СБЕРБАНК ТЕХНОЛОГИИ

# Структурные шаблоны

# Шаблоны проектирования

## 2. Структурные шаблоны

| ШАБЛОН | ОПИСАНИЕ | ЧТО ПОЗВОЛЯЕТ МЕНЯТЬ |
|---|---|---|
| Адаптер | Преобразует интерфейс класса в другой интерфейс ожидаемый клиентом. | Интерфейс |
| Компоновщик | Собирает объекты в древовидные структуры для представления иерархий часть-целое при этом структура сохраняет интерфейс объекта. | Структуру и состав объекта |
| Декоратор | Динамически добавляет дополнительные возможности объекту. | Дополнительные возможности объекта |
| Фасад | Предоставляет объединённый интерфейс для набора интерфейсов в подсистеме. | Интерфейс к подсистеме |
| Заместитель | Предоставляет заменитель другого объекта для контроля доступа к нему. | Местоположение объекта и то как происходит доступ к объекту |

**Структурные шаблоны используются для композиции классов и объектов для формирования больших структур.**

# ADAPTER

Адаптер (Adapter или Wrapper): Преобразует интерфейс класса в другой интерфейс ожидаемый клиентом.

Задача: Класс поддерживает требуемые данные и поведение, но имеет неподходящий интерфейс.

Способ решения: Адаптер предусматривает создание класса-оболочки с требуемым интерфейсом.

```
public interface Engine {
    void start();
    void stop();
    void increasePower();
    void decreasePower();

    int getSize();
    boolean isTurbo();
}
```

# ADAPTER

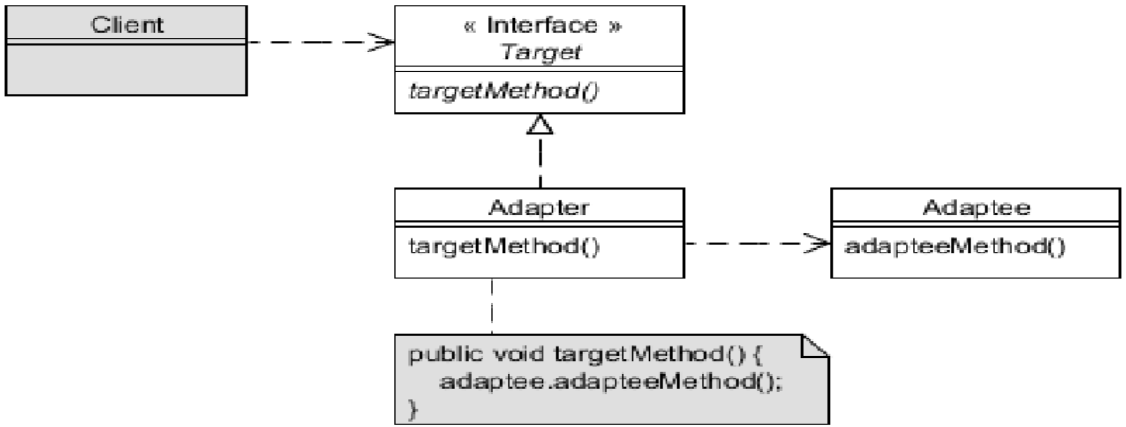
```
public interface DriverControl {
    void ignitionOn();
    void ignitionOff();
    void accelerate();
    void brake();
}
```

# ADAPTER

```
public class StandardDriverControl implements DriverControl {
    private final Engine engine = new StandardEngine();

    public void ignitionOn() {
        engine.start();
    }

    public void ignitionOff() {
        engine.stop();
    }

    public void accelerate() {
        engine.increasePower();
    }

    public void brake() {
        engine.increasePower();
    }
}
```

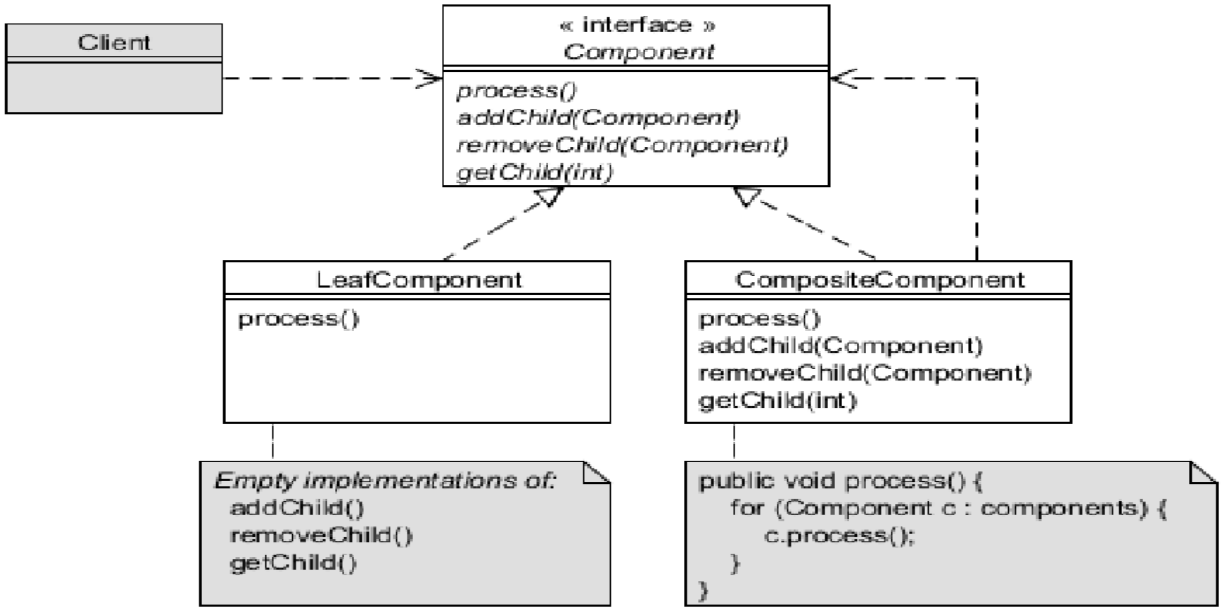
# ADAPTER

```
DriverControl driverControl = new StandardDriverControl();
driverControl.ignitionOn();
driverControl.accelerate();
driverControl.brake();
driverControl.ignitionOff();
```

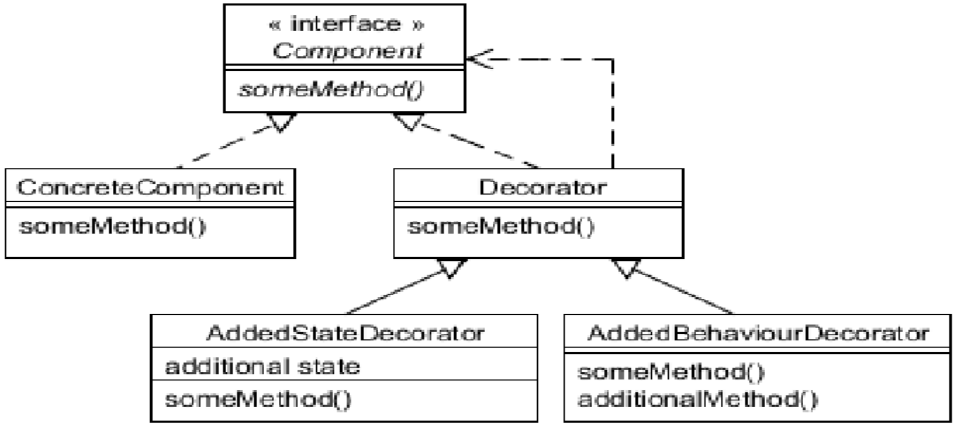

Client
« Interface »
Target
targetMethod()
Adapter
targetMethod()
Adaptee
adapteeMethod()
public void targetMethod() {
adaptee.adapteeMethod();
}

# COMPOSITE

Компоновщик: Собирает объекты в древовидные структуры для представления иерархий часть-целое. Позволяет клиентам однообразно обращаться с индивидуальными объектами и композициями.


Client
« interface »
Component
process()
addChild(Component)
removeChild(Component)
getChild(int)
LeafComponent
process()
CompositeComponent
process()
addChild(Component)
removeChild(Component)
getChild(int)
Empty implementations of:
addChild()
removeChild()
getChild()
public void process() {
for (Component c : components) {
c.process();
}
}

# Декоратор
# (Decorator)

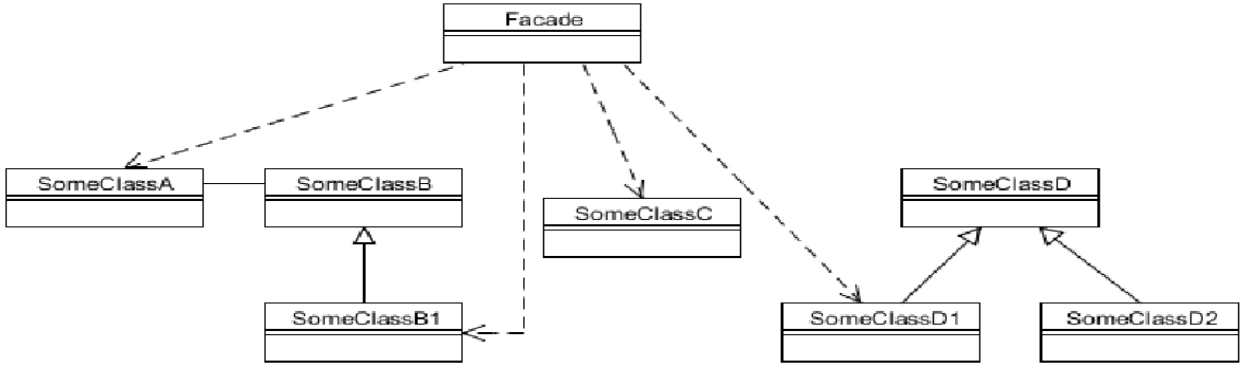



**Шаблон Декоратор:** Динамически добавляет дополнительные возможности объекту. Является гибкой альтернативой созданию подклассов для расширения функциональности.


Пакет java.io. FilterOutputStream – абстрактный декоратор. OutputStream – компонент. FileOutputStream – конкретный компонент. BufferedOutputStream – конкретный декоратор.

# Фасад

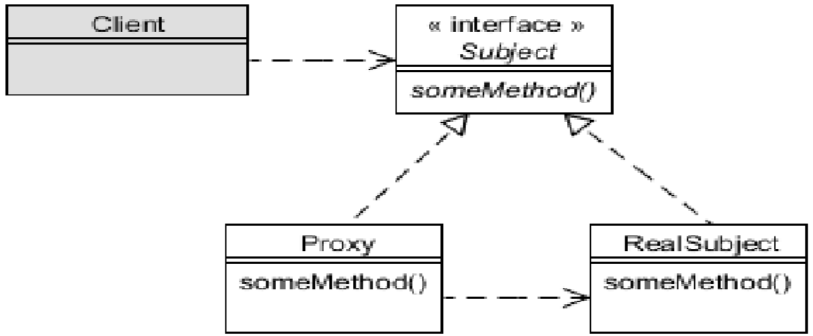
# (Facade)




| | |
|---|---|
| | Шаблон Фасад: Предоставляет объединённый интерфейс для набора интерфейсов в подсистеме. Определяет интерфейс более высокого уровня который делает использование подсистемы проще. |

# Заместитель

# (Proxy)




| | |
|---|---|
| | Шаблон Заместитель: Предоставляет заменитель другого объекта для контроля доступа к нему. |