using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Memento

{

class Originator

{

private string \_state;

public Originator(string state)

{

\_state = state;

Console.WriteLine("Originator : My Initial state is : "+\_state);

}

public void DoSomething()

{

Console.WriteLine("Originator : I am doing something important");

this.\_state = this.GenerateRandomString(30);

}

public string GenerateRandomString(int length = 30)

{

string allowedSymbols = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

string result = "";

while (length>0)

{

result += allowedSymbols[new Random().Next(0, allowedSymbols.Length)];

Thread.Sleep(12);

length--;

}

return result;

}

public IMemento Save()

{

return new TextMemento(this.\_state);

}

public void Restore(IMemento memento)

{

if(!(memento is TextMemento))

{

throw new Exception("Unknown memento class " + memento.ToString());

}

this.\_state = memento.GetState();

}

}

public interface IMemento

{

string GetName();

string GetState();

DateTime GetDate();

}

public class TextMemento : IMemento

{

private string \_state;

private DateTime \_date;

public TextMemento(string state)

{

this.\_state = state;

this.\_date = DateTime.Now;

}

public DateTime GetDate()

{

return \_date;

}

public string GetName()

{

return $"{\_date} / {\_state.Substring(0, 9)}";

}

public string GetState()

{

return \_state;

}

}

class CareTaker

{

private List<IMemento> \_mementos = new List<IMemento>();

private Originator \_originator = null;

public CareTaker(Originator originator)

{

\_originator = originator;

}

public void Undo()

{

if (\_mementos.Count == 0)

{

return;

}

//var memento = \_mementos.Last();

//this.\_mementos.Remove(memento);

var memento = \_mementos[Index];

Index--;

Console.WriteLine("Care Taker : Restoring state" + memento.GetName());

try

{

\_originator.Restore(memento);

}

catch (Exception)

{

this.Undo();

}

}

public void Redo()

{

var memento = \_mementos[Index];

Index++;

Console.WriteLine("Care Taker : Restoring state" + memento.GetName());

}

public int Index { get; set; } = -1;

public void BackUp()

{

Console.WriteLine("\nCareTaker Saving Originator state . . . ");

this.\_mementos.Add(\_originator.Save());

Index++;

}

public void ShowHistory()

{

Console.WriteLine("CareTaker : Here\'s the list of mementos");

foreach (var item in \_mementos)

{

Console.WriteLine(item.GetName()); ;

}

}

}

class Program

{

static void Main(string[] args)

{

Originator originator = new Originator("Super duper super puper super");

CareTaker careTaker = new CareTaker(originator);

careTaker.BackUp();

originator.DoSomething();

Thread.Sleep(1500);

careTaker.BackUp();

originator.DoSomething();

Thread.Sleep(1500);

careTaker.BackUp();

originator.DoSomething();

Thread.Sleep(1500);

Console.WriteLine();

careTaker.ShowHistory();

Console.WriteLine("\nClient : Now , let\'s rollback!\n");

careTaker.Undo();

Console.WriteLine();

careTaker.Redo();

Console.WriteLine();

careTaker.Undo();

Console.WriteLine();

careTaker.ShowHistory();

careTaker.Undo();

Console.WriteLine();

careTaker.Undo();

Console.WriteLine();

}

}

}