Programare orientata pe obiecte

Tema – Voucher Management Service (VMS)

Publicarea enuntului: 09.12.2019

Data ultimei modificari a enuntului: 16.12.2019

Termen de predare: 10.01.2020, ora 23:55

Nu se accepta temele trimise dupa termenul de predare !

Responsabili tema:

Ioana Craciun, Gabriel Mocanu, Mihai Nan

Catalin Icleanu, Paul Ungureanu

Profesor titular: Carmen Odubasteanu

Facultatea de Automatica si Calculatoare, Universitatea POLITEHNICA din Bucuresti

Anul universitar 2019 – 2020, Seria CC

1. Obiective

In urma realizarii acestei teme, studentul va fi capabil:
• sa aplice corect principiile programarii orientate pe obiecte studiate in cadrul cursului;
• sa construiasca o ierarhie de clase, pornind de la un scenariu propus;
• sa utilizeze un design orientat-obiect;
• sa implementeze o serie de structuri de date folosind programarea orientata pe obiecte;
• sa trateze exceptiile ce pot interveni in timpul rularii unei aplicatii;
• sa transpuna o problema din viata reala intr-o aplicatie ce se poate realiza folosind notiunile dobandite in cadrul cursului.

2. Descriere

In marketing, un voucher reprezinta un cupon de reducere utilizat pentru obtinerea unui discount la achizitionarea unor anumitor produse de la un anumit comerciant sau producator.

In era digitalizarii, voucherele pot fi si sub forma unor coduri de reducere ce pot fi aplicate la cumparaturile online.

Scopul acestei teme este de a implementa un serviciu de voucher management – crearea de vouchere specifice pentru anumite campanii, distribuirea acestora utilizatorilor si posibilitatea de a fi folosite de catre acestia, aplicand notiunile studiate in cadrul cursului.

3. Descrierea claselor

1. **VMS** – aceasta este clasa care modeleaza colectia de date VMS si este caracterizata de multimea de campanii existente, precum si de utilizatorii care pot primi diverse vouchere, fiind posibile urmatoarele operatii:
    - **getCampaigns()** – intoarce toate campaniile din colectie
    - **getCampaign(Integer id)** – intoarce campania cu id-ul specificat
    - **addCampaign(Campaign campaign)** – adauga o campanie noua
    - **updateCampaign(Integer id, Campaign campaign)** – modifica campania cu id-ul specificat (se pot modifica doar campaniile avand statusul NEW – toate campurile mai putin campul corespunzator numarului de vouchere disponibile sau STARTED – doar campurile corespunzatoare numarului total de vouchere si data de finalizare a campaniei). De asemenea, numarul total de vouchere nu poate fi setat cu o valoare mai mica decat cea a numarului de vouchere distribuite. Valoarea campului corespunzator numarului de vouchere disponibile va fi modificata in functie de noua valoare a campului care precizeaza numarul total de vouchere care pot fi generate.
    - **cancelCampaign(Integer id)** – inchide campania cu id-ul specificat (se pot inchide doar campaniile avand statusul NEW si STARTED)
    - **getUsers()** – intoarce toti utilizatorii care exista in colectia de date a VMS
    - **addUser(User user)** – adauga un utilizator nou

2. **Campaign** – o campanie este reprezentata printr-o clasa care contine drept atribute: ID-ul campaniei (identificator unic), numele si descrierea campaniei, data de start si data de finalizare a campaniei, numarul total de vouchere care pot fi distribuite, numarul curent de vouchere disponibile, statusul campaniei (**va contine o enumerare, CampaignStatusType, cu urmatoarele constante: NEW, STARTED, EXPIRED, CANCELLED**), dictionarul de vouchere care au fost distribuite si utilizatorii care au primit cel putin un voucher corespunzator

campaniei respective (observers), precum si tipul strategiei care va fi detaliata ulterior. Urmatoarele operatii sunt posibile:

- **getVouchers()** – intoarce toate voucherele corespunzatoare campaniei
- **getVoucher(String code)** – intoarce voucherul cu codul specificat
- **generateVoucher(String email, String voucherType, float value)** – genereaza un voucher cu un cod unic avand detaliile precizate pe care il distribuie utilizatorului avand adresa de e-mail specificata, daca aceasta este valida. ID-ul voucherului este generat incremental pornind de la valoarea 1.(ex: primul voucher generat va avea ID 1, al doilea ID 2, samd.)
- **redeemVoucher(String code, LocalDateTime date)** – marcheaza voucherul avand codul specificat ca fiind utilizat, daca data este valida in conformitate cu perioada de desfasurare a campaniei, precum si cu statusul acesteia, iar voucherul nu a mai fost utilizat
- **getObservers()** – intoarce toti utilizatorii care au primit cel putin un voucher in cadrul campaniei
- **addObserver(User user)** – metoda apelata pentru a inregistra un utilizator ca un observator (aceasta metoda trebuie apelata atunci cand utilizatorul primeste cel putin un voucher corespunzator campaniei respective)
- **removeObserver(User user)** – metoda apelata pentru a sterge un utilizator ca observator (aceasta metoda trebuie apelata cand utilizatorul nu mai are in inbox niciun voucher apartinand campaniei)
- **notifyAllObservers(Notification notification)** – trimite notificarea catre fiecare observator (aceasta metoda trebuie apelata daca este modificata sau inchisa campania curenta)

3. **Voucher** – un voucher este reprezentat printr-o clasa abstracta care contine drept atribute: id-ul voucherului (identificator unic), codul voucherului (identificator unic), statusul voucherului (**va contine o enumerare, VoucherStatusType, cu urmatoarele constante: USED, UNUSED**), data utilizarii, email-ul catre care a fost distribuit voucherul, precum si ID-ul campaniei in cadrul careia a fost generat.

4. **GiftVoucher**, **LoyalityVoucher** – aceste clase reprezinta tipuri de vouchere si extind clasa abstracta Voucher

   - Clasa GiftVoucher contine un atribut care specifica suma disponibila care poate fi utilizata in cadrul campaniei, o singura data
   - Clasa LoyalityVoucher contine un atribut care specifica o reducere care poate fi aplicata in cadrul campaniei, o singura data

5. **Notification** – fiecare utilizator detine o colectie de notificari pe care le primeste atunci cand este modificata sau inchisa o campanie pentru care a primit cel putin un voucher. Clasa Notification va contine drept atribute: tipul notificarii (**va contine o enumerare, NotificationType, cu urmatoarele constante: EDIT, CANCEL**), data la care a fost trimisa notificarea, ID-ul campaniei si lista codurilor corespunzatoare voucherelor pe care utilizatorul le-a primit in cadrul campaniei respective.

6. **User** – un utilizator este reprezentat printr-o clasa care contine drept atribute: ID-ul user-ului (identificator unic), nume, adresa de e-mail unica si parola, tipul user-ului (**va contine o enumerare, UserType, cu urmatoarele constante: ADMIN, GUEST**), dictionarul de vouchere primite si o colectie de notificari. Un utilizator obisnuit poate doar citi informatiile referitoare la campanii si nu are acces la colectia de vouchere corespunzatoare fiecareia. Acesta poate citi detaliile despre voucherele primite si poate prezenta codul unic unui administrator pentru a-l folosi. Un administrator poate citi si poate modifica detaliile specifice unei campanii, poate citi informatiile despre voucherele distribuite si le poate marca ca fiind utilizate.

7. **ArrayMap** – reprezinta o clasa generica, care modeleaza un dictionar realizat ca o colectie de obiecte ArrayMapEntry. Clasa ArrayMap **extinde** clasa abstracta AbstractMap, implementand urmatoarele metode:
   - V put(K key, V value)
   - boolean containsKey(Object key)
   - V get(Object key)
   - int size()
   - Set<Entry<K, V>> entrySet

   De asemenea, clasa ArrayMap contine o clasa interna ArrayMapEntry care **implementeaza** interfata Map.Entry. Acesata clasa implementeaza urmatoarele metode:
   - K getKey()
   - V getValue()
   - V setValue(V value)
   - String toString()
   - boolean equals(Object o)
   - int hashCode()

8. **UserVoucherMap** – aceasta clasa va extinde clasa ArrayMap, in care cheia este reprezentata de ID-ul unei campanii, iar valoarea este reprezentata de colectia de vouchere primite in cadrul campaniei respective. Acesata clasa va implementa metoda:
   - boolean addVoucher(Voucher v)

9. **CampaignVoucherMap** – aceasta clasa va extinde clasa ArrayMap, in care cheia este reprezentata de email-ul unui utilizator, iar valoarea este reprezentata de colectia de vouchere distribuite in cadrul campaniei respective. Acesata clasa va implementa metoda:
   - boolean addVoucher(Voucher v)

4. Sabloane de proiectare

Un sablon de proiectare descrie o problema care se intalneste in mod repetat in proiectarea programelor si, de asemenea, in solutia generala pentru problema respectiva, astfel incat sa poata fi utilizata oricand, dar nu in acelasi mod de fiecare data. Solutia este exprimata folosind clase si obiecte. Atat descrierea problemei cat si a solutiei sunt abstracte astfel incat sa poata fi folosite in situatii diferite.

Scopul sabloanelor de proiectare este de a asista rezolvarea unor probleme similare cu unele deja intalnite si rezolvate anterior. Ele ajuta la crearea unui limbaj comun pentru comunicarea experientei despre aceste probleme, precum si solutiile lor.

- **Singleton pattern** – pentru a asigura faptul ca fiecare utilizator are acces la informatiile aceleasi colectii de date este necesara mentinerea unei referinte catre un obiect de tip VMS, in mai multe clase. Astfel, se utilizeaza sablonul Singleton pentru a restrictiona numarul de instantieri ale clasei VMS la un singur obiect. Pentru a nu consuma inutil resursele sistemului, in implementarea voastra, se va folosi instantierea intarziata.
    - **getInstance()**

- **Observer pattern –** design pattern-ul Observer defineste o relatie de dependenta unul la mai multi intre obiecte, astfel incat atunci cand un obiect isi schimba starea, toti dependentii lui sunt notificati si actualizati automat. Acest pattern implica existenta unui obiect denumit subiect care are asociata o lista de obiecte dependente, numite observatori, pe care le apeleaza automat de fiecare data cand se intampla o actiune. In cadrul acestei aplicatii, se utilizeaza pattern-ul Observer pentru a notifica utilizatorii de fiecare data cand se modifica sau se inchide o campanie pentru care acestia detin cel putin un voucher.
    - **update(Notification notification) – Observer - User**
    - **addObserver(User user) – Subject – Campaign**
    - **removeObserver(User user) – Subject – Campaign**
    - **notifyAllObservers(Notification) – Subject – Campaign**

5. Cerinte

1. Cerinta 1 - Implementarea si testarea aplicatiei (fara interfata grafica) - 120 puncte

Pentru a putea realiza o testare a aplicatiei, trebuie implementata o clasa **Test**, ce contine metoda **main** care parseaza o serie de fisiere de intrare si realizeaza o serie de operatii indicate. Rezultatul acestor operatii va fi scris intr-un fisier de iesire, **output.txt**.

Formatul fisierelor – formatul fiecarui fisier de intrare ce se va oferi pentru verificarea functionalitatii aplicatiei este:

- **campaigns.txt –** pe prima linie este specificat numarul de campanii N, pe cea de-a doua linie se gaseste data curenta a aplicatiei, iar pe urmatoarele N linii sunt specificate campaniile existente astfel:

  ID;NAME;DESCRIPTION;START_DATE;END_DATE;BUDGET;STRATEGY_TYPE

  La instantierea unui obiect de tipul Campaign, valoarea campului corespunzator numarului de vouchere disponibile va fi setata ca fiind egala cu valoarea campului corespunzator numarului total de vouchere care pot fi distribuite, iar campul status va fi setat in conformitate cu valorile START_DATE, END_DATE, respectiv data curenta a aplicatiei.

- **users.txt** – pe prima linie a fisierului se gaseste N, numarul total de utilizatori, iar pe urmatoarele N linii sunt specificati utilizatorii: **ID;NAME;PASSWORD;EMAIL;TYPE**

- **events.txt** – pe prima linie a fisierului se gaseste data curenta a aplicatiei, pe cea de-a doua linie este specificat numarul de evenimente N, iar pe urmatoarele N linii sunt descrise acestea. Evenimentele posibile sunt urmatoarele:
  - **userId;addCampaign;campaignId;campaignName;campaignDescription;startDate;endDate;budget;strategy** – adauga campania indicata in colectia de campanii, daca utilizatorul este administrator
  - **userId;editCampaign;campaignId;campaignName;campaignDescription;startDate;endDate;budget** – modifica detaliile campaniei specificate, daca aceasta actiune este permisa si daca utilizatorul este administrator
  - **userId;cancelCampign;campaignId** – inchide campania indicata, daca aceasta actiune este permisa si daca utilizatorul este administrator
  - **userId;generateVoucher;campaignId;email;voucherType;value** – genereaza un voucher avand detaliile specificate si il distribuie utilizatorului avand adresa de e-mail precizata, daca utilizatorul este administrator
  - **userId;redeemVoucher;campaignId;voucherId;localDate** – marcheaza voucher-ul indicat ca fiind utilizat, daca aceasta actiune este permisa si daca utilizatorul este administrator
  - **userId;getVouchers** – intoarce toate voucherele din inbox-ul utilizatorului precizat, daca acesta este unul obisnuit
  - **userId;getObservers;campaignId** – intoarce observatorii campaniei indicate, daca utilizatorul este administrator
  - **userId;getNotifications** – intoarce notificarile utilizatorului specificat, daca acesta este unul obisnuit
  - **userId**;**getVoucher;campaignId** – intoarce voucher-ul distribuit conform strategiei definite in cadrul fisierului campaigns.txt pentru campania specificata, daca utilizatorul este administrator

2. Cerinta 2 – Interfata Grafica – 80 puncte

Interfata grafica se realizeaza folosind **componenta Swing**. Din aceasta interfata vor putea executa comenzi atat administratorii, cat si clientii obisnuiti.

Prima pagina ai interfetei grafice va cuprinde un **buton de incarcare a fisierelor campaigns.txt si users.txt**. Pe baza acestor fisiere se va crea instanta obiectului de tip VMS si ierarhia de clase adiacente. De asemenea, vor exista si **doua casete text**, pentru **introducerea numelui si a parolei**, precum si a **unui buton de logare**.

Pentru administrarea colectiei de campanii se vor implementa urmatoarele pagini:

1. **pagina de afisare si administrare a campaniilor din colectie** cu urmatoarele operatii:
   - **afisarea campaniilor existente,** sub forma de tabel, cu posibilitatea de ordonare alfabetica a acestora dupa nume, precum si dupa data de start (crescator/descrescator)
   - **adaugarea unei campanii** (daca campania exista deja, administratorul va fi atentionat)
   - **editarea unei campanii**
   - **inchiderea unei campanii**
   - **vizualizarea detaliilor despre o campanie**

2. **pagina de afisare si administrare a voucherelor corespunzatoare unei campanii din colectie** cu urmatoarele operatii:
    - **afisarea voucherelor existente**, sub forma de tabel
    - **generarea unui voucher nou**
    - **cautarea unui voucher pe baza codului si marcarea acestuia ca fiind utilizat**

Pentru paginile destinate unui utilizator obisnuit se vor implementa urmatoarele pagini:

1. **pagina de vizualizare a campaniilor din colectie**
2. **pagina de vizualizare a colectiei de vouchere personale**

Observatie! In cazul verificarii functionalitatilor din interfata grafica, data curenta a aplicatiei este momentul de timp la care rulati aplicatia.

6. Bonusuri

- **Strategy pattern - 15 puncte**

Sablonul architectural Strategy a fost conceput pentru a putea selecta algoritmul optim pentru gestionarea fiecarei strategii de campanie utilizata, in functie de dorintele administratorului. In cazul aplicatiei, avem o lista de campanii, fiecare continand o colectie de vouchere si dorim sa selectam cat mai profitabil clientii carora le trimitem vouchere. Astfel, acest sablon ii ofera administratorului posibilitatea de a alege singur ce strategie foloseste pentru a selecta clientii care vor primi vouchere:

    - **execute(Campaign c)** – Strategy: Strategy A, Strategy B, Strategy C
    - **executeStrategy()** – Campaign
    - **Strategy A** – selecteaza aleator o adresa de e-mail din cele existente deja in colectia de observatori si ii trimite un voucher de tipul GiftVoucher, in valoare de 100
    - **Strategy B** – selecteaza adresa de e-mail care a utilizat cele mai multe vouchere din cadrul campaniei respective si ii trimite un voucher de tipul LoyalityVoucher, cu o reducere de 50%
    - **Strategy C** – selecteaza adresa de e-mail care a primit cele mai putine vouchere din colectia de observatori si ii trimite un voucher de tipul GiftVoucher, in valoare de 100

- **Distribuirea multipla de vouchere – 15 puncte**

Deoarce este foarte ineficient pentru un administrator sa genereze si sa trimita cate un voucher pe rand, se doreste implementarea functionalitatii de distribuire multipla a voucherelor. Acest lucru se realizeaza citind un fisier, **emails.txt**, care contine pe prima linie numarul de e-mail-uri existente in fisier, iar pe urmatoarele linii cate o adresa de e-mail, tipul voucherului generat, precum si valoarea acestuia. Pentru fiecare adresa de e-mail valida se va genera si se va distribui voucherul corespunzator.

- **Implementarea functionalitatii de verificare si editare periodica a statusului camapaniilor, in functie de data curenta si cea de start/finalizare a acestora – 10 puncte**
Hint: TimerTask

- **Interfata grafica - 10 puncte**

  In ceea ce priveste interfata grafica, se propun urmatoarele bonusuri:

  - pagina de generare a voucherelor va contine si functionalitatea de sugerare a unei adrese de e-mail existente in colectia de utilizatori, conform strategiei administratorului
  - o pagina pentru fiecare campanie in care se vor afisa observatorii acesteia, precum numarul total de vouchere utilizate in campania respectiva
  - pagina in care un utilizator isi poate accesa lista de notificari primite

7. Punctaj

- **Cerinta 1 – 120 puncte**
- **Cerinta 2 – 80 puncte**
- **Bonus Design Pattern Strategy – 15 puncte**
- **Bonus Distribuire Multipla – 15 puncte**
- **Bonus TimerTask – 10 puncte**
- **Bonus Interfata grafica – 10 puncte**

8. Restrictii si precizari

- Tipizati orice colectie folosita in cadrul implementarii.
- Respectati specificatiile detaliate in enuntul temei si folositi hint-urile mentionate.
- **Tema este individuala! Toate solutiile trimise vor fi verificate, folosind o aplicatie pentru detectarea plagiatului.**
- **Important! Tema se va prezenta personal la ultimul laborator din semestru!**
- Tema se va incarca pe moodle pana la termenul specificat in pagina de titlu. Se va trimite o arhiva .zip ce va avea un nume de forma grupa_Nume_Prenume.zip (ex. 326CC_Popescu_Andreea.zip) si care va contine urmatoarele:
  - un folder SURSE ce contine doar sursele Java, un Makefile care sa compileze aceste surse si sa permită rularea clasei Test, avand cel putin 2 reguli (build si run) si fisierele de test
  - un folder PROIECT ce contine proiectul NetBeans, Eclipse sau IntelliJ IDEA
  - un fisier README in care veti specifica numele, grupa, gradul de dificultate al temei, timpul alocat rezolvarii si veti detalia succint modul de implementare, specificand si alte obervatii, daca este cazul – lipsa acestui fisier duce la o depunctare de 10 puncte