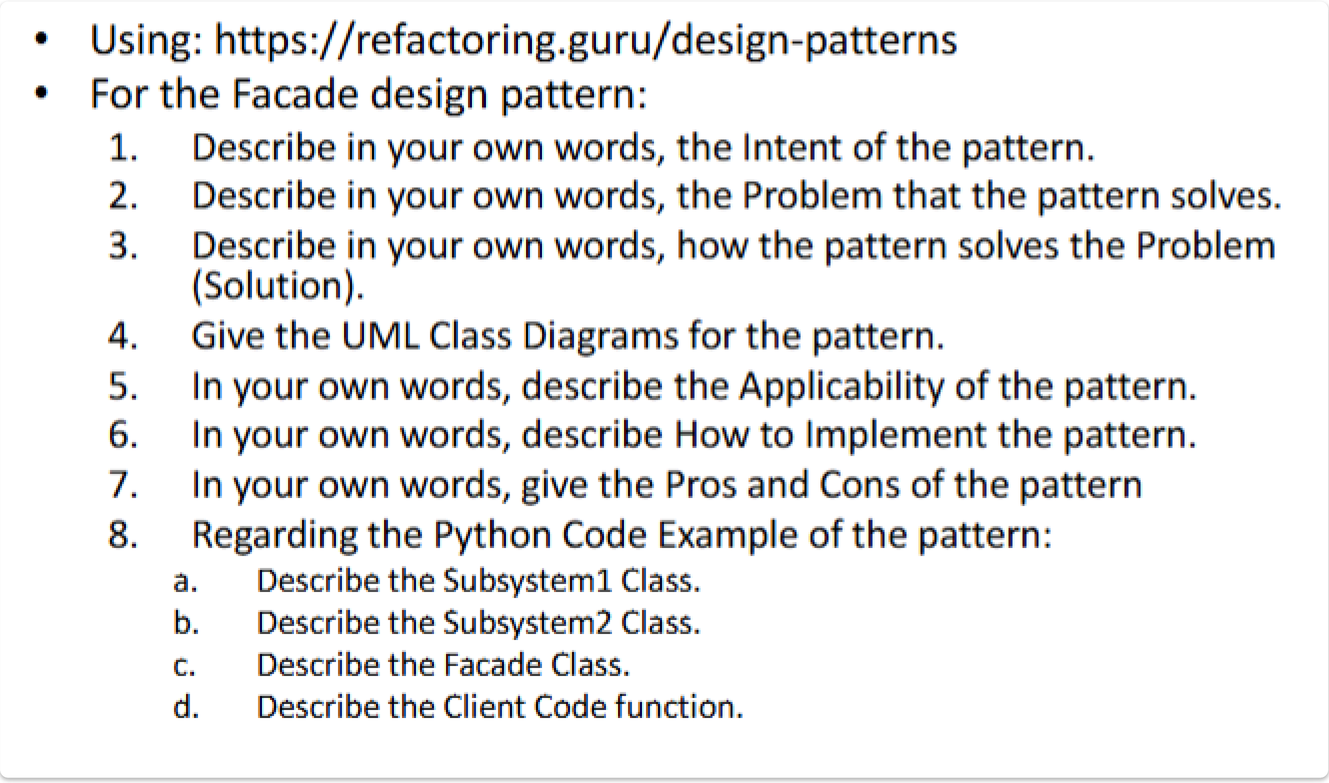
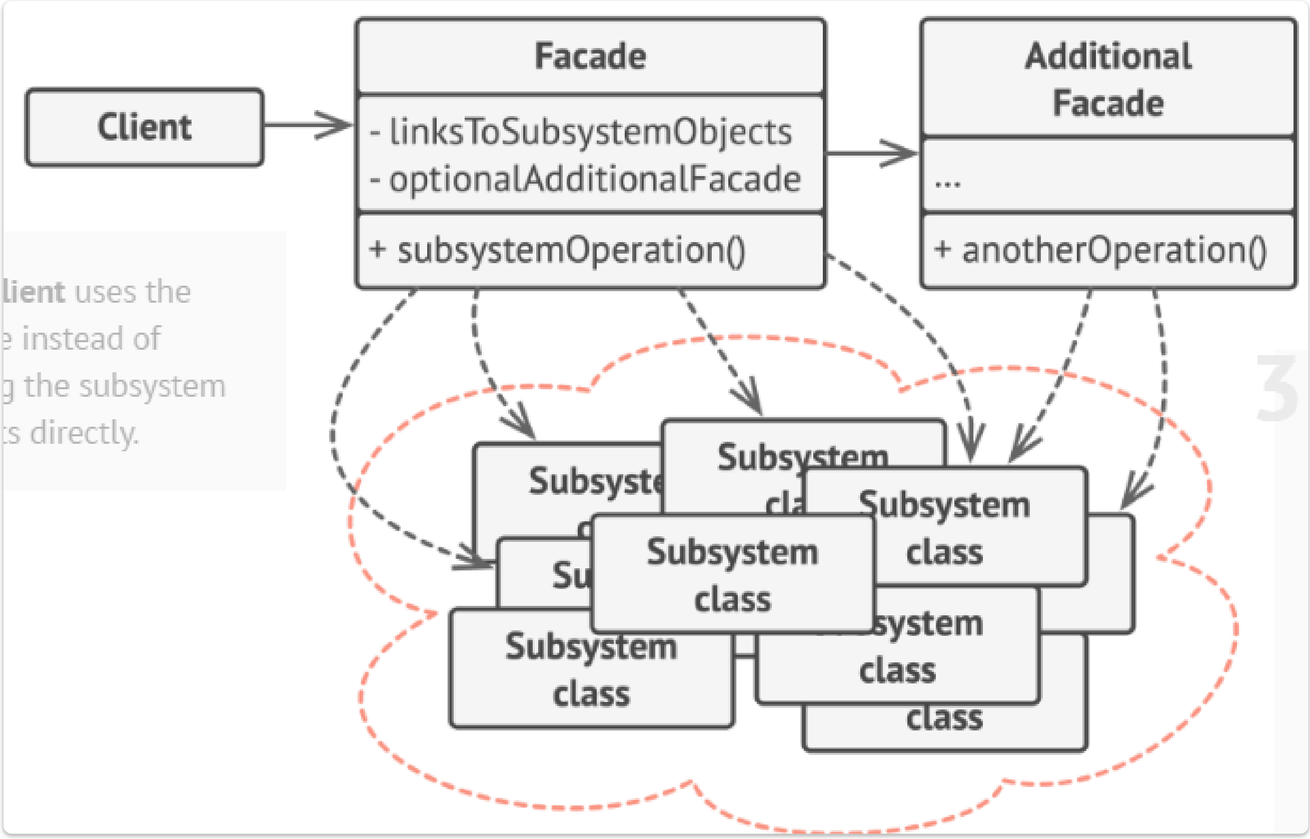
Design Patterns



- Using: https://refactoring.guru/design-patterns
- For the Facade design pattern:
  1. Describe in your own words, the Intent of the pattern.
  2. Describe in your own words, the Problem that the pattern solves.
  3. Describe in your own words, how the pattern solves the Problem (Solution).
  4. Give the UML Class Diagrams for the pattern.
  5. In your own words, describe the Applicability of the pattern.
  6. In your own words, describe How to Implement the pattern.
  7. In your own words, give the Pros and Cons of the pattern
  8. Regarding the Python Code Example of the pattern:
     a. Describe the Subsystem1 Class.
     b. Describe the Subsystem2 Class.
     c. Describe the Facade Class.
     d. Describe the Client Code function.

1. Provides a simple interface that isn’t too specific to outline a set of libraries or frameworks
2. Messy code due to many libraries or complex sets of classes that need to be done in the right order.
3. Introduces a facade class that handles all of the logic and order to the class structure, then builds up subsystems behind it. The client uses the facade instead of calling the subsystems directly.

4. 

5. When you want to have a simple interface for a somewhat complex subsystem, if you can condense the inputs into a facade to handle the inputs better, it makes the subsystems less complicated
6. Check if you can make your current interface simpler, if so, implement a new facade class that redirects calls from the client to the subsystems. Ideally all code should go through the facade and to the subsystems
7. Pros: You can isolate the code to be less complex. Cons: It can grow too large and just be another complex object to maintain
8. .
    1. Subsystem1 Class: Accepts requests from client or facade. Provides methods that can be called directly. Not directly connected to the faced, but can be called by it
    2. Subsystem 2 Class: Also independent but with different methods. Can also be used on its own or by the facade
    3. Facade Class: Uses Subsystem1 and Subsystem2. Its constructor can create and use Subsystem classes. Its operation class calls subsystem methods. Provides a unified operation for clients and does not directly show the operations of the subsystems
    4. Cient Code: Shows how the client interacts. Calls only the facades operation() method to make it simple. Stays clean and decoupled. The client only needs one high-level call to run everything else.