**The Memento design pattern** helps you save and restore an object's state without exposing its internal details. It's like taking a snapshot of something so you can go back to it later, even if the thing you're snapshotting is changing.

**Originator (Calculator) Class:**

* Imagine you have a calculator that can perform calculations and keep track of the history. This calculator is the "originator."
* It has methods to add calculations to its history and display that history.

**Memento (CalculatorMemento) Class:**

* In the real world, you might take a photo to capture a moment. Here, the CalculatorMemento class is like a snapshot of the calculator's history at a particular point.
* It holds the history, but it doesn't reveal how the calculator works or what's inside it.

**Caretaker (HistoryTracker) Class:**

* Imagine you have a friend who keeps your photo album. That friend is like the "caretaker" in this pattern.
* The HistoryTracker class keeps track of snapshots (mementos) of the calculator's history.

**Client Code (Main Method):**

* Here's where everything comes together.
* Create a calculator and a history tracker.
* Add calculations to the calculator's history and save snapshots of its state at different points.
* Display the history.
* Restore the calculator's history from a saved snapshot using the history tracker.

In simple English, the Memento pattern lets an object (like your calculator) take snapshots of its own state. These snapshots are stored in a separate place (the caretaker), allowing you to revert the object back to previous states whenever you want. It's like having a time machine for objects, so you can go back to how they were before. This can be useful for implementing undo features, tracking changes, or managing application history.

**Step 1: Create the Originator (Calculator) Class**

The Calculator class maintains calculation history and has methods to add calculations, save history, restore history from a memento, and display history.

class Calculator

{

private List<string> calculationHistory = new List<string>();

public void AddCalculation(string calculation)

{

calculationHistory.Add(calculation);

}

public void DisplayHistory()

{

Console.WriteLine("Calculation History:");

foreach (var calculation in calculationHistory)

{

Console.WriteLine(calculation);

}

}

}

**Step 2: Create the Memento (CalculatorMemento) Class**

The CalculatorMemento class represents a snapshot of the Calculator's state (calculation history) at a specific point in time.

class CalculatorMemento

{

private readonly List<string> state;

public CalculatorMemento(List<string> history)

{

state = new List<string>(history);

}

public List<string> GetState()

{

return state;

}

}

**Step 3: Create the Caretaker (HistoryTracker) Class**

The HistoryTracker class manages and keeps track of multiple mementos, representing different states of the Calculator.

class HistoryTracker

{

private List<CalculatorMemento> mementos = new List<CalculatorMemento>();

public void SaveMemento(CalculatorMemento memento)

{

mementos.Add(memento);

}

public CalculatorMemento GetMemento(int index)

{

return mementos[index];

}

}

**Step 4: Client Code (Main Method)**

In the Main method, you create instances of the Calculator and HistoryTracker classes. You perform the following actions:

* Add calculations to the calculator and save the history using the SaveMemento method.
* Display the history using the DisplayHistory method.
* Restore the history using the RestoreHistory method by retrieving a memento from the HistoryTracker.

static void Main(string[] args)

{

Calculator calculator = new Calculator();

HistoryTracker history = new HistoryTracker();

calculator.AddCalculation("5 + 3 = 8");

history.SaveMemento(calculator.SaveHistory());

calculator.AddCalculation("10 \* 2 = 20");

history.SaveMemento(calculator.SaveHistory());

calculator.DisplayHistory();

calculator.RestoreHistory(history.GetMemento(0));

calculator.DisplayHistory();

calculator.RestoreHistory(history.GetMemento(1));

calculator.DisplayHistory();

}

**Explanation**:

* The Calculator class (originator) maintains a history of calculations and methods to interact with it.
* The CalculatorMemento class (memento) holds the state (calculation history) of the Calculator at a specific point.
* The HistoryTracker class (caretaker) stores and manages multiple mementos.
* In the Main method, you create instances and perform actions to add calculations, save mementos, display history, and restore history. This demonstrates how mementos can capture and restore the state of the Calculator object.

The Memento pattern helps maintain a history of states and provides a way to revert to previous states, which can be useful for undo functionality or tracking changes in an application's state.