Thinking about Abstraction

- Abstraction → Conceptualisation, modeling.
- Eg. ‘Red means stop, green means go ahead with caution’.

What’s a good Abstraction?

- In OOP — ‘What’s relevant to the application on hand with reference to the real world?’
- Good Abstraction (GA) has **minimum essential details**. Any detail that isn’t part of the abstraction should be removed/postponed.
- GA represents only what we care about, not the real world object. It represents our perception of the real-world object within the confines of our application.
- In OOP, we use objects to represent abstractions.
- How we model things may depend not on the object itself, but the usage pattern.
    - If we discard the usage pattern and look at the model in isolation, we end up creating a model that is overly complex and often irrelevant, and somewhat hard to use for a certain usage pattern.
    - If we let the usage pattern influence the model, then the abstraction is there to serve the usage rather than being self-serving.
- Abstractions shouldn’t be self-serving, it should be serving the usage of the models in the applications.

Domain

- Domain of the application — people in hospital vs those in a college vs those in a sports team. All are people, but the domain is different.
    - In medical domain, patients can be represented.
    - In a college system, teachers and students can be represented.

  - In a sports team, players/athlete can be represented.
- A car can be represented in different ways depending on the domain of the application.
  - An insurance company will represent a car very differently than a financial domain or a manufacturing domain.
- Definition of the abstraction is important — it may mean different things to people in different fields.
- Best to define the abstraction so everyone is one the same page. A term can mean different things to people in different fields.

Context

- Context — focused area within a domain.
- Domain is big, context is the smaller part about which you care.
- Eg. If you’re talking about taxation laws, you may not need to know/care about everything tax law, because it may not even be applicable.

Bounded Context

- A much narrower region in which we are interested.
- Certain ‘behaviours’ can be ignored.

Domain Driven Design

- Erik Evans book, recommended.