Levelup #6

Избавляясь от одного бага, мы наживаем еще двоих

Проекты

Design Patterns

Классификация паттернов

Порождающие

Отвечают за удобное и безопасное создание новых объектов или даже целых семейств объектов.

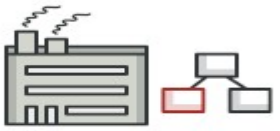
Структурные

Отвечают за построение удобных в поддержке иерархий классов.

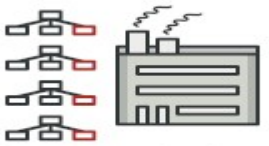
Поведенческие

Решают задачи эффективного и безопасного взаимодействия между объектами программы.

Порождающие паттерны

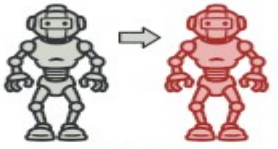


Фабричный метод
Factory Method

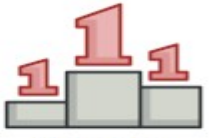


Абстрактная фабрика
Abstract Factory

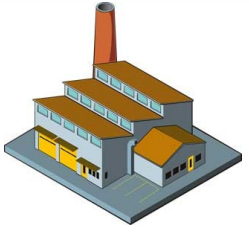
Строитель
Builder



Прототип
Prototype



Одиночка
Singleton

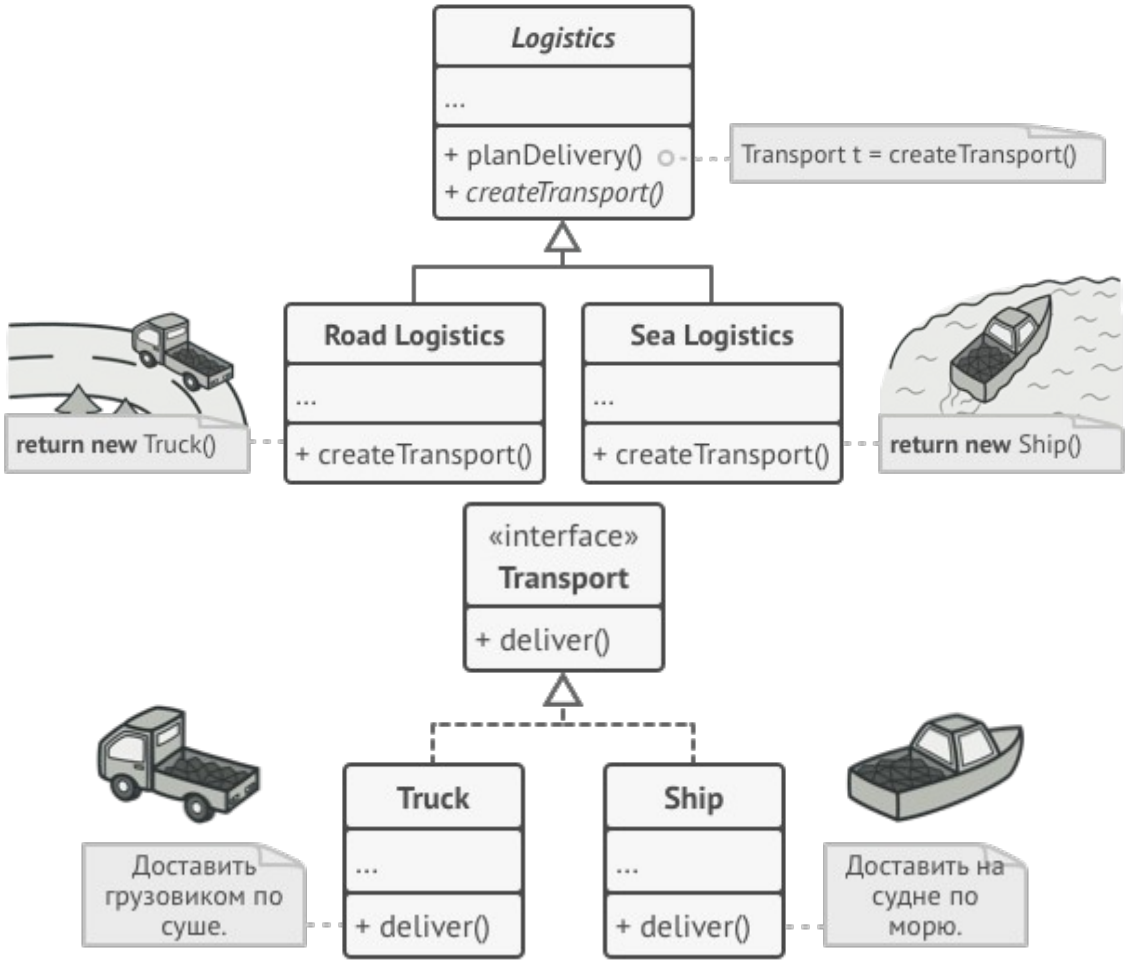


Фабрика
Factory

Фабрика

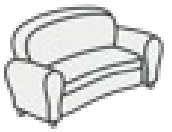
```
public class UserFactory {
    public static function create(String type) {
        switch (type) {
            case "user": return new User();
            case "customer": return new Customer();
            case "admin": return new Admin();
            default:
                throw new Exception("Wrong user type passed.");
        }
    }
}
```

Фабричный метод

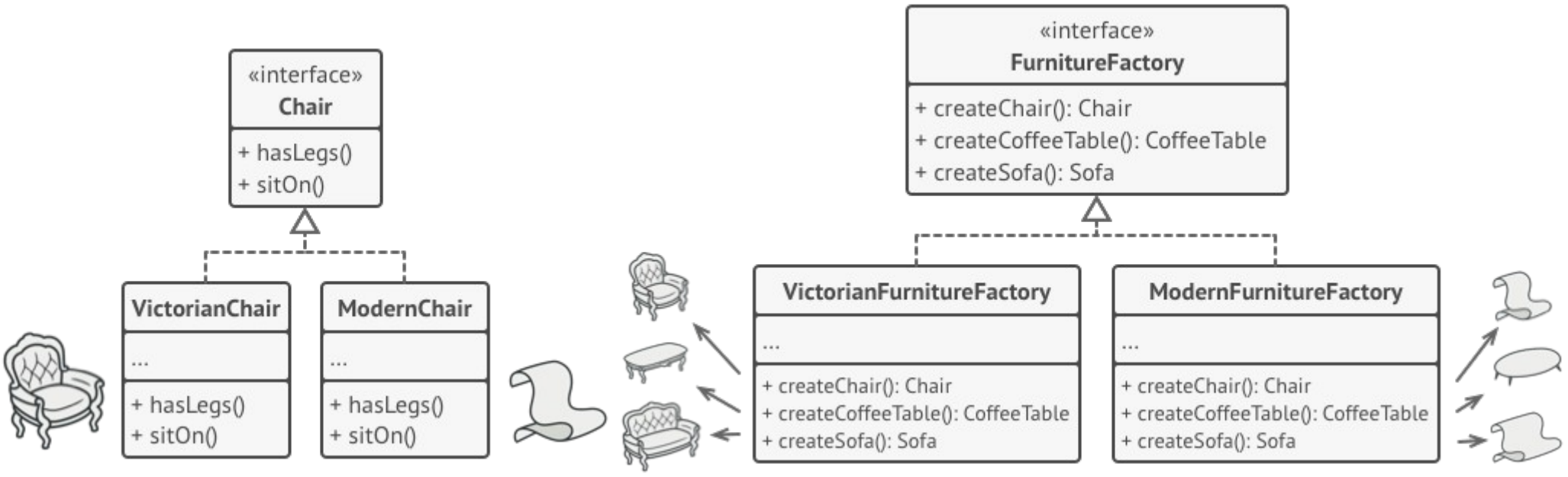


```java
interface Product { }

class ConcreteProductA implements Product { }

class ConcreteProductB implements Product { }

abstract class Creator {
    public abstract Product factoryMethod();
}

class ConcreteCreatorA extends Creator {
    @Override
    public Product factoryMethod() { return new ConcreteProductA(); }
}

class ConcreteCreatorB extends Creator {
    @Override
    public Product factoryMethod() { return new ConcreteProductB(); }
}

public class FactoryMethodExample {
    public static void main(String[] args) {
        // an array of creators
        Creator[] creators = {new ConcreteCreatorA(), new ConcreteCreatorB()};
        // iterate over creators and create products
        for (Creator creator: creators) {
            Product product = creator.factoryMethod();
            System.out.printf("Created {%s}\n", product.getClass());
        }
    }
}
```

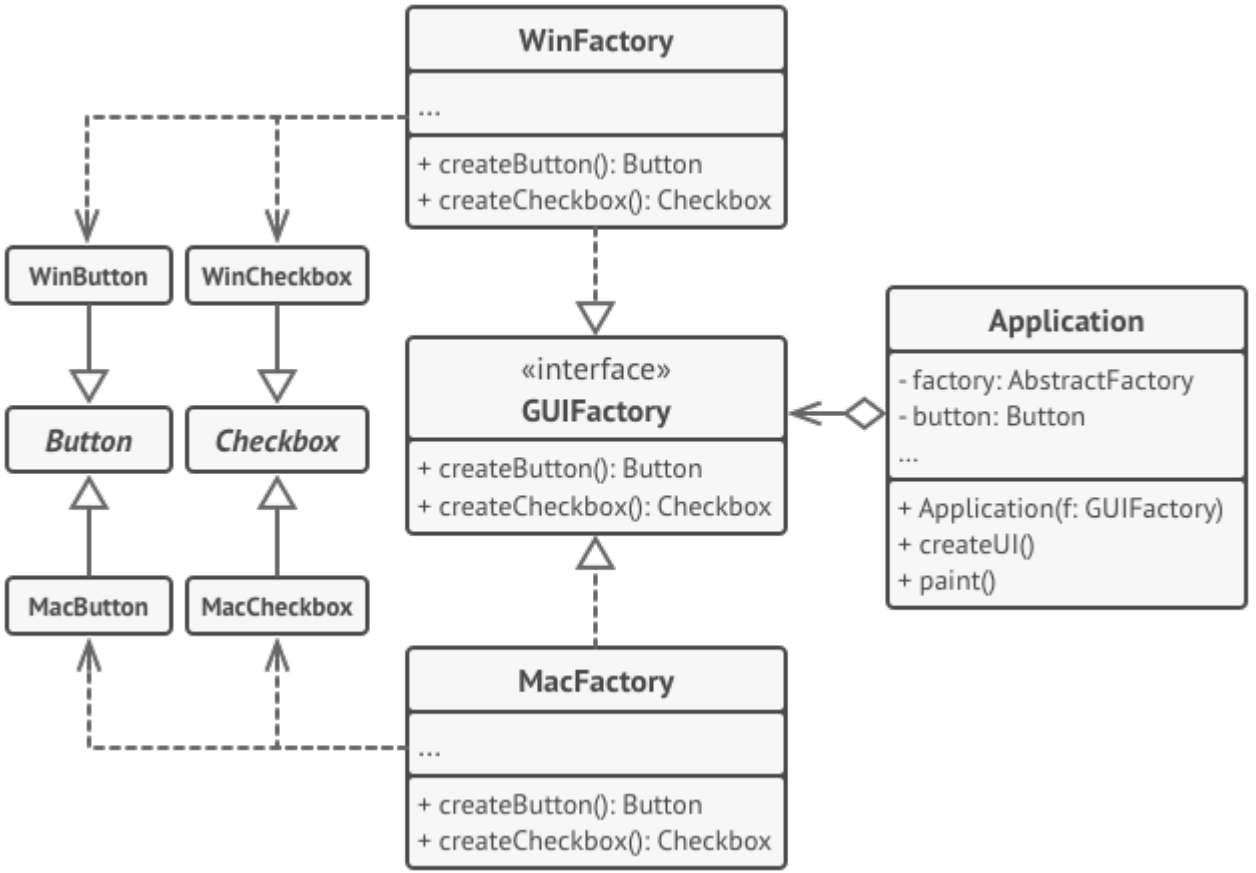
Абстрактная фабрика

	Кресло	Диван	Столик
Ар-деко			
Виктори-анский			
Модерн			

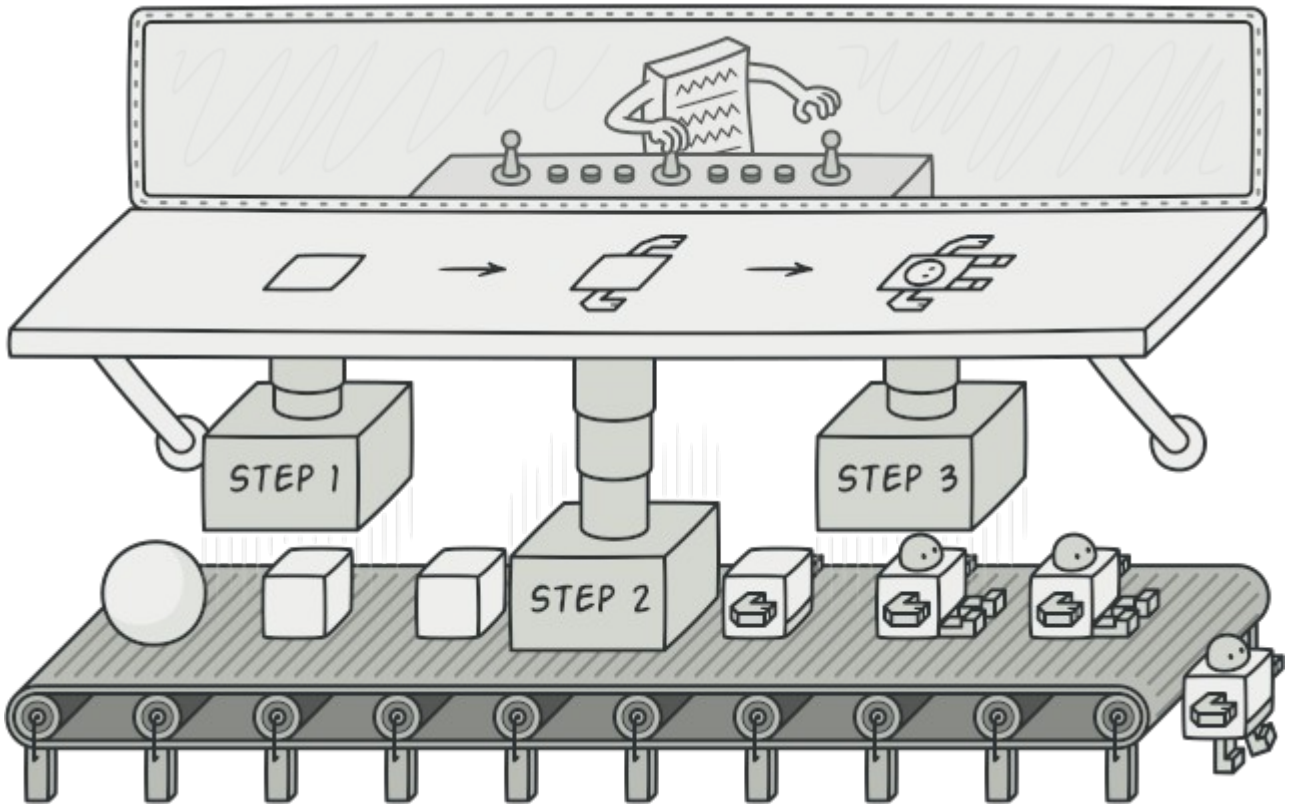
Абстрактная фабрика



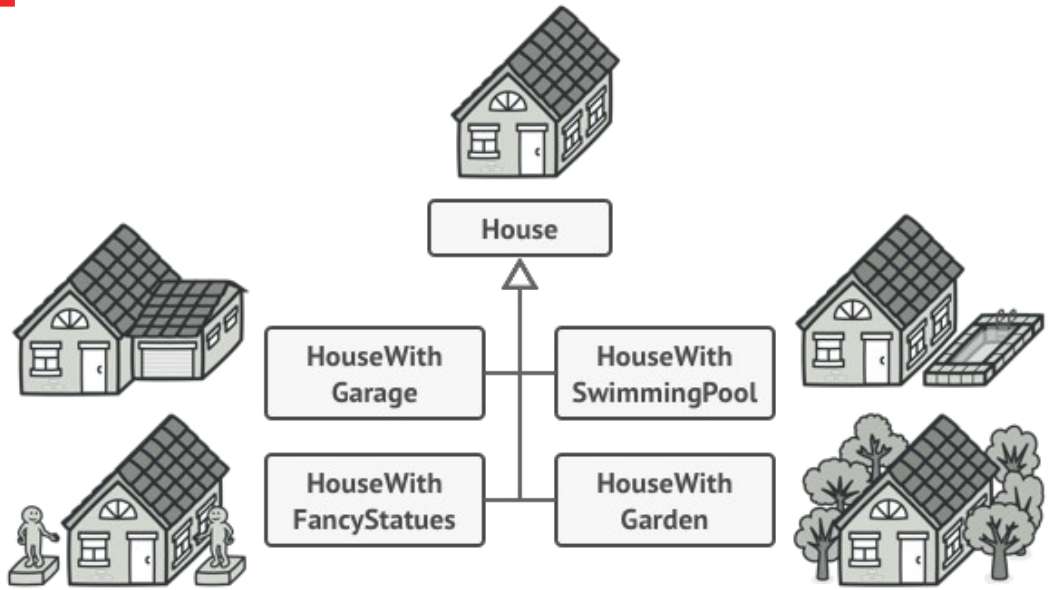
Абстрактная фабрика

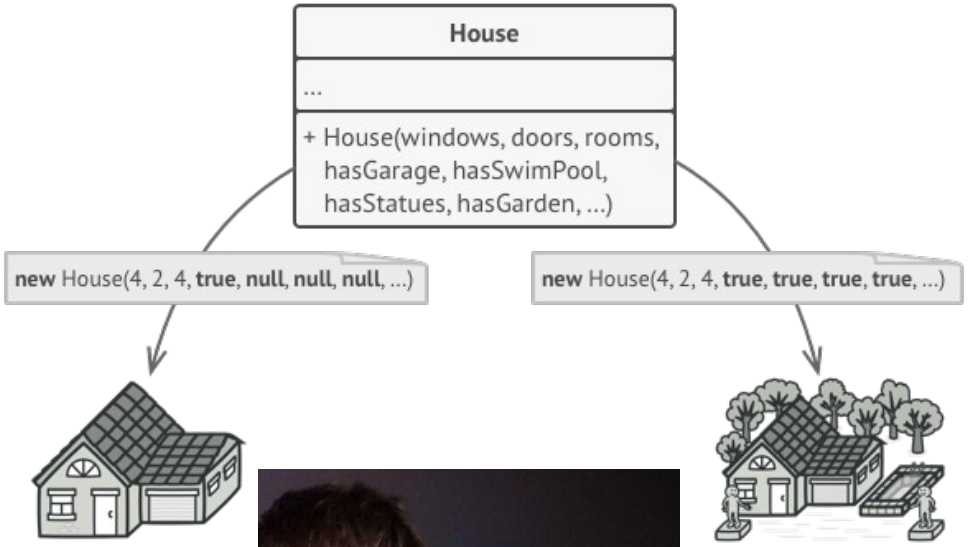


Строитель



Строитель


House
HouseWith Garage
HouseWith SwimmingPool
HouseWith FancyStatues
HouseWith Garden


House
...
+ House(windows, doors, rooms, hasGarage, hasSwimPool, hasStatues, hasGarden, ...)
new House(4, 2, 4, true, null, null, null, ...)
new House(4, 2, 4, true, true, true, true, ...)

Строитель

Строитель

```
public class MemeBuilder() {
   private int orLevel;
   private String author;
   private Image image;

   public MemeBuilder setOrLevel(int orLevel) {
      this.orLevel = orLevel;
      return this;
   }
   public MemeBuilder setAuthor(String author) {
      this.author = author;
      return this;
   }
   public MemeBuilder setImage(Image image) {
      this.image = image;
      return this;
   }
   public Meme build() {
      return new Meme(orLevel, author, image);
   }
}
```

```
Meme hikaloMeme = new MemeBuilder()
     .setAuthor("Гикало 9 и 3/4")
     .setImage(memeImage)
     .setOrLevel(10)
     .build();
```

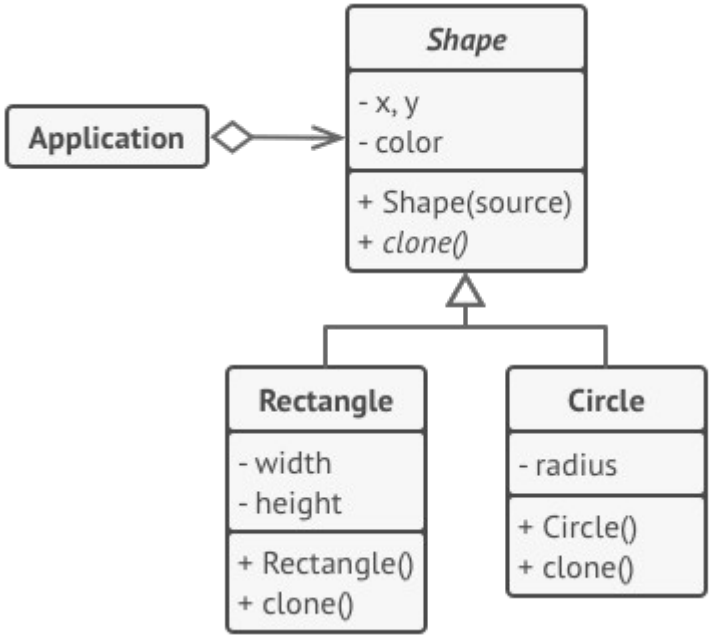
Элегантный строитель

```
public class Meme() {
   private int orLevel;
   private String author;
   private Image image;

   /* ПРИВАТНЫЙ КОНСТРУКТОР И ГЕТТЕРЫ ПРОПУЩЕНЫ */

   public static Builder newBuilder() {
      return new Meme().new Builder();
   }

   public class Builder {
      private Builder() {
      }

      public MemeBuilder setOrLevel(int orLevel) {
         Meme.this.orLevel = orLevel;
         return this;
      }
      public MemeBuilder setAuthor(String author) {
         Meme.this.author = author;
         return this;
      }
      public MemeBuilder setImage(Image image) {
         Meme.this.image = image;
         return this;
      }
      public Meme build() {
         return Meme.this;
      }
   }
}
```

```
Meme hikaloMeme = Meme.newBuilder()
     .setAuthor("Гикало 9 и 3/4")
     .setImage(memeImage)
     .setOrLevel(10)
     .build();
```

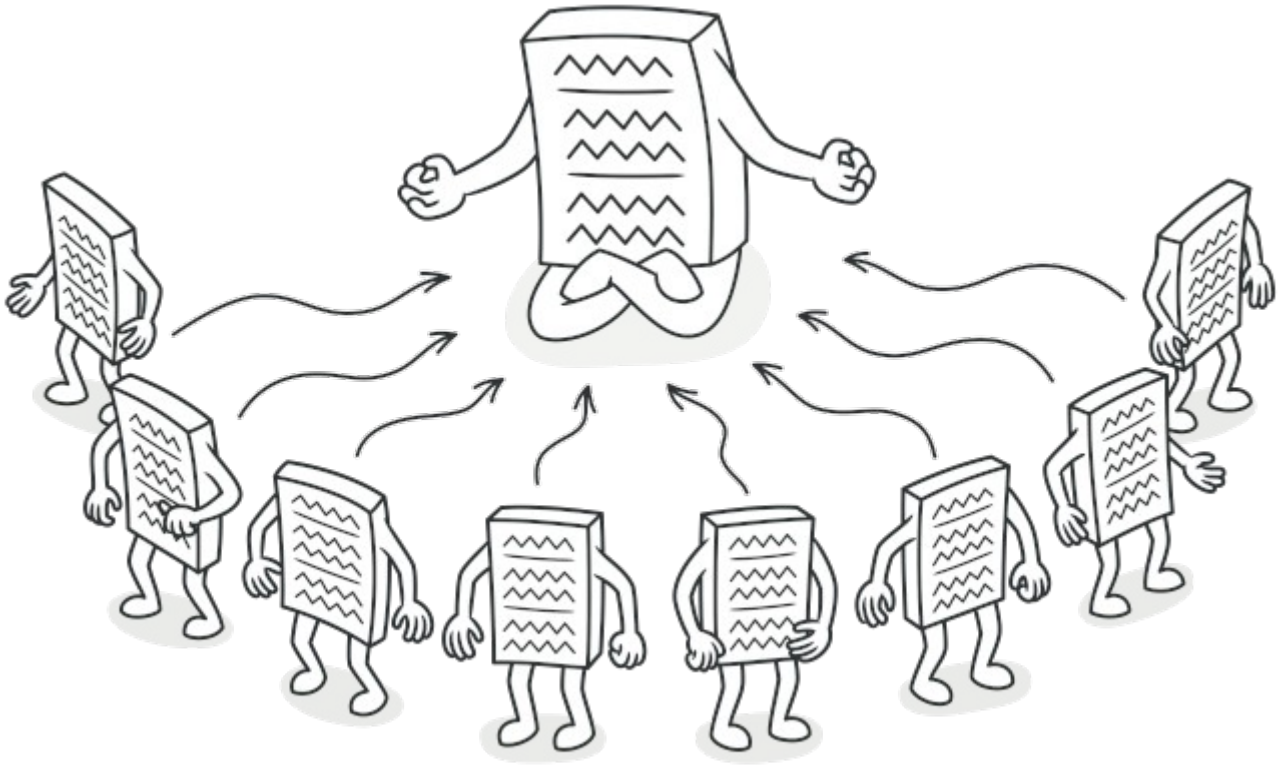
Прототип

Прототип


Application
Shape
- x, y
- color
+ Shape(source)
+ clone()
Rectangle
- width
- height
+ Rectangle()
+ clone()
Circle
- radius
+ Circle()
+ clone()

Одиночка (Singleton)

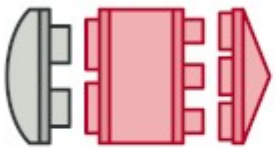
Одиночка



Одиночка

```
public class Singleton {
        private static volatile Singleton instance;

        public static Singleton getInstance() {
                Singleton localInstance = instance;
                if (localInstance == null) {
                        synchronized (Singleton.class) {
                                localInstance = instance;
                                if (localInstance == null) {
                                        instance = localInstance = new Singleton();
                                }
                        }
                }
                return localInstance;
        }
}
```

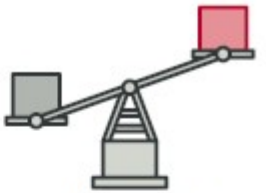
Структурные паттерны



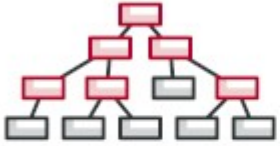
Адаптер
Adapter

Мост
Bridge

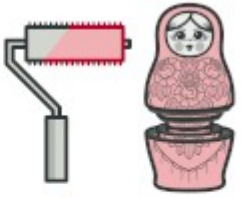
Фасад
Facade



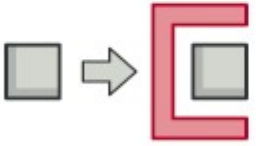
Легковес
Flyweight



Компоновщик
Composite



Декоратор
Decorator

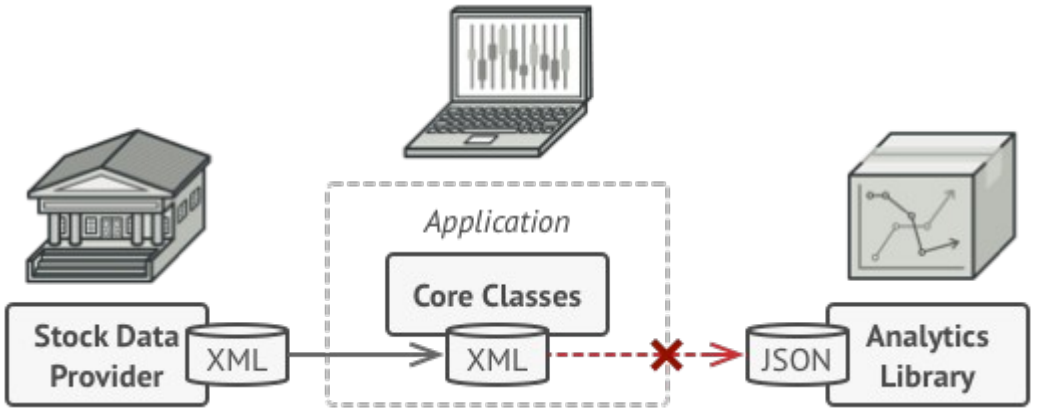


Заместитель
Proxy

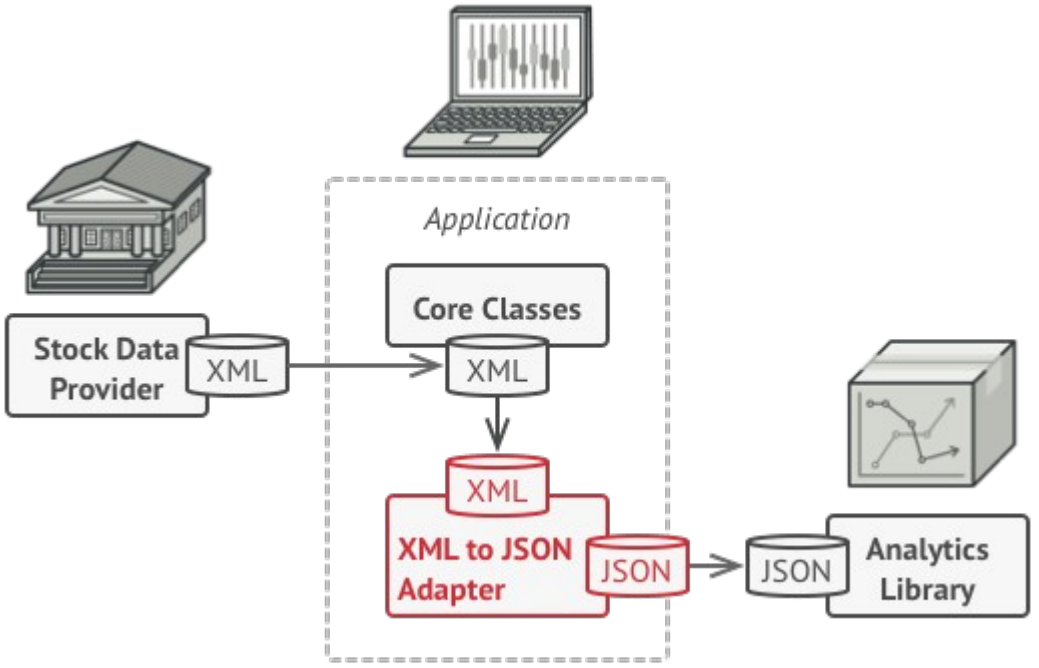

Что мы говорим богу
говнокода?
Не сегодня
(но это не точно)

Адаптер

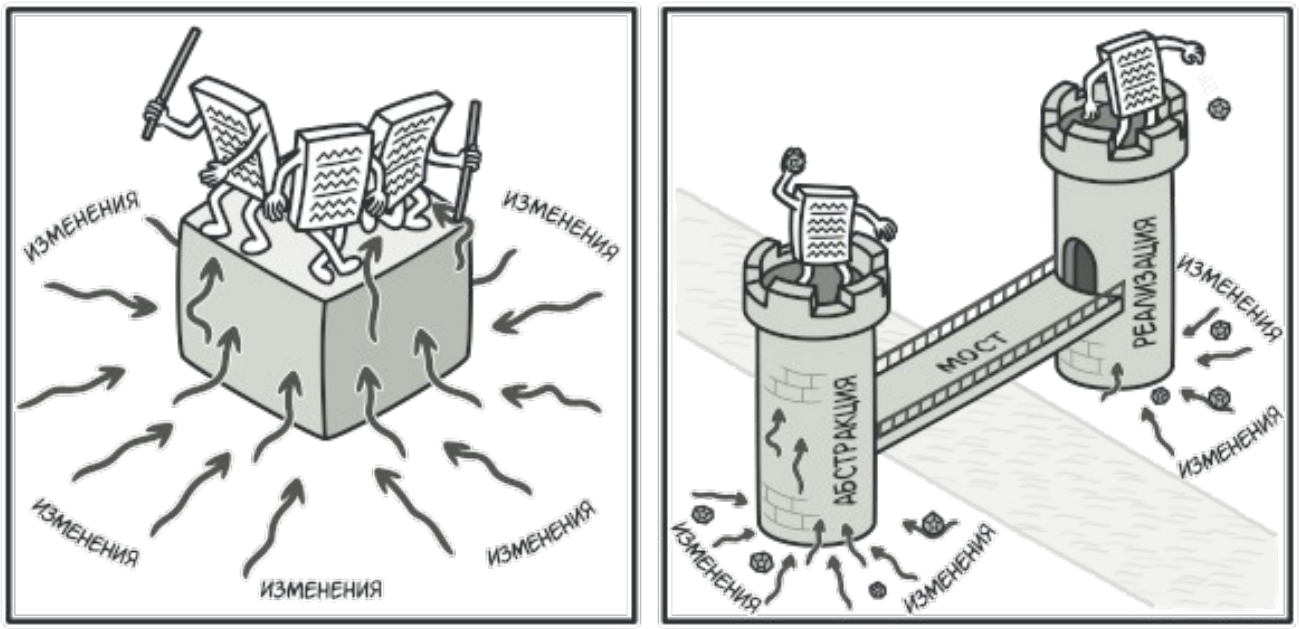
Адаптер


Application
Core Classes
Stock Data Provider
XML
XML
JSON
Analytics Library

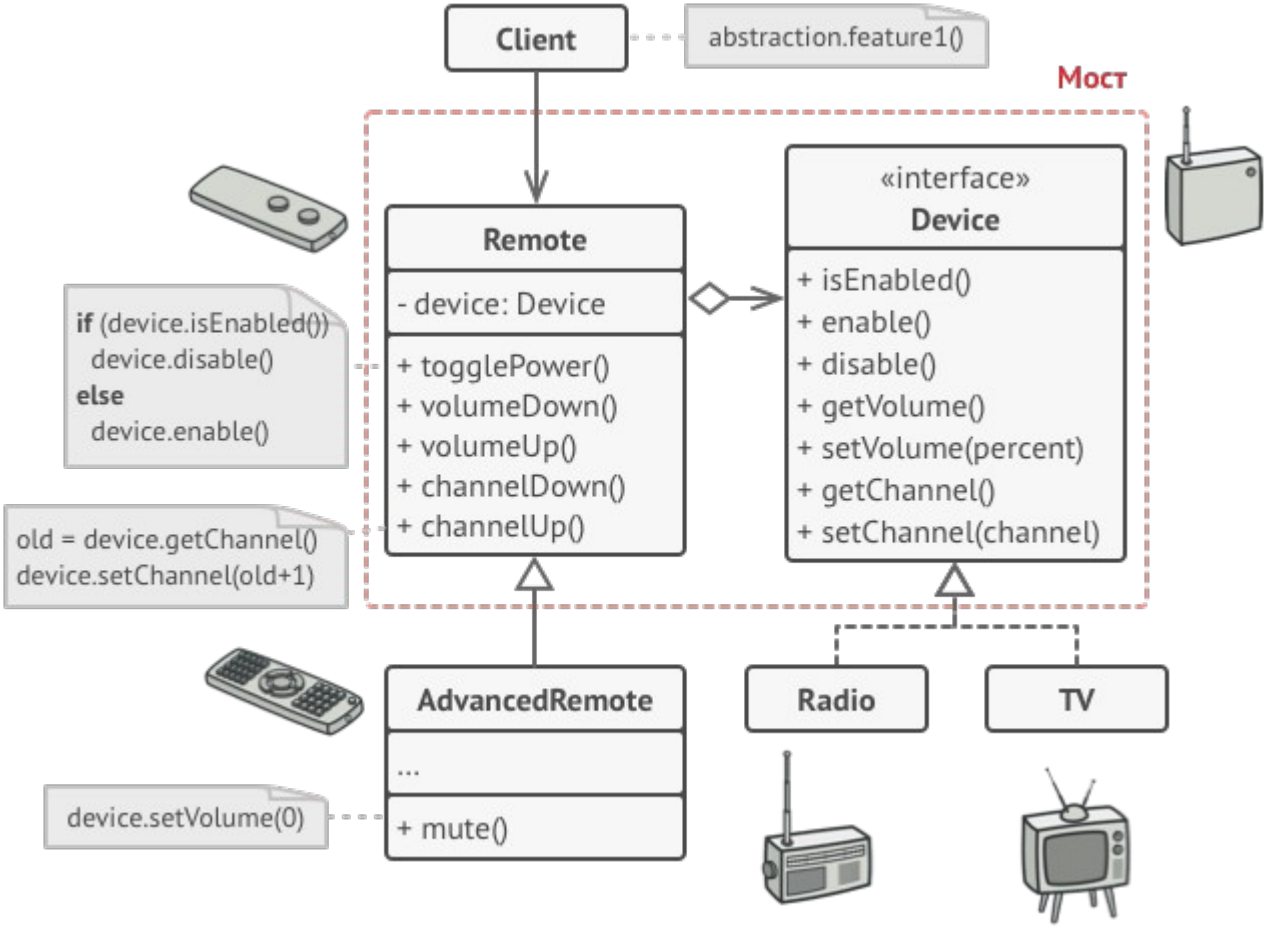
Адаптер


Application
Core Classes
Stock Data Provider
XML
XML
XML
XML to JSON Adapter
JSON
JSON
Analytics Library

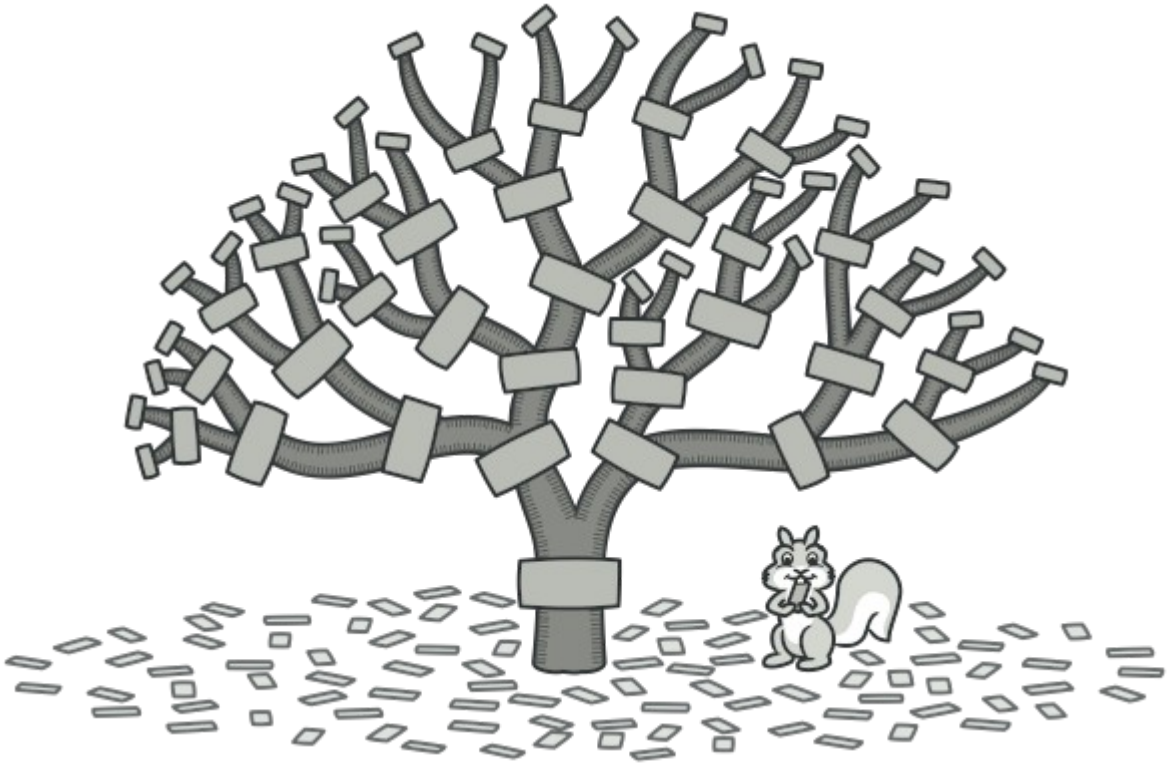
Мост


ИЗМЕНЕНИЯ
ИЗМЕНЕНИЯ
ИЗМЕНЕНИЯ
ИЗМЕНЕНИЯ
ИЗМЕНЕНИЯ
АБСТРАКЦИЯ
МОСТ
РЕАЛИЗАЦИЯ
ИЗМЕНЕНИЯ
ИЗМЕНЕНИЯ
ИЗМЕНЕНИЯ
ИЗМЕНЕНИЯ

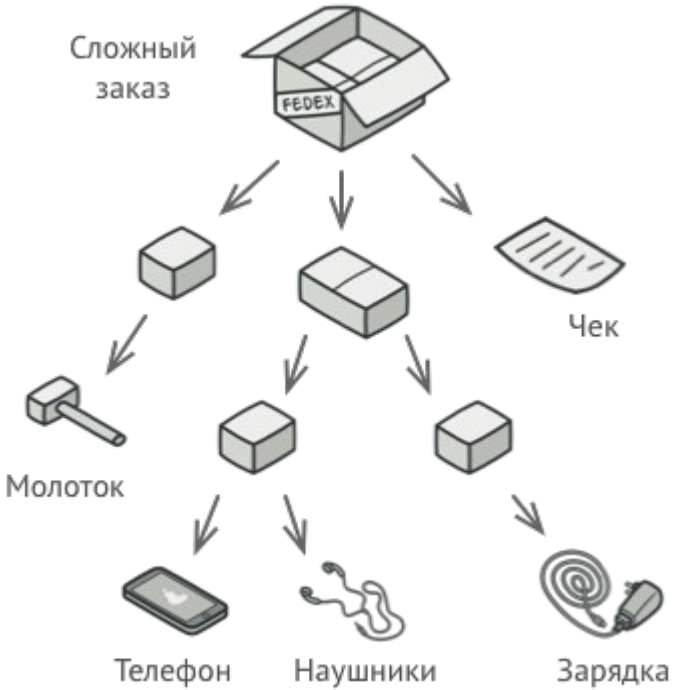
Мост


Client
abstraction.feature1()
Мост
«interface»
Device
+ isEnabled()
+ enable()
+ disable()
+ getVolume()
+ setVolume(percent)
+ getChannel()
+ setChannel(channel)
Remote
- device: Device
+ togglePower()
+ volumeDown()
+ volumeUp()
+ channelDown()
+ channelUp()
if (device.isEnabled())
device.disable()
else
device.enable()
old = device.getChannel()
device.setChannel(old+1)
AdvancedRemote
...
+ mute()
device.setVolume(0)
Radio
TV

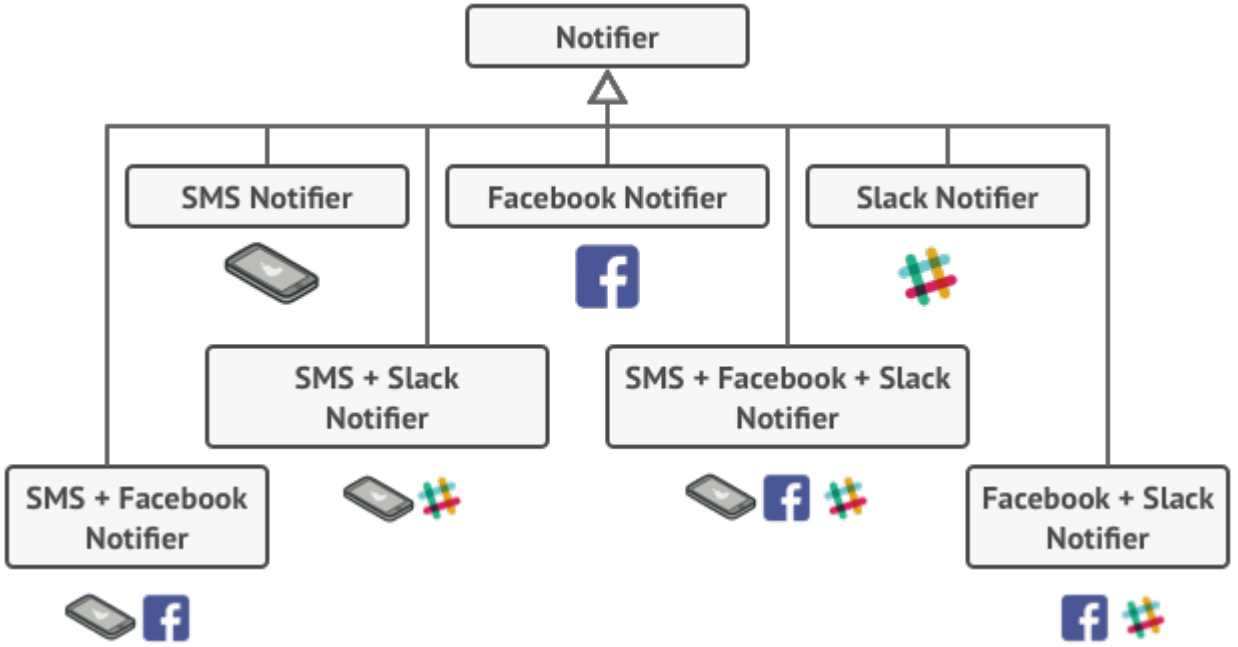
Компоновщик



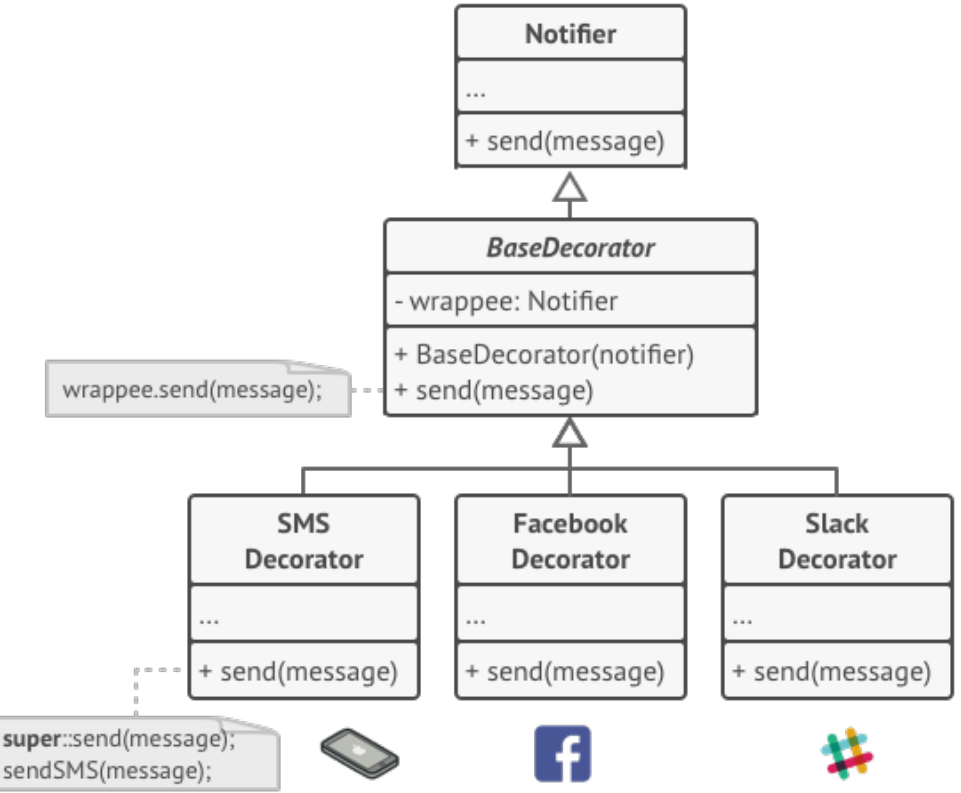
Компоновщик



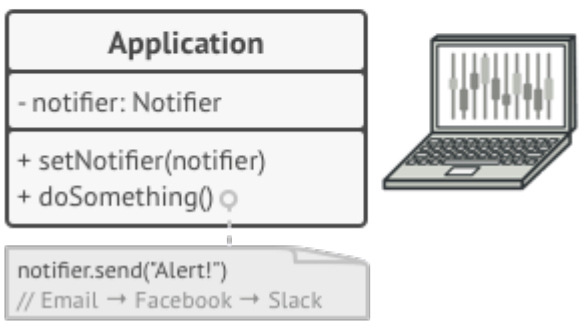
Декоратор


Notifier
SMS Notifier
Facebook Notifier
Slack Notifier
SMS + Slack Notifier
SMS + Facebook + Slack Notifier
SMS + Facebook Notifier
Facebook + Slack Notifier

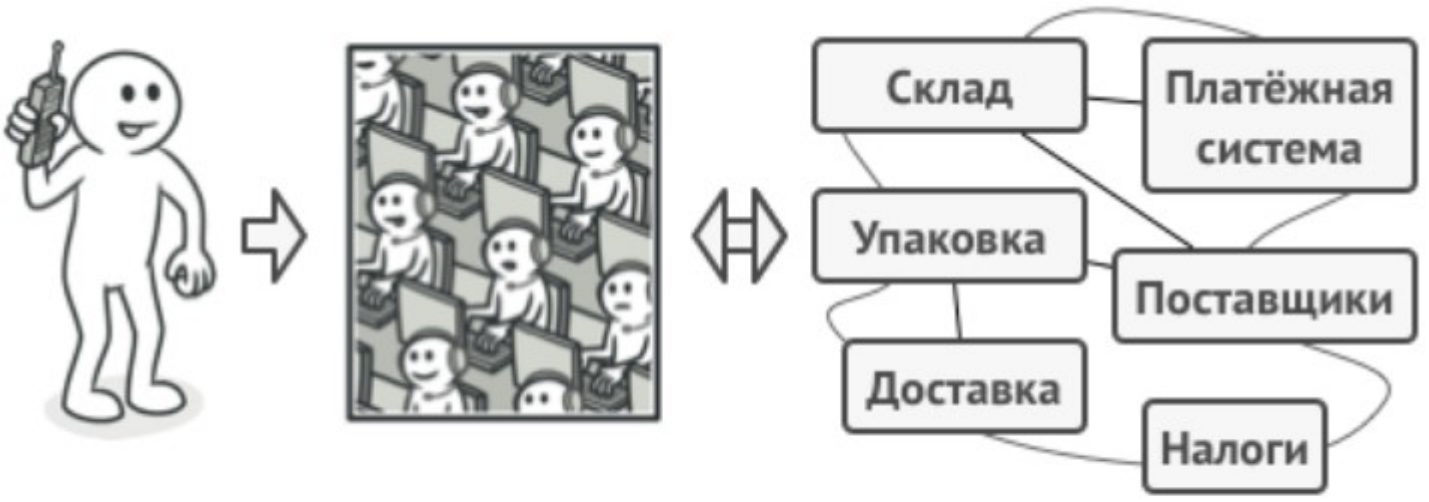
Декоратор




```
stack = new Notifier()
if (facebookEnabled)
    stack = new FacebookDecorator(stack)
if (slackEnabled)
    stack = new SlackDecorator(stack)

app.setNotifier(stack)
```



Фасад

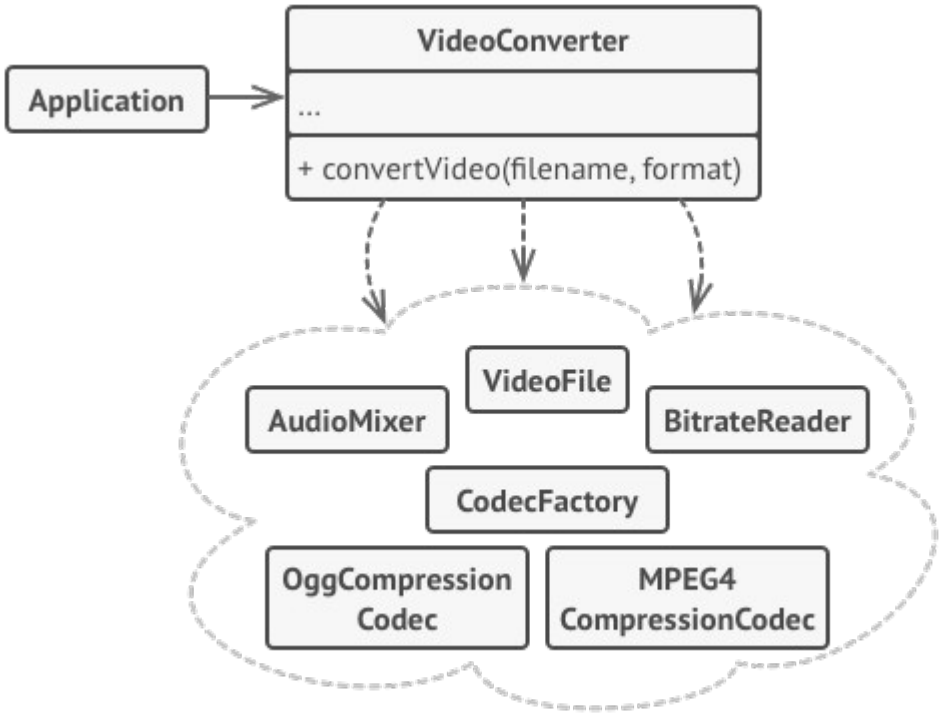


Пример телефонного заказа.

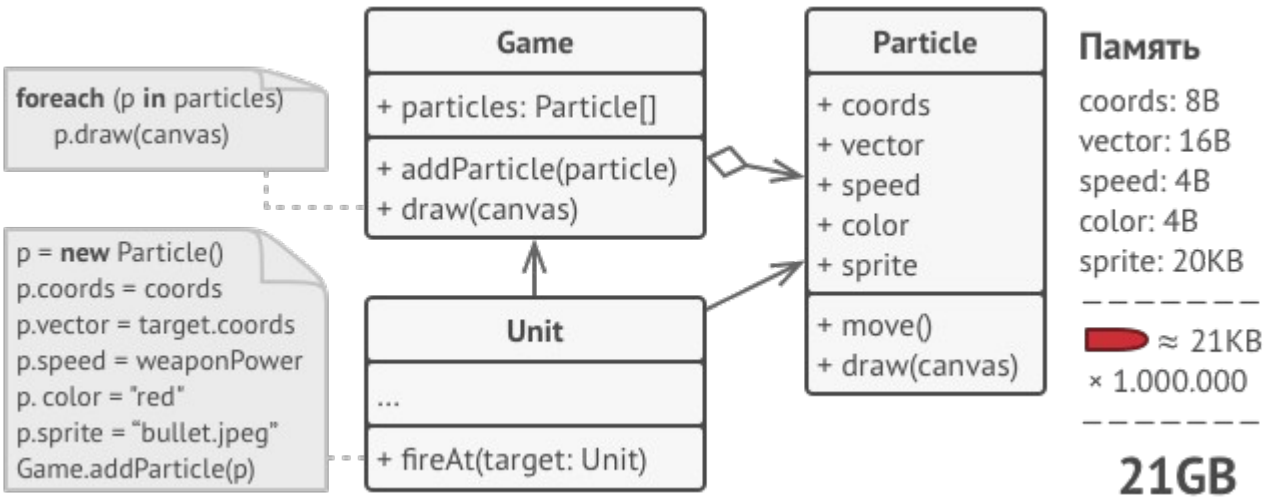
Фасад


armyDestroyer.destroy()
Fly
Arrive
OpenMouse
...
meme-arsenal.ru

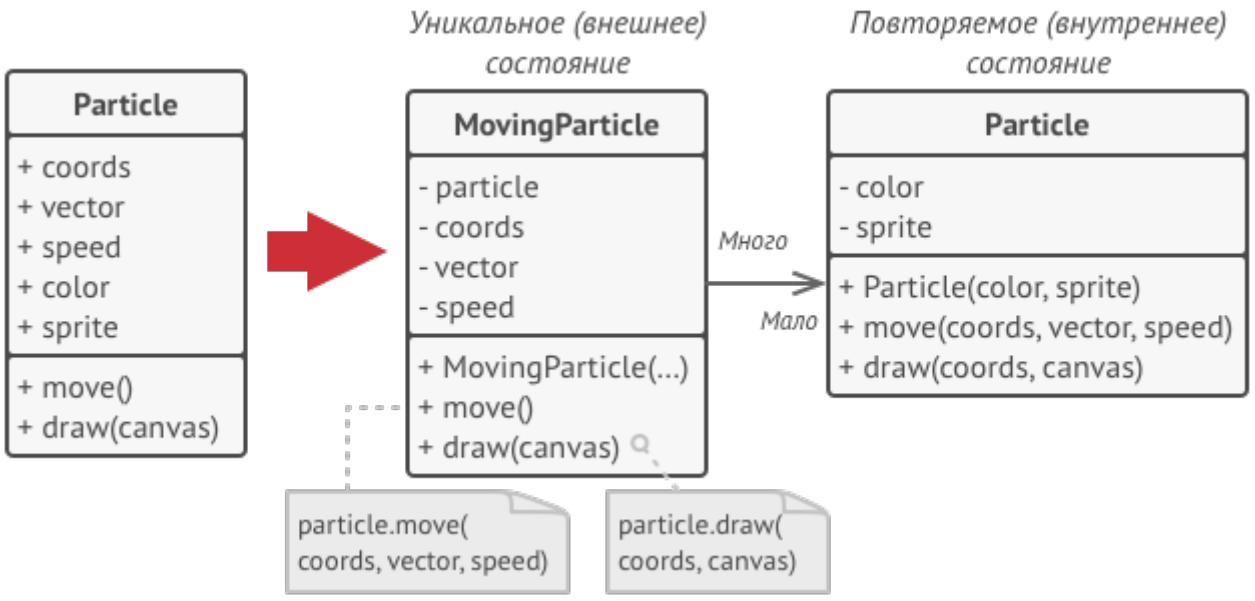
Фасад


Application
VideoConverter
...
+ convertVideo(filename, format)
VideoFile
AudioMixer
BitrateReader
CodecFactory
OggCompression Codec
MPEG4 CompressionCodec

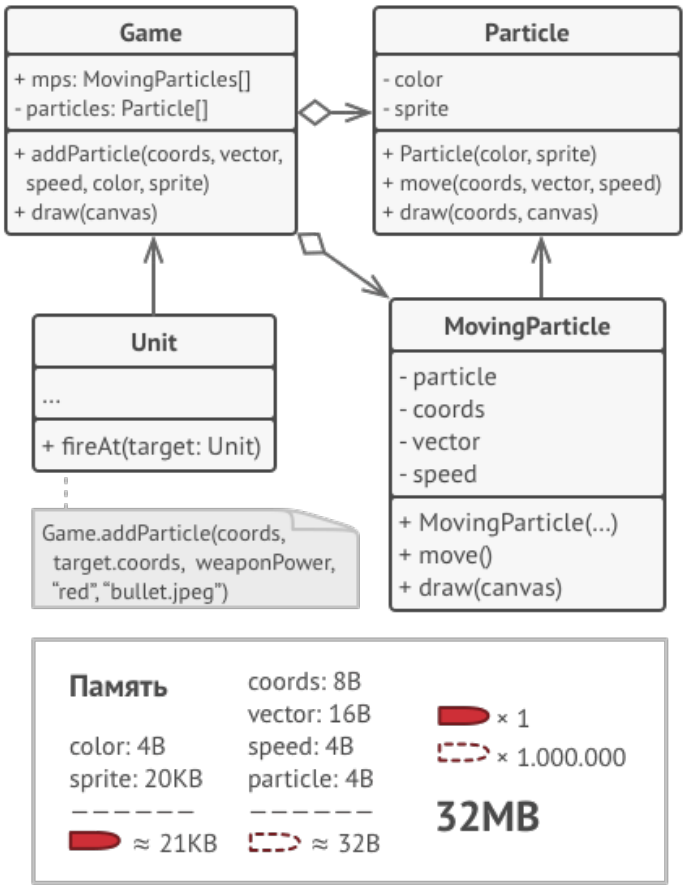
Легковес


foreach (p in particles)
p.draw(canvas)
p = new Particle()
p.coords = coords
p.vector = target.coords
p.speed = weaponPower
p. color = "red"
p.sprite = "bullet.jpeg"
Game.addParticle(p)
Game
+ particles: Particle[]
+ addParticle(particle)
+ draw(canvas)
Unit
...
+ fireAt(target: Unit)
Particle
+ coords
+ vector
+ speed
+ color
+ sprite
+ move()
+ draw(canvas)
Память
coords: 8B
vector: 16B
speed: 4B
color: 4B
sprite: 20KB
≈ 21KB
× 1.000.000
21GB

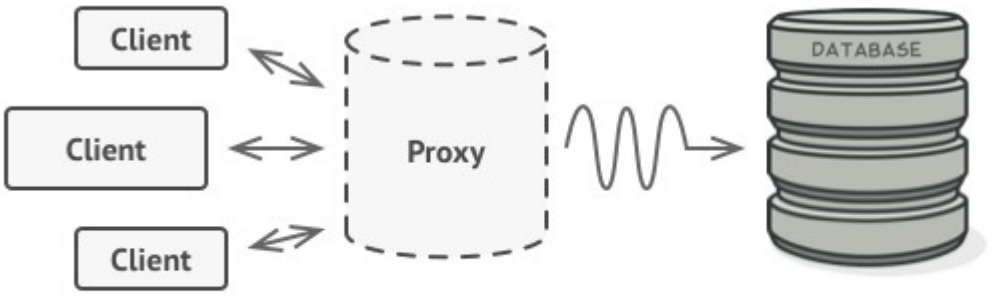
Легковес


Particle
+ coords
+ vector
+ speed
+ color
+ sprite
+ move()
+ draw(canvas)
Уникальное (внешнее) состояние
MovingParticle
- particle
- coords
- vector
- speed
+ MovingParticle(...)
+ move()
+ draw(canvas)
Много
Мало
Повторяемое (внутреннее) состояние
Particle
- color
- sprite
+ Particle(color, sprite)
+ move(coords, vector, speed)
+ draw(coords, canvas)
particle.move(coords, vector, speed)
particle.draw(coords, canvas)

Легковес

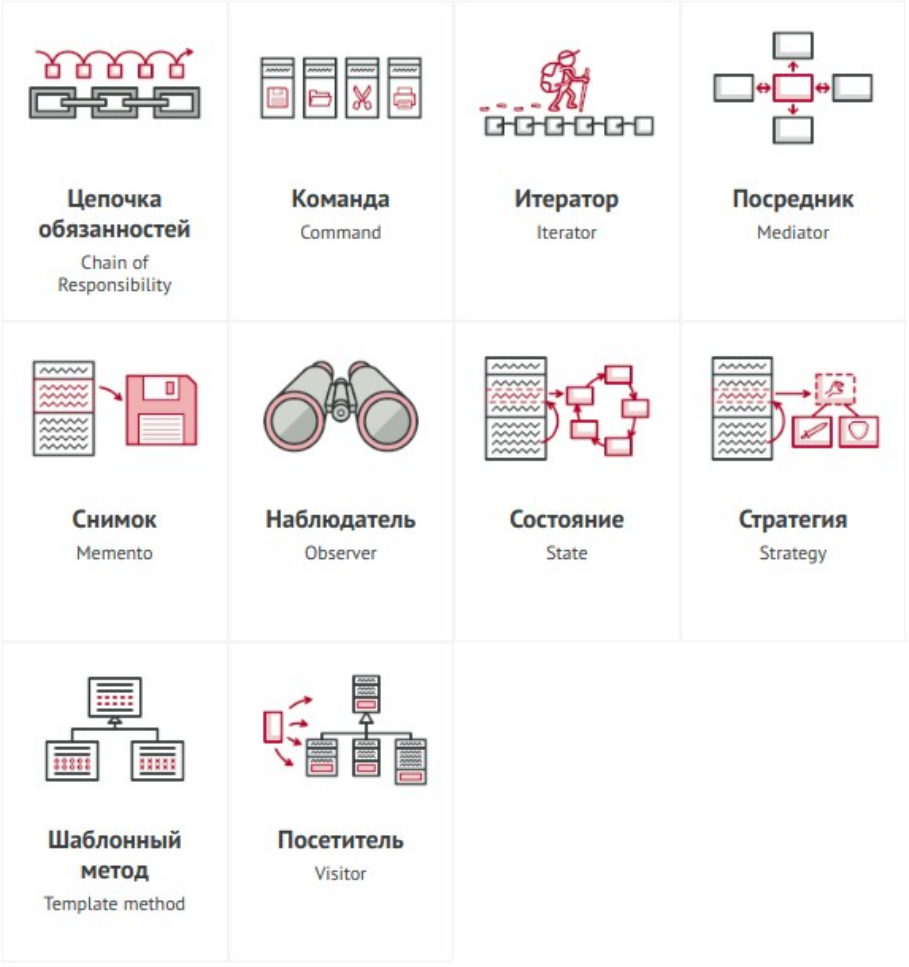

Game
+ mps: MovingParticles[]
- particles: Particle[]
+ addParticle(coords, vector, speed, color, sprite)
+ draw(canvas)
Particle
- color
- sprite
+ Particle(color, sprite)
+ move(coords, vector, speed)
+ draw(coords, canvas)
Unit
...
+ fireAt(target: Unit)
Game.addParticle(coords, target.coords, weaponPower, "red", "bullet.jpeg")
MovingParticle
- particle
- coords
- vector
- speed
+ MovingParticle(...)
+ move()
+ draw(canvas)
Память
color: 4B
sprite: 20KB
≈ 21KB
coords: 8B
vector: 16B
speed: 4B
particle: 4B
≈ 32B
× 1
× 1.000.000
32MB

Заместитель



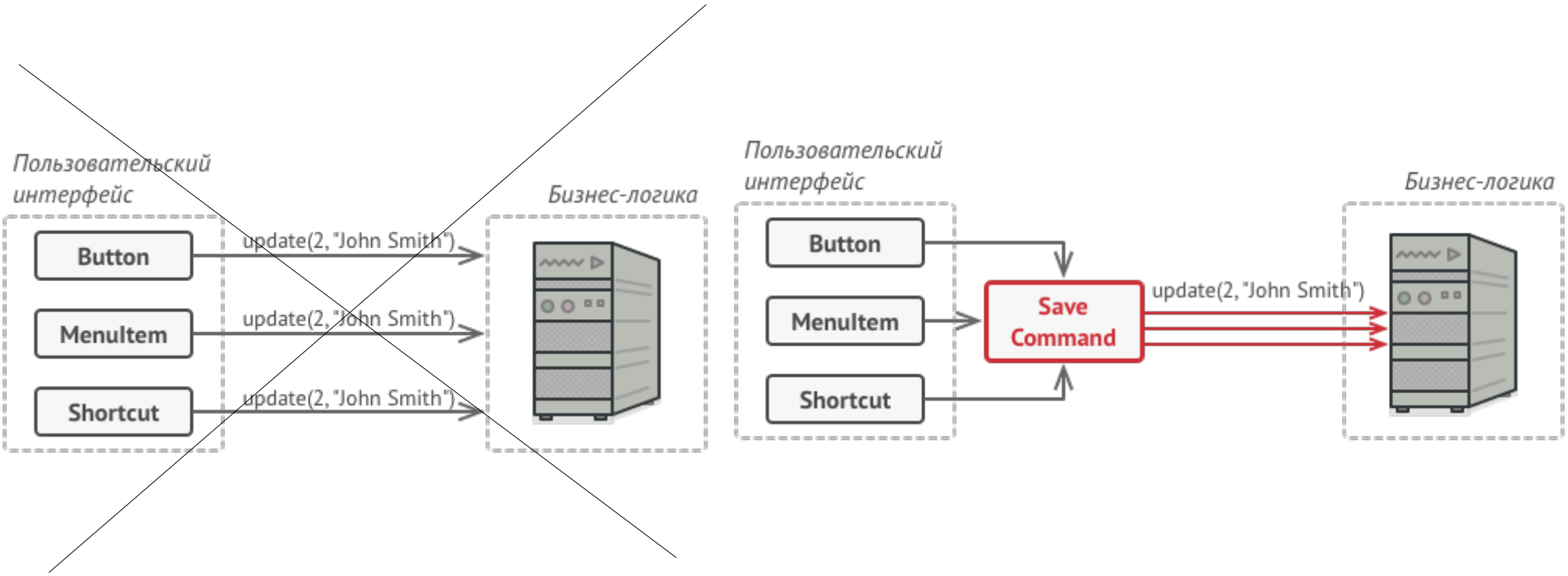
Заместитель «притворяется» базой данных, ускоряя работу за счёт ленивой инициализации и кеширования повторяющихся запросов.

Поведенческие паттерны

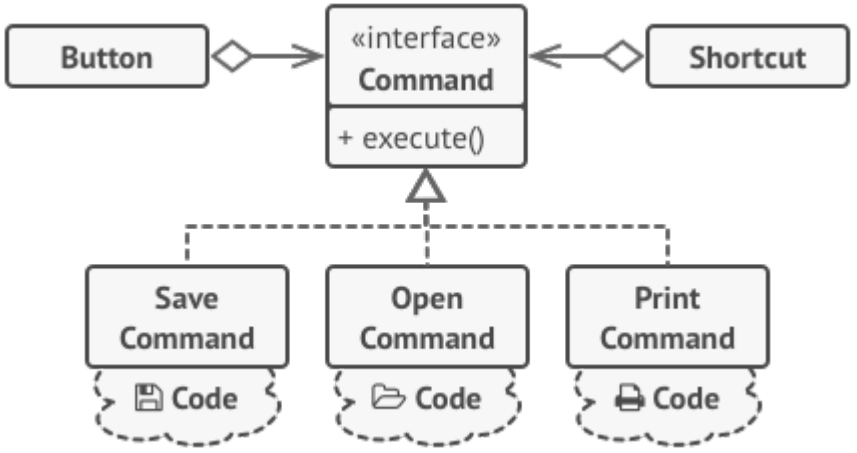


Цепочка обязаностей

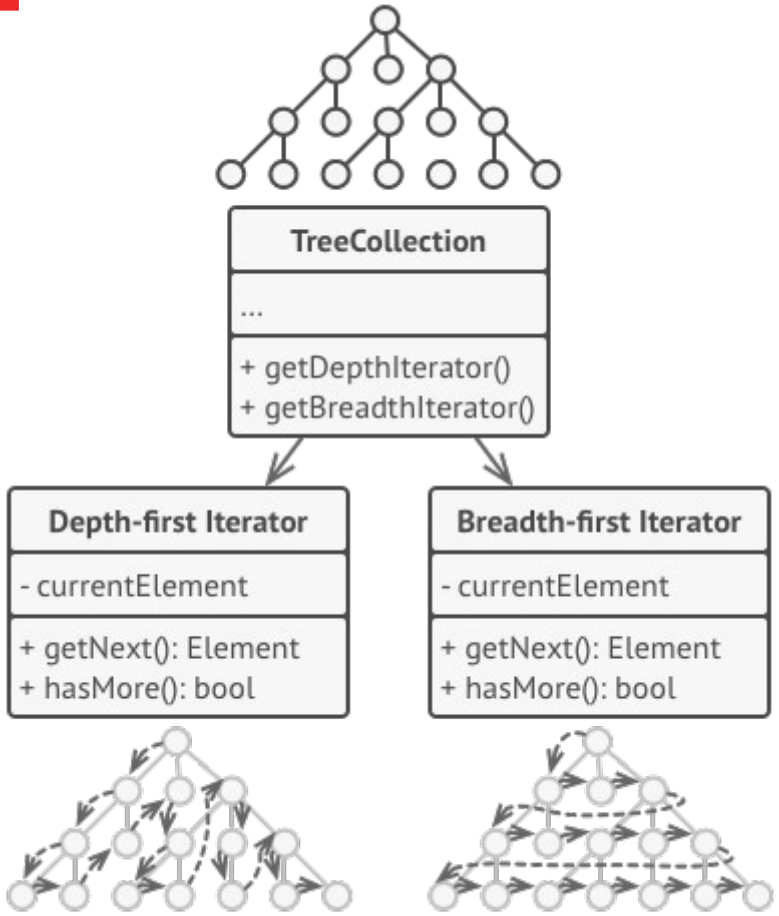
Команда

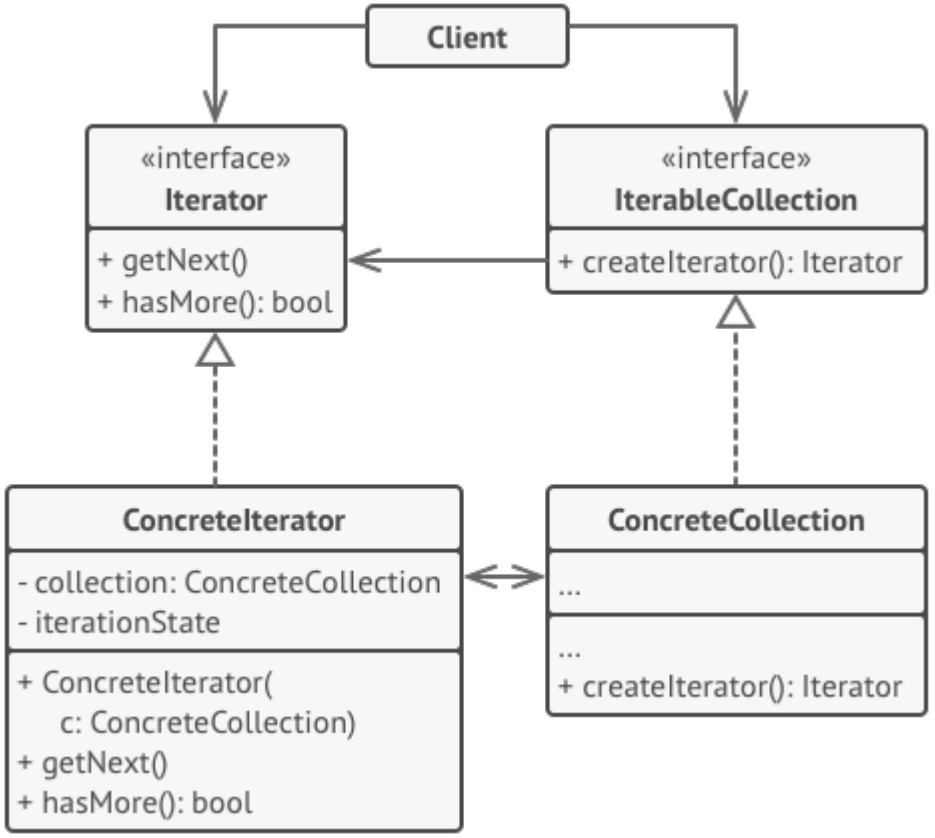

Пользовательский интерфейс
Button
MenuItem
Shortcut
update(2, "John Smith")
update(2, "John Smith")
update(2, "John Smith")
Бизнес-логика
Пользовательский интерфейс
Button
MenuItem
Shortcut
Save Command
update(2, "John Smith")
Бизнес-логика

Команда

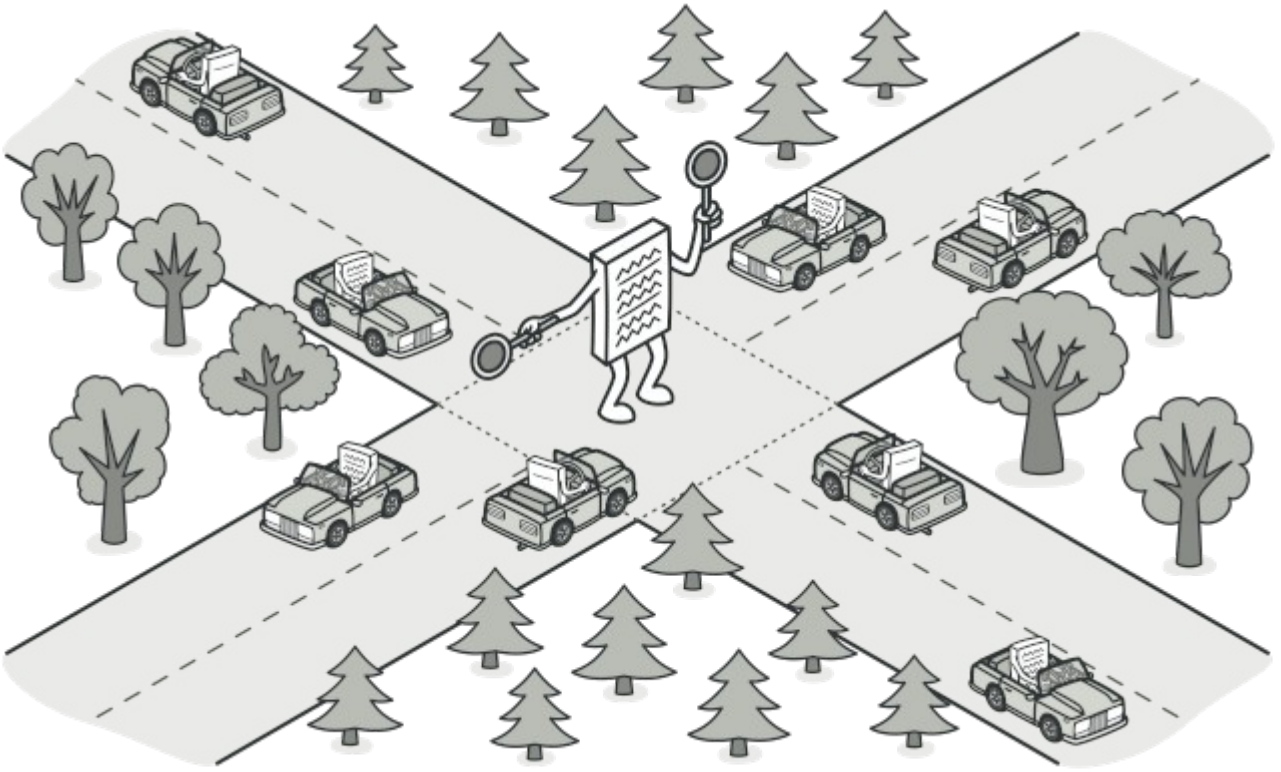

Button
«interface»
Command
+ execute()
Shortcut
Save
Command
Code
Open
Command
Code
Print
Command
Code

Итератор

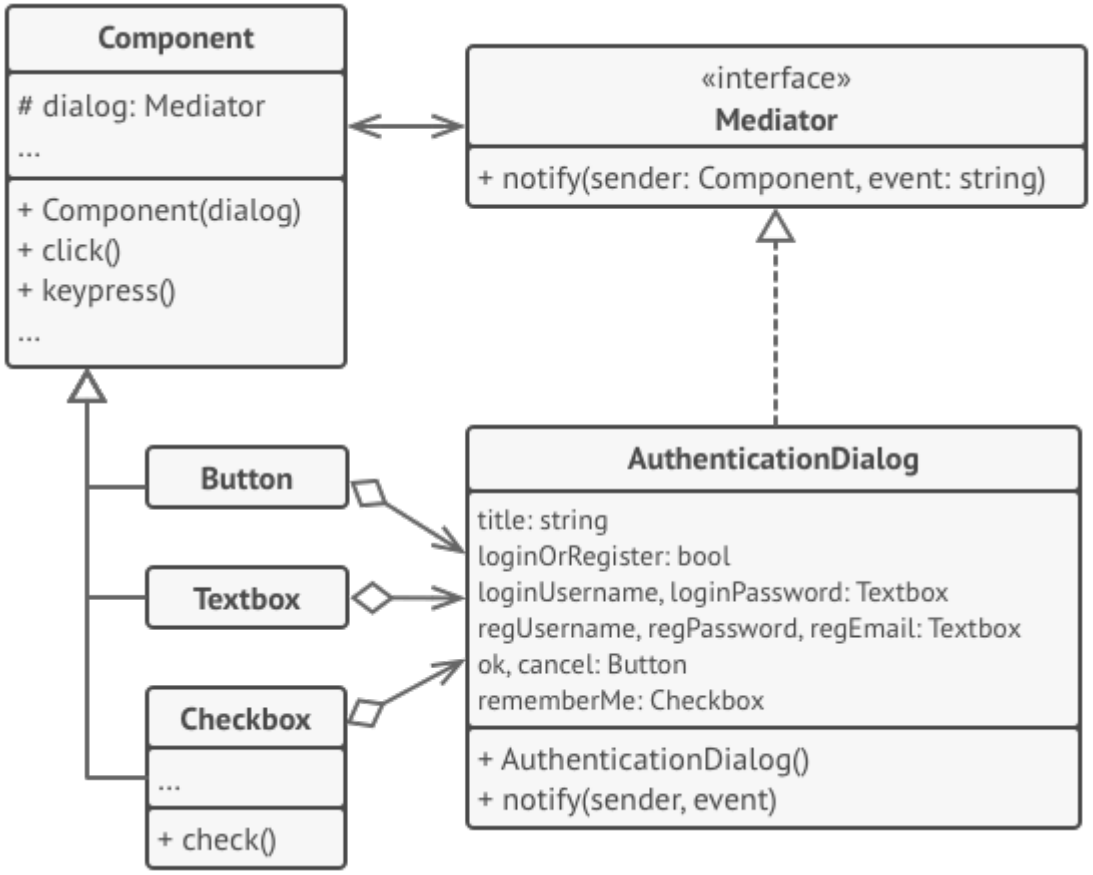

TreeCollection
...
+ getDepthIterator()
+ getBreadthIterator()
Depth-first Iterator
- currentElement
+ getNext(): Element
+ hasMore(): bool
Breadth-first Iterator
- currentElement
+ getNext(): Element
+ hasMore(): bool


Client
«interface»
Iterator
+ getNext()
+ hasMore(): bool
«interface»
IterableCollection
+ createIterator(): Iterator
ConcreteIterator
- collection: ConcreteCollection
- iterationState
+ ConcreteIterator(
c: ConcreteCollection)
+ getNext()
+ hasMore(): bool
ConcreteCollection
...
...
+ createIterator(): Iterator

Посредник

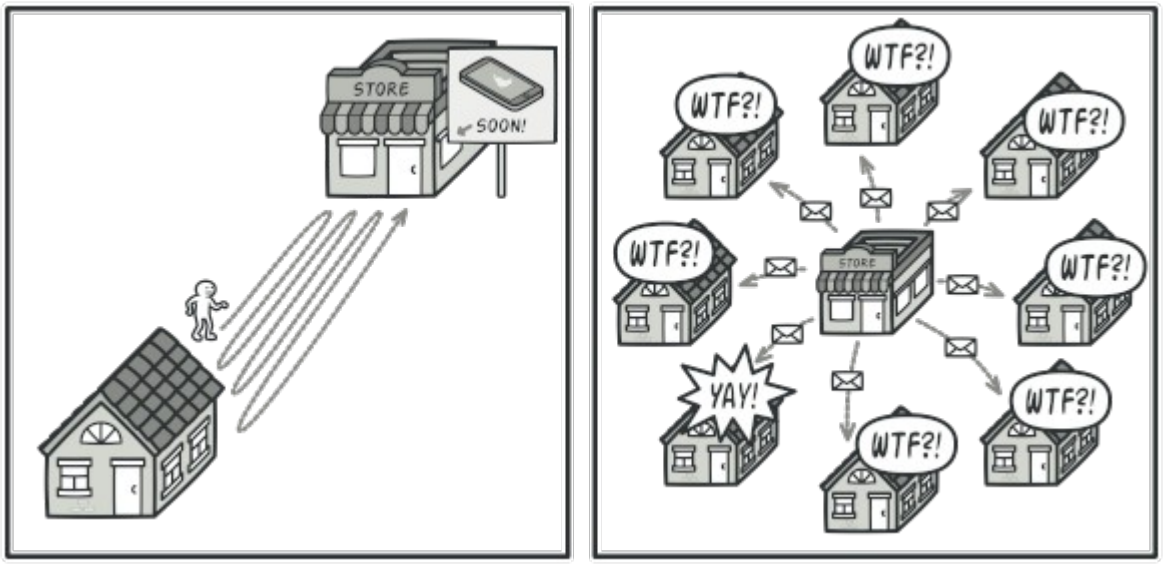


Посредник

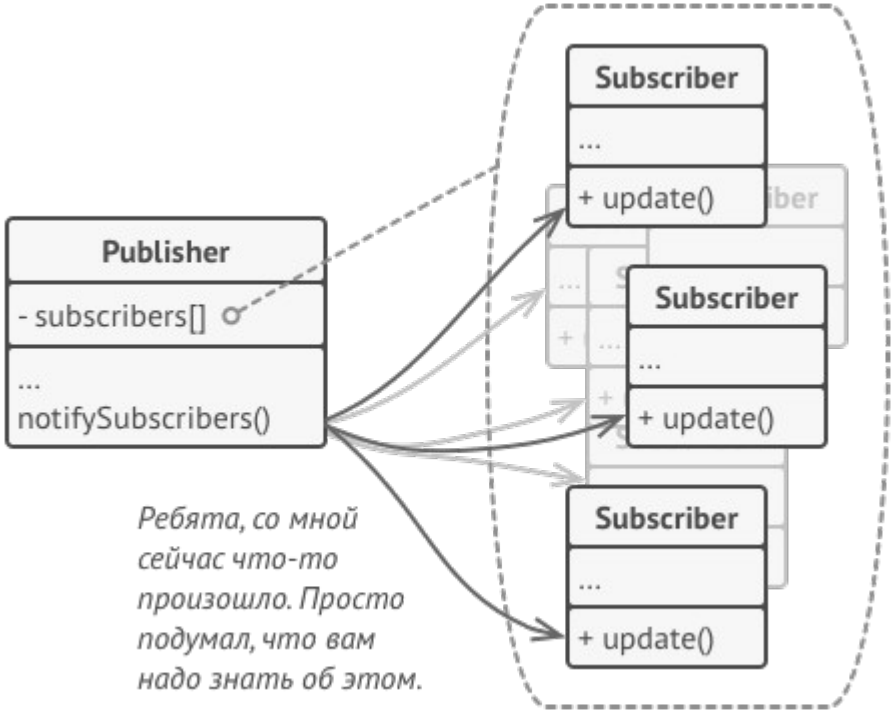

Component
dialog: Mediator
...
+ Component(dialog)
+ click()
+ keypress()
...
«interface»
Mediator
+ notify(sender: Component, event: string)
Button
Textbox
Checkbox
...
+ check()
AuthenticationDialog
title: string
loginOrRegister: bool
loginUsername, loginPassword: Textbox
regUsername, regPassword, regEmail: Textbox
ok, cancel: Button
rememberMe: Checkbox
+ AuthenticationDialog()
+ notify(sender, event)

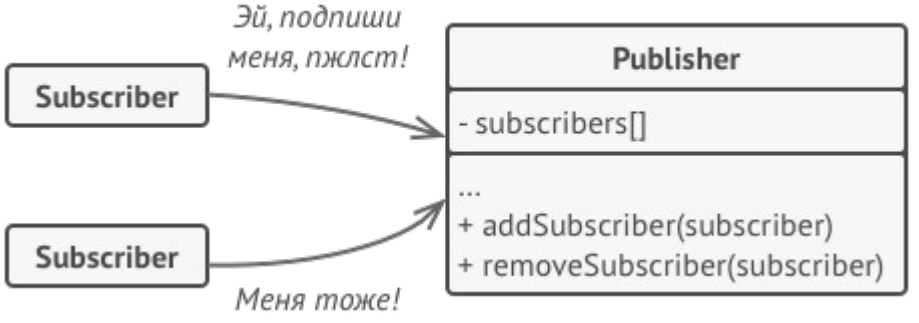
Наблюдатель

Наблюдатель

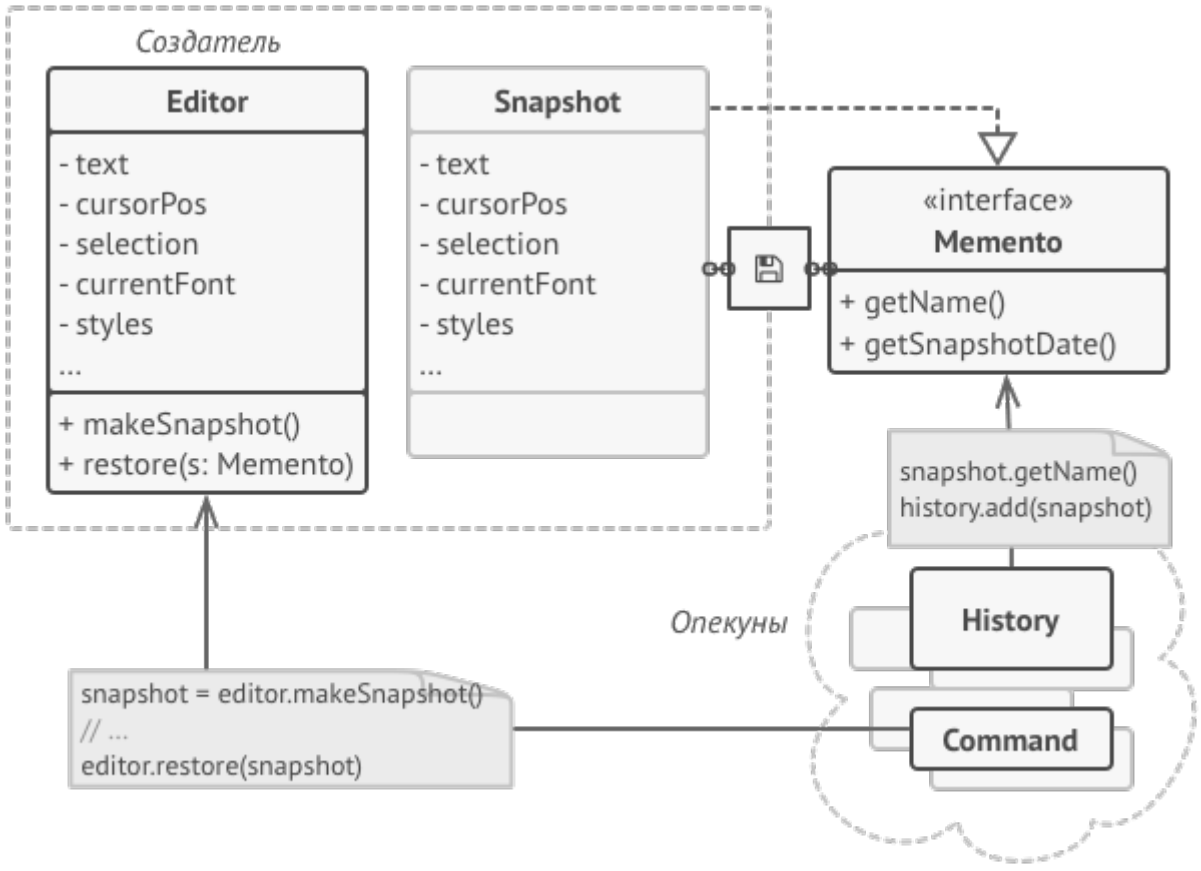


Наблюдатель



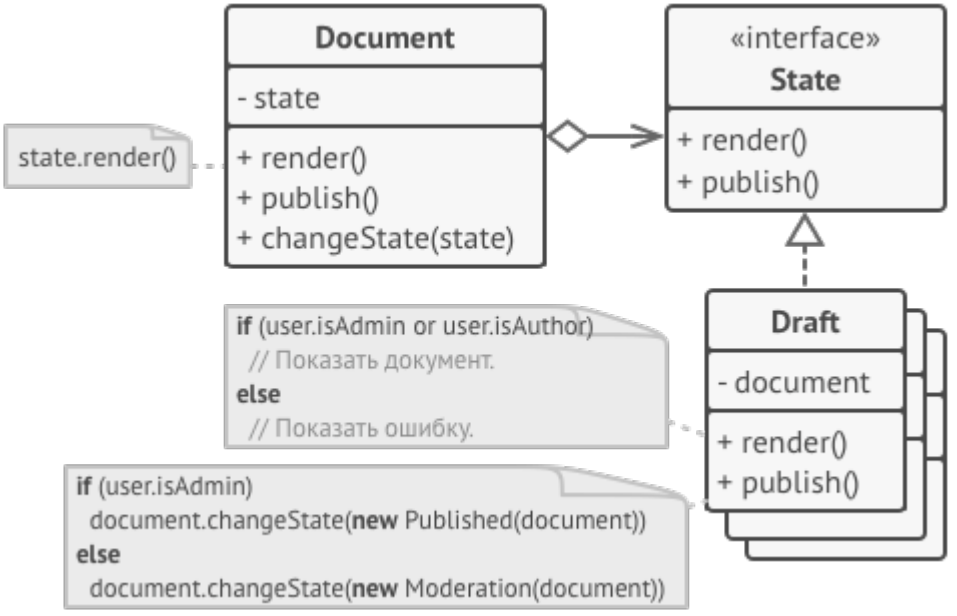


Снимок

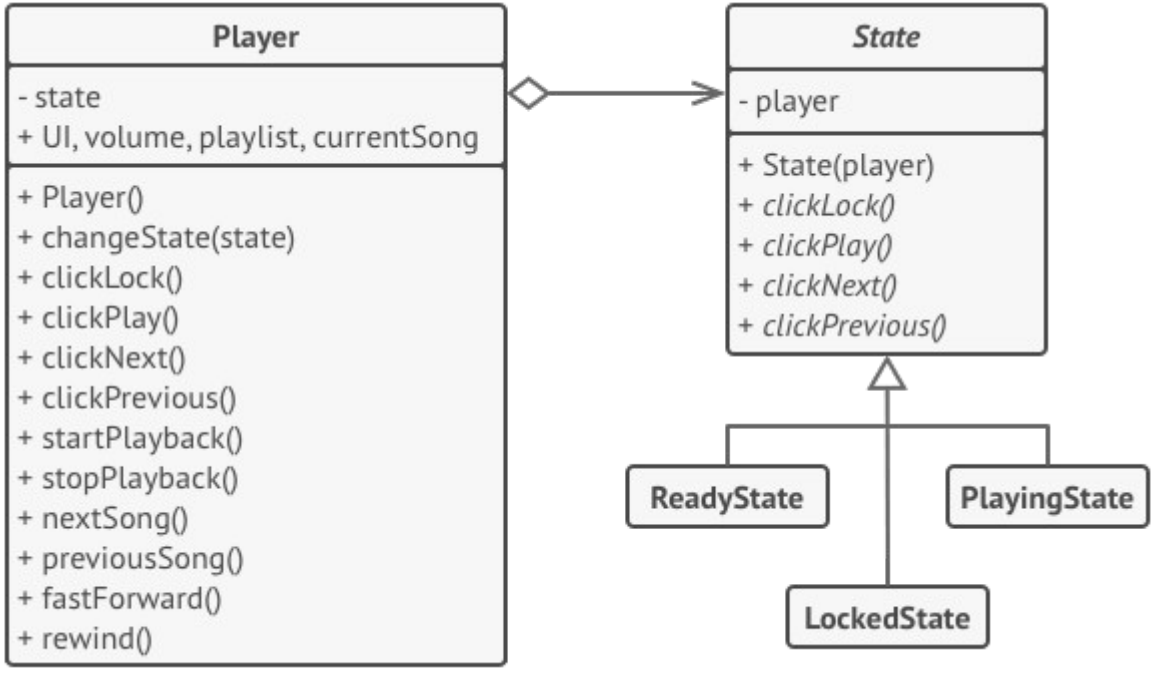

Создатель
Editor
- text
- cursorPos
- selection
- currentFont
- styles
...
+ makeSnapshot()
+ restore(s: Memento)
Snapshot
- text
- cursorPos
- selection
- currentFont
- styles
...
«interface»
Memento
+ getName()
+ getSnapshotDate()
snapshot.getName()
history.add(snapshot)
Опекуны
History
Command
snapshot = editor.makeSnapshot()
// ...
editor.restore(snapshot)

Состояние

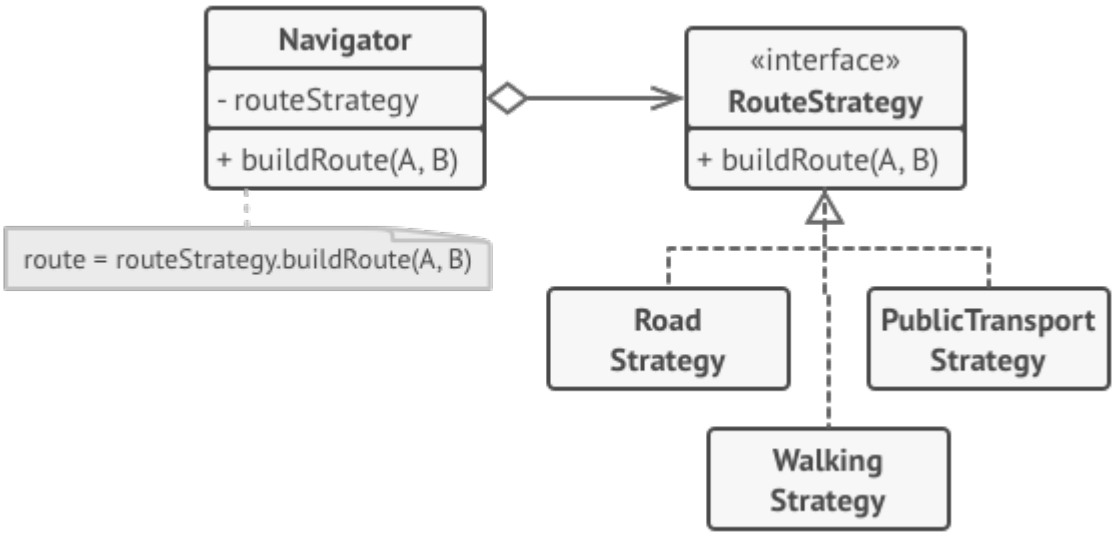

Черновик
Опубликован пользователем
Публикация отклонена
Публикация просрочена
Модерация
Опубликован админом
Публикация одобрена
Опубликован


Document
- state
+ render()
+ publish()
+ changeState(state)
«interface»
State
+ render()
+ publish()
state.render()
if (user.isAdmin or user.isAuthor)
// Показать документ.
else
// Показать ошибку.
Draft
- document
+ render()
+ publish()
if (user.isAdmin)
document.changeState(new Published(document))
else
document.changeState(new Moderation(document))

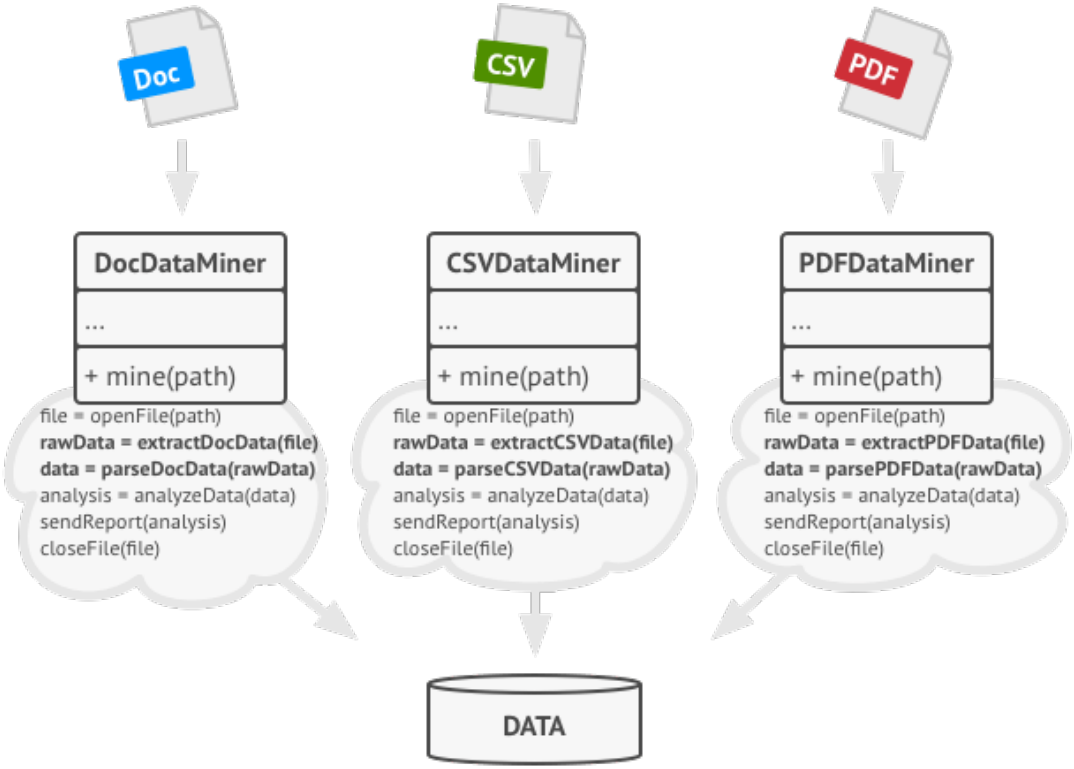
Состояние


Player
- state
+ UI, volume, playlist, currentSong
+ Player()
+ changeState(state)
+ clickLock()
+ clickPlay()
+ clickNext()
+ clickPrevious()
+ startPlayback()
+ stopPlayback()
+ nextSong()
+ previousSong()
+ fastForward()
+ rewind()
State
- player
+ State(player)
+ clickLock()
+ clickPlay()
+ clickNext()
+ clickPrevious()
ReadyState
PlayingState
LockedState

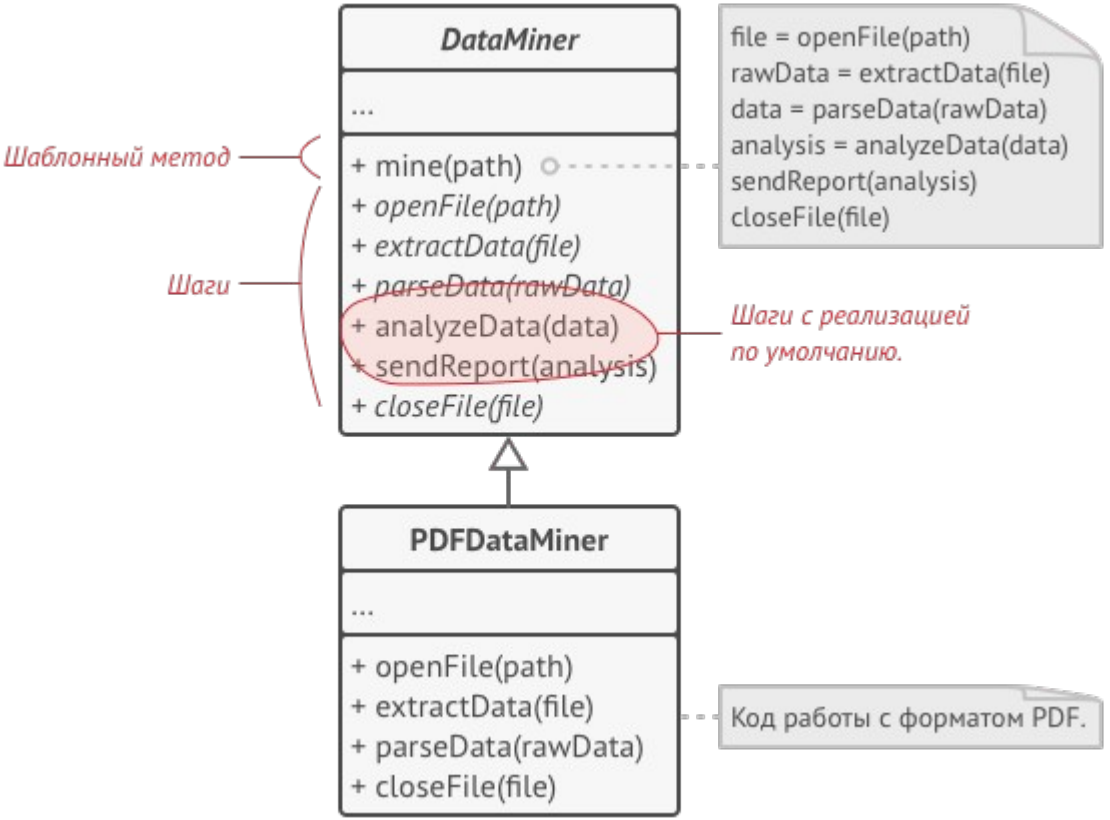
Стратегия


Navigator
- routeStrategy
+ buildRoute(A, B)
«interface»
RouteStrategy
+ buildRoute(A, B)
route = routeStrategy.buildRoute(A, B)
Road Strategy
PublicTransport Strategy
Walking Strategy

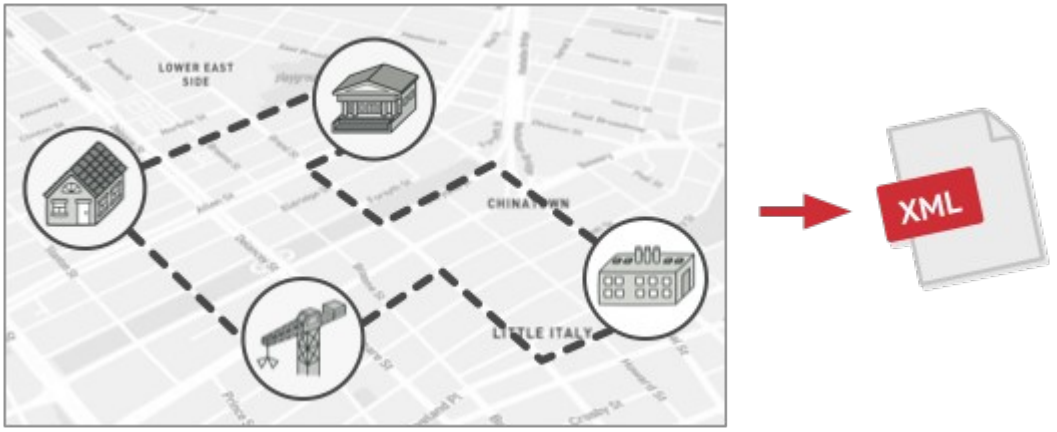
Шаблонный метод

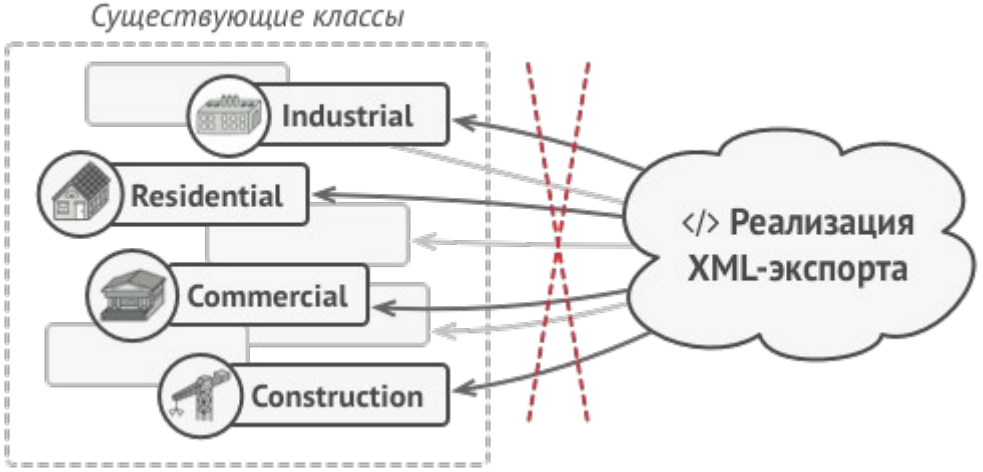


Шаблонный метод



Посеситель

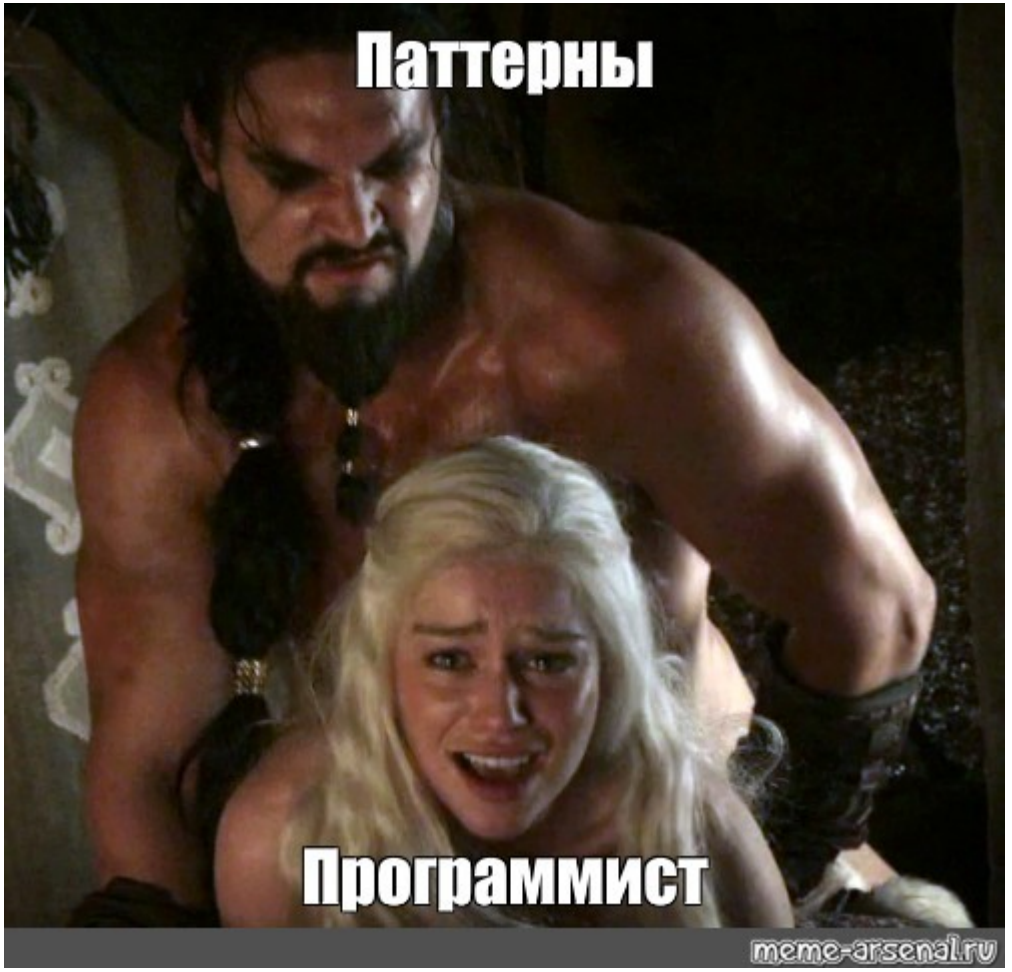

LOWER EAST SIDE
CHINATOWN
LITTLE ITALY
XML


Существующие классы
Industrial
Residential
Commercial
Construction
</> Реализация XML-экспорта

Посеситель

```
class ExportVisitor implements Visitor is
    method doForCity(City c) { ... }
    method doForIndustry(Industry f) { ... }
    method doForSightSeeing(SightSeeing ss) { ... }
    // ...
```

```
foreach (Node node in graph)
    if (node instanceof City)
        exportVisitor.doForCity((City) node)
    if (node instanceof Industry)
        exportVisitor.doForIndustry((Industry) node)
    // ...
```


Паттерны
Программист
meme-arsenal.ru


Когда
освоил паттерны
meme-arsenal.ru