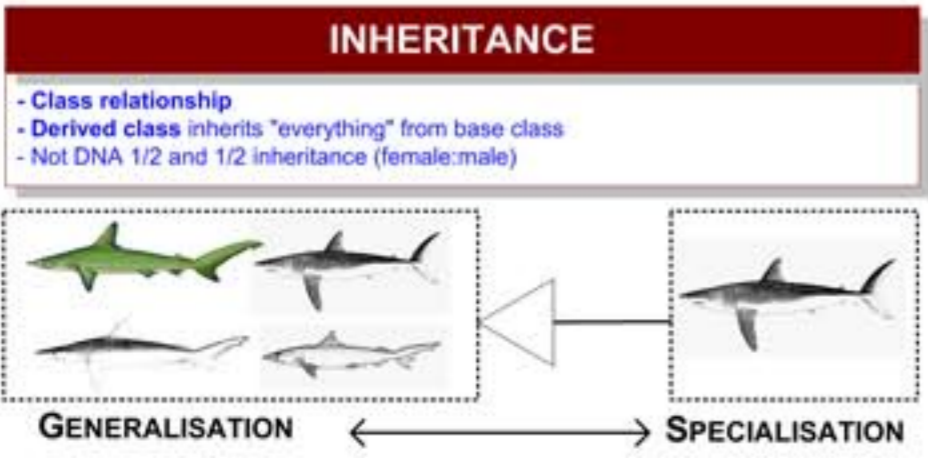
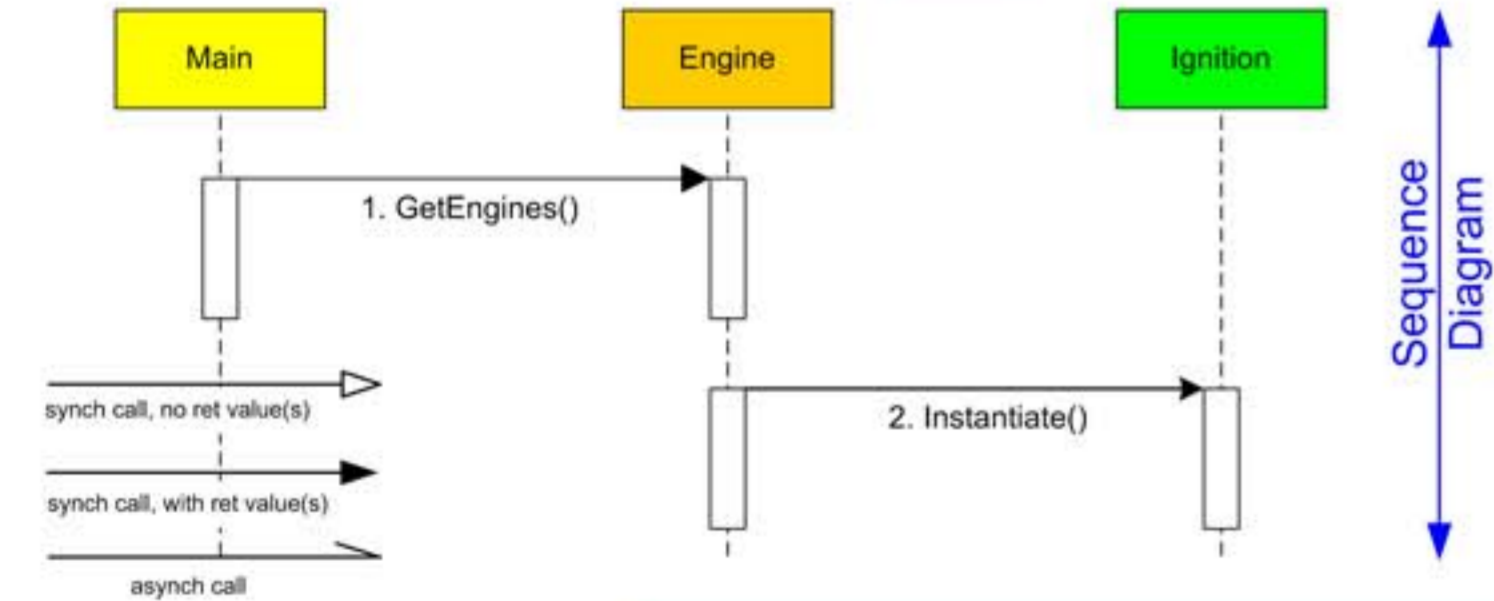
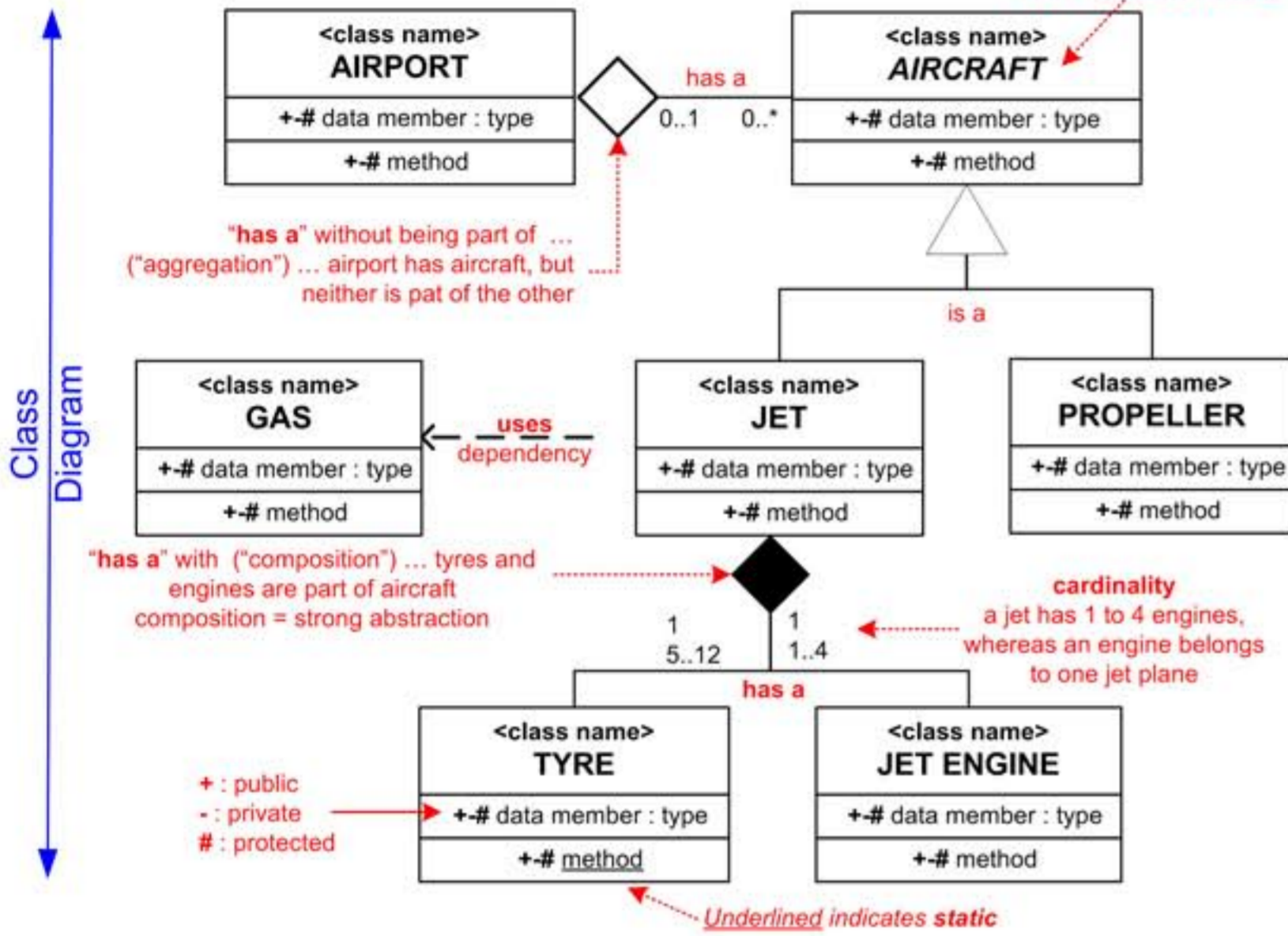


CONCEPTUAL
Represents the concepts in the domain under study, with little or no regard for technology

SPECIFICATION
Looking at interfaces of software, not the implementation.

IMPLEMENTATION
Coding time...

- Perspectives
- Class**: Repository for methods and definition of data members.
 - Object**: Entity with responsibilities, implemented by a class.
 - Instance**: Example of a class ... always an object.
 - Encapsulation**: Any kind of hiding ... data and functionality.
 - Inheritance**: A special kind of another class. Super and Sub Classes.
 - Polymorphism**: Get appropriate behaviour to derived class being referred to.
 - Perspectives**: Different perspectives for looking at objects



STRUCTURAL - HANDLING INTERFACES

Facade

Intent: Simplify usage of an existing system. Use only a subset of a complex system. -or- Interact with a system in a particular way. Solution: Façade presents a new interface for an existing system.

Adapter

Intent: Match an existing object beyond your control. System has right data and behaviour, but wrong interface. Adapter provides a wrapper interface.

Bridge

Intent: Decouple set of implementations from objects using them. Derivations of abstract class must use multiple implementations without a class number explosion. Bridge defines an interface for all implementations to use and have derivations and abstract class use that.

Decorator

Intent: Attach additional responsibility to object dynamically. Although an object full fills basic functions, you need to add additional functionality to occur before or after base. Solution: Extend functionality of object without subclassing.

CREATIONAL - INSTANTIATION

Singleton

Intent: You want only one of an object. Several objects need to refer to the one and same thing. Guarantees one instance.

Factory

Intent: Define an interface for creating an object, but let subclasses decide which class to instantiate. Class needs to instantiate derivation of another class but does not know which one. Derived class makes decision on which class to instantiate and how.

Abstract Factory

Intent: Want to have families or sets of objects for particular clients. Families of related objects need to be instantiated. Coordinates the creation of families of objects.

I am ALIVE principle

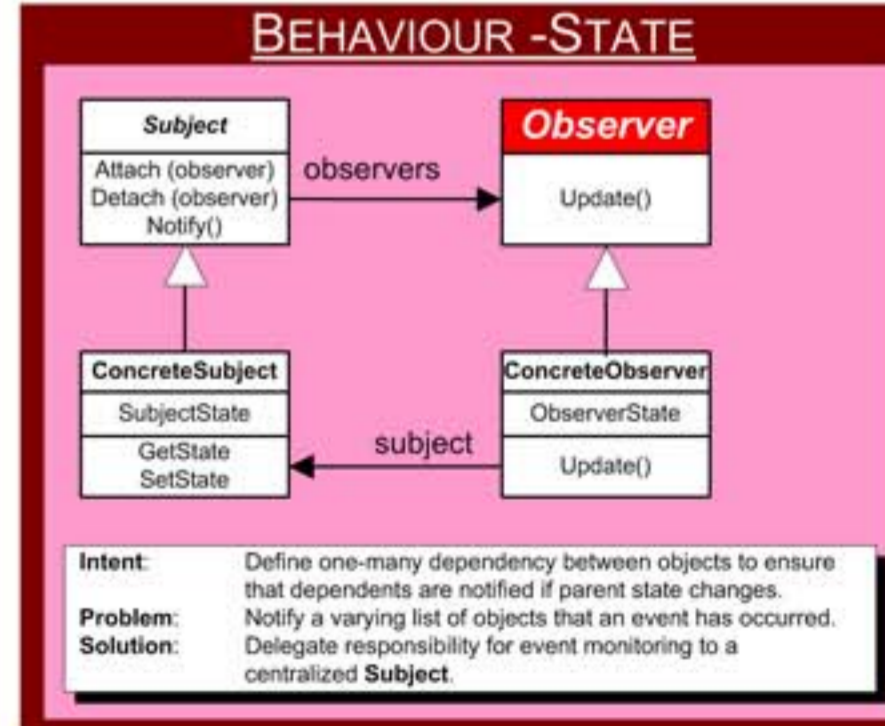
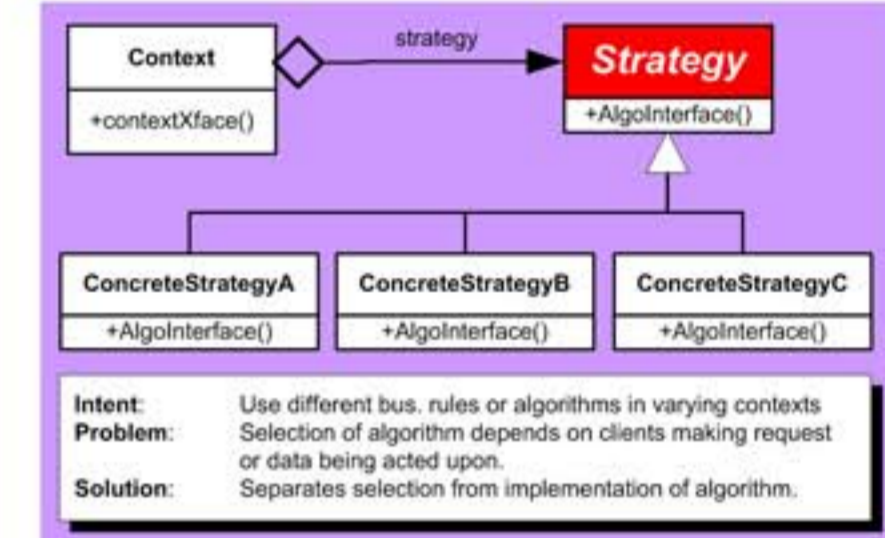
Keep family **POLITICS** at bay

- loosely coupled
- no mother-in-law objects

Data is **PRIVATE**, Data is **PRIVATE**

Containment is better than Inheritance

No **BIG-BANG** theories please



- GOLDEN RULES**
1. **OC**P Principle → Open to extension, Closed for modification
 2. Loosely coupled and tightly cohesive ... low dependency on others, but work well with others
 3. One way dependencies
 4. Base classes should be abstract
 5. **Avoid**: Rigidity, Fragility and Immobility