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

Приспособленец (Flyweight)

Паттерн Flyweight относится к структурным паттернам уровня объекта.

Паттерн Flyweight используется для эффективной поддержки большого числа мелких объектов, он позволяет повторно использовать мелкие объекты в различном контексте.

Требуется для реализации:

1. Класс FlyweightFactory, являющейся модифицированным паттерном фабрики, для создания приспособленцев;
2. Базовый абстрактный класс Flyweight, для описания общего интерфейса приспособленцев;
3. Класс ConcreteFlyweight реализующий приспособленца, который будет замещать собой одинаковые мелкие объекты.

Суть в том, что мы можем запрашивать приспособленцев у фабрики по запросу, в свою очередь она будет отдавать те объекты, которые уже были созданы, или создавать новые. Это означает, что мы будем использовать уже созданные объекты, а не создавать ещё больше, если объекты под наши нужны уже имеются. Также стоит обратить внимание, что приспособленцы имеют внутреннее и внешние состояние. Фабрика находит приспособленцев по внутреннему состоянию, а внешнее состояние передается в его методы.

[!] 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.

```
//flyweight.go
// Package flyweight is an example of the Flyweight Pattern.
package flyweight

import "fmt"

// Flyweighter interface
type Flyweighter interface {
	Draw(width, height int, opacity float64) string
}

// FlyweightFactory implements a factory.
// If a suitable flyweighter is in pool, then returns it.
type FlyweightFactory struct {
	pool map[string]Flyweighter
}

// GetFlyweight creates or returns a suitable Flyweighter by state.
func (f *FlyweightFactory) GetFlyweight(filename string) Flyweighter {
	if f.pool == nil {
		f.pool = make(map[string]Flyweighter)
	}
	if _, ok := f.pool[filename]; !ok {
		f.pool[filename] = &ConcreteFlyweight{filename: filename}
	}
	return f.pool[filename]
}

// ConcreteFlyweight implements a Flyweighter interface.
type ConcreteFlyweight struct {
	filename string // internal state
}

// Draw draws image. Args width, height and opacity is external state.
func (f *ConcreteFlyweight) Draw(width, height int, opacity float64) string {
	return fmt.Sprintf("draw image: %s, width: %d, height: %d, opacity: %.2f", f.filename, width, height, opacity)
}
```

```
//flyweight_test.go
package flyweight

import (
	"strconv"
	"testing"
)

func TestFlyweight(t *testing.T) {
	var testCases = []struct {
		filename string
		width    int
		height   int
		opacity  float64
		expect   string
	}{
		{"cat.jpg", 100, 100, 0.95, "draw image: cat.jpg, width: 100, height: 100, opacity: 0.95"},
		{"cat.jpg", 200, 200, 0.75, "draw image: cat.jpg, width: 200, height: 200, opacity: 0.75"},
		{"dog.jpg", 300, 300, 0.50, "draw image: dog.jpg, width: 300, height: 300, opacity: 0.50"},
	}

	var factory = new(FlyweightFactory)

	for i, tt := range testCases {
		t.Run("case "+strconv.Itoa(i), func(t *testing.T) {
			var flyweight = factory.GetFlyweight(tt.filename)
			var result = flyweight.Draw(tt.width, tt.height, tt.opacity)
			if result != tt.expect {
				t.Errorf("Expect result to equal %s, but %s.\n", tt.expect, result)
			}
		})
	}
}
```

Revision #1

Created 3 July 2022 10:32:33 by gasick

Updated 3 July 2022 10:33:15 by gasick