

FACULTATEA CALCULATOARE, INFORMATICA SI MICROELECTRONICA
UNIVERSITATEA TEHNICA A MOLDOVEI

ANALIZA, PROIECTAREA ȘI PROGRAMAREA ORIENTATĂ PE
OBIECTE

LUCRAREA DE LABORATOR#2

Implementare principiilor SOLID

Autor:

Marina JECHIU

lector asistent:

Mihail PECARI

Laboratory work #2

1 Scopul lucrării de laborator

Scopul lucrării de laborator este de a implementa într-un limbaj de programare principiile single responsibility și interface segregation.

2 Sarcina de lucru

În cadrul laboratorului e nevoie de realizat un software care o să conțină realizarea principiilor S (Single responsibility) și I (Interface segregation). Tematica este la discreția dumneavoastră (Nu se accepta folosirea lucrării precedente). Source code-ul este obligatoriu de încărcat pe github. La susținerea laboratorului prezenta raportului e obligatorie.

3 Noțiuni teoretice

Single responsibility principle Orice context (clasă, funcție, variabilă etc.) trebuie să aibă o unică responsabilitate, iar această responsabilitate trebuie să fie în întregime încapsulată în context. Toate serviciile sale trebuie să fie orientate pentru a servi această unică responsabilitate.

O “responsabilitate” poate fi definită și ca “motiv de schimbare”. Ca exemplu, putem considera un program simplu care generează și tipărește un raport. Un astfel de modul ar putea fi modificat din două motive: fie se schimbă conținutul raportului, fie se schimbă formatul raportului. Principiul responsabilității unice afirmă că aceste două aspecte ale problemei sunt două responsabilități separate și trebuie tratate în clase sau module diferite. A trata cele două lucruri împreună în aceeași clasă sau modul ar reprezenta o problemă de design. Astfel, este important să avem o clasă concentrată pe o unică responsabilitate pentru a o face mai robustă.

Interface segregation principle Acest principiu afirmă că niciun client nu trebuie să fie forțat să depindă de metode pe care nu le utilizează. Interfețele trebuie separate în alte interfețe mai mici și mai specifice.

4 Laboratory work implementation

4.1 Analiza lucrării de laborator

<https://github.com/MarinaJechiuTI154/APPO/>

Pentru realizarea sarcinilor înaintate spre îndeplinire a fost utilizat limbajul Java, un limbaj specific paragmei programării orientate pe obiecte. Pentru o structurare logică mai bună, fiecare funcțional a fost integrat într-un pachet separat.

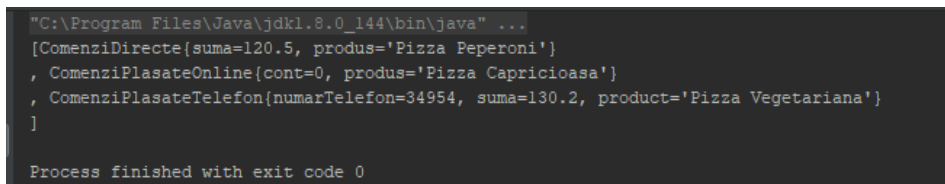
Tema aleasă pentru crearea claselor este o aplicație o aplicție de gestionare a comenzilor. Aceasta are ca scop preluarea, înregistrarea și procesarea comenzilor. Totodată, oferă posibilitatea procesării comenzilor pe mai multe căi: direct(preluare de la clienți), prin telefon și online.

Pentru respectarea principiului interface segregation au fost create interfețe specifice fiecărui tip de comandă. Pentru respectarea principiului single responsibility am creat clase care au o anumită, unică responsabilitate în cadrul sistemului.

Pentru a exemplifica funcționalitatea acestor obiecte, cât și relațiile dintre acestea au fost create obiecte concrete. Pentru o evidență sistematică a comenzilor, acestea sunt stocate într-un ArrayList. Avantajele folosirii acestor colecții este ușurința de stocare, prelucrare și ștergere a elementelor în cadrul lor.

În figurile de mai jos sunt prezentate imagini ce exemplifică funcționalitatea aplicației.

4.2 Imagini



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...  
[ComenziDirecte{suma=120.5, produs='Pizza Pepperoni'}  
, ComenziPlasateOnline{cont=0, produs='Pizza Capricioasa'}  
, ComenziPlasateTelefon{numarTelefon=34954, suma=130.2, product='Pizza Vegetariana'}  
]  
  
Process finished with exit code 0
```

Figure 4.1 – Afișarea comenzilor în proces de realizare

Concluzie

Sarcina de bază de implementare a celor 2 principii single responsibility și interface segregation a fost respectată și implementată. Acestea au fost implementate într-o aplicație de gestionare a comenzilor. Funcționalitatea softului a fost arătată în cadrul clasei Test.