17. Шаблон Стратегия (Strategy)

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ

ГЛ. АС. ЕМИЛ ДОЙЧЕВ

Общи сведения

- ✓ **Вид:** поведенчески за обекти
- ✓ **Цел:** Дефинира семейство алгоритми, капсулира всеки от тях, и ги прави взаимнозаменяеми. Този шаблон позволява алгоритмите да се сменят независимо от клиентите, които ги използват
- ✓ **Известен и като:** Policy

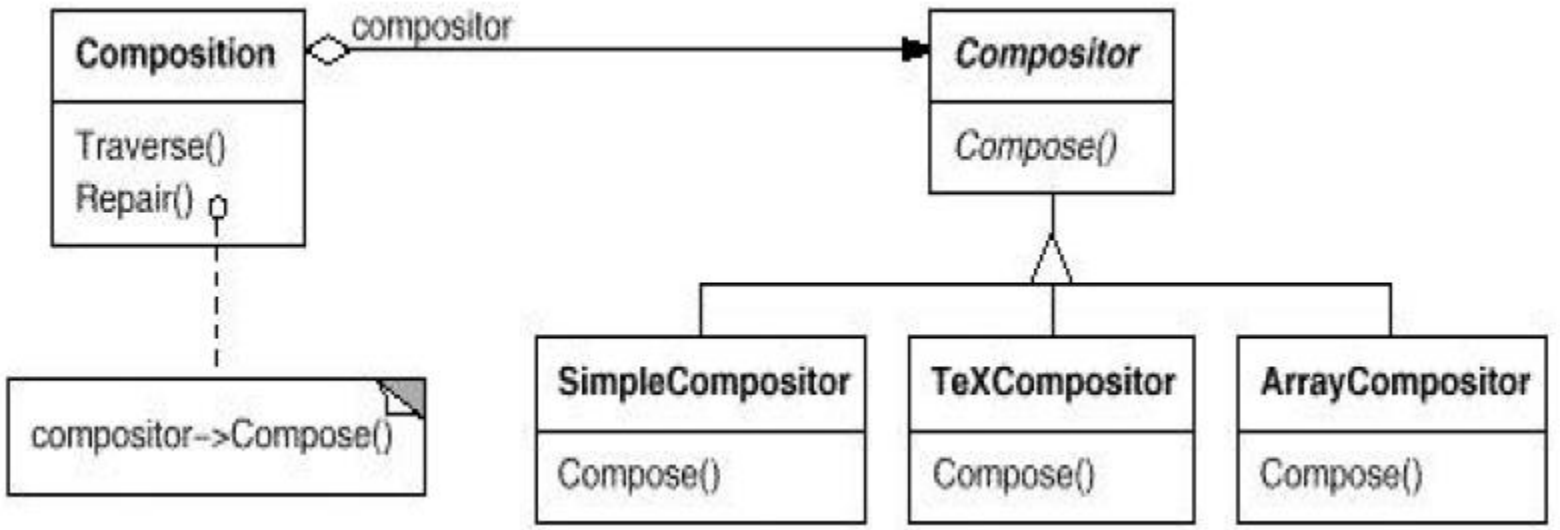
Мотивация

✓ Например при разделянето на поток от текст на редове съществуват множество алгоритми, някои по-прости, други по-сложни. Ако се опитаме да вложим всички тях в класовете на клиента, ще възникнат няколко проблема:

- Клиентите, които имат нужда от разделяне на текста на редове ще станат по-сложни, ако включим в тях и кода за имплементацията му. Това прави клиентите по-големи и по- трудни за поддръжка, особено ако подържат множество алгоритми за разделяне на редове.
- Тъй като различните алгоритми са подходящи са различни случаи, не е добре да се поддържат много алгоритми за разделяне на редове, ако не се използват всичките.
- Трудно се добавят нови и се променят съществуващи алгоритми, когато разделянето на редове е неразделна част от клиента.

Мотивация

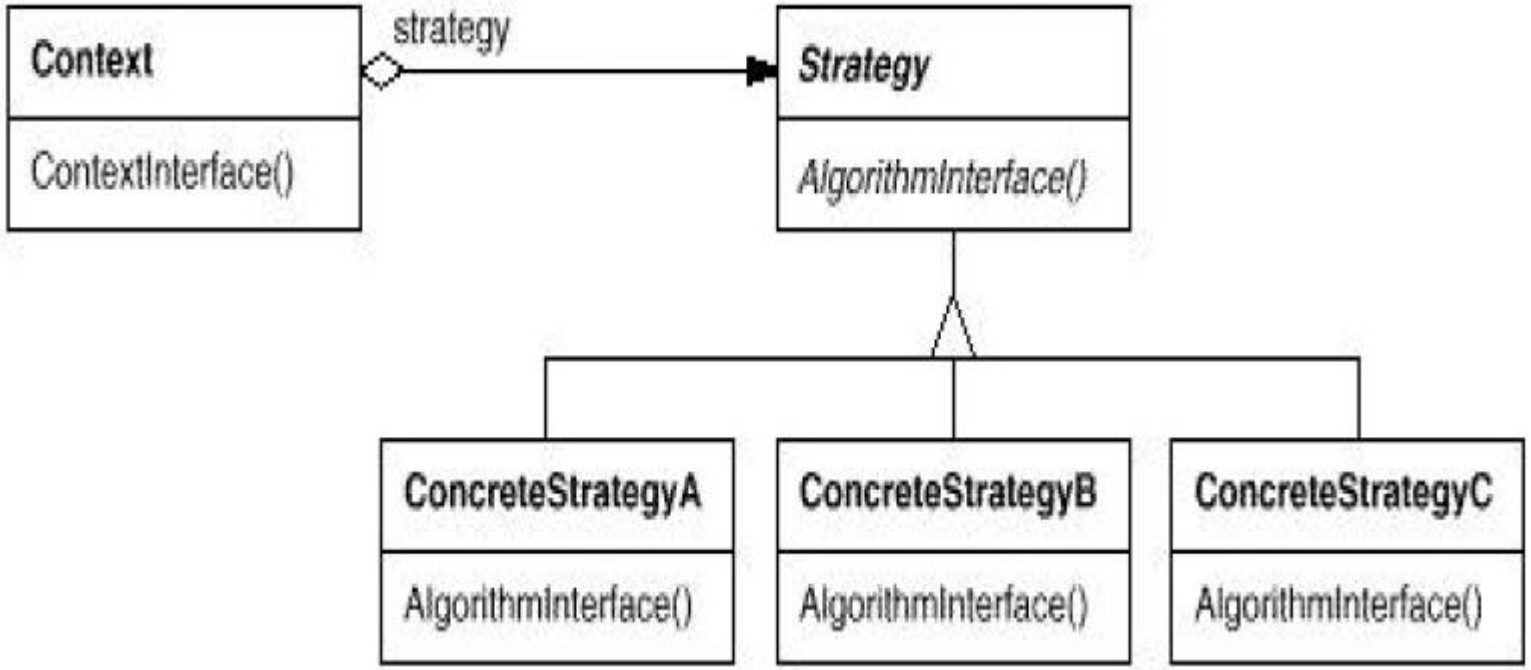
- ✓ Тези проблеми могат да бъдат избегнати, ако дефинираме класове, които да капсулират различните алгоритми за разделяне на редове. Капсулиран по този начин алгоритъм се нарича **стратегия**.



Приложимост

- ✓ **Приложимост:** Шаблонът Стратегия се използва в следните случаи:
  - Много свързани класове се различават само по поведението си.
  - Необходими са различни варианти на един алгоритъм.
  - Алгоритъм използва данни, които не трябва да са известни на клиентите. В този случай шаблонът Стратегия се използва за да се предотврати даването на достъп до структури от данни, които са сложни и специфични за самия алгоритъм.
  - Клас дефинира много поведения, които се реализират чрез множество условни конструкции. Вместо да се използват условни конструкции съответните разклонения се преместват в собствени Strategy класове.

Структура


Context
ContextInterface()
strategy
Strategy
AlgorithmInterface()
ConcreteStrategyA
AlgorithmInterface()
ConcreteStrategyB
AlgorithmInterface()
ConcreteStrategyC
AlgorithmInterface()

Участници

- ✓ **Strategy** (Compositor) – декларира интерфейс, общ за всички поддържани алгоритми. Context използва този интерфейс, за да извика алгоритъма, дефиниран от ConcreteStrategy.
- ✓ **Concrete Strategy** (SimpleCompositor, TeXCompositor, ArrayCompositor) – имплементира алгоритъма посредством интерфейса Strategy.
- ✓ **Context** (Composition) – конфигурира се чрез обект ConcreteStrategy и пази референция към обект Strategy. Може и да дефинира интерфейс, с който да даде възможност на Strategy да осъществява достъп до неговите данни.

Взаимодействия

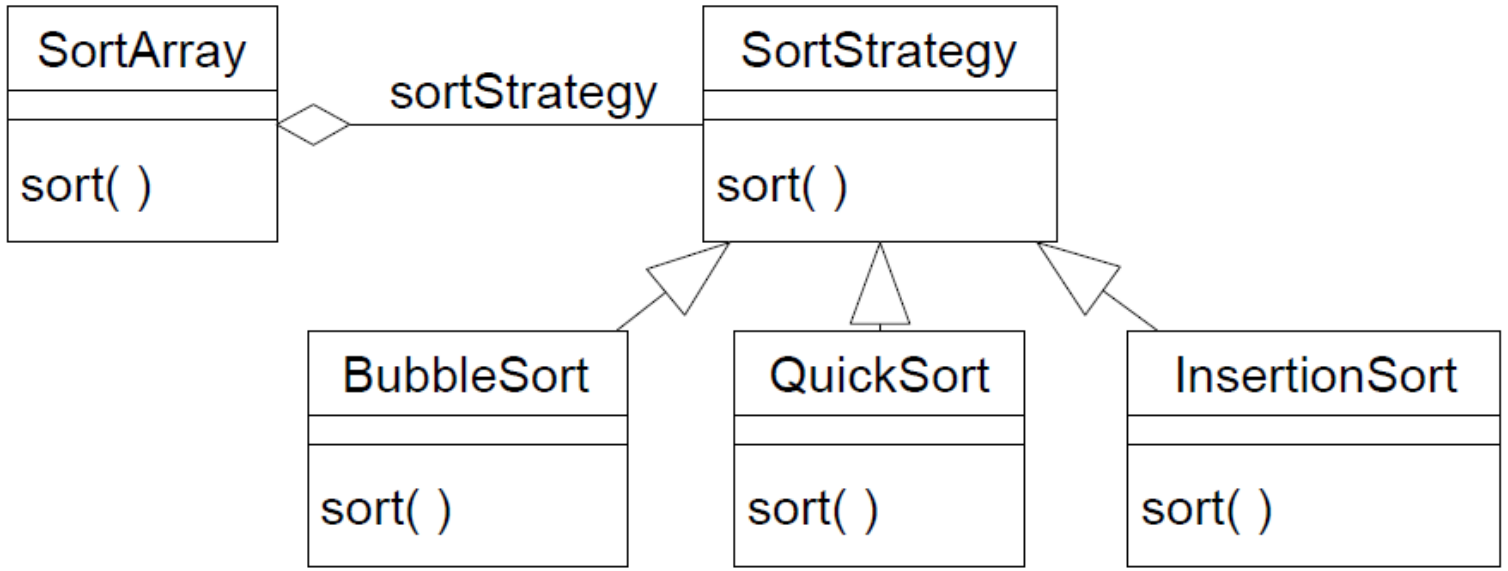
- ✓ Strategy и Context си взаимодействат, за да имплементират избрания алгоритъм. Контекстът може да предаде данните, необходими на алгоритъма, или да предаде самия себе си като аргумент на операциите на Стратегията.
- ✓ Контекстът препредава заявки от клиентите към стратегията, които обикновено могат да избират от цяло семейство конкретни стратегии.

Следствия

- ✓ **Предимства**
  - Предоставя алтернатива на наследяването на класа Context за да се реализират множество алгоритми или поведения.
  - Премахва големите условни конструкции.
  - Предоставя избор от имплементации на едно и също поведение.
- ✓ **Недостатъци**
  - Увеличава броя на обектите.
  - Всички алгоритми трябва да използват един и същи Strategy интерфейс

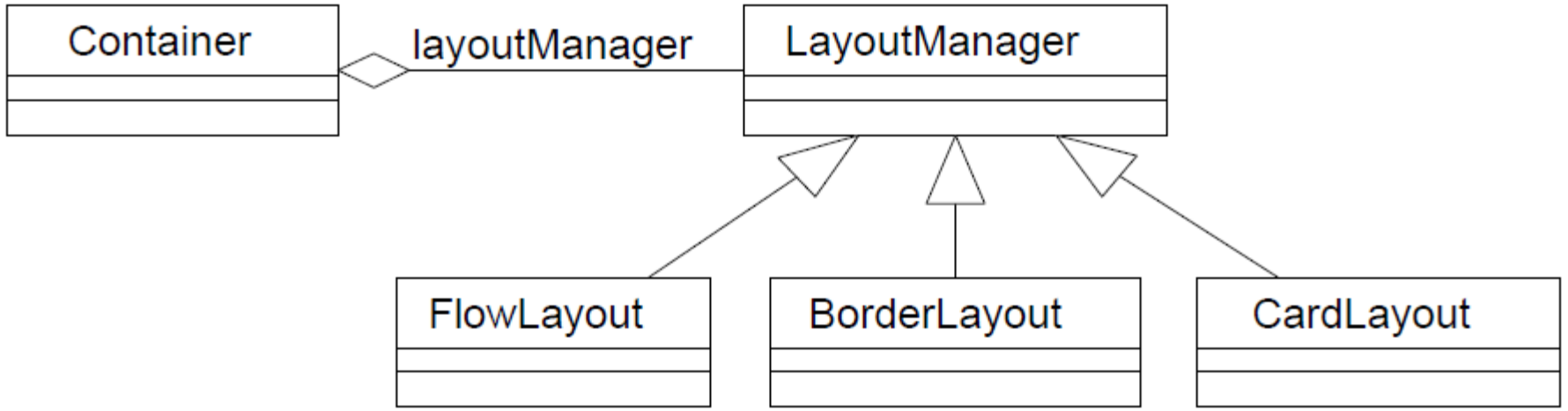
Пример 1

- ✓ **Проблем**: Клас трябва да определи по време на изпълнение какъв алгоритъм за сортиране да използва. Налични са различни видове алгоритми за сортиране.
- ✓ **Решение**: Капсулиране на различните алгоритми за сортиране чрез прилагане на шаблона Стратегия.



Пример 2

- ✓ **Проблем**: Контейнер за графични обекти на потребителския интерфейс (GUI контейнер) трябва да определи по време на изпълнение каква стратегия да използва за разполагане на графичните компоненти. Налични са множество такива стратегии.
- ✓ **Решение**: Капсулиране на различните стратегии за разполагане на компоненти чрез прилагане на шаблона Стратегия.
- ✓ Това прави Java AWT със своите LayoutManagers!



Пример 2 (продължение)

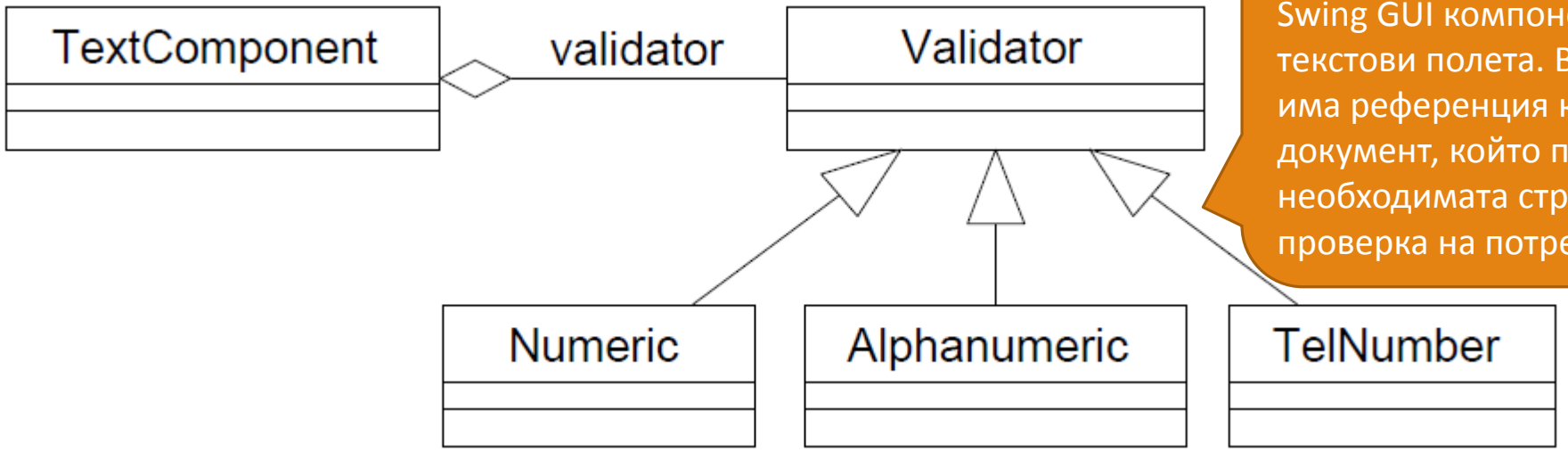
- ✓ Малко код

```
Frame f = new Frame();
f.setLayout(new FlowLayout());
f.add(new Button("Press"));
```

```
private void setupWindow() {
      ...

      setLayout(new BorderLayout());
      add(canvas, BorderLayout.CENTER);

      ...
      add(toolbox, BorderLayout.SOUTH);

      ...
}
```

Пример 3

- ✓ **Проблем**: Графичен компонент – текстова кутия (text box) трябва да определи по време на изпълнение каква стратегия да използва за проверка на въведените от потребителя данни. Налични са различни такива стратегии за проверка на входа за: цифрови полета, буквено-цифрови полета, полета за телефонен номер, за e-mail и т.н.
- ✓ **Решение**: Капсулиране на различните стратегии за проверка чрез прилагане на шаблона Стратегия.



Този подход се използва от Java Swing GUI компонентите за текстови полета. Всеки компонент има референция към модел на документ, който предоставя необходимата стратегия за проверка на потребителския вход.

Шаблонът Null Object

- ✓ Понякога Context може да не иска да използва стратегията предоставена от съдържащия се в него обект Strategy. В този случай се казва, че Context иска да използва “do nothing” стратегия.
- ✓ Едно от възможните решения е да се присвои `null` на референцията към Strategy обекта. В този случай Context трябва винаги да проверява за `null` преди да извика операция на Strategy обекта:

```
if (strategy != null)
   strategy.doOperation();
```

Шаблонът Null Object

- ✓ Другият вариант за постигане на този ефект е да се създаде клас за стратегия “do-nothing”, който имплементира всички нужни операции на интерфейса Strategy, но тези операции не извършват нищо. По този начин клиентите няма да е необходимо да правят разлика между обектите strategy, които реално извършват нещо и тези, които не извършват нищо.
- ✓ Използването на “do-nothing” обект за тази цел е познато като Null Object Pattern.

Край: Шаблон Стратегия

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ