Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**Lucrare de laborator nr.3**

*Disciplina: Tehnici și mecanisme de proiectare software*

**Tema:** Șabloane de proiectare comportamentale



**Studentul: gr. TI-206 Raileanu Daniel**

**Coordonator: asist. univ. Buldumac Oleg**

**Chișinău 2023**

**Sarcina:** Să se implementeze la nivel de cod 5 șabloane de proiectare comportamentale din cele 11.

Sabloanele comportamentale reprezintă un set de soluții de proiectare care se concentrează pe gestionarea comportamentului și interacțiunii între obiecte. Putem separa responsabilitățile și comportamentele diferitelor obiecte, promovând astfel coeziunea și cuplarea slabă în cadrul aplicațiilor.

**Chain of Responsibility**

Acest sablon permite transmiterea unei solicitări de la un obiect la altul într-o succesiune predefinită până când un obiect din lanț o gestionează. Fiecare obiect din lanț are posibilitatea de a trata solicitarea sau de a o transmite înainte.

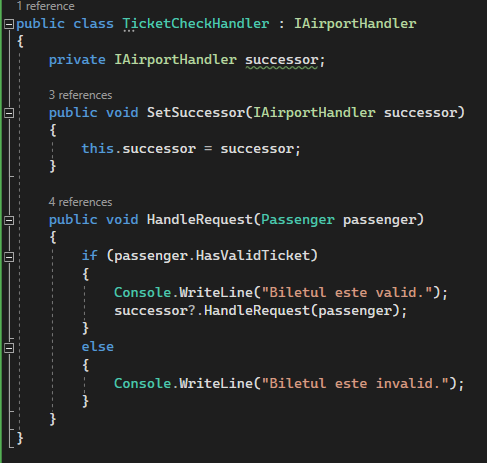


Figura 1 – Implementarea sablonului Chain of Responsability

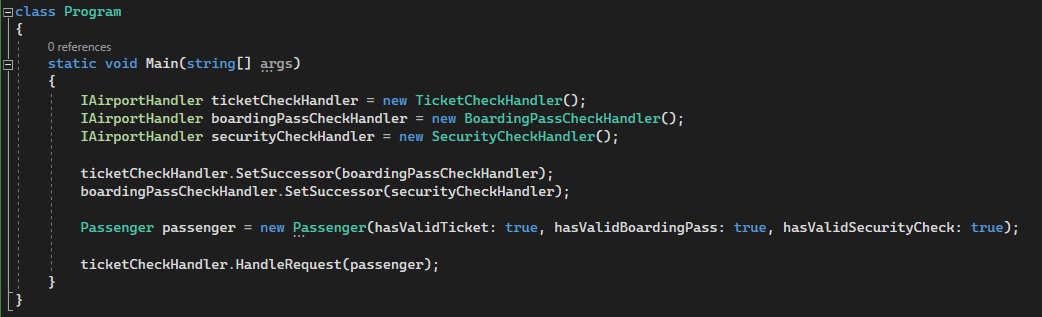


Figura 2 – Crearea/primirea instanței

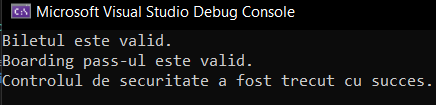


Figura 3 – Rezultatele in urma executarii programului

**Command**

Sablonul Command transformă o solicitare într-un obiect independent care conține toate informațiile necesare pentru a executa acea solicitare. Acesta separă solicitarea de executarea efectivă, permițând manipularea și gestionarea ulterioară a comenzilor. Astfel, permite crearea de operațiuni complexe, invocarea acestora într-o ordine specifică și posibilitatea de anulare a comenzilor efectuate.



Figura 4 – Implementarea sablonului Command

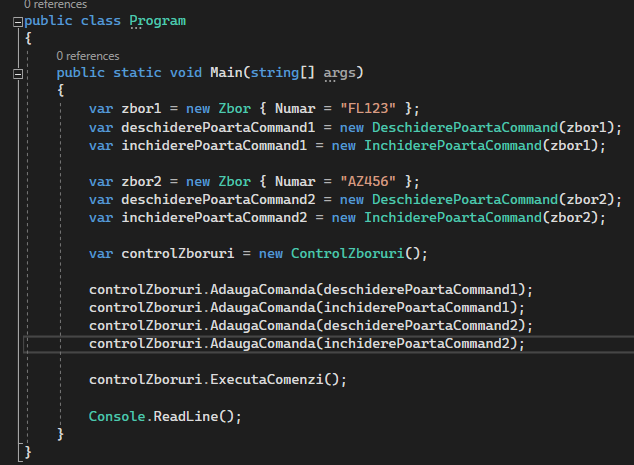


Figura 5 – Crearea/primirea instanței

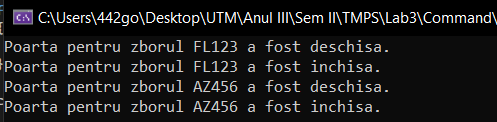


Figura 6 – Rezultatele in urma executarii programului

**Mediator**

Sablonul Mediator promovează comunicarea între obiecte prin intermediul unui obiect mediator centralizat, eliminând astfel dependențele directe între obiecte. Mediatorul permite interacțiunea în mod slab cuplat între obiecte și facilitează extensibilitatea sistemului prin adăugarea sau modificarea mediatorilor existenți.



Figura 7 – Implementarea sablonului Mediator

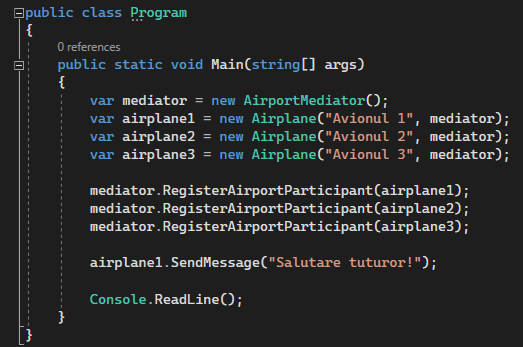


Figura 8 – Crearea/primirea instanței

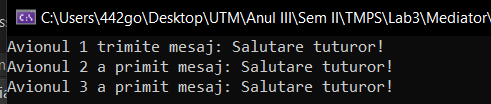


Figura 9 – Rezultatele in urma executarii programului

**Strategy**

Acest sablon permite definirea unei familii de algoritmi, încapsularea fiecărui algoritm într-un obiect separat și permițându-le să fie interschimbabile. Astfel, clientul poate alege un algoritm din familia specificată și îl poate schimba în timpul execuției. Aceasta promovează flexibilitatea, modularitatea și extensibilitatea codului.

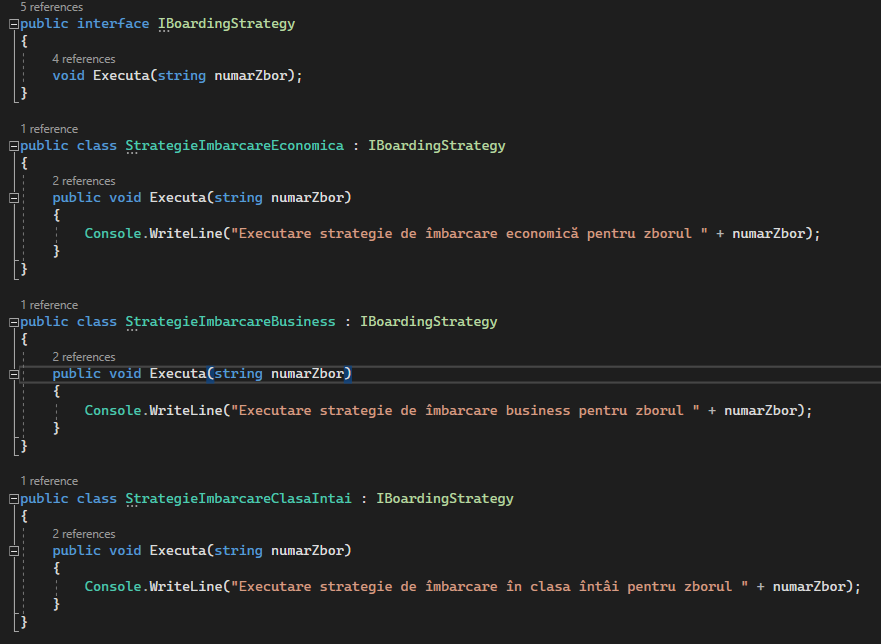


Figura 10 – Implementarea sablonului Strategy

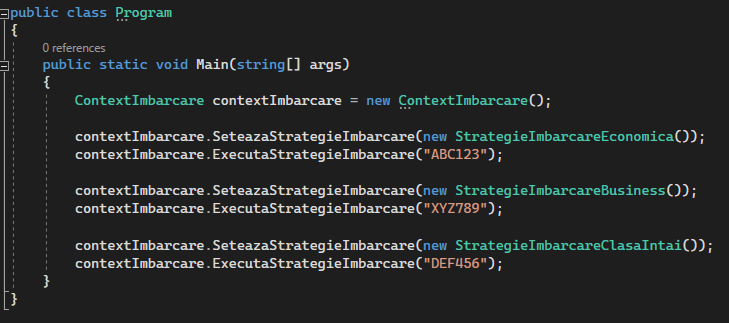


Figura 11 – Crearea/primirea instanței

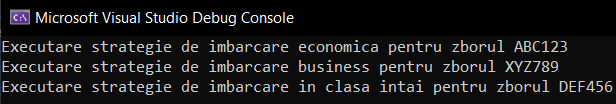


Figura 12 – Rezultatele in urma executarii programului

**Template Method**

Sablonul Template Method definesc un schelet algoritmic într-o clasă de bază, lăsând implementarea anumitor pași specifici derivatei. Clasa de bază stabilește structura generală a algoritmului, dar lasă derivatele să-și implementeze pașii specifici. Astfel, permite refolosirea codului și flexibilitatea în implementarea diferitelor variante ale algoritmului.

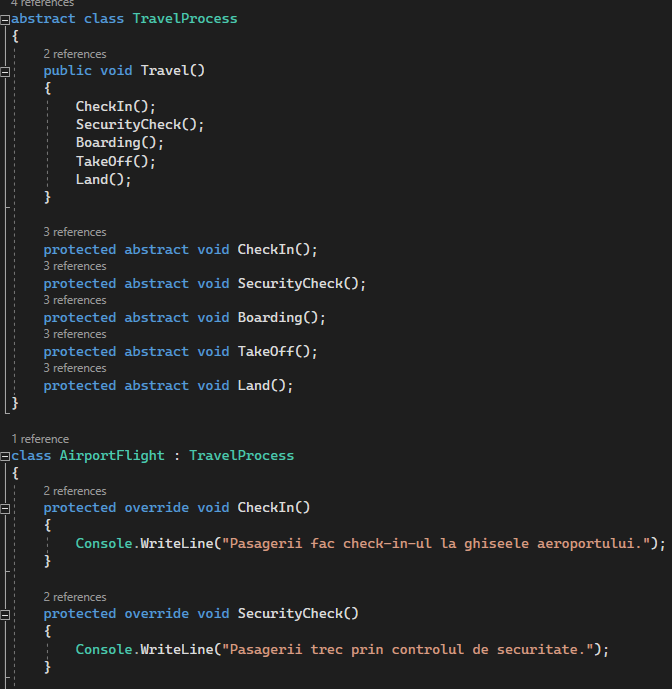


Figura 13 – Implementarea sablonului Template Method

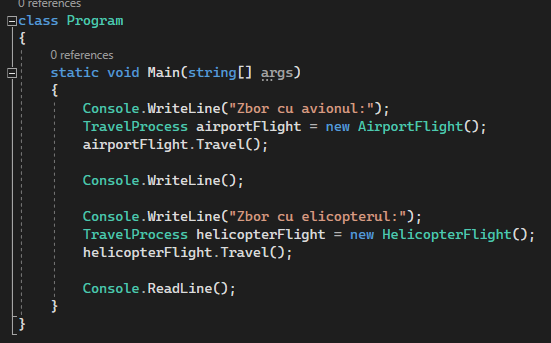


Figura 14 – Crearea/primirea instanței

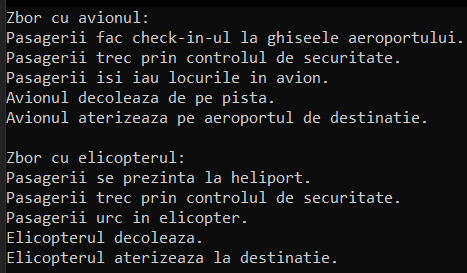


Figura 15 – Rezultatele in urma executarii programului

**Concluzie**

În urma efectuării lucrării de laborator am învățat despre sabloanele comportamentale scopul lor în programare. Sabloanele comportamentale reprezintă un set de soluții de proiectare care se concentrează pe gestionarea comportamentului și interacțiunii între obiecte într-un sistem software. Acestea facilitează structurarea și organizarea codului într-un mod flexibil, modular și ușor de înțeles, permițând schimbări și extensii ulterioare cu efort redus.

Chain of responsibility permite transmiterea unei solicitări de la un obiect la altul într-o succesiune predefinită până când un obiect din lanț o gestionează. Fiecare obiect din lanț are posibilitatea de a trata solicitarea sau de a o transmite înainte. Este util în situații în care mai mulți obiecte pot trata o solicitare și ordinea de tratament poate varia.

Sablonul Command transformă o solicitare într-un obiect independent care conține toate informațiile necesare pentru a executa acea solicitare. Acesta separă solicitarea de executarea efectivă, permițând manipularea și gestionarea ulterioară a comenzilor. Astfel, permite crearea de operațiuni complexe, invocarea acestora într-o ordine specifică și posibilitatea de anulare a comenzilor efectuate.

Sablonul Mediator promovează comunicarea între obiecte prin intermediul unui obiect mediator centralizat, eliminând astfel dependențele directe între obiecte. Mediatorul permite interacțiunea în mod slab cuplat între obiecte și facilitează extensibilitatea sistemului prin adăugarea sau modificarea mediatorilor existenți.

Acest sablon permite definirea unei familii de algoritmi, încapsularea fiecărui algoritm într-un obiect separat și permițându-le să fie interschimbabile. Astfel, clientul poate alege un algoritm din familia specificată și îl poate schimba în timpul execuției. Aceasta promovează flexibilitatea, modularitatea și extensibilitatea codului.

Sablonul Template Method definesc un schelet algoritmic într-o clasă de bază, lăsând implementarea anumitor pași specifici derivatei. Clasa de bază stabilește structura generală a algoritmului, dar lasă derivatele să-și implementeze pașii specifici. Astfel, permite refolosirea codului și flexibilitatea în implementarea diferitelor variante ale algoritmului.

În concluzie, prin utilizarea lor într-un mod corespunzător, putem obține un design software robust și scalabil, care să îndeplinească cerințele și să faciliteze evoluția sistemelor software.

**Bibliografie**

https://github.com/442god/TMPS