Behavioral Pattern: Memento

Kevin Dockx
Architect

@KevinDockx https://www.kevindockx.com

Coming Up

Describing the memento pattern

- Storing a command's state to support undo actions

Structure of the memento pattern

Coming Up

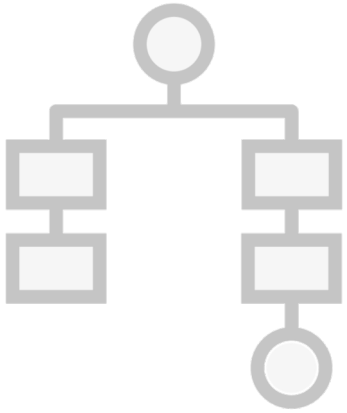
Use cases for this pattern

Pattern consequences

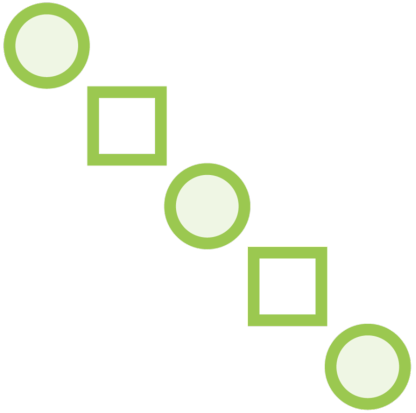
Related patterns

Describing the Memento Pattern

Creational



Structural



Behavioral

Memento

The intent of this pattern is to capture and externalize an object's internal state so that the object can be restored to this state later, without violating encapsulation.

Describing the Memento Pattern

Supporting undo when working with commands

- Consider the command as a class that has internal state that needs to be stored

```
public class AddEmployeeToManagerList : Icommand
{
       private readonly IEmployeeManagerRepository _employeeManagerRepository;
       private readonly int _managerId;
       private readonly Employee? _employee;

// implementation ... }
```

Describing the Memento Pattern

```
public class AddEmployeeToManagerList : Icommand
{
       private readonly IEmployeeManagerRepository _employeeManagerRepository;
       private readonly int _managerId;
       private readonly Employee? _employee;

// implementation ... }
```

Describing the Memento Pattern

```
public class CommandManager
{
	private readonly Stack<ICommand> _commands = new Stack<ICommand>();

	public void Invoke(ICommand command)
	{ // implementation
	}

	public void Undo()
	{ // implementation
	}
}
```

Describing the Memento Pattern

```
public class CommandManager
{
	private readonly Stack<ICommand> _commands = new Stack<ICommand>();

	public void Invoke(ICommand command)
	{ // implementation
	}

	public void Undo()
	{ // implementation
	}
}
```

Describing the Memento Pattern

```
public class AddEmployeeToManagerList : Icommand
{
       private readonly IEmployeeManagerRepository _employeeManagerRepository;
       private readonly int _managerId;
       private readonly Employee? _employee;

// implementation ... }
```

Describing the Memento Pattern

```
public class AddEmployeeToManagerList : Icommand
{
       private readonly IEmployeeManagerRepository _employeeManagerRepository;
       public readonly int _managerId;
       public readonly Employee? _employee;

// implementation ... }
```

Describing the Memento Pattern

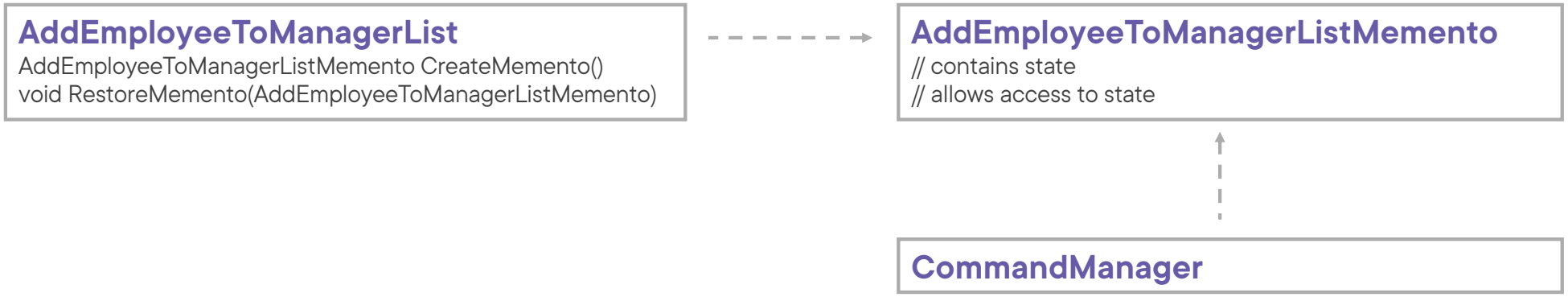
Making internal state public breaks encapsulation

Describing the Memento Pattern

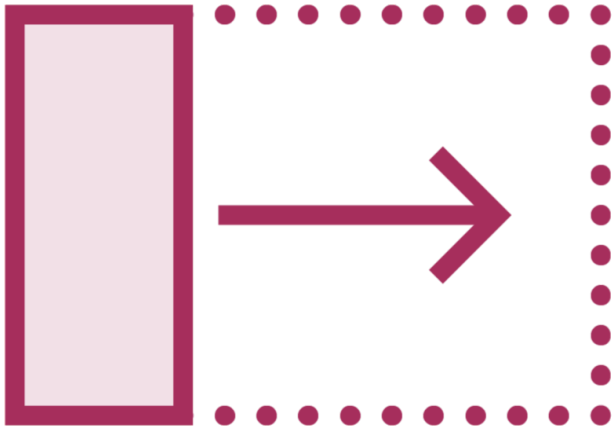
AddEmployeeToManagerListMemento
// contains state
// allows access to state

Describing the Memento Pattern

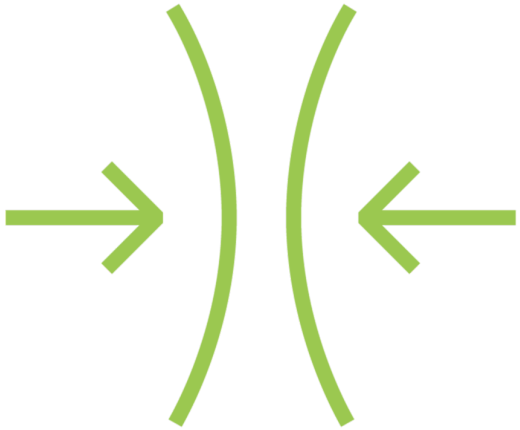
Describing the Memento Pattern



Provide 2 Interfaces to the Memento

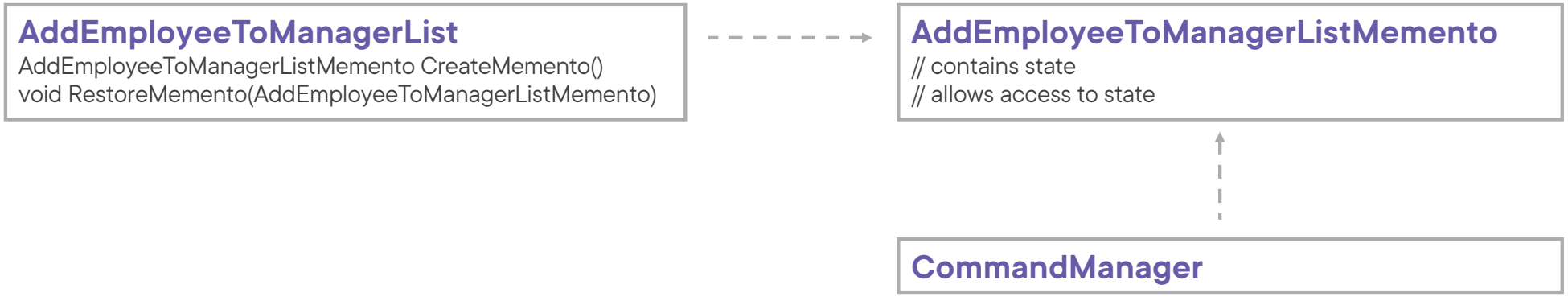


Wide interface, used by the command to create the memento

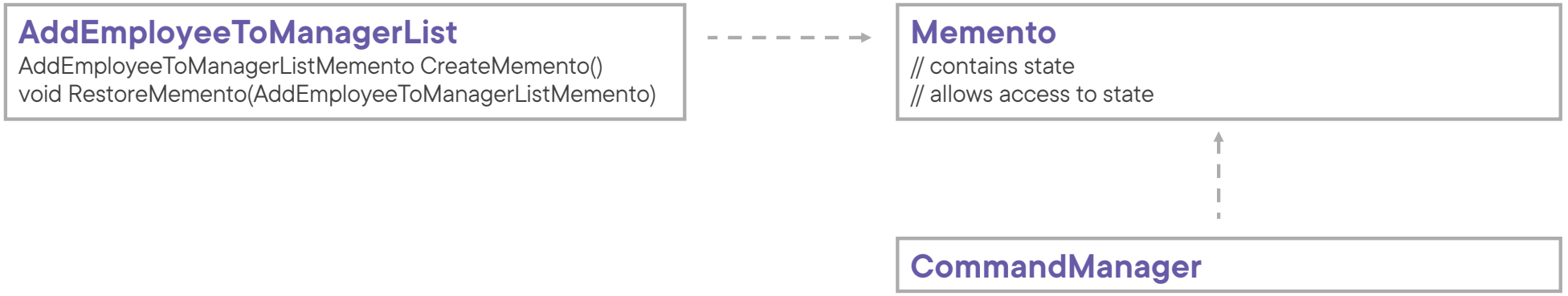


Narrow interface, used by the command manager

Structure of the Memento Pattern

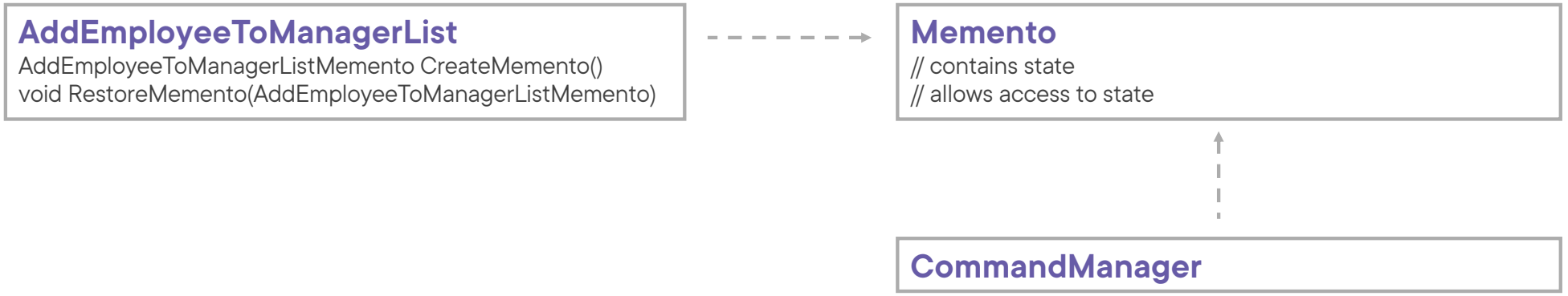


Structure of the Memento Pattern

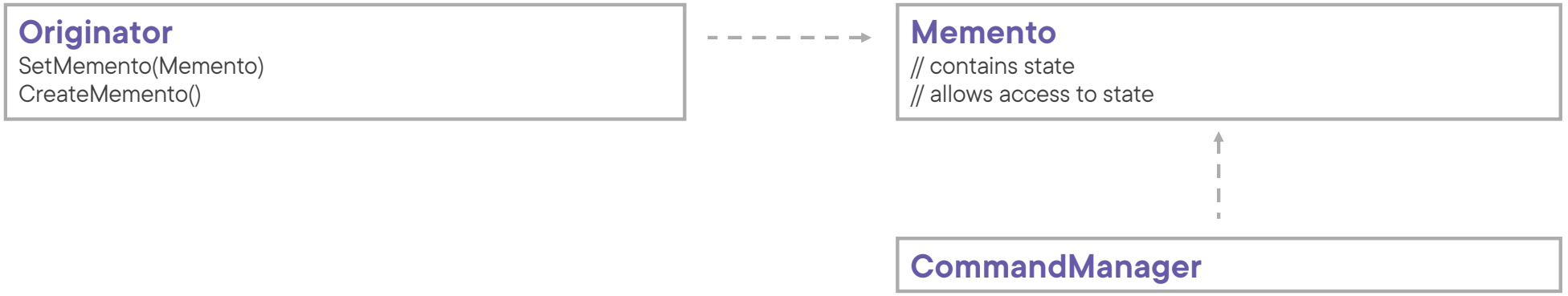


Memento stores the internal state of the originator. The state should be protected against access by other objects as much as possible.

Structure of the Memento Pattern



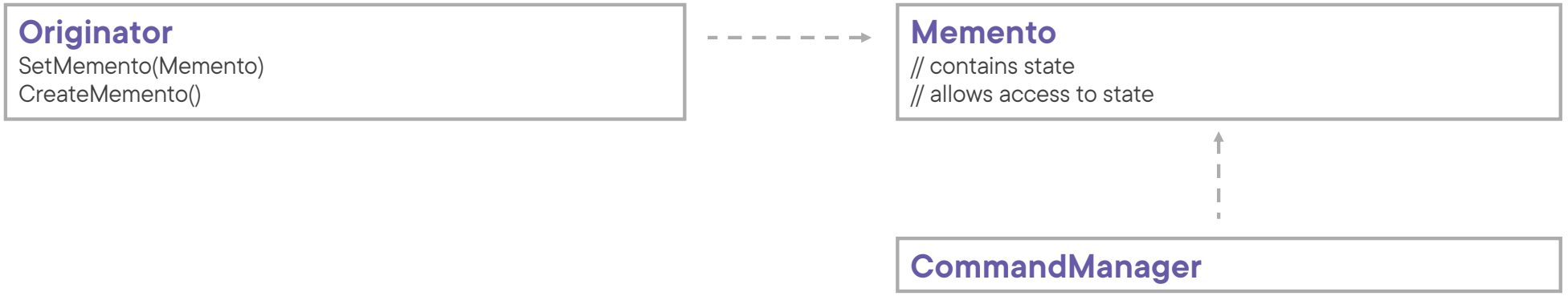
Structure of the Memento Pattern



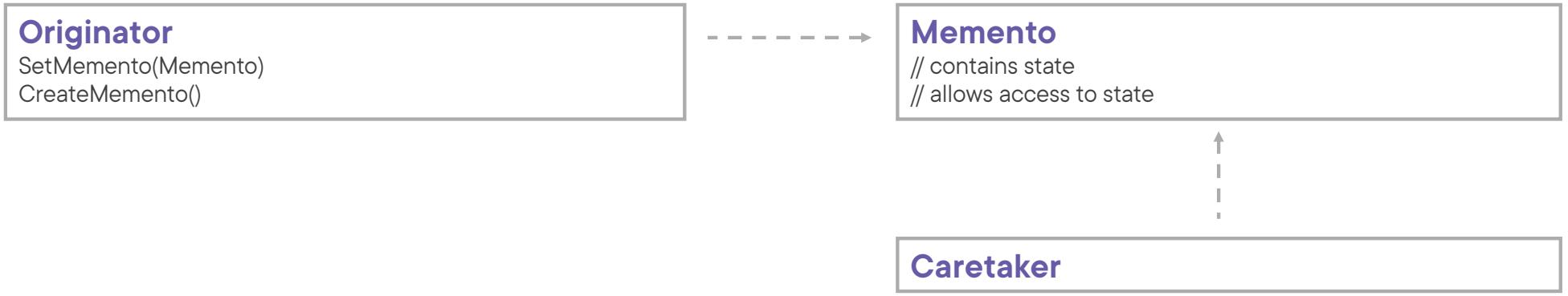
Originator creates a **Memento** with a snapshot of its internal state. It also uses the **Memento** to restore its internal state.

Caretaker keeps the **Memento** safe, and shouldn't operate on or examine its contents.

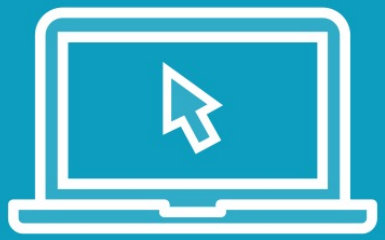
Structure of the Memento Pattern



Structure of the Memento Pattern



Demo



Implementing the memento pattern

Use Cases for the Memento Pattern

When part of an object's state must be saved so it can be restored later on

AND when a direct interface to obtaining the state would expose implementation details and break encapsulation

Pattern Consequences

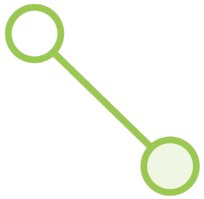
It preserves encapsulation boundaries

It simplifies the originator

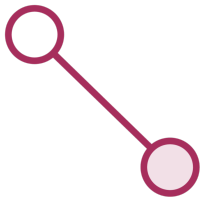
Using mementos might be expensive

It can introduce complexity to your code base

Related Patterns



Command
Can use a memento to store and restore its state



Iterator
Memento can be used to capture the current iteration state and potentially roll it back

Summary

Intent of the memento pattern:

- To capture and externalize an object's internal state so that the object can be restored to this state later, without violating encapsulation

Summary

Implementation:

- Make the distinction between a narrow and a wider interface to the memento

Up Next:
Behavioral Pattern: Mediator