**Ministerul Educaţiei al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică şi Microelectronică**

**Catedra Automatică şi Tehnologii Informaţionale**

**RAPORT**

Lucrare de laborator nr 2

Disciplina: Ingineria produselor program

Tema: „Șabloane de proiectare structurale”

**A efectuat:**

Vovc Artemie st. TI-133

**A verificat:**

Eugenia Latu lect. asist.

**Chișinău 2016**

Cuprins

[1 Sarcina 3](#_Toc462488303)

[2 Șabloane de proiectare structurale implementare 4](#_Toc462488304)

[2.1 Adaptorul 4](#_Toc462488305)

[2.2 Fațada 4](#_Toc462488306)

[2.3 Punte 5](#_Toc462488307)

[2.4 Compoziție 6](#_Toc462488308)

[2.5 Proxy 7](#_Toc462488309)

[Concluzia 9](#_Toc462488310)

[Bibliografia 10](#_Toc462488311)

[Anexe A 11](#_Toc462488312)

[Anexa B 12](#_Toc462488313)

[Anexa C 13](#_Toc462488314)

# 1 Sarcina

De creat un program în care vor interacționa cinci șabloane de proiectare structurale.

# 2 Șabloane de proiectare structurale implementare

Şabloanele structurale ale claselor descriu modul de utilizare a moştenirii în scopul compunerii claselor.

# 2.1 Adaptorul

Şablonul Adaptor converteşte interfaţa unei clase în altă interfaţă pe care o aşteaptă clientul. Adaptorul permite să funcţioneze împreună clase care altfel nu ar putea din cauza interfeţelor incompatibile.

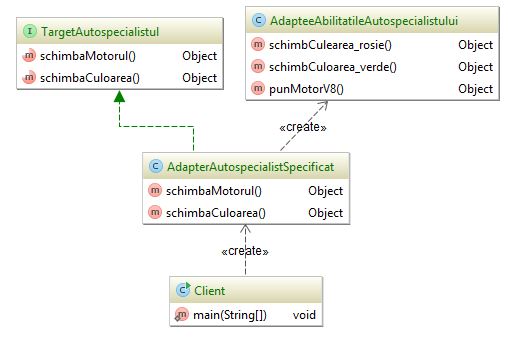


Figura 2.1 - adaptorul

# 2.2 Fațada

Şablonul Faţadă asigură o interfaţă unificată la o mulţime de interfeţe dintr-un subsistem. Faţada defineşte o interfaţă de nivel mai înalt care face subsistemul mai uşor de utilizat.

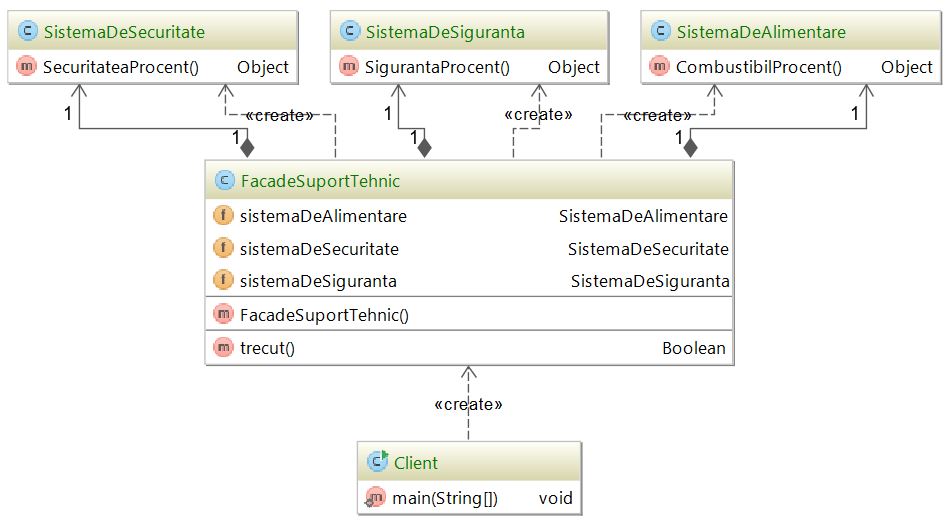


Figura 2.2 – fațada

Faţada nu numai că simplifică o interfaţă, ci şi decuplează clientul de subsistemul de componente

-subsistemul poate fi de asemenea accesat direct, faţada nu încapsulează subsistemul

Atât Faţada cât şi Adaptorul pot împacheta (pot fi wrappere pentru) una sau mai multe clase

-scopul Adaptorului este să convertească interfaţa

-scopul Faţadei este să simplifice interfaţa

# 2.3 Punte

Şablonul Punte decuplează o abstracţiune de implementarea ei, astfel încât cele două să poată varia în mod independent

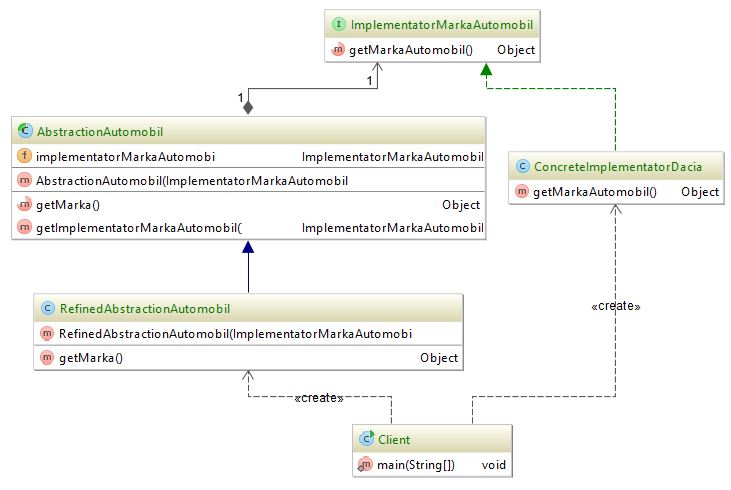


Figura 2.3 – punte

# 2.4 Compoziție

Realizeaza compunerea obiectelor intr-o structura de arbore, pentru a reprezenta ierarhii de tipul 'intreg-parti componente'. Da clientilor posibilitatea de a trata in mod uniform atat obiectele individuale, cat si structurile compuse de obiecte.

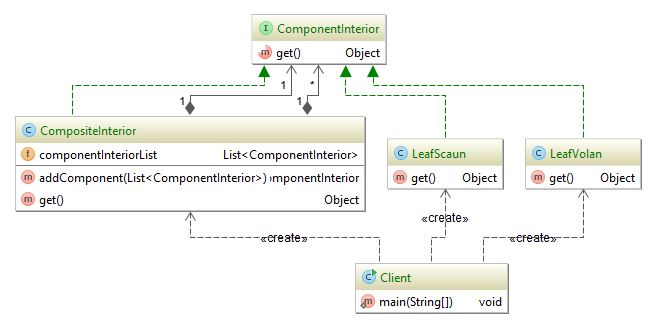


Figura 2.4 – compoziție

# 2.5 Proxy

Un proxy , în forma sa cea mai generală , este o clasă funcțională ca o interfață la altceva . Proxy-ul ar putea interfață la orice: o conexiune de rețea , un obiect mare în memorie , un fișier sau o altă resursă care este costisitoare sau imposibil de duplicat .

Pe scurt , un proxy este un obiect înveliș sau un agent care este numit de către client pentru a avea acces la obiectul de servire reală din spatele scenei . Folosirea proxy poate fi pur și simplu redirecționarea către obiectul real , sau poate furniza logica suplimentară .

În proxy poate fi furnizată funcționalități suplimentare , de exemplu, cache-ul atunci când operațiunile efectuate pe obiectul real sunt intensive de resurse , sau verificarea precondiții înainte de a se invocă operații asupra obiectului real . Pentru client , utilizarea unui obiect proxy este similară cu utilizarea obiectului real , deoarece ambele pune în aplicare aceeași interfață .

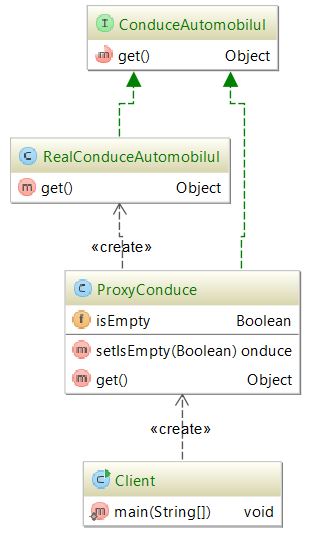


Figura 2.5 – proxy

# Concluzia

Lucrarea dată a avut ca scop să ne facă cunoscuți cu șabloanele de proiectare structurale, care sunt des utilizate în industria de dezvoltare a aplicațiilor. Am studiat cinci șabloane de proiectare structurale: adaptor composite, proxy, bridge, facade. Fiecare tip de șablon are ajunsurile și neajunsurile. La general șabloanele studiate rezolvă unele probleme structurale ale unui sistem informațional la nivel de implementare.

# Bibliografia

1. Adaptor. [Resursa electronică].-regim de acces:

https://en.wikipedia.org/wiki/Adapter\_pattern

1. Composite. [Resursa electronică].-regim de acces:

https://en.wikipedia.org/wiki/Composite\_pattern

1. Bridge. [Resursa electronică].-regim de acces:

https://en.wikipedia.org/wiki/Bridge\_pattern

1. Facade. [Resursa electronică].-regim de acces:

https://en.wikipedia.org/wiki/Facade\_pattern

1. Proxy. [Resursa electronică].-regim de acces:

https://en.wikipedia.org/wiki/Proxy\_pattern

## Anexe A

**Client**

**package** com.lab2ipp;  
  
**import** com.lab2ipp.adapter.AdapterAutospecialistSpecificat;  
**import** com.lab2ipp.adapter.TargetAutospecialistul;  
**import** com.lab2ipp.bridge.AbstractionAutomobil;  
**import** com.lab2ipp.bridge.ConcreteImplementatorDacia;  
**import** com.lab2ipp.bridge.RefinedAbstractionAutomobil;  
**import** com.lab2ipp.composite.ComponentInterior;  
**import** com.lab2ipp.composite.CompositeInterior;  
**import** com.lab2ipp.composite.LeafScaun;  
**import** com.lab2ipp.composite.LeafVolan;  
**import** com.lab2ipp.facade.FacadeSuportTehnic;  
**import** com.lab2ipp.proxy.ConduceAutomobilul;  
**import** com.lab2ipp.proxy.ProxyConduce;  
  
**import** java.util.Arrays;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** Client {  
 **public static void** main(String[] args) {  
 */\*\*  
 \* ADAPTER  
 \*/* TargetAutospecialistul targetAutospecialistul = **new** AdapterAutospecialistSpecificat();  
 System.***out***.println(**"Motorul: "** + targetAutospecialistul.schimbaMotorul() + **" Culoarea: "** + targetAutospecialistul.schimbaCuloarea());  
 */\*\*  
 \* BRIDGE  
 \*/* AbstractionAutomobil abstractionAutomobil = **new** RefinedAbstractionAutomobil(**new** ConcreteImplementatorDacia());  
 System.***out***.println(**"Automobilul: "** + abstractionAutomobil.getMarka());  
 */\*\*  
 \* COMPOSITE  
 \*/* ComponentInterior componentInterior = **new** CompositeInterior().addComponent(Arrays.*asList*(**new** LeafVolan(), **new** LeafScaun()));  
 System.***out***.println(componentInterior.get());  
 */\*\*  
 \* FACADE  
 \*/* FacadeSuportTehnic facadeSuportTehnic = **new** FacadeSuportTehnic();  
 **if**(facadeSuportTehnic.trecut()) System.***out***.println(**"Automobilul a trecut suportul tehnic."**);  
 */\*\*  
 \* PROXY  
 \*/* ConduceAutomobilul conduceAutomobilul = **new** ProxyConduce().setIsEmpty(**true**);  
 System.***out***.println(**"Conduc automobilul: "** + conduceAutomobilul.get());  
  
 }  
}

## Anexa B

**Interfețele**

**package** com.lab2ipp.adapter;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public interface** TargetAutospecialistul {  
 Object schimbaMotorul();  
 Object schimbaCuloarea();  
}

**package** com.lab2ipp.bridge;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public interface** ImplementatorMarkaAutomobil {  
 Object getMarkaAutomobil();  
}

**package** com.lab2ipp.composite;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public interface** ComponentInterior {  
 Object get();  
}

**package** com.lab2ipp.proxy;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public interface** ConduceAutomobilul {  
 Object get();  
}

## Anexa C

**Clase**

**package** com.lab2ipp.adapter;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** AdapteeAbilitatileAutospecialistului {  
 **public** Object schimbCulearea\_rosie(){  
 **return "rosie"**;  
 }  
 **public** Object schimbCuloarea\_verde(){  
 **return "verde"**;  
 }  
 **public** Object punMotorV8(){  
 **return "V8"**;  
 }  
}

**package** com.lab2ipp.adapter;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** AdapterAutospecialistSpecificat **implements** TargetAutospecialistul {  
 **public** Object schimbaMotorul() {  
 **return new** AdapteeAbilitatileAutospecialistului().punMotorV8();  
 }  
  
 **public** Object schimbaCuloarea() {  
 **return new** AdapteeAbilitatileAutospecialistului().schimbCulearea\_rosie();  
 }  
}

**package** com.lab2ipp.bridge;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public abstract class** AbstractionAutomobil {  
 **private** ImplementatorMarkaAutomobil **implementatorMarkaAutomobil**;  
  
 **public** AbstractionAutomobil(ImplementatorMarkaAutomobil implementatorMarkaAutomobil) {  
 **this**.**implementatorMarkaAutomobil** = implementatorMarkaAutomobil;  
 }  
  
 **public abstract** Object getMarka();  
  
 **public** ImplementatorMarkaAutomobil getImplementatorMarkaAutomobil(){**return this**.**implementatorMarkaAutomobil**;}  
}

**package** com.lab2ipp.bridge;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** ConcreteImplementatorDacia **implements** ImplementatorMarkaAutomobil {  
 **public** Object getMarkaAutomobil() {  
 **return "Dacia"**;  
 }  
}

**package** com.lab2ipp.bridge;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** RefinedAbstractionAutomobil **extends** AbstractionAutomobil {  
  
 **public** RefinedAbstractionAutomobil(ImplementatorMarkaAutomobil implementatorMarkaAutomobil) {  
 **super**(implementatorMarkaAutomobil);  
 }  
  
 **public** Object getMarka() {  
 **return this**.getImplementatorMarkaAutomobil().getMarkaAutomobil();  
 }  
}

**package** com.lab2ipp.composite;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** CompositeInterior **implements** ComponentInterior {  
  
 **private** List<ComponentInterior> **componentInteriorList** = **new** ArrayList<ComponentInterior>();  
  
 **public** ComponentInterior addComponent(List<ComponentInterior> componentInteriorList){  
 **this**.**componentInteriorList**.addAll(componentInteriorList);  
 **return this**;  
 }  
  
 **public** Object get() {  
 System.***out***.print(**"Componentele interiorului: "**);  
 **this**.**componentInteriorList**.forEach(componentInterior -> System.***out***.print(componentInterior.get()+**" "**));  
 **return ""**;  
 }  
}

**package** com.lab2ipp.composite;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** LeafScaun **implements** ComponentInterior {  
 **public** Object get() {  
 **return "scaun"**;  
 }  
}

**package** com.lab2ipp.composite;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** LeafVolan **implements** ComponentInterior{  
  
 **public** Object get() {  
 **return "volan"**;  
 }  
}

**package** com.lab2ipp.facade;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** FacadeSuportTehnic {  
 **private** SistemaDeAlimentare **sistemaDeAlimentare**;  
 **private** SistemaDeSecuritate **sistemaDeSecuritate**;  
 **private** SistemaDeSiguranta **sistemaDeSiguranta**;  
  
 **public** FacadeSuportTehnic(){  
 **sistemaDeAlimentare** = **new** SistemaDeAlimentare();  
 **sistemaDeSecuritate** = **new** SistemaDeSecuritate();  
 **sistemaDeSiguranta** = **new** SistemaDeSiguranta();  
 }  
  
 **public** Boolean trecut(){  
 **if**( **sistemaDeAlimentare**.CombustibilProcent().equals(**"48%"**) &&  
 **sistemaDeSiguranta**.SigurantaProcent().equals(**"80%"**) &&  
 **sistemaDeSecuritate**.SecuritateaProcent().equals(**"75%"**)  
 ) **return true**;  
 **return false**;  
 }  
}

**package** com.lab2ipp.facade;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** SistemaDeAlimentare {  
 **public** Object CombustibilProcent(){**return "48%"**;}  
}

**package** com.lab2ipp.facade;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** SistemaDeSecuritate {  
  
 **public** Object SecuritateaProcent(){**return "75%"**;}  
  
}

**package** com.lab2ipp.facade;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** SistemaDeSiguranta {  
  
 **public** Object SigurantaProcent(){**return "80%"**;}  
  
}

**package** com.lab2ipp.proxy;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** ProxyConduce **implements** ConduceAutomobilul {  
 **private** Boolean isEmpty = **false**;  
  
 **public** ProxyConduce setIsEmpty(Boolean isEmpty){**this**.isEmpty = isEmpty;**return this**;}  
  
 @Override  
 **public** Object get() {  
 **if**(isEmpty) **return "nu poti conduce, nu ai motorina"**;  
 **return new** RealConduceAutomobilul().get();  
 }  
}

**package** com.lab2ipp.proxy;  
  
*/\*\*  
 \* Created by Artemie on 24.09.2016.  
 \*/***public class** RealConduceAutomobilul **implements** ConduceAutomobilul {  
 @Override  
 **public** Object get() {  
 **return "conduc"**;  
 }  
}