Если вы видите что-то необычное, просто сообщите мне.

Хранитель (Memento)

Паттерн Memento относится к поведенческим паттернам уровня объекта.

Паттерн Memento получает и сохраняет за пределами объекта его внутреннее состояние так, чтобы позже можно было восстановить объект в таком же состоянии. Если клиенту в дальнейшем нужно "откатить" состояние исходного объекта, он передает Memento обратно в исходный объект для его восстановления.

Паттерн оперирует тремя объектами:

1. Хозяин состояния (Originator);
2. Хранитель (Memento) - Хранит в себе состояние объекта-хозяина класса Originator;
3. Смотритель (Caretaker) - Отвечает за сохранность объекта-хранителя класса Memento.

Требуется для реализации:

1. Класс Originator, у которого есть какое-то меняющиеся состояние, а так же он может создавать и принимать хранителей (Memento) своего состояния;
2. Класс Memento, реализует хранилище для состояния Originator;
3. Класс Caretaker, получает и хранит объект-хранитель (Memento), пока он не понадобится хозяину.

[!] В описании паттерна применяются общие понятия, такие как Класс, Объект, Абстрактный класс. Применимо к языку Go, это Пользовательский Тип, Значение этого Типа и Интерфейс. Также в языке Go за место общепринятого наследования используется агрегирование и встраивание.

```
//memento.go
// Package memento is an example of the Memento Pattern.
package memento
```

```
// Originator implements a state master.
type Originator struct {
	State string
}

// CreateMemento returns state storage.
func (o *Originator) CreateMemento() *Memento {
	return &Memento{state: o.State}
}

// SetMemento sets old state.
func (o *Originator) SetMemento(memento *Memento) {
	o.State = memento.GetState()
}

// Memento implements storage for the state of Originator
type Memento struct {
	state string
}

// GetState returns state.
func (m *Memento) GetState() string {
	return m.state
}

// Caretaker keeps Memento until it is needed by Originator.
type Caretaker struct {
	Memento *Memento
}
```

```
//memento_test.go
package memento

import (
	"testing"
)

func TestMomento(t *testing.T) {

	originator := new(Originator)
```

```
	caretaker := new(Caretaker)

	originator.State = "On"

	caretaker.Memento = originator.CreateMemento()

	originator.State = "Off"

	originator.SetMemento(caretaker.Memento)

	if originator.State != "On" {
		t.Errorf("Expect State to %s, but %s", originator.State, "On")
	}
}
```

Revision #1

Created 2022-07-03 10:15:20 UTC by gasick

Updated 2022-07-03 10:16:08 UTC by gasick