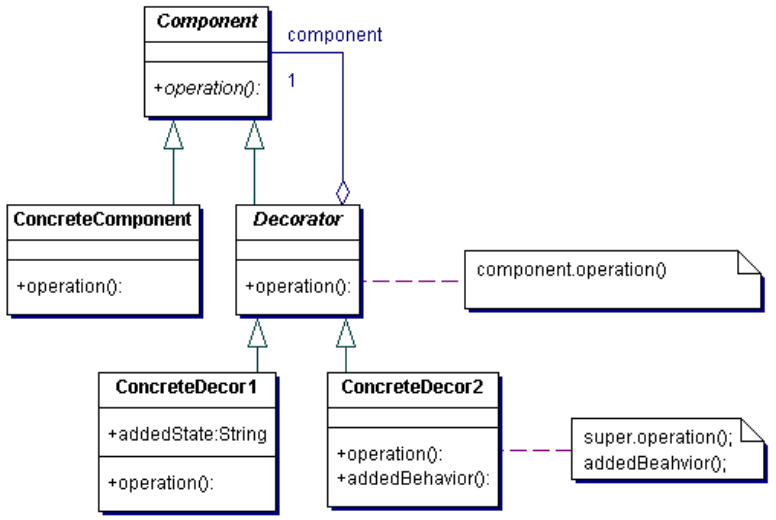
Tecniche di Progettazione: Design Patterns

Esercitazione

Re-write the following code using Decorator pattern.

```
class A {f(){print("A");};}
class AB extends A {f(){super.f(); print("B");}}
class AC extends A {f(){super.f(); print("C");}}
class  ABC extends A {f(){super.f(); print("B"); print("C");}}
class CBA extends A {f(){print("C"); print("B"); super.f()}}
   ….
   A a;
      a = new A(); a.f();
      a = new AB(); a.f();
      a = new AC(); a.f();
      a = new ABC(); a.f();
      a = new CBA();  a.f();
```

Decorator: struttura



Solution

class Comp{ *f()* }
class Nullcomp extends Comp {f() \\ a capo}
class Dec extends Comp {Comp c; *f()*}
Class Adec extends Dec { f(){c.f(); print("A");}}
Class Bdec extends Dec { f(){c.f(); print("B");}}
Class Cdec extends Dec { f(){c.f(); print("C");}}

….
Comp a;
a = new Adec(new Nullcomp()); a.f();
a = new Bdec (new Adec(new Nullcomp())); a.f();
a = new Cdec (new Adec(new Nullcomp())); a.f();
a = new Cdec(new Bdec (new Adec(new Nullcomp()))); a.f();
a = new Adec(new Bdec (new Cdec(new Nullcomp()))); a.f();

Adapter

Given the following Stack class and Set interface

a) Write a class adapter to adapt Stack to Set
b) Write an object adapter to adapt Stack to Set

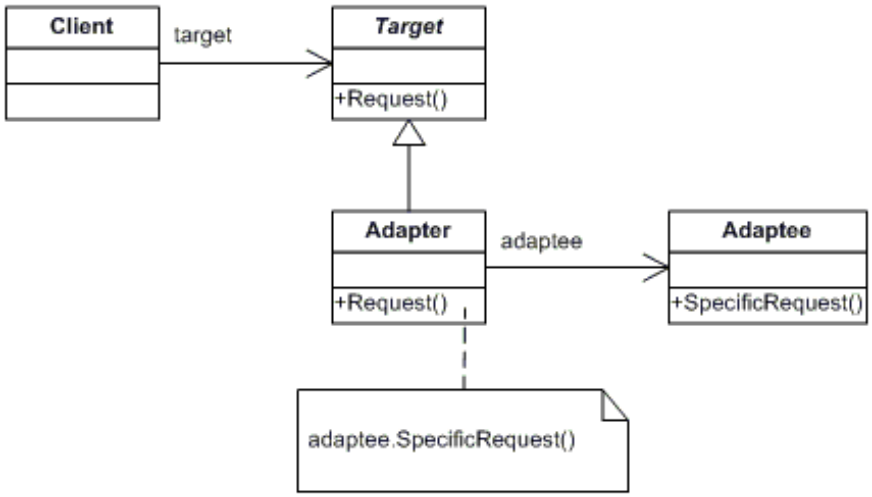
```
interface Set {
   public int size();
   public boolean isEmpty();
   public boolean member(int x); //check if x is a member of the set
   public void remove(int x); //remove x from the set if x is a member of the
  set
   }
```


Adapter cont'd

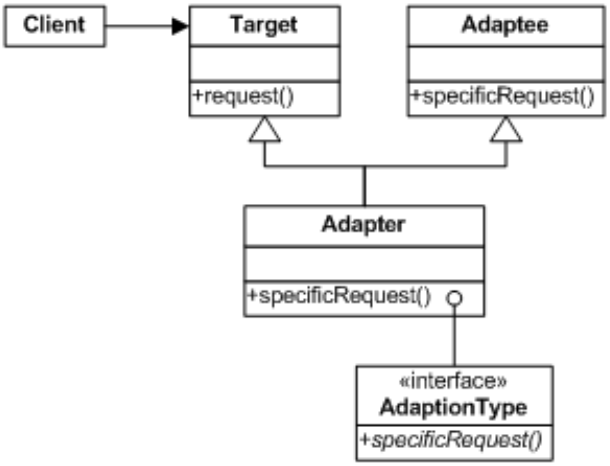
```
class Stack {
  private int[] contents;
  private int top = 0;
  public Stack(int maxSize) {contents = new int[maxSize];}
  public int size() {return top;}
  public boolean isEmpty() {return top = = 0;}
  public int top() {return contents[top-1];}
  public void pop() {top--;}
  public void push(int value) {contents[top++] = value;}
}
```

Note that in the adapters, you are not allowed to use any data structure other than stack, such as array and Vector.

An object adapter relies on object composition



A class adapter uses multiple inheritance to adapt one interface to another



- Soluzione: cartella AdapterSetStack