Adapter

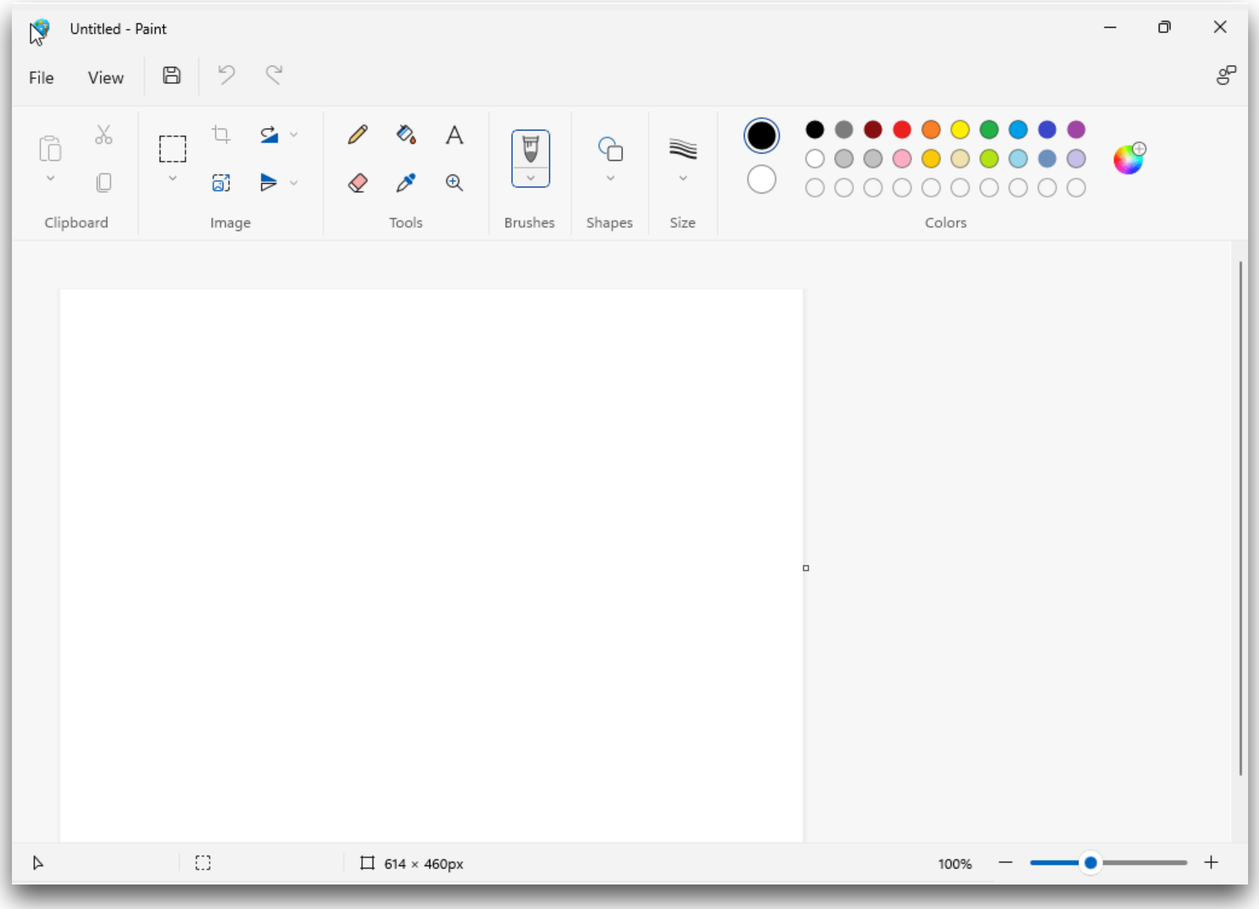
Angelo Afeltra

Intent

Converte l’interfaccia di una classe in un’altra interfaccia che i client si aspettano, in modo tale che possano lavorare insieme.

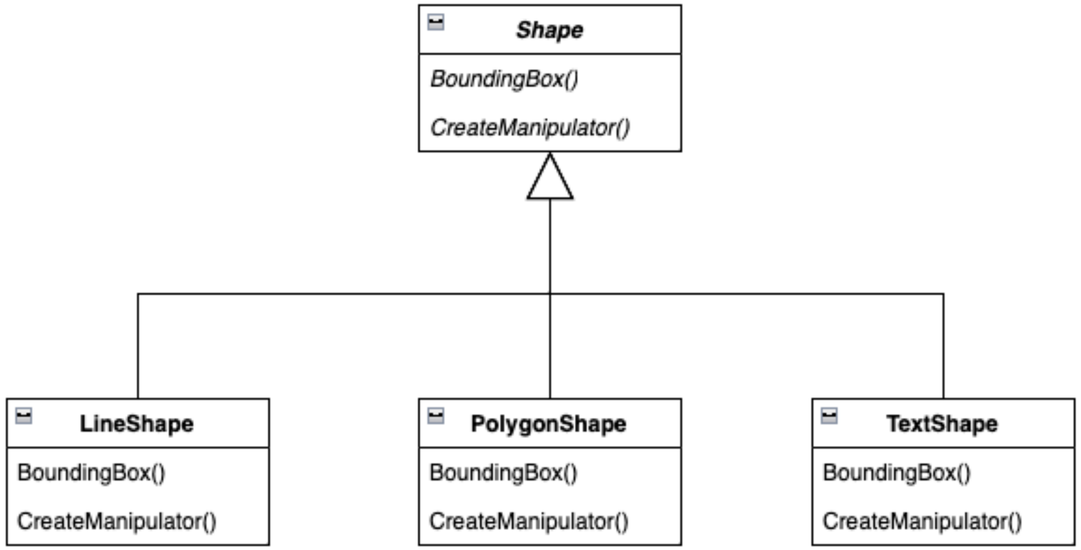
Problema

Immaginiamo di dover sviluppare un editor grafico, che consente agli utenti di disegnare e disporre elementi grafici come linee, poligoni, testo ecc.



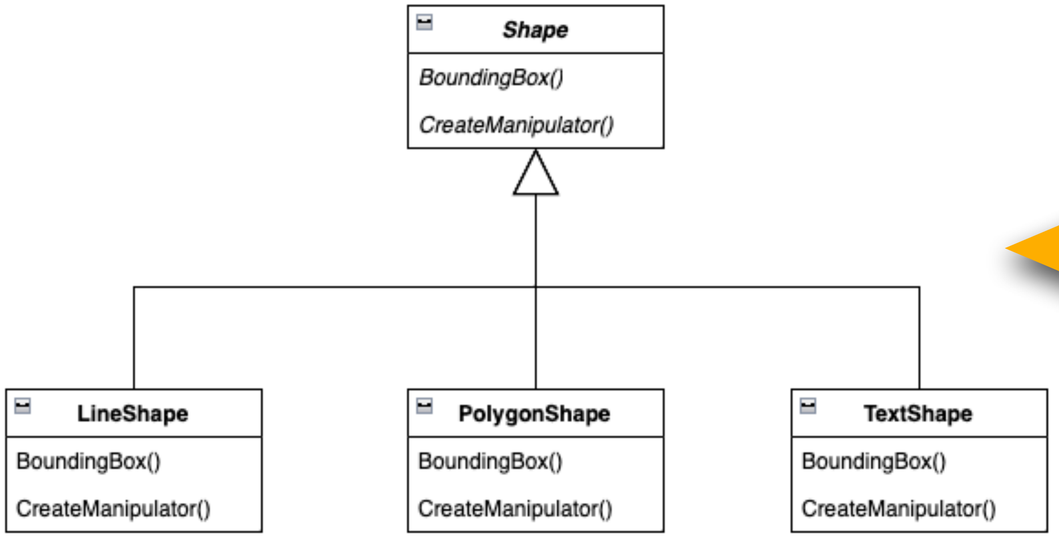
Problema

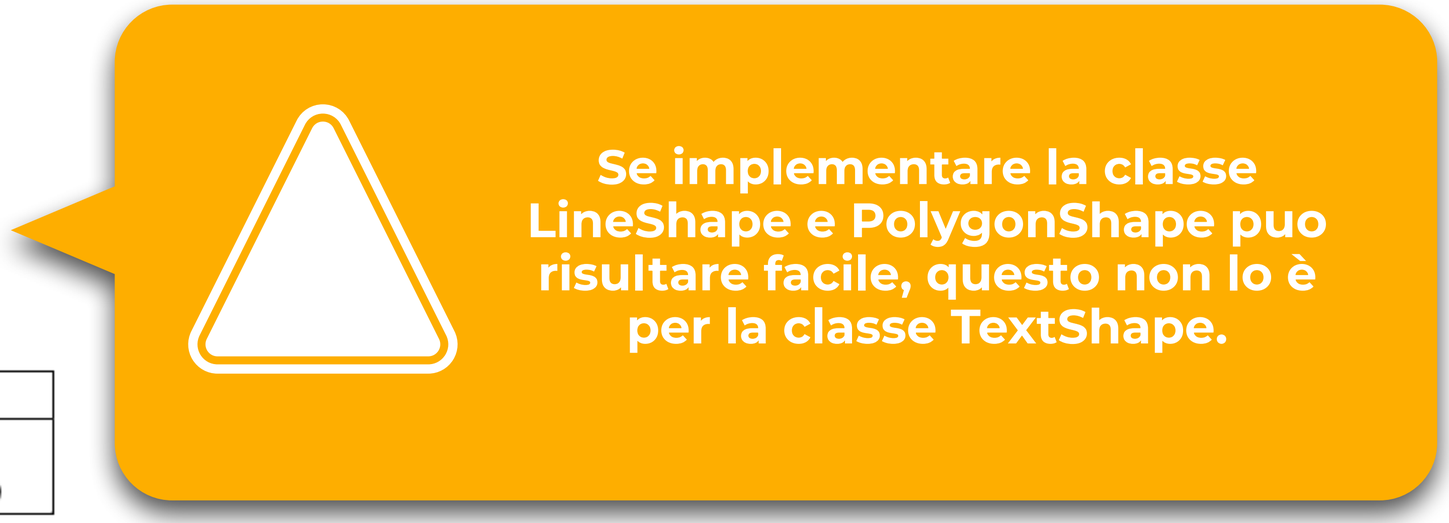
Per quanto riguarda gli elementi grafici, possiamo pensare di realizzare l'astrazione tramite una classe astratta Shape e poi sviluppare per ognuno di essi la propria sottoclasse.



Problema

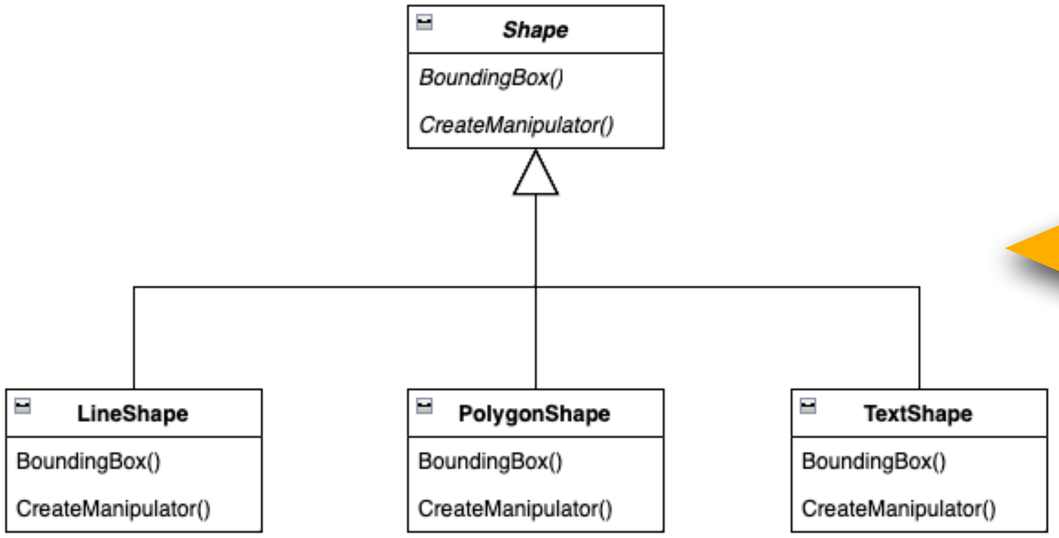
Per quanto riguarda gli elementi grafici, possiamo pensare di realizzare l'astrazione tramite una classe astratta Shape e poi sviluppare per ognuno di essi la propria sottoclasse.





Problema

Per quanto riguarda gli elementi grafici, possiamo pensare di realizzare l'astrazione tramite una classe astratta Shape e poi sviluppare per ognuno di essi la propria sottoclasse.



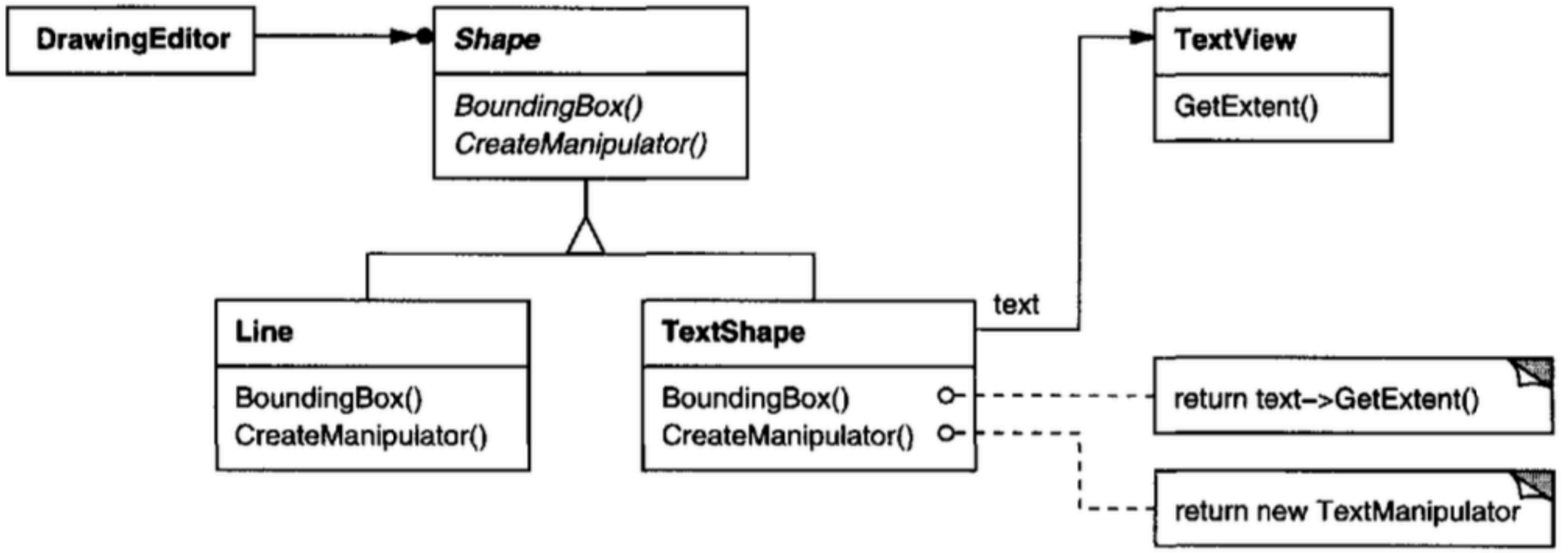
Per semplificare la progettazione possiamo pensare di utilizzare una classe gia implementata in un altro editor gragico, TextView.

Problema

Può capitare che non è possibile riutilizzare la classe TextView in quanto essa non è una sottoclasse di Shape. In tal caso si può risolvere il problema costruendo TextShape in modo tale che adatti l'interfaccia di TextView a Shape.

Questo può essere fatto, componendo un istanza di TextView all'interno di TextShape e implementando TextShape in termini d'interfaccia

Motivazione



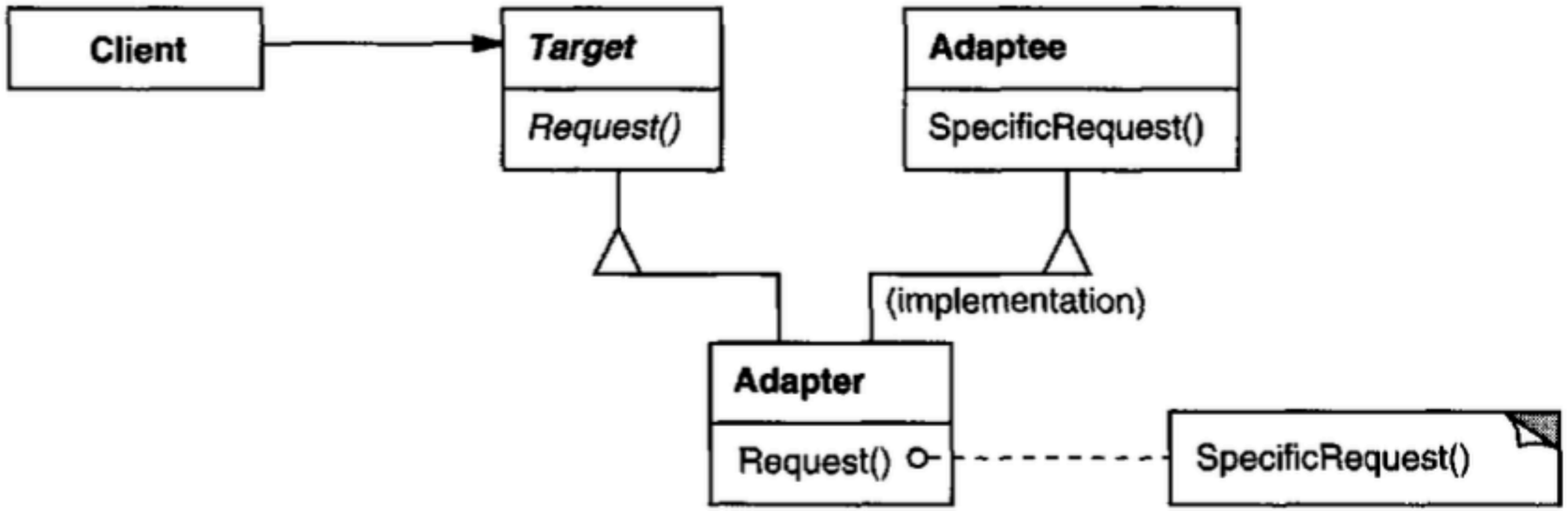
Quando usare l’Adapter?

Si desidera utilizzare una classe esistente e la sua interfaccia non corrisponde a quella necessaria.

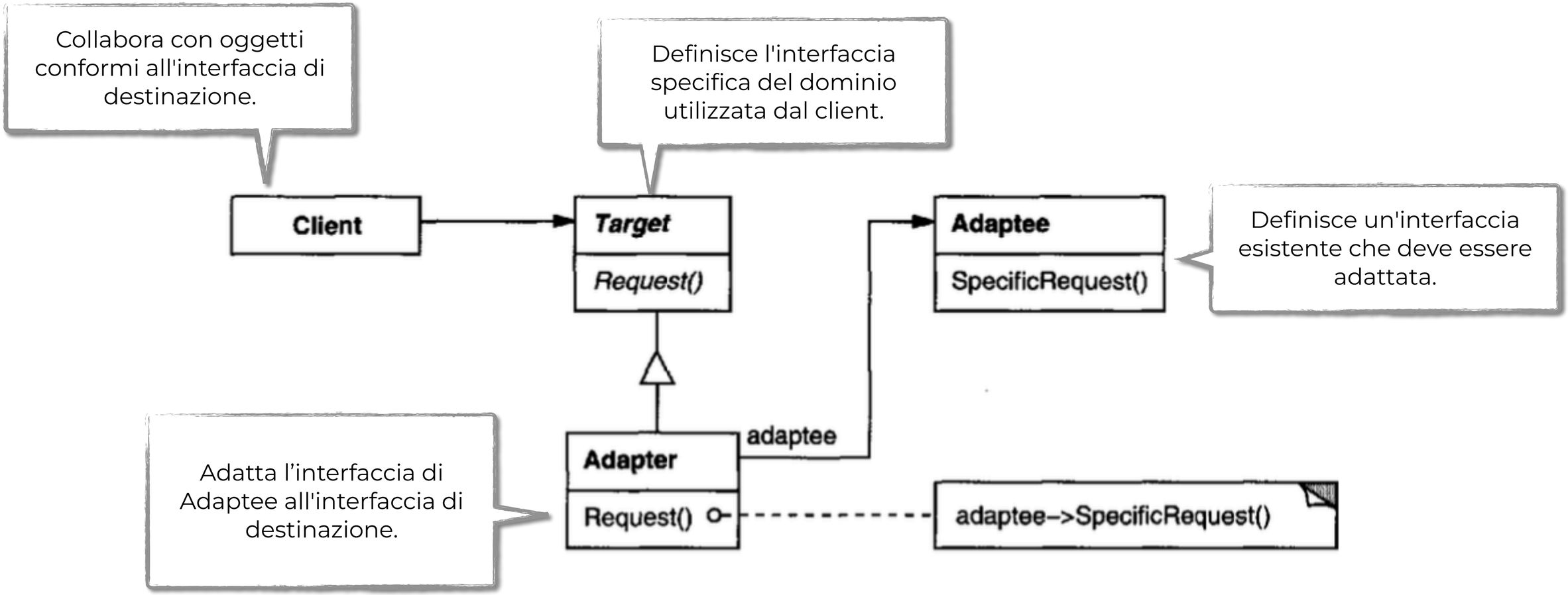
Vuoi creare una classe utilizzabile che coopera con classi non correlate o non viste, cioè classi che non hanno necessariamente interfacce compatibili.

È necessario utilizzare diverse sottoclassi esistenti, ma non è pratico adattare la loro interfaccia sottoclassando tutte. Un adattatore per oggetti può adattare l'interfaccia della sua classe padre.

Struttura - Class Adapter



Struttura - Object Adapter



Conseguenze

PRO

- Principio di responsabilità unica
- Principio aperto/chiuso

CONTRO

EMPTY

Conseguenze

PRO

- Principio di responsabilità unica
- Principio aperto/chiuso

CONTRO

Può separare l'interfaccia o il codice di conversione dei dati dalla logica di bussines primaria del programma.

Conseguenze

PRO

Principio di responsabilità unica

Principio aperto/chiuso

CONTRO

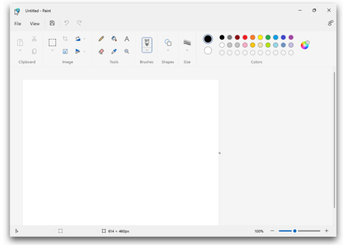
È possibile introdurre nuovi tipi di adapters nel programma senza rompere il codice client esistente, purché funzionino con l’ adapters attraverso l'interfaccia client.

Intent

Converte l'interfaccia di una classe in un'altra interfaccia che i client si aspettano, in modo tale che possano lavorare insieme.

Problema

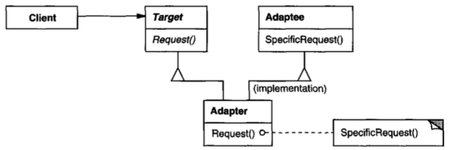
Immaginiamo di dover sviluppare un editor grafico, che consente agli utenti di disegnare e disporre elementi grafici come linee, poligoni, testo ecc.



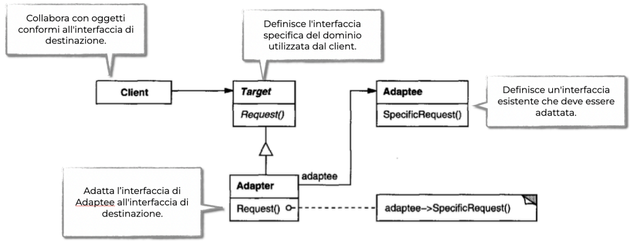
Problema

Per quanto riguarda gli elementi grafici, possiamo pensare di realizzare l'astrazione tramite una classe astratta Shape e poi sviluppare per ognuno di essi la propria sottoclasse.

Struttura - Class Adapter



Struttura - Object Adapter



Conseguenze

PRO

- Principio di responsabilità unica
- Principio aperto/chiuso

CONTRO

EMPTY

Adapter