Assignment : Understanding Inheritance vs. Composite

Objective:

Design and implement a system to demonstrate your understanding of the differences between inheritance and the composite design pattern. This system should clearly showcase scenarios where each approach is appropriate and how they can be used to solve different problems.

Scenario:

Imagine you are designing a simple graphic editor application. This application needs to manage different shapes (like circles, rectangles, and lines) and groups of shapes. The goal is to demonstrate how inheritance and composite can be used to manage these shapes and their groups.

Requirements:

Using Inheritance:

Create a base class Shape with methods for common behaviors (e.g., draw(), move(x, y)).
Implement concrete shape classes like Circle, Rectangle, and Line that inherit from Shape.

Using Composite:

Create a CompositeShape class that also inherits from Shape but can contain multiple Shape objects.
Implement methods in CompositeShape to add, remove, and manage child shapes. Ensure that composite shapes can also perform common behaviors by delegating to their child shapes.
Demonstration:

Provide examples showing the use of inheritance to create individual shapes.
Provide examples showing the use of the composite pattern to create groups of shapes and perform operations on these groups.
Highlight the differences in flexibility, reusability, and scalability between the two approaches.
Instructions:

Define the base Shape class with methods draw() and move(x, y).
Implement concrete shape classes (Circle, Rectangle, Line) using inheritance.
Implement the CompositeShape class to manage groups of shapes.
In the main class, create examples that illustrate the differences and use cases for inheritance and composite.
Example Output:

Using Inheritance:

Drawing individual shapes: Circle, Rectangle, Line.
Moving individual shapes: Circle, Rectangle, Line.
Using Composite:

Creating a group of shapes and performing operations on the group:
Drawing a composite shape containing a Circle and a Rectangle.

Moving a composite shape containing a Circle and a Line.

Adding a new shape to an existing composite and performing operations.

Submission:

Submit your code along with a brief explanation of how you implemented both inheritance and composite patterns, the reasoning behind your design choices, and the differences in flexibility and reusability between the two approaches.