

Ministerul Educației al Republicii Moldova  
Universitatea Tehnică a Moldovei  
Facultatea Calculatoare, Informatica și Microelectronică  
Catedra Automatica și Tehnologii Informaționale

# Raport

Lucrarea de laborator nr. 2

La disciplina: Ingineria Produselor Program

Tema: Șabloane structurale

A efectuat: st.gr.TI-143 Cornita Constantin

A verifica: lector asistent, Chetrusca Ecaterina

Chișinău 2016

## Scopul și sarcina

De studiat 5 șabloane structurale și de implementat în proiect propriu.

## Noțiuni teoretice

Sabloanele structurale studiază modelele cu privire la structura sistemului bazat pe clase și obiecte.

În acest caz, pot fi folosite următoarele mecanisme:

- Moștenire, atunci când clasa de bază definește interfața iar subclasele – realizarea. Structuri obținute în baza moștenirii se primnesc statice.
- Compoziția, atunci când structurile sunt construite prin combinarea mai multor clase de obiecte. Compoziția produce structuri care pot fi schimbate în timpul.

## Adaptorul (Adapter)

Acest șablon convertește interfața unei clase în altă interfață pe care o așteaptă clientul. Adaptorul permite să funcționeze împreună clase care altfel nu ar putea din cauza interfețelor incompatibile.

**Adaptorul** permite utilizarea unor clase fără a modifica nici codul client nici codul țintă.

Diagrama de clasa pentru Adapter Pattern

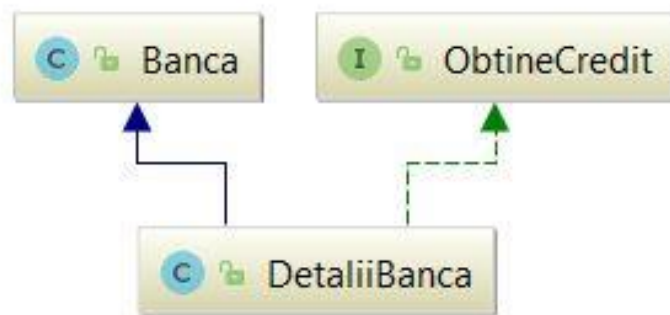


Fig.1 Adapter Patern

## Composite Pattern

Acest șablon compune obiecte în structuri de tip arbore pentru a reprezenta ierarhii parte-întreg. Șablonul permite clienților să trateze uniform obiecte individuale și compuneri de obiecte.

Pot fi ignorate diferențele dintre obiectele compuse și obiectele elementare.

Composite Pattern are 2 responsabilități:

- Gestionarea ierarhiei
- Executarea operațiilor legate de meniuri

Compromis pentru creșterea transparenței

Șablonul Composite este uneori utilizat împreună cu șablonul Decorator

Când se utilizează împreună Decoratori și Compuneri, ei vor avea de obicei o clasă părinte comună.

Decoratorii vor trebui să implementeze interfața Component cu operații precum Add, Remove și GetChild.

Diagrama de clasa pentru Composite Pattern

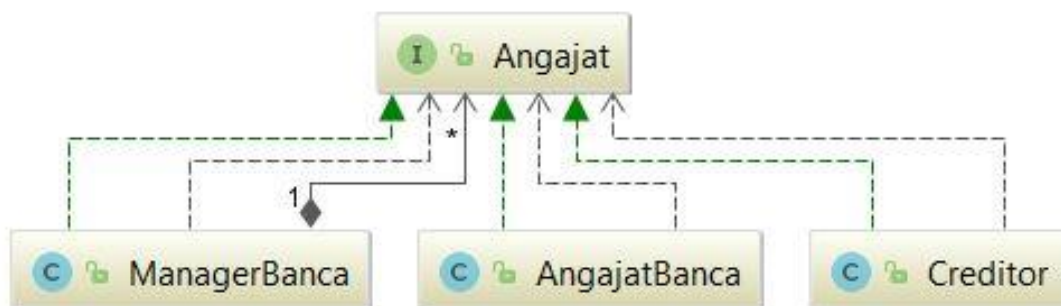


Fig.2 Composite Pattern

## Decorator Pattern

Sablonul Decorator atașează responsabilități suplimentare unui obiect în mod dinamic. Decoratorii asigură o alternativă flexibilă la derivare pentru extinderea funcționalității.

Decoratorii au același tip ca și obiectele decorate. Se poate de trimis ca parametru un obiect decorat în locul obiectului inițial.

Decoratorul folosește moștenirea pentru a prelua tipul obiectului decorat și compunerea pentru a-i schimba comportamentul. Se pot utiliza unul sau mai mulți decoratori pentru un obiect. Obiectele pot fi decorate dinamic în momentul execuției. Decoratorul își poate adăuga propriul comportament înainte sau după delegare. Decoratorul permite adăugarea de noi funcționalități fără modificări în cod. Decoratorii pot fi creați cu șabloanele Fabrică și Constructor

Digrama de clase pentru șablonul Decorator Pattern

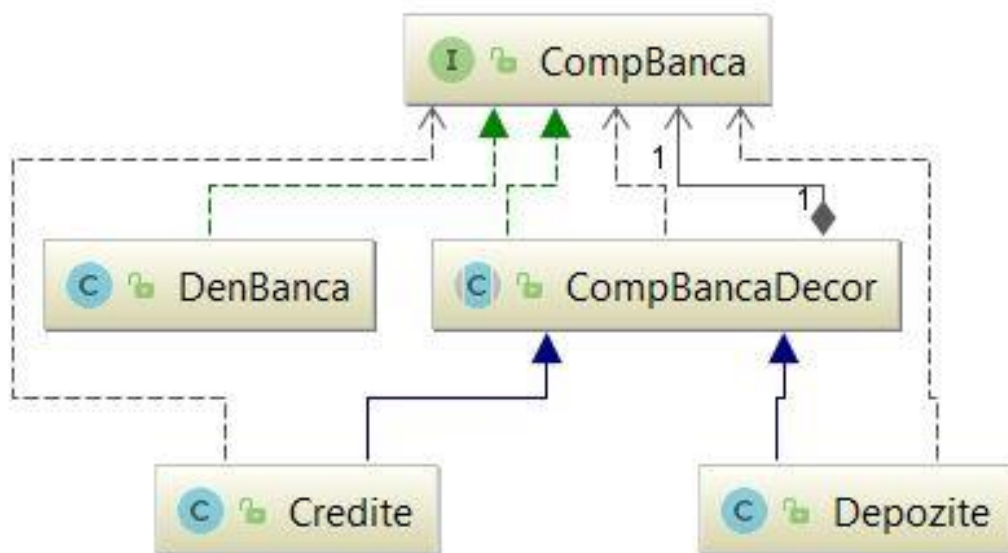


Fig. 3 Decorator Pattern

## Facade Pattern

**Fațada (Façade)** - asigură o interfață unificată la o mulțime de interfețe dintr-un subsistem. Fațada definește o interfață de nivel mai înalt care face subsistemul mai ușor de utilizat.

Fațada nu numai că simplifică o interfață, ci și decuplează clientul de subsistemul de componente  
- Subsistemul poate fi de asemenea accesat direct, fațada nu încapsulează subsistemul.

Atât **Fațada** cât și **Adaptorul** pot împacheta (pot fi wrappere pentru) una sau mai multe clase

- Scopul adaptorului este să convertească interfața;
- Scopul fațadei este să simplifice interfața.

Diagrama de clasă pentru fațada Pattern

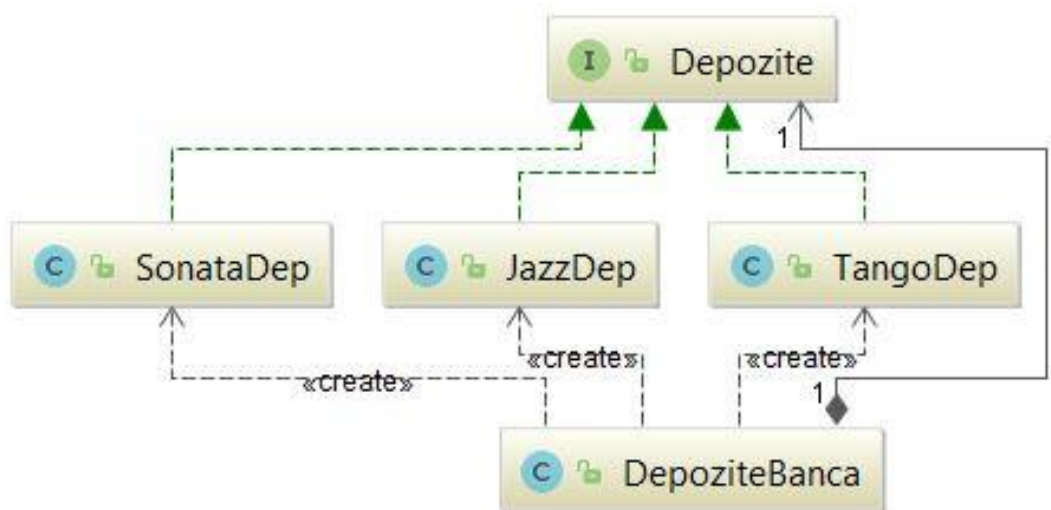


Fig.4 Facade Pattern

## Bridge Pattern

Sablonul Bridge are o structura similara unui obiect adaptor, dar are un cu totul alt scop: el este destinat separarii unei interfete de implementarea ei, astfel incat cele doua sa poata fi modificate independent, cu usurinta. Scopul unui adaptor este sa schimbe interfata unui obiect existent.

Sablonul Proxy defineste un reprezentant sau un surogat pentru un alt obiect si nu-i modifica interfata.

Decupleaza o abstractizare de implementarea ei, astfel incat cele doua sa poata varia independent.  
Aplicabilitate

Sablonul Bridge se utilizeaza cand:

- dorim sa evitam o legatura permanenta intreo abstractizare si implementarea acesteia. Astfel de cazuri ar putea fi, de exemplu, cand implementarea trebuie selectata sau schimbata a executie.
- atat abstractizarile, cat si implementarile acestora trebuie sa fie extensibile prin generarea subclaselor. In acest caz, sablonul Bridge, permite sa combinam abstractizari si implementari diferite si sa le extindem imediat.
- modificarile in implementarea unei abstractizari nu trebuie sa afecteze clientii; cu alte cuvinte, codul client nu trebuie sa fie recompilat.
- 

Digrama de clasa pentru Bridge Pattern

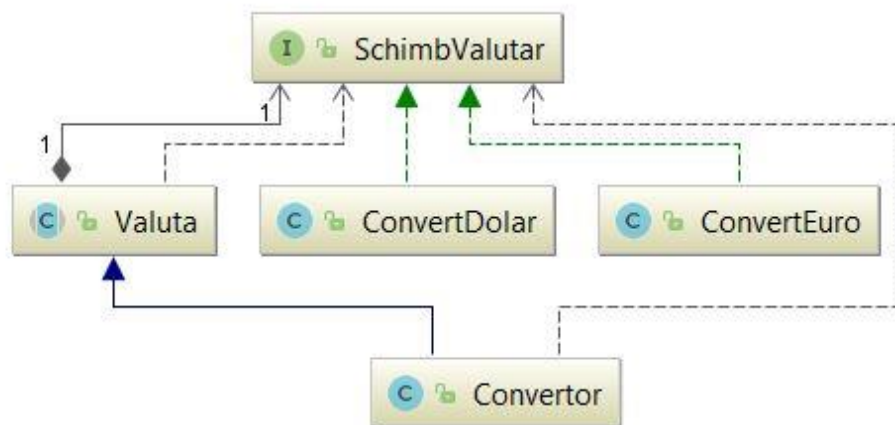
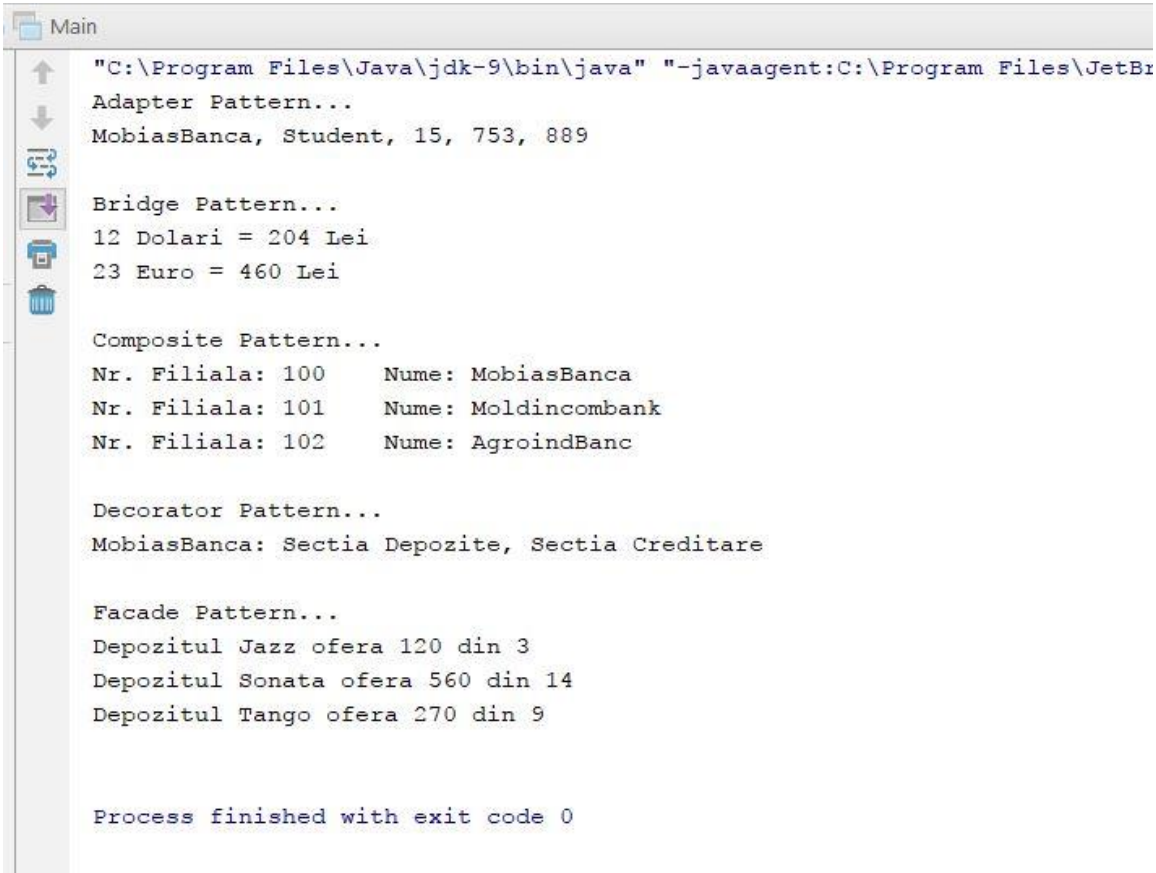


Fig. 5 Bridge Pattern

## Rezultatul programului



```
"C:\Program Files\Java\jdk-9\bin\java" "-javaagent:C:\Program Files\JetBr
Adapter Pattern...
MobiasBanca, Student, 15, 753, 889

Bridge Pattern...
12 Dolari = 204 Lei
23 Euro = 460 Lei

Composite Pattern...
Nr. Filiala: 100      Nume: MobiasBanca
Nr. Filiala: 101     Nume: Moldincombank
Nr. Filiala: 102     Nume: AgroindBanc

Decorator Pattern...
MobiasBanca: Sectia Depozite, Sectia Creditare

Facade Pattern...
Depozitul Jazz ofera 120 din 3
Depozitul Sonata ofera 560 din 14
Depozitul Tango ofera 270 din 9

Process finished with exit code 0
```

Fig.6 Rezultat

## Concluzie

Sabloanele structural se ocupa de modul in care sunt compuse clasele si obiectele pentru a forma structuri mai complexe. Sabloanele structurale de clasa folosesc mostenirea pentru a compune interfete sau implementari. Acest tip de sablon este in special util pentru a face sa functioneze impreuna bibliotecile intre clase dezvoltate independent. Un exemplu este forma de clasa Adapter. In general, un adaptor realizeaza o interfata conforma cu o alta, furnizand astfel o abstractizare uniforma pentru interfete diferite. Un adaptor de clasa realizeaza acest lucru prin mostenirea privata de la clasa adaptat. Apoi, adaptorul exprima interfata sa in termenii adaptatului.

.



**Anexa A:**

Repoztorul distant: <https://github.com/CornitaConstantin/Lab2IPP>