Design patterns - singleton & factory

SOLID (Single responsibility, Open/closed, Liskov-substitution, Interface segregation, Dependency-inversion)

- coding standard which helps developers avoid bad design in development
- makes code more extensible, more logical, and easier to read
- badly designed software can become inflexible and brittle
- small changes in software can result in bugs that break other parts of code
- Single Responsibility
  - classes should only have 1 responsibility
  - changes to part of software should only effect specification of 1 class
- Open/Closed
  - class should be open for extensions but closed for modifications
  - modules and classes must be designed so that new functionality can be added when new requirements are generated
  - can use inheritance and implement interfaces to achieve this
- Liskov-Substitution
  - derived classes must implement all methods and fields of parent classes
  - after method and field implementation any derived class can be used instead of parent class and will behave same way
  - ensures derived class does not affect behavior of parent class
  - derived class must be substitutable for base class
- Interface Segregation

    - all should have specific purpose or responsibility
    - classes that do not share interface purpose should not be forced to implement the interface
    - larger interfaces are more likely to include methods not all classes can implement
- Dependency-Inversion
    - high-level modules/classes implement business rules or logic in system (front-end)
    - low-level modules/classes deal with more detailed operations (writing to database or passing messages to OS or services)
    - must keep high-level and low-level modules/classes loosely coupled as much as possible
        - make both dependent on abstractions instead of knowing eachother

---

Project 1 Domain Requirements

Review Session