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. Explain how this design solves the problem.
6. 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. (14 points) As a rule of thumb, we do not hard code configuration information, such as database URL, inside the code. Such information is written in a configuration file.

   Your job is to implement a configuration manager for our multi-threaded application. We are using this manager so that configuration information could be used across the components of the system. Any object in the system should be able to call a lookup function of this configuration manager using a key. We want this manager to be initialized lazily.

   Give a design using a creational design pattern. (address all items 1-6)

2. (16 points) We are developing an academic publication exchange application on the web. Our team is responsible for the back end of the system. There are two core components of this system: Books and Journals. (Conference Proceedings are also books.) In this system, once a Book or a Journal object is created, it should not be modified. This is to ensure data consistency and to prevent potential issues with concurrent access in a multi-threaded environment.

   Each book has the following properties:

   1. title: The title of the book. A string or char*
   2. List of authors: The authors of the book. (each element is an object of type Contributor).
   3. publisher: The publisher of the book. There exist a Publisher class.
   4. isbn: The ISBN number of the book. An integer
   5. publicationYear: The year the book was published. //type is your choice
   6. numberOfPages: The number of pages in the book. An integer

   For the current development cycle, we only need to implement the Book class.

   We need the Book class to be an immutable class, i.e. no setter methods, but only getter methods. The client code, let's say a Library, wants to create Book instances. Only the title, number of pages, and the publisher are mandatory, other fields are optional. There is one constraint on these optional fields. The publication year and the publisher should be set before ISBN since we will validate the given ISBN value using these two values.

   Give a design so that Library can have immutable Book instances. I do not want the library object to be exposed to this complicated logic to create a Book instance. So please have a dedicated object that reads a file; creates an immutable Book instance; and returns this instance to the Library.

   (address all items 1-6)

3. (15 points) We have implemented an inventory system for books. When a book enters the inventory, a book instance is created. Then, the system takes care of the visualizations, reporting, and all kinds of inventory management tasks regardless of how a book is represented.

   The inventory management system is quite lightweight. The customers are very satisfied with the functionalities of our inventory management system, so we want to broaden our customer base. We want to repurpose the system for other kinds of small products, like vinyl records or plush toys, in the future for different businesses.

   This is a common strategy in business and technology, where existing resources or systems are repurposed to meet new needs or opportunities. We will sell our software as a vinyl inventory system to one customer, as a book inventory system to another. (similar to all the clicker games like crab wars, idle miner, adventure capitalist etc.)

   Which creational pattern can help us for this purpose? Show how to customize it as a vinyl inventory. (Please do not use if-then-else or switch to select the product type.)

   (address all items 1-6)

Table 1: Grading Rubric for **14** 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 (5 points)	0	missing
	+2	includes all participants (including client) that play a role in the pattern
	+2	all class relations are correct
	+1	includes all class members that are related to the pattern
4 (4 points)	0	missing
	+1	includes all pattern related methods and attributes
	+2	includes client usage
	+1	correctly implements and uses all pattern related methods
5 (1 points)	0	missing
	+1	uses the specific names and roles of their designed classes and their collaborations to demonstrate how the problem is systematically solved.
5 (2 points)	0	missing
	+1	correctly identifies at least two relevant SOLID principles and accurately assesses whether the design adheres to them.
	+2	correctly identifies all of the relevant SOLID principles and accurately assesses whether the design adheres to them, providing specific examples from their code to justify the evaluation.

Table 2: Grading Rubric for **16** 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 (5 points)	0	missing
	+2	includes all participants (including client) that play a role in the pattern
	+2	all class relations are correct
	+1	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 or not enough detail
	+2	uses the specific names and roles of their designed classes and their collaborations to demonstrate how the problem is systematically solved.
5 (2 points)	0	missing
	+1	correctly identifies at least two relevant SOLID principles and accurately assesses whether the design adheres to them.
	+2	correctly identifies all of the relevant SOLID principles and accurately assesses whether the design adheres to them, providing specific examples from their code to justify the evaluation.

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 (5 points)	0	missing
	+2	includes all participants (including client) that play a role in the pattern
	+2	all class relations are correct
	+1	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 (1 points)	0	missing
	+1	uses the specific names and roles of their designed classes and their collaborations to demonstrate how the problem is systematically solved.
5 (2 points)	0	missing
	+1	correctly identifies at least two relevant SOLID principles and accurately assesses whether the design adheres to them.
	+2	correctly identifies all of the relevant SOLID principles and accurately assesses whether the design adheres to them, providing specific examples from their code to justify the evaluation.