adapter pattern

Marco Ferraioli, Dario Tecchia

02/11/2015

Università degli Studi di Salerno
marcoferraioli@live.com; dariotecchia@gmail.com
marcoferraioli.com

Keynote

Keynote

- Introduzione Pattern Strutturali

Keynote

- Introduzione Pattern Strutturali
- Introduzione all'Adapter Pattern

Keynote

- Introduzione Pattern Strutturali
- Introduzione all'Adapter Pattern
    - Struttura dell'Adapter Pattern

Keynote

- Introduzione Pattern Strutturali
- Introduzione all'Adapter Pattern
    - Struttura dell'Adapter Pattern
    - Tipologie di Adapter Pattern

Keynote

- Introduzione Pattern Strutturali
- Introduzione all'Adapter Pattern
    - Struttura dell'Adapter Pattern
    - Tipologie di Adapter Pattern
- Esempi

Keynote

- Introduzione Pattern Strutturali
- Introduzione all'Adapter Pattern
    - Struttura dell'Adapter Pattern
    - Tipologie di Adapter Pattern
- Esempi
- Conclusioni

introduzione pattern strutturali

Introduzione Pattern Strutturali

Introduzione Pattern Strutturali

- I pattern strutturali consentono di riutilizzare degli oggetti esistenti fornendo agli utilizzatori un'interfaccia più adatta alle loro esigenze

Introduzione Pattern Strutturali

- I pattern strutturali consentono di riutilizzare degli oggetti esistenti fornendo agli utilizzatori un'interfaccia più adatta alle loro esigenze
- Possono essere basati su classi o oggetti

Introduzione Pattern Strutturali

Introduzione Pattern Strutturali

- Design Pattern basati su classi:

Introduzione Pattern Strutturali

- Design Pattern basati su classi:
  - utilizzano l'ereditarietà per generare classi che combinano le proprietà di classi base

Introduzione Pattern Strutturali

- Design Pattern basati su classi:
  - utilizzano l'ereditarietà per generare classi che combinano le proprietà di classi base
- Design Pattern basati su oggetti:

Introduzione Pattern Strutturali

- Design Pattern basati su classi:
  - utilizzano l'ereditarietà per generare classi che combinano le proprietà di classi base
- Design Pattern basati su oggetti:
  - ci permettono di comporre oggetti per realizzare nuove funzionalità

Introduzione Pattern Strutturali

- Design Pattern basati su classi:
  - utilizzano l'ereditarietà per generare classi che combinano le proprietà di classi base
- Design Pattern basati su oggetti:
  - ci permettono di comporre oggetti per realizzare nuove funzionalità
  - da flessibilità alla composizione che viene effettuata a run-time, cosa impossibile da realizzare con le classi

introduzione all'adapter pattern

Introduzione all'Adapter Pattern

Introduzione all'Adapter Pattern

- L'adapter Pattern, conosciuto anche come Wrapper, si pone come soluzione astratta al problema dell'interoperabilità tra interfacce differenti

adapter pattern

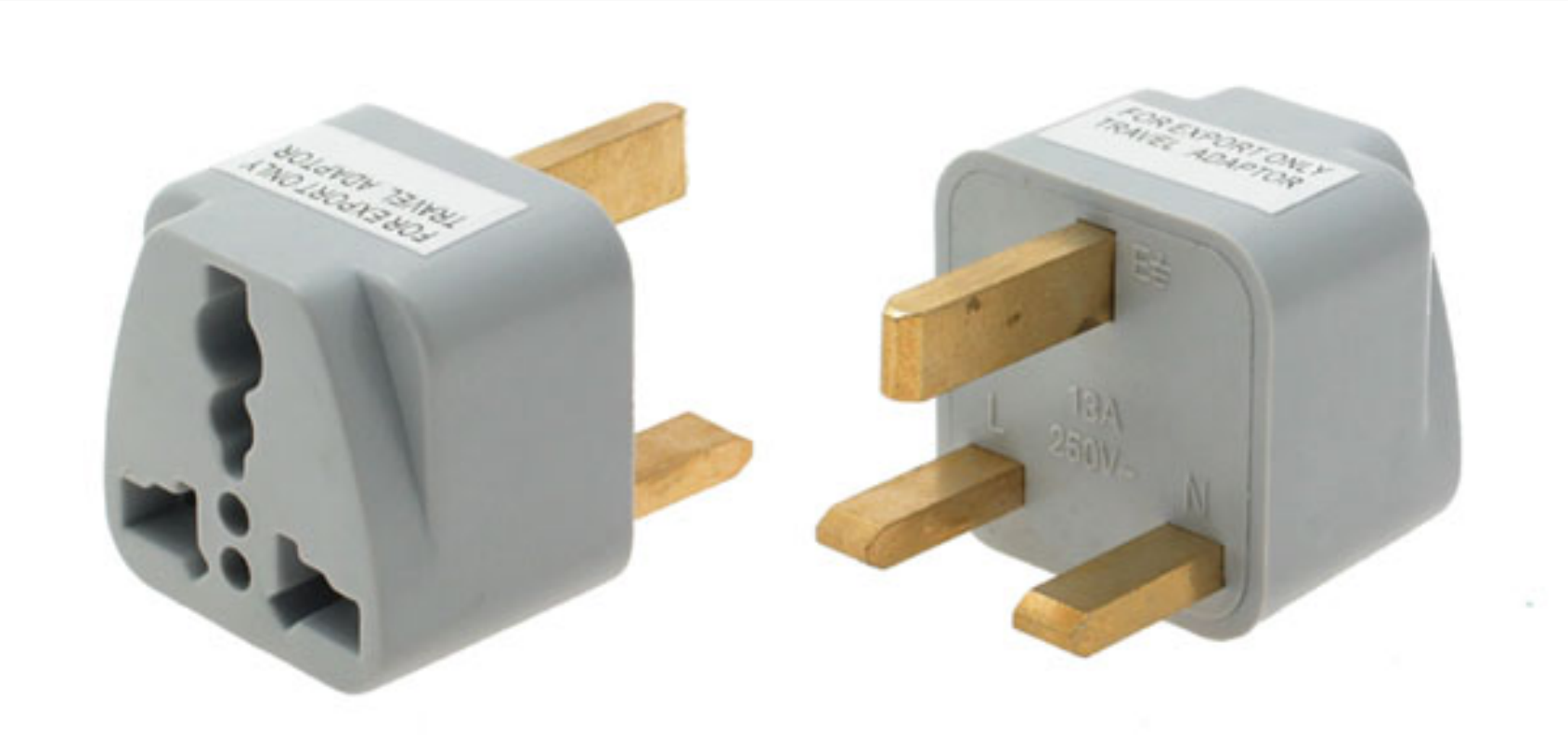
Introduzione all'Adapter Pattern

- L'adapter Pattern, conosciuto anche come Wrapper, si pone come soluzione astratta al problema dell'interoperabilità tra interfacce differenti
- Il problema si presenta ogni qual volta in un progetto software si devono utilizzare sistemi di supporto (es. librerie) la cui interfaccia non è perfettamente compatibile con del codice precedentemente scritto

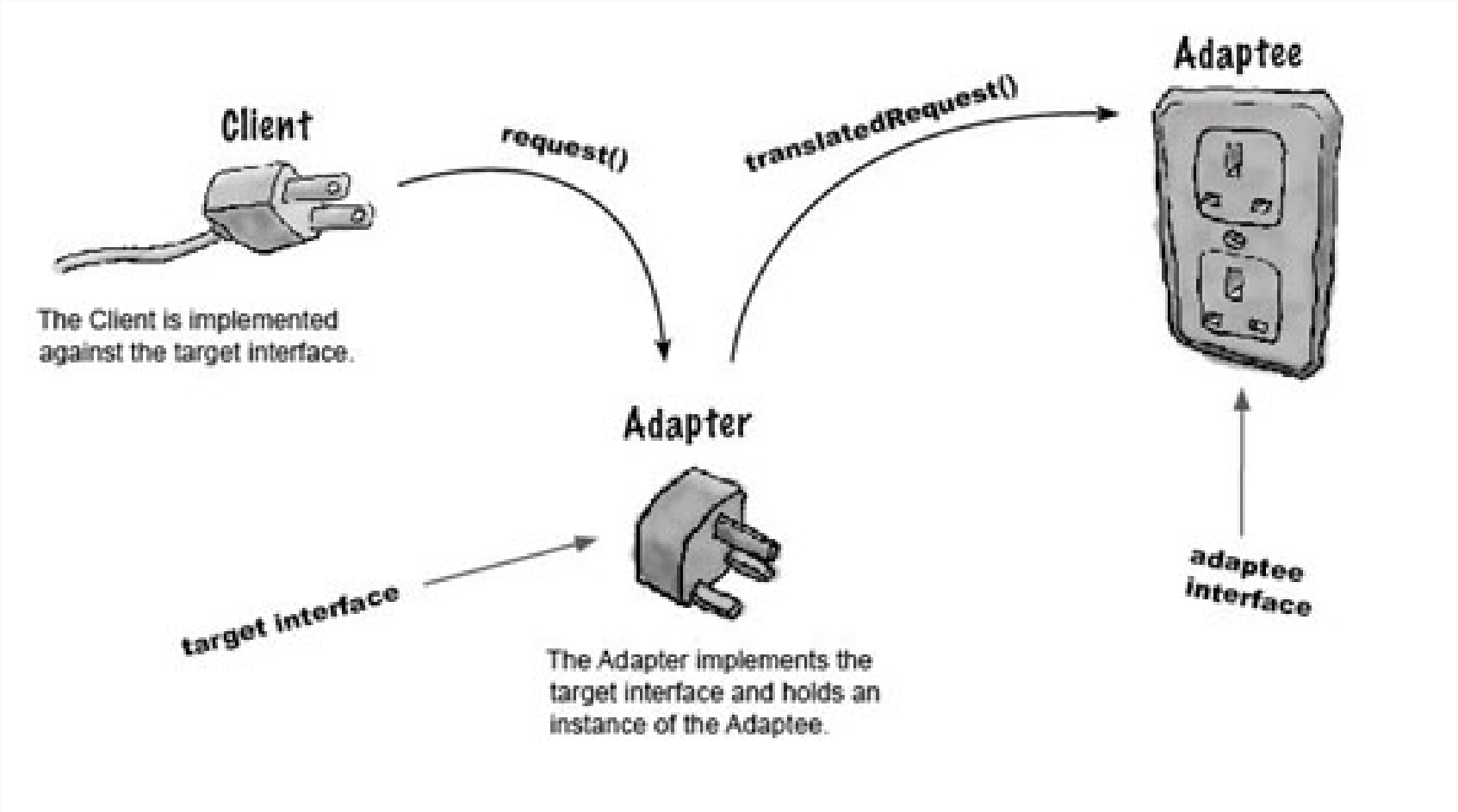
Esempi di Adapter nella vita reale

adapter pattern

adapter pattern



adapter pattern



struttura dell'adapter pattern

Struttura dell'Adapter Pattern: Composizione

Struttura dell'Adapter Pattern: Composizione

- **Adaptee:** definisce l'interfaccia di un diverso dominio applicativo da dover adattare per l'invocazione da parte del Client

Struttura dell'Adapter Pattern: Composizione

- **Adaptee:** definisce l'interfaccia di un diverso dominio applicativo da dover adattare per l'invocazione da parte del Client
- **Adapter:** definisce l'interfaccia compatibile con il Target che maschera l'invocazione dell'Adaptee

Struttura dell'Adapter Pattern: Composizione

- **Adaptee:** definisce l'interfaccia di un diverso dominio applicativo da dover adattare per l'invocazione da parte del Client
- **Adapter:** definisce l'interfaccia compatibile con il Target che maschera l'invocazione dell'Adaptee
- **Target:** definisce l'interfaccia specifica del dominio applicativo utilizzata dal Client

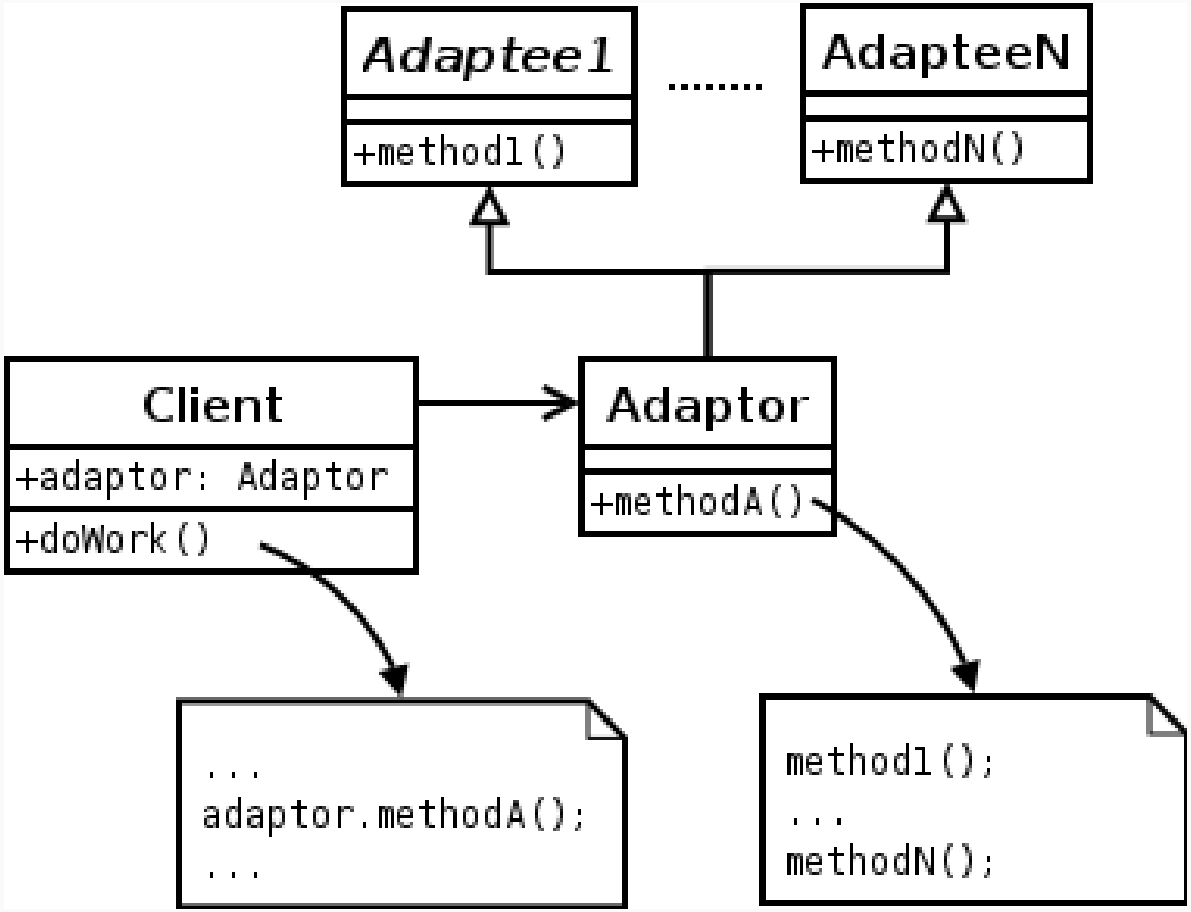
Struttura dell'Adapter Pattern: Composizione

- **Adaptee:** definisce l'interfaccia di un diverso dominio applicativo da dover adattare per l'invocazione da parte del Client
- **Adapter:** definisce l'interfaccia compatibile con il Target che maschera l'invocazione dell'Adaptee
- **Target:** definisce l'interfaccia specifica del dominio applicativo utilizzata dal Client
- **Client:** colui che effettua l'invocazione all'operazione di interesse

tipologie di adapter pattern

adapter pattern

Class Adapter



adapter pattern

Class Adapter

adapter pattern

Class Adapter

- Prevede un rapporto di ereditarietà tra Adapter e Adaptee (Adapter specializza Adaptee)

adapter pattern

Class Adapter

- Prevede un rapporto di ereditarietà tra Adapter e Adaptee (Adapter specializza Adaptee)
- Non è possibile creare un Adapter che specializzi più Adaptee

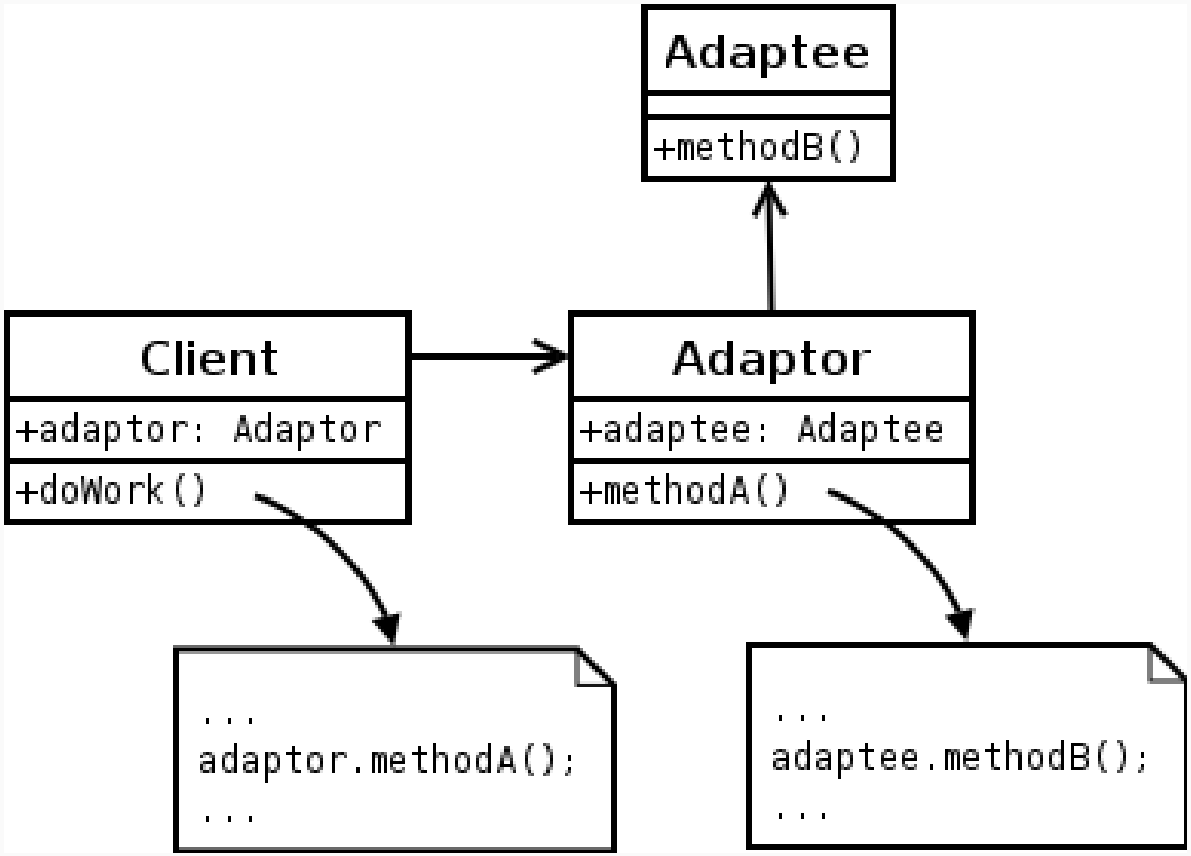
adapter pattern

Class Adapter

- Prevede un rapporto di ereditarietà tra Adapter e Adaptee (Adapter specializza Adaptee)
- Non è possibile creare un Adapter che specializzi più Adaptee
- Se esiste una gerarchia di Adaptee occorre creare una gereachia di Adapter

adapter pattern

Object Adapter



Object Adapter

adapter pattern

Object Adapter

- Prevede un rapporto di associazione tra Adapter e Adaptee (Adapter instanzia Adaptee)

adapter pattern

Object Adapter

- Prevede un rapporto di associazione tra Adapter e Adaptee (Adapter instanzia Adaptee)
- È possible avere un Adapter associato con più Adaptee

esempi

conclusioni

adapter pattern

Design Pattern Collegati

adapter pattern

Design Pattern Collegati

- Bridge Pattern

Design Pattern Collegati

- Bridge Pattern
  - Ha una struttura simile a quella dell'Object Adapter, ma il Bridge ha scopi differenti: separa l'interfaccia dalla propria implementazione

Design Pattern Collegati

- Bridge Pattern
  - Ha una struttura simile a quella dell'Object Adapter, ma il Bridge ha scopi differenti: separa l'interfaccia dalla propria implementazione
- Decorator

Design Pattern Collegati

- Bridge Pattern
  - Ha una struttura simile a quella dell'Object Adapter, ma il Bridge ha scopi differenti: separa l'interfaccia dalla propria implementazione
- Decorator
  - Il decorator è simile all'Adapter Pattern, migliora gli oggetti senza cambiarne l'interfaccia, quindi la sua implementazione risulta più trasparente

Design Pattern Collegati

- Bridge Pattern
  - Ha una struttura simile a quella dell'Object Adapter, ma il Bridge ha scopi differenti: separa l'interfaccia dalla propria implementazione
- Decorator
  - Il decorator è simile all'Adapter Pattern, migliora gli oggetti senza cambiarne l'interfaccia, quindi la sua implementazione risulta più trasparente
- Proxy

Design Pattern Collegati

- Bridge Pattern
  - Ha una struttura simile a quella dell'Object Adapter, ma il Bridge ha scopi differenti: separa l'interfaccia dalla propria implementazione
- Decorator
  - Il decorator è simile all'Adapter Pattern, migliora gli oggetti senza cambiarne l'interfaccia, quindi la sua implementazione risulta più trasparente
- Proxy
  - Definisce un rappresentante di un altro oggetto e non cambia la sua interfaccia

Materiale:
https://github.com/paranoiasystem/SeminarioTPA

adapter pattern

adapter pattern

Materiale sotto licenza GNU GPL

Fine, Domande?