
- County

Methods
- String wholeAddress(); No parameters, returns the address in standard form

Plan
Attributes
- Plantype
- Date issued
- Date ended (may be null)
- Whether it is the Policyholder's current plan (this must be true when at the head of Policyholder.plans, just a double check due to importance)
- Whether the deductible has been entirely paid for, for the current year
- The amount of the deductible that has been paid
- Whether the plan is a family plan
- How much this Policyholder is charged per month

Methods
- double amountUntilDeductiblePaid(); No parameters, returns the amount this Policyholder must pay until the deductible is entirely paid

Plantype is a Plan
Attributes
- Name
- Level (Bronze level plan, Silver level plan, etc.)
- Standard cost per month of this Plantype
- Annual deductible for an individual plan
- Annual deductible for a family plan
- The max out of pocket for an individual plan
- The max out of pocket for a family plan
- Cost of a primary office visit
- Cost of a specialist office visit

Visit contains a Doctor, contains a Hospital
Attributes
- Doctor who this Visit was with
- Occurring date
- Reason for this Visit
- List of notes
- Resolution (e.g., another Visit scheduled, a prescription, advice, referral, etc.)

Methods

- addNote(String); Any notes the Doctor may have during this Visit, no return