SOEN 387 Web-based Enterprise Application Design

Stuart Thiel

Concordia University
Department of Computer & Software Engineering

Fall, 2015

Outline

Domain Model

What are RDGs again?

- They hold raw data from DB records
- They provide DB interaction behaviour

What is Domain Logic?

- Behaviour associated with elements specific to the application
- Elements that have meaning to the users of the application
- Not specifically programmatic classes

What If We Want Domain Logic?

- What if we wanted to do something?
- What if we wanted to qualify data?
- What if we wanted to compare things?
- Does *.compareTo*() or *.equals* belong in RDG?

Active Record

- So, if we just add Domain Logic to RDGs, we get the Active Record pattern
- Popular with Microsoft for a long time. . . maybe still
- It works, but low cohesion in those classes
  - Domain Logic
  - Raw Database Data
  - DB interaction behaviour

POJO to the Rescue

- So why not make a POJO that represents the record in memory
- Keep it totally separate from the DB stuff
- Then it could hold Domain Logic and that would make sense

What About RDG Database Behaviour?

- We still need to interact with the DB
- In exactly the same way. . .
- But we don't want to store the data in whatever does it

Table Data Gateway

- A Table Data Gateway (TDG) fits the bill
- Abstract Class with static methods to do everything
- Takes raw data and adjusts DB
- Takes DB and returns raw data (RecordSets)
- Doesn't know or care about these POJOs

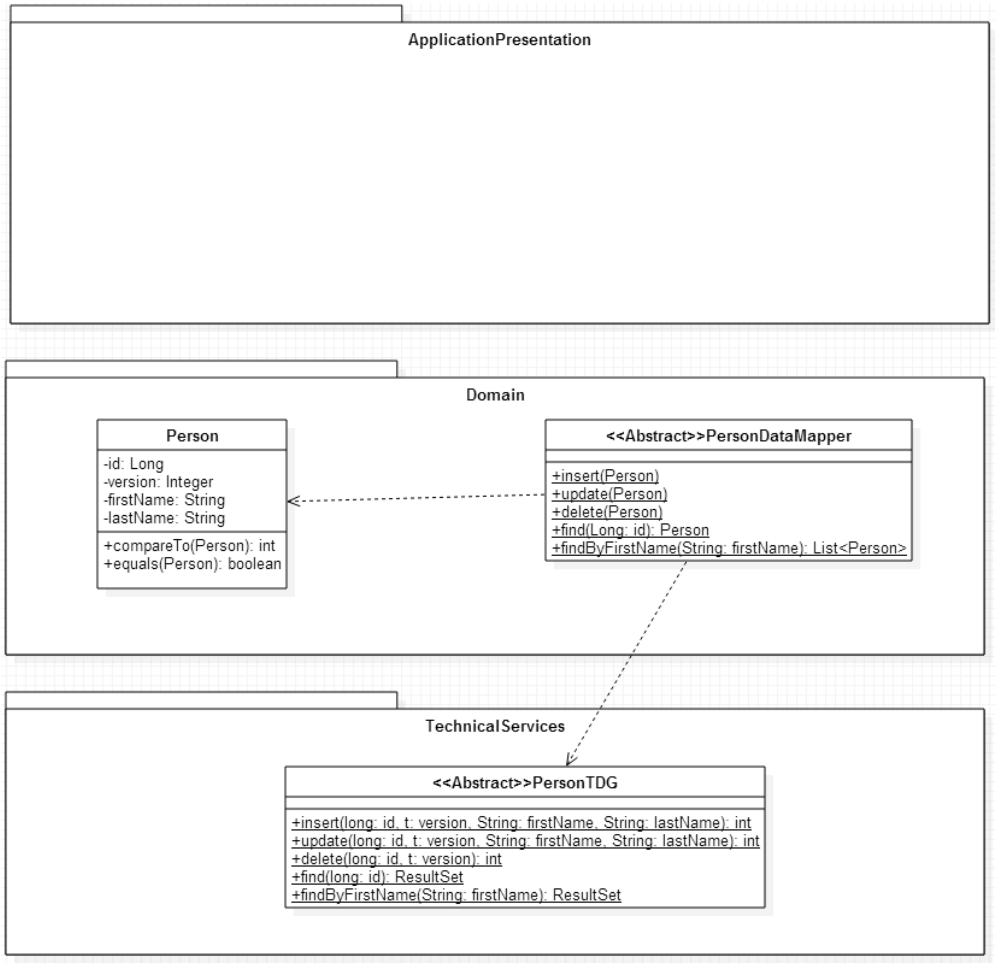
How to Get to the POJO?

- So where do we get the POJO?
- The TDG is at a lower level, POJO is clearly in Domain
- We need something that can use the TDG as a service to decide which POJOs to make

Data Mapper

- The Data Mapper fits this well!
- It takes raw data/POJOs in finders and uses the TDG service to get ResultSets
- It then converts the ResultSets into one or more POJO as needed
- It takes the related POJOs and pulls out necessary raw data for insert/update/delete using the TDG service
- One Mapper per type of POJO
- Inheritance? Think about it, but we'll do that later.

Diagram of POJO/TDG/DM

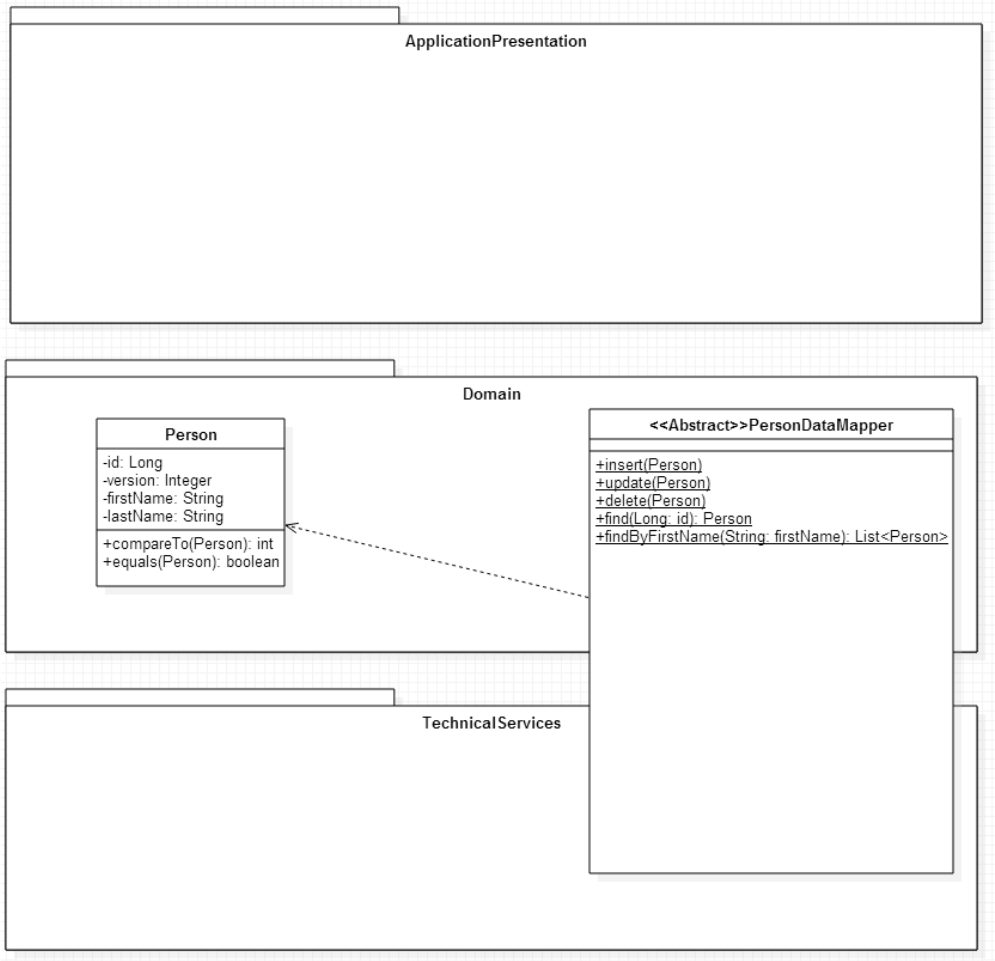


- Splits off the database behaviour from everything else

Fowler's Joint TDG/Data Mapper

- Fowler Merges the Data Mapper and TDG and calls it a Data Mapper
- It spans two Layers. . . is that good?
- Larman identifies that it's easy and sensible to just split it into the two we use.

Diagram of Fowler's DM



- Does everything that Larman's version does, just all in one class

Making a Domain Model Diagram to Pick POJOs

- How do we know which POJOs to use?
- This is the most fundamental question in your webapp architecture
- Talk with client, understand requirements
- Make a Domain Model Diagram that makes sense to them

SpaceTime Example

- Forget Players for now, let's talk about the game
- We conceptually have Teams
- We conceptually have Pilots
- Are Teams first-class objects?
- Are Pilots first-class objects

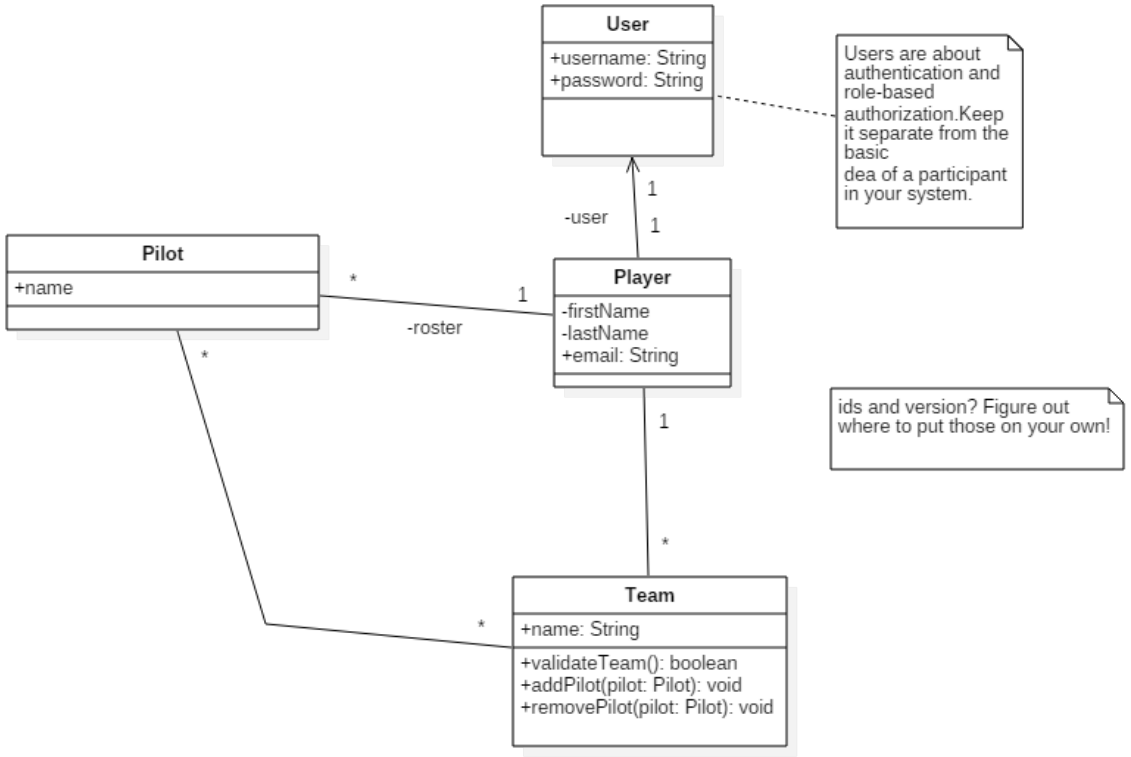
What Behaviours for Teams/Pilots?

- What are the behaviours in this game?
  - Pilots join teams
  - Teams group pilots for the player
  - Team composition varies
  - There are unspecified restrictions on composition
  - Teams will compete

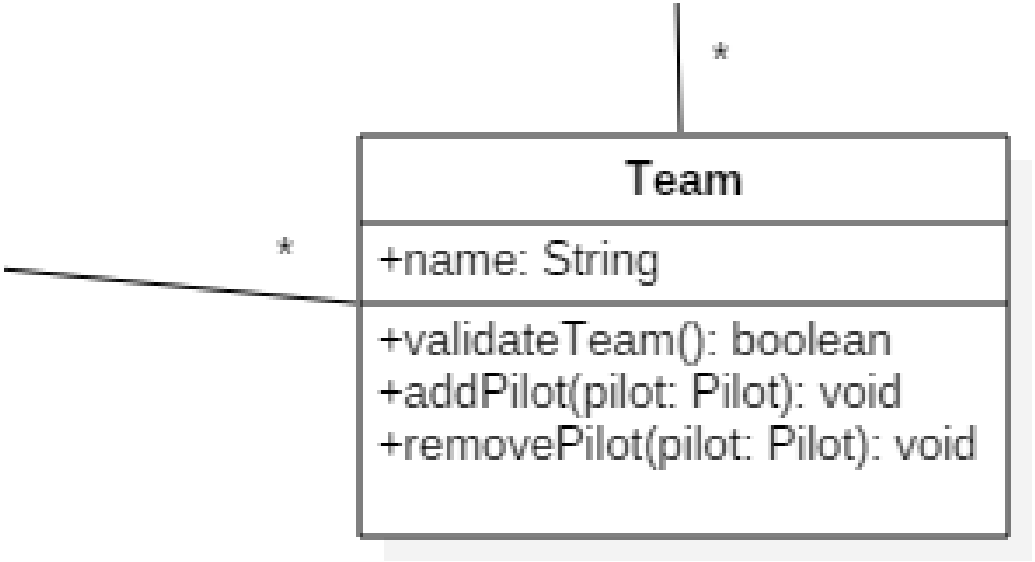
Where Does Team/Pilot Behaviour Belong?

- So, should the Teams track winning?
- Should the Pilots track everything else?

Assignment OO Design Diagram



Conservative OO Design Diagram



What Does Team Record Store?

- Teams store all their pilots
- Does it make sense to do that in a "Team" table?
- What about team names
- Are the rules for validating a team stored here?
- Are records of wins/losses described in the diagrams?
- How do we know who was on a team when it won/lost?
- Other temporal issues: tracking history

What Does Team DM Do?

- Let us leave history and tracking aside for now
- Does it need to validate the rules before saving a Team to the DB?
- What records does it need to store?
- When reading from the database, what should it provide for the Team Domain Object?
- Maybe a list of pilots?
- Is that structural or behavioural? Maybe we can consider that later...

Team Behaviour

- Decide if a Team is valid
- Add Pilots
- Remove Pilots
- . . . leave the rest for now