14. Шаблон Верига отговорности (Chain of Responsibility)

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ

ГЛ. АС. Д-Р ЕМИЛ ДОЙЧЕВ

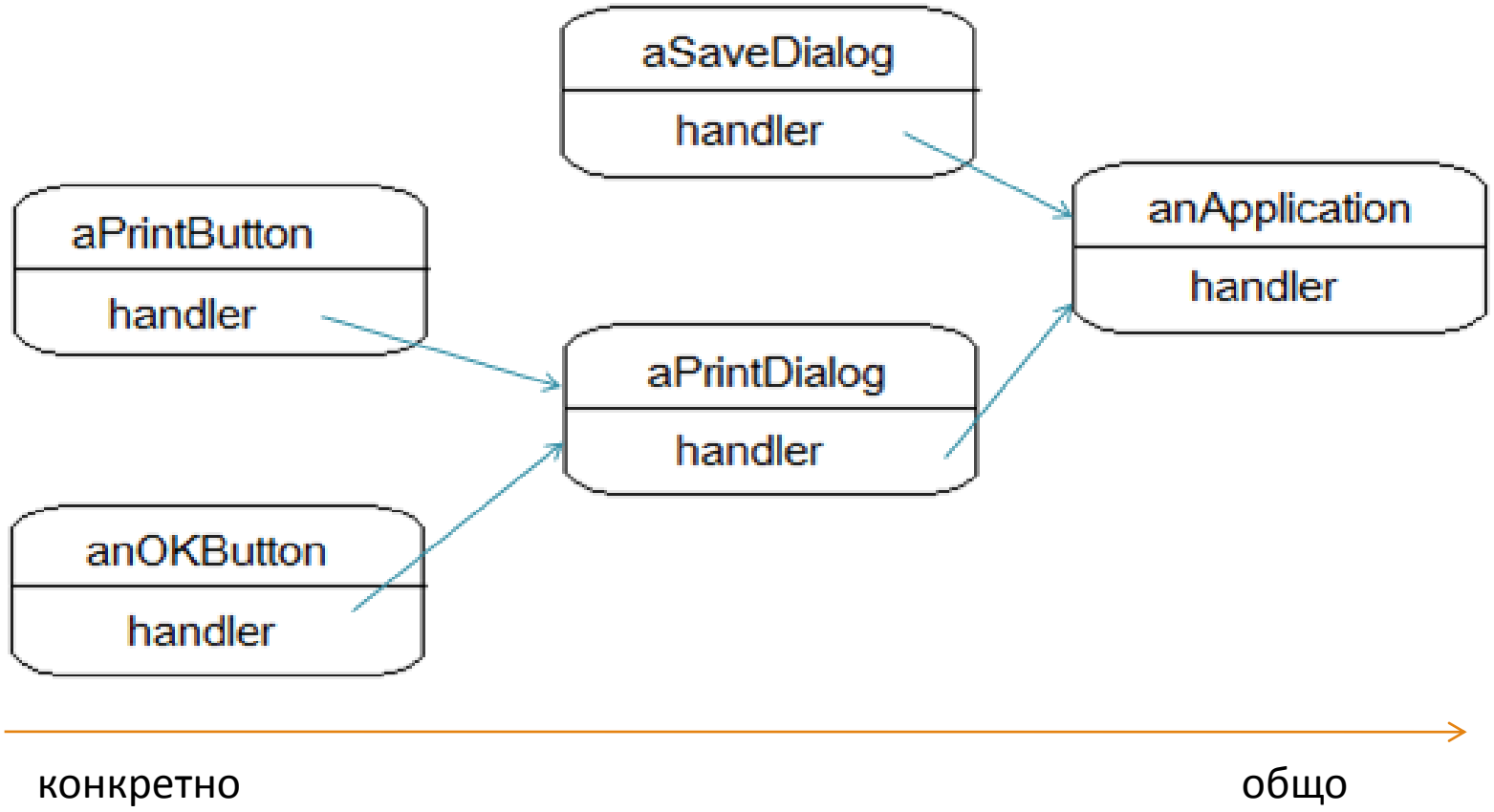
Общи сведения

- ✓ **Вид:** Поведенчески за обекти
- ✓ **Цел:** Избягва обвързването на изпращача на дадена заявка с получателя ѝ, като дава възможност на няколко обекта да обработят заявката. Свързва заедно приемащите обекти и предава заявката по веригата, докато някой от тях я обработи.

Мотивация

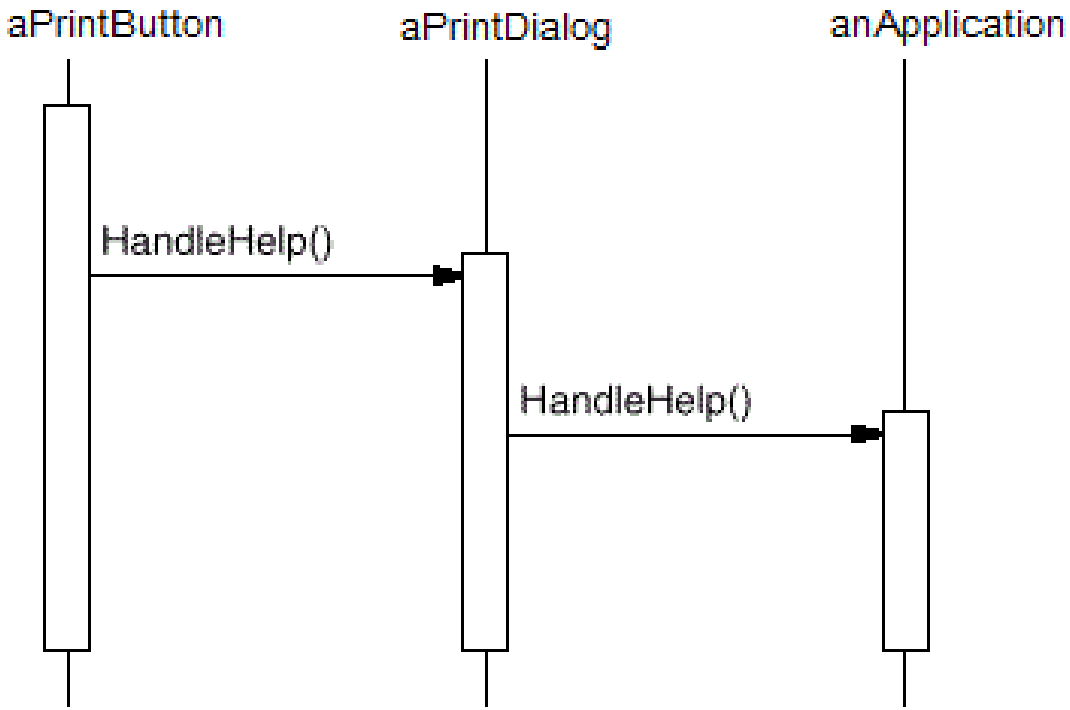
- ✓ Нека имаме контекстно-зависима помощна система за графичен потребителски интерфейс.
- ✓ Обектът, който реално предоставя помощната информация не е известен на обекта (напр. бутон), който изпраща заявката за помощ.
- ✓ Това е възможно като се използва верига от обекти за да се разделя изпращача на заявката от получателя. Заявката се изпраща през веригата докато някой от обектите я обработи.
- ✓ Първият обект във веригата получава заявката и или я обработва, или я предава на следващия кандидат от веригата, който прави същото. Обектът отправил заявката няма никаква представа, кой ще я обработи – т.е. заявката е с неявен получател (implicit receiver).
- ✓ Всеки обект във веригата споделя общ интерфейс за обработка на заявките и за достъп до наследниците си във веригата.

Мотивация


aSaveDialog
handler
aPrintButton
handler
anApplication
handler
aPrintDialog
handler
anOKButton
handler
конкретно
общо

Мотивация

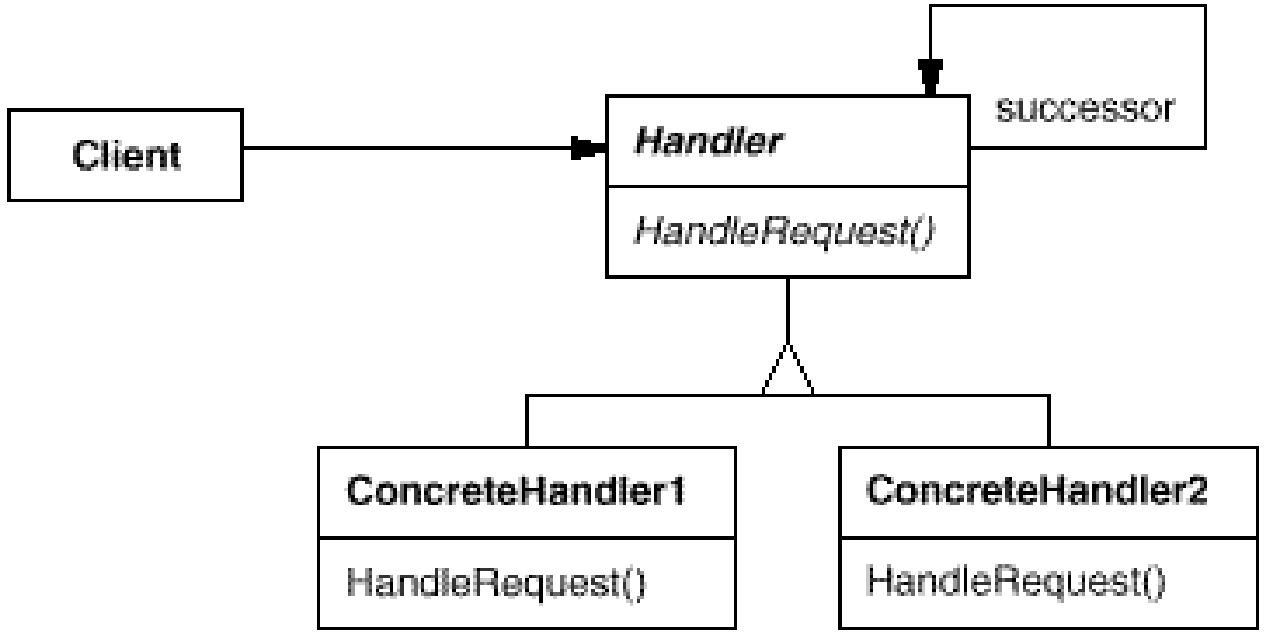
- ✓ Например: потребителя е натиснал бутон с надпис „Print”. Бутонът се съдържа в диалогов прозорец, инстанция на PrintDialog, който знае към кой обект на приложение принадлежи.
- ✓ В този случай, нито aPrintButton, нито aPrintDialog обработват заявката. Тя спира в anApplication, който може да я обработи или да я игнорира.
- ✓ Клиентът, издал заявката, няма директна връзка с обекта, който ще я изпълни.



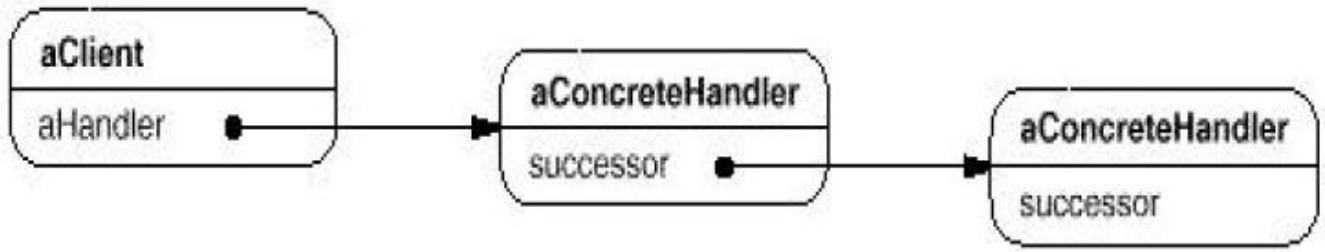
Приложимост

- ✓ **Приложимост:** Шаблонът Верига отговорности се използва в следните случаи:
  - Повече от един обект може да обработва дадена заявка, а изпълнителят не се знае *предварително*. Изпълнителя трябва да се установи автоматично.
  - Искате да изпратите заявка към няколко обекта, без да задавате изрично получателя.
  - Множеството обекти, които обработват дадена заявка, трябва да се задава динамично.

Структура


Client
Handler
HandleRequest()
successor
ConcreteHandler1
HandleRequest()
ConcreteHandler2
HandleRequest()

Структура


aClient
aHandler
aConcreteHandler
successor
aConcreteHandler
successor

Участници

- ✓ **Handler** (HelpHandler)
  - дефинира интерфейс за обработка на заявки.
  - (незадължително) имплементира връзката към следващия елемент.
- ✓ **ConcreteHandler** (PrintButton, PrintDialog)
  - обработва заявките, за които е отговорен (ако може да ги обработи).
  - ако не може да обработи заявката я предава на следващия елемент.
- ✓ **Client**
  - инициира заявката към ConcreteHandler обект от веригата.

Взаимодействия

- ✓ Когато даден клиент издаде заявката, тя се разпространява по веригата, докато някой ConcreteHandler обект поеме отговорност за обработката ѝ.

Следствия

- ✓ **Предимства**
  - **Слабо обвързване** – не е нужно обектите да знаят кой друг обект обработва дадена заявка, с което се опростяват взаимовръзките между обектите.
  - **Допълнителна гъвкавост при възлагането на отговорности на обекти** – добавяне или промяна на отговорности по време на изпълнение на заявката.
- ✓ **Недостатъци**
  - Обратната връзка не е сигурна – тъй като заявката няма изричен получател, няма гаранция, че ще бъде обработена.

Пример

- ✓ Приложеният проект

Край: Шаблон Верига отговорности

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ