Formating Instructions: Please use this provided LaTeX template to complete your homework. When including figures such as UMLs, it is highly encouraged to use tools (graphviz, drawio, tikz, etc..). Figures may be hand draw, however, these may not receive credit if the grader cannot read it. For ease of grading, when including code please have it as part of the same pdf as the question while also including correct formatting/indent, preferably syntax highlighting. Latex includes the minted, or lstlisting package as a helpful tool. For this assignment all code must be full code (no pseudo-code) and be written in either Java or C++. However, implements may ignore all logic not relevant to the design pattern with simple print out statements of "[BLANK] logic done here"

Question Instructions: In this homework assignment, you will apply one or more Structural patterns discussed in the lectures.
This is a group assignment that requires two students per group.
For each question:

1. Give the name of the design pattern(s) you are applying to the problem.
2. Present your reasons why this pattern will solve the problem. Please be specific to the problem and do not give general applicability statements. If there is an alternative pattern, explain why you preferred this one..
3. Show you design with a UML class diagram. If the pattern collaborations would be more visible with another diagram (e.g. sequence diagram), give that diagram as well.
   (a) Your diagram should show every participant in the pattern including the pattern related methods.
   (b) In pattern related classes, give the member (method and attribute) names that play a role in the pattern and effected by the pattern. Optionally, include the member names mentioned in the question. You are encouraged to omit the other methods and fields.
   (c) For the non-pattern related classes, you are not expected to give detailed class names etc. You may give a high-level component, like "UserInterface" or "DBManagement"
4. Give Java or C++ code for your design showing how you have implemented the pattern.
   (a) Pattern related methods and attributes should appear in the code
   (b) Client usage of the pattern should appear in the code
   (c) Non-pattern related parts of the methods could be a simple print. (e.g. "System.out.println()", "cout")
5. Evaluate your design with respect to SOLID principles. Each principle should be addressed, if a principle is not applicable to the current pattern, say so.

1. (25 points) We are developing a software application for an energy company, and we need to access the weather history data from a database. We make queries by date or by location. An example weather data is: "Golden CO, November 10th, 2021, daytime, average temperature 54°F, 57% Humidity". This information does not change once it is inserted into the weather history database.

   (a) (10 points) For performance considerations, we do not want to access the database each time we need the weather data in our application. Since the weather data does not change once it enters the history, we could use caching (preloading the data) and return the cashed values whenever weather information is requested instead of database access. We access the database only when the requested data is not in the cache. Whether the data is cached or not should be a concern for the rest of the application. Suggest a structural design pattern for this purpose. (address all items 1-5)

   (b) (7 points) We have switched to a remote weather database. Currently, the remote connection, sending and receiving data using HTTP or Rest API request are all over the codebase. Suggest a structural pattern so that the rest of the application is not dependent on such remote procedures. (address items 1-4)

   (c) (8 points) I want to restructure my code. Currently there are five interconnected classes that are responsible for weather data access and processing. I want to give simplified access to these from my energy related components. (You may assume any name for the five interconnected classes). Suggest a structural design pattern that would simplify this and discuss advantages and disadvantages of your design. (address items 1,2,3,5)

2. (15 points) Your team has designed a system to manage the security of a hierarchical network of computers, implemented using the Composite pattern. The network consists of a main server (a composite node) connected to several sub-servers (composite nodes), each connected to multiple client computers (leaf nodes). The system supports the following requirements:

   Security Checks: Each component (main server, sub-server, or client) supports basic security checks: scanning for malware, updating antivirus software, and applying security patches. As an admin, you can initiate these checks at any component (e.g., the main server for the entire network, a sub-server for its sub-network, or a single client). Dynamic Additional Checks: Certain components require additional security measures based on specific conditions:

   Servers handling sensitive data must perform an additional encryption check. Any component that has failed any of the last five security checks requires an extra deep security scan. If a component passes five consecutive checks, this scan is removed. (Note: This is a simplified scenario for educational purposes.)

   Flexibility: The system must support adding new types of security checks and new component types in the future without modifying existing code.

   Given that the network hierarchy is implemented using the Composite pattern, design a solution using a structural design pattern to handle the dynamic addition and removal of security checks as described. Justify your choice of pattern, explaining how it complements the Composite structure and meets the flexibility requirements.

   (address all items 1-5)

Use Table 2 for Question1 and Table 3 for Question2.

Table 1: Grading Rubric for **10** points questions

1 (1 point)	0	missing or incorrect
	+1	correct pattern
2 (1 point)	0	missing
	+1	the reason provided correctly describes an advantage of the pattern and is specifically beneficial to this scenario
3 (3 points)	0	missing
	+1	includes all participants (including client) that play a role in the pattern
	+1	all class relations are correct
	+1	includes all class members that are related to the pattern
4 (3 points)	0	missing
	+1	includes all pattern related methods and attributes
	+1	includes client usage
	+1	correctly implements and uses all pattern related methods
5 (2 points)	0	missing
	+2	correctly lists multiple ways the pattern benefits a user

Table 2: Grading Rubric

Grading Rubric for Part A - use the grading rubric for 10 points questions		
Grading Rubric for Part B - 7 points		
1 (1 point)	0	missing or incorrect
	+1	correct pattern
2 (1 point)	0	missing
	+1	the reason provided correctly describes an advantage of the pattern and is specifically beneficial to this scenario
3 (2 points)	0	missing
	+1	includes all participants (including client) that play a role in the pattern
	+1	all class relations are correct
4 (3 points)	0	missing
	+1	includes all pattern related methods and attributes
	+1	includes client usage
	+1	correctly implements and uses all pattern related methods
Grading Rubric for Part C - 8 points		
1 (1 point)	0	missing or incorrect
	+1	correct pattern
2 (2 point)	0	missing
	+1	the reason provided correctly describes an advantage of the pattern and is specifically beneficial to this scenario
3 (3 points)	0	missing
	+1	includes all participants (including client) that play a role in the pattern
	+1	all class relations are correct
	+1	includes all class members that are related to the pattern
5 (2 points)	0	missing
	+2	correctly lists multiple ways the pattern benefits a user

Table 3: Grading Rubric for **15** points questions

1 (1 point)	0	missing or incorrect
	+1	correct pattern
2 (1 point)	0	missing
	+1	the reason provided correctly describes an advantage of the pattern and is specifically beneficial to this scenario
3 (6 points)	0	missing
	+2	includes all participants (including client) that play a role in the pattern
	+2	all class relations are correct
	+2	includes all class members that are related to the pattern
4 (5 points)	0	missing
	+1	includes all pattern related methods and attributes
	+2	includes client usage
	+2	correctly implements and uses all pattern related methods
5 (2 points)	0	missing
	+2	correctly lists multiple ways the pattern benefits a user