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

Мост (Bridge)

Паттерн Bridge относится к структурным паттернам уровня объекта.

Паттерн Bridge позволяет разделить объект на абстракцию и реализацию так, чтобы они могли изменяться независимо друг от друга.

Если для одной абстракции возможно несколько реализаций, то обычно используют наследование. Однако такой подход не всегда удобен, так как наследование жестко привязывает реализацию к абстракции, что затрудняет независимую модификацию и усложняет их повторное использование.

Паттерн следует применять, когда у нас имеется абстракция и несколько её реализаций. Разумеется, нет смысла отделять абстракцию от реализации, если реализация может быть только одна.

Я не нашел не одного адекватного описания паттерна "Мост". Все что мне встречалось, либо не соответствует действительности и примеры высосаны из пальца или очень размыты. Из того, что я понял и могу объяснить на пальцах - Мост это хитрая агрегация. Класс реализующий изделие, реализует интерфейс агрегируемого класса, который подсовывается на этапе создания экземпляра класса изделия.

Как я понял... у нас есть 3 машины и 3 разных двигателя. Каждый двигатель подходит к каждой машине, т.е. она реализует его интерфейс. Если делать это наследованием, мы получим 9 разных классом. Получается у каждой машины 3 модификации. Это неудобно, поэтому мы будем подсовывать двигатель на этапе создания машины. Так же каждый двигатель, может работать на разном топливе, дизель или бензин, что бы не плодить 6 разных реализаций, при создании двигателя мы будем подсовывать в него тип топлива.

Для реализации паттерна в этом примере необходимо в базовом классе автомобилей добавить поле для хранения указателя на тип реализации, значение которого класс будет получать в своём конструкторе, и вызывать по необходимости методы вложенного объекта.

Требуется для реализации:

1. Базовый абстрактный класс (в нашем случаем описывающий автомобиль);
2. Класс реализующий базовый класс. В нем есть свойство в которое мы будем подсовывать указатель на используемый двигатель (машина может работать с любым из представленных двигателей);
3. Абстракция двигателя;
4. Реализация двигателя.

В общем свойство хранящее указатель на используемый объект и есть мост. Мы в него можем подсовывать разные объекты, главное, что бы они имели одинаковый интерфейс.

[!] 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 заместо общепринятого наследования используется агрегирование и встраивание.

```
//bridge.go
// Package bridge is an example of the Bridge Pattern.
package bridge

// Carer provides car interface.
type Carer interface {
Rase() string
}

// Enginer provides engine interface.
type Enginer interface {
GetSound() string
}

// Car implementation.
type Car struct {
engine Enginer
}

// NewCar is the Car constructor.
func NewCar(engine Enginer) Carer {
```

```
	return &Car{
		engine: engine,
	}
}

// Rase implementation.
func (c *Car) Rase() string {
	return c.engine.GetSound()
}

// EngineSuzuki implements Suzuki engine.
type EngineSuzuki struct {
}

// GetSound returns sound of the engine.
func (e *EngineSuzuki) GetSound() string {
	return "SssuuuuZzzuuuuKkiiiii"
}

// EngineHonda implements Honda engine.
type EngineHonda struct {
}

// GetSound returns sound of the engine.
func (e *EngineHonda) GetSound() string {
	return "HhoooNnnnnnnnnDddaaaaaaa"
}

// EngineLada implements Lada engine.
type EngineLada struct {
}

// GetSound returns sound of the engine.
func (e *EngineLada) GetSound() string {
	return "PhhhhPhhhhPhPhPhPhPh"
}
```

```
//bridge_test.go
package bridge
```

```
import (
	"testing"
)

func TestBridge(t *testing.T) {

	expect := "SssuuuuZzzuuuuKkiiiii"

	car := NewCar(&EngineSuzuki{})

	sound := car.Rase()

	if sound != expect {
		t.Errorf("Expect sound to %s, but %s", expect, sound)
	}
}
```

Revision #1

Created 2022-07-03 10:29:08 UTC by gasick

Updated 2022-07-03 10:29:44 UTC by gasick