Il pattern Adapter

a cura di **Angelo Furfaro**

da “Design Patterns”, Gamma et al.
“Patterns in Java”, Grand

Dipartimento di
Ingegneria Informatica, Elettronica, Modellistica e Sistemistica
Università della Calabria, 87036 Rende(CS) - Italy
Email: a.furfaro@unical.it
Web: http://angelo.furfaro.dimes.unical.it

Adapter

Classificazione

- Scopo: strutturale
- Raggio d'azione: classi e oggetti

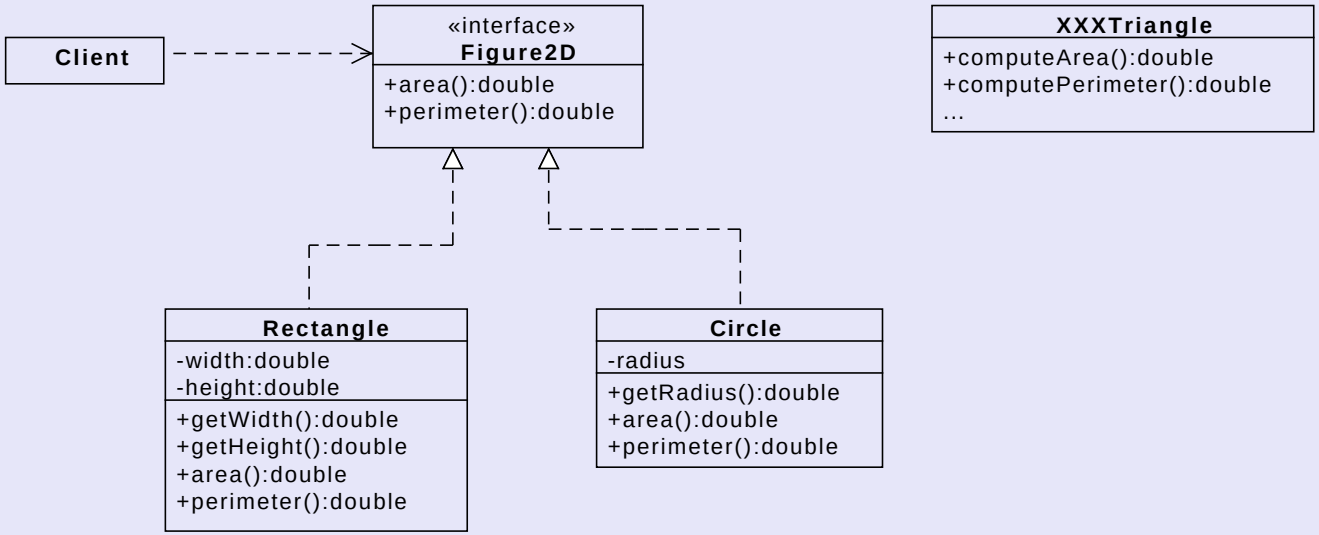
Altri nomi

Wrapper

Scopo

- Convertire l'interfaccia di una classe in un'altra interfaccia richiesta dal client.
- Adapter consente a classi diverse di cooperare insieme quando ciò non sarebbe possibile a causa di interfacce incompatibili.

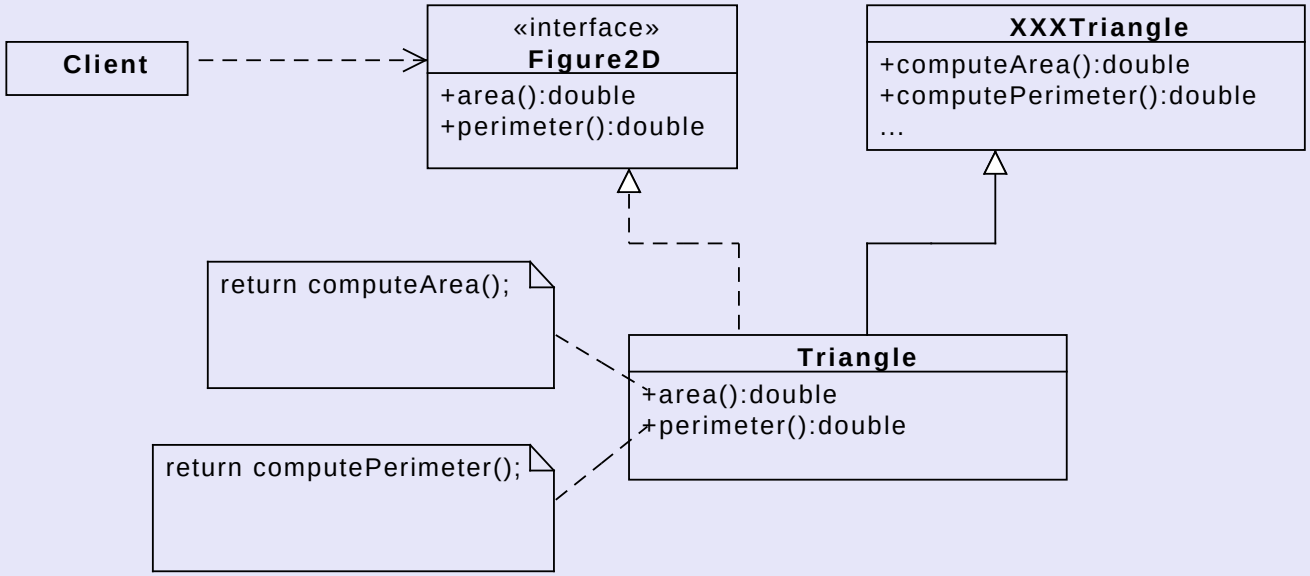
Problema



- A volte una classe preesistente, progettata per essere riutilizzata, non può essere riusata solo perchè la sua interfaccia non è compatibile con quella specifica richiesta dall'applicazione
- Nell'esempio, la classe `XXXTriangle` non può essere riusata dove ci si aspetta l'interfaccia `Figure2D` perchè non è compatibile con essa
- Modificare `XXXTriangle` per renderla conforme a `Figure2D` non è una buona soluzione: la si legherebbe ad un contesto specifico ed inoltre il sorgente potrebbe non essere disponibile.

Soluzione

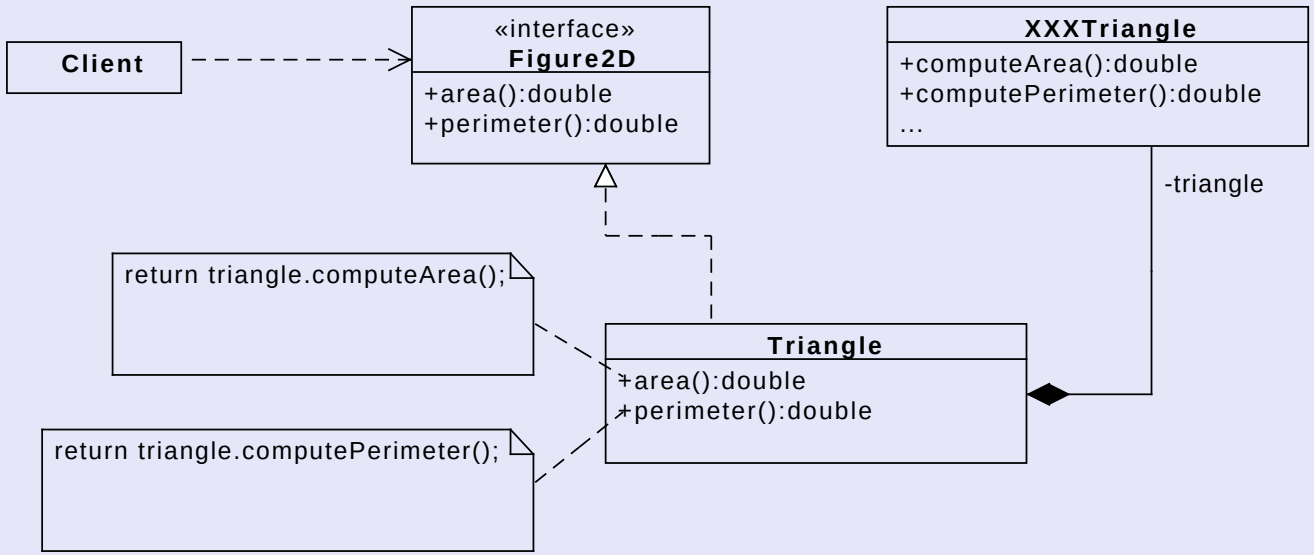
Class adapter



- Si introduce una classe `Triangle` che sia al contempo erede di `XXXTriangle` e conforme all'interfaccia `Figure2D`. L'implementazione dei metodi di `Figure2D` sfrutta quelli di `XXXTriangle`
- L'ereditarietà non è sempre possibile.
- Nel caso di Java:
  - se `XXXTriangle` è final
  - se `Figure2D` fosse una classe e non un'interfaccia

Soluzione

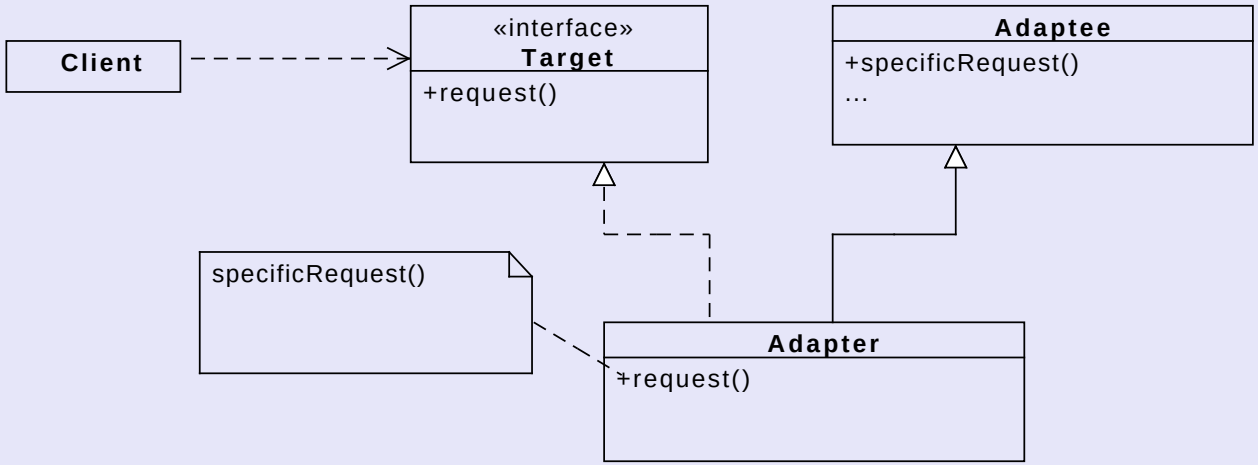
Object adapter



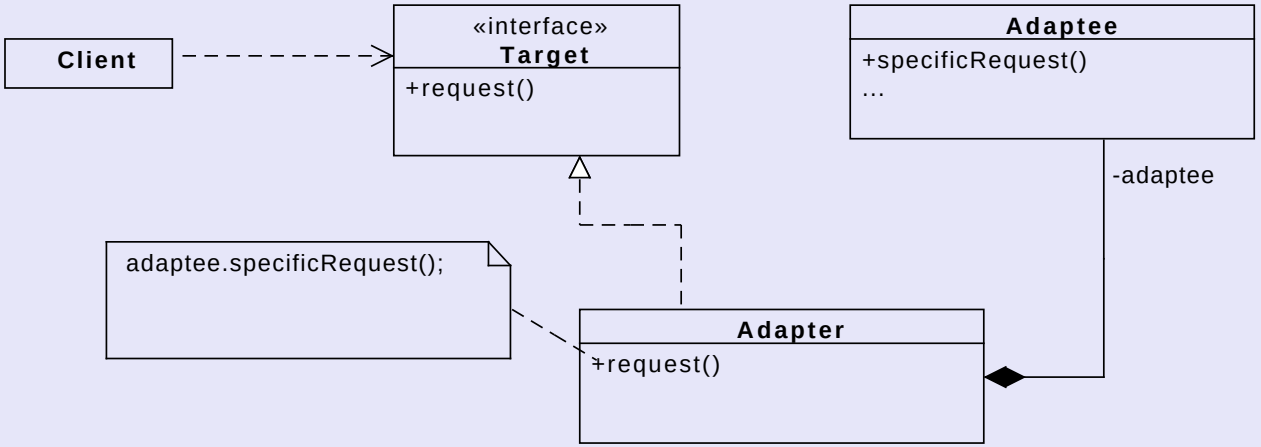
- Si introduce una classe `Triangle` che implementa l'interfaccia `Figure2D` e le cui istanze incapsulano un oggetto di tipo `XXXTriangle`
- L'implementazione dei metodi di `Figure2D` *delega* l'esecuzione all'oggetto incapsulato
- Non ci sono problemi di ereditarietà: è sempre applicabile

Struttura

Class Adapter



Object Adapter



Struttura

Partecipanti

- **Target** (`Figure2D`): definisce l'interfaccia specifica del domino utilizzata dal client
- **Client**: collabora con oggetti compatibili con l'interfaccia `Target`
- **Adaptee** (`XXXTriangle`): individua un'interfaccia che deve essere adattata
- **Adapter** (`Triangle`): adattata l'interfaccia `Adaptee` all'interfaccia `Target`

Collaborazioni

- I client invocano le operazioni su un'istanza di `Adapter`.
- L'adapter a sua volta invoca operazioni di `Adaptee` per soddisfare la richiesta.

Conseguenze

Class Adapter

- ☺ Adatta l'interfaccia di `Adaptee` all'interfaccia `Target` basandosi su una classe concreta.
- ☹ Non può essere utilizzata se `Adaptee` è astratta oppure è un'interfaccia.
- ☺ Consente ad `Adapter` di sovrascrivere parte del comportamento di `Adaptee` essendo una sua sottoclasse.
- ☺ Non occorrono ulteriori indirezioni per raggiungere l'oggetto adattato.

Object Adapter

- ☺ Permette ad un singolo `Adapter` di operare con `Adaptee` e le sue sottoclassi, se esistono. Può in tal caso aggiungere funzionalità a tutti gli `Adaptee`.
- ● Rende difficile sovrascrivere il comportamento di `Adaptee` non essendo una sua sottoclasse.
- ☹ Aggiunge un livello di indirezione per raggiungere l'oggetto adattato.

Implementazione

```
public interface Figure2D{
   double area();
   double perimeter();
}
```

```
public class Rectangle implements Figure2D{
 private double width;
 private double height;
  ...
 public Rectangle(double w,double h){
    width=w; height=h;
 }
 public getHeight(){
  return height;
 }
 public getWidth(){
  return width;
 }
 public double area(){
    return width*height;
 }
 public double perimeter(){
    return 2*(width+height);
 }
}
```

Implementazione

```
public class XXXTriangle ...{
   ...
  public double computeArea(){...}
  public double computePerimeter(){...}
}
```

Class Adapter

```
public class Triangle extends XXXTriangle implements Figure2D{

 ...
 public Triangle (...) {
   super(...);
 }

 public double area(){
   return computeArea();
 }
 public double perimeter(){
   return computePerimeter();
 }
}
```

Implementazione

Object Adapter

```
public class Triangle implements Figure2D{
 private XXXTriangle triangle;
  ...
 public Triangle (...) {
    triangle =new XXXTriangle(...);
 }

 public double area(){
    return triangle .computeArea();
 }
 public double perimeter(){
    return triangle .computePerimeter();
 }
}
```

Codice Cliente

```
Figure2D[] figure= new Figure2D[3];
figure[0]=new Rectangle(3.0,7.0);
figure[1]=new Triangle(3.0,4.0,5.0) ;
figure[2]=new Circle(5.0);
 ...
double am=areaMedia(figure);
 ...
```

Pattern Correlati

- Bridge
- Decorator
- Proxy