Filtrare content în timp real

Tema 2

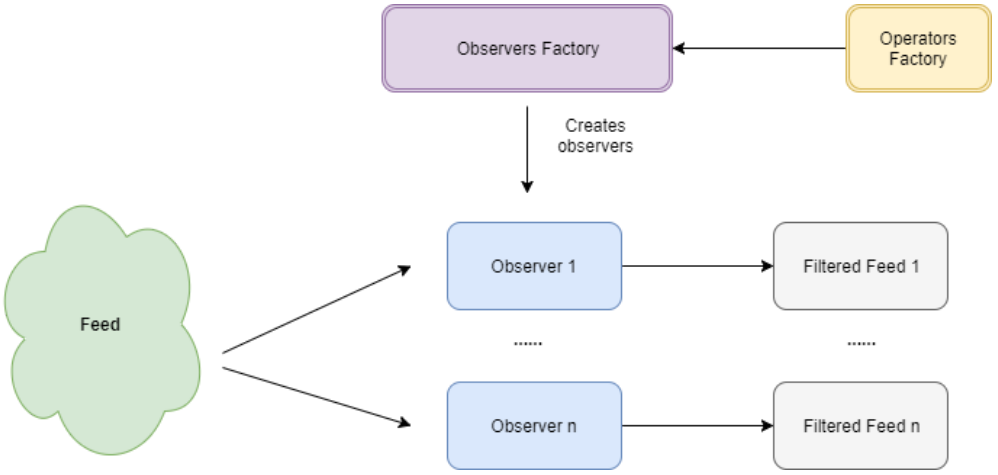
Responsabili: Rareş Tăerel şi Ştefan Ruşeţi

1 Introducere

Scopul acestei teme este de a vă familiariza cu programarea orientată pe obiecte şi cu limbajul Java. De asemenea se vor utiliza diverse design pattern-uri.

Evoluţia acţiunilor în cadrul unei burse de valori, postările de pe o reţea de socializare, ştirile din cadrul unei publicaţii, toate acestea reprezintă un flux de date, accesibil unui număr mare de utilizatori. Pentru prezentarea unui conţinut personalizat, este necesară o filtrare particulară în funcţie de preferinţele fiecărui utilizator.

2 Cerinţă



Cerinţa temei este de a se implementa o modalitate de filtrare a unui feed pentru diverşi utilizatori în funcţie de anumite expresii de filtrare. Este recomndată utilizarea factory design pattern împreună cu singleton design pattern pentru crearea de observatori şi de operatori pentru filtrare. Fiecare observator va trebui să menţină pentru fiecare stock filtrat numele, ultima valoare, fluctuaţia în procente dintre ultimele două valori şi numărul de schimări. Se recomandă utilizarea unei colecţii de tipul TreeMap pentru persistenţa informaţiilor despre stocurile filtrare în cadrul fiecărui observer. În cadrul acestei teme,

chiar dacă nu au fost studiate la laborator este permisă şi chiar încurajată utilizarea colecţiilor.

3 Testare

Tema va fi testată pe **VMChecker** fiind disponibil şi un checker local (va apărea în curând), astfel încât va trebui să aveţi şi un makefile cu o regulă *build* pentru compilare şi una *run* care să ruleze clasa şi care conţine metoda ”main”, pentru a uşura testarea automata. Datele de input se citesc de la System.in şi se scriu la System.out.

Tipuri de comenzi**:**

create_obs obs_id Filter	Comanda va crea un observator cu id-ul **obs_id** şi ce va filtra content-ul în funcţie de filtrul **Filter** . **Filter** – expresie formată din operatori utilizată pentru filtrarea content-ului **Filter :=** nil Expr **Expr :=** (Expr) Expr && Expr Expr || Expr Op name/value filter_value **Op :-** eq, ne, gt, ge, lt, le **(Important)** Pentru **value** se vor aplica toţi operatorii. Pentru **name** se vor aplica doar operatorii **eq** şi **ne**.
delete_obs obs_id	Comanda va şterge observatorul cu id-ul **obs_id**
print obs_id	Comanda va printa content-ul filtrat al observatorului cu id-ul **obs_id** în formatul prezentat în exemplul următor. Stock-urile vor fi sortate crescător după nume, iar informaţiile pentru fiecare stock vor fi afişate pe o singură linie.

feed name value	Comanda reprezintă notificarea observerilor cu privire la o nouă schimbare a stock-ului **name** cu valoare **value**

Exemplu format comandă **print obs_id**:

input	***output***
feed name1 value1 feed name1 value2 feed name1 value3 print obs_id	obs **obs_id**: **name1 value3 increase nr_of_changes** **increase** - fluctuaţia preţului de la ultima valoare afişată de observator în procente (negativă sau pozitivă). Dacă nu fost executată o comandă **print** înainte, se va afişa 0.0% **nr_of_changes** - numărul total de feed-uri detectate de către observer pentru stock-ul name1 de la ultima afişare (sunt incluse şi cele care nu trec de filtru). În cazul în care, este prima afişare a stock-ului name1 pentru observer obs_id, se va afişa numărul total de feed-uri, în exemplu curent 3. În cazul în care un observer are mai multe stock-uri de afişat, se vor afişa în ordine alfabetică după **name.** (Pentru a menţine lista de stock-uri sortată, se recomandă folosirea unui Treemap) La executarea comenzii print, dacă ultima valoare pentru un anumit stock nu respectă filtrul, acel stock nu se va afişa.

4 Exemplu rulare

input
begin create_obs 10 nil create_obs 11 (eq name GOLD) create_obs 12 ((le value 4.8) || (ge value 4.9)) && (eq name GOLD) feed GOLD 4.6 feed GOLD 4.8 print 11 feed GOLD 4.7 print 11 feed GOLD 4.75 delete_obs 11 create_obs 11 (eq name GOLD) feed GOLD 4.9 feed EUR 5.8 feed OIL 3.7 print 10 print 11 feed EUR 5.7 feed OIL 4.0 feed GOLD 4.85 print 12 end

output
obs 11: GOLD 4.80 0.00% 2 obs 11: GOLD 4.70 -2.08% 1 obs 10: EUR 5.80 0.00% 1 obs 10: GOLD 4.90 0.00% 5 obs 10: OIL 3.70 0.00% 1 obs 11: GOLD 4.90 0.00% 1

5 Constrângeri

1) **obs_id –** un id de tipul integer.
2) Pentru rezolvarea temei se vor utiliza design pattern-urile: Observer, Singleton, Factory şi Visitor unde este posibil.
3) Pentru parsarea expresiei se poate utilza orice algoritm astfel încât să se obţină un arbore de expresie pe care se va aplica visitor design pattern pentru evaluare.
4) Nu vor exista situaţii în care să se creeze/şterge un observator cu un id existent/ienxistent. Este însă posibil să se adauge un observator cu un id care a fost şters înainte.
5) Toate valorile citite/afişate vor fi rotunjite la 2 zecimale.
6) Fluxul de date de intrare începe cu **begin** şi se termină cu **end.**

6 Punctaj

Punctajul pe tema va consta din:

- 90% testare automată
- 10% coding style + readme + JavaDoc
- Code inspection (**-100%**)

7 Resurse

- Observer design pattern
- Factory design pattern
- Singleton design pattern
- Visitor design pattern
- Algoritm rezolvare expresie filtru