PROIECT FINAL JAVA

ENUNT

Sa se construiasca o aplicatie cu interfata grafica ce implementeaza o agenda de contacte. Aplicatia va porni initial in mod "shareware" (cu functionalitate partiala si afisare de reclame) urmand ca, dupa ce utilizatorul introduce codul de inregistrare corect, sa comute in modul "full" in care reclamele nu mai sunt afisate si functionalitatea este deplina.

Contactele

Fiecare contact va avea nume, prenume, telefon (care poate fi fix sau mobil) si data nasterii. Input-ul de la utilizator este validat astfel incat sa nu se poata crea contacte cu date de intrare invalide, dupa cum urmeaza:

- numele si prenumele trebuie sa aiba cel putin doua litere
- data nasterii trebuie sa fie una valida, in format ZZ.LL.AAAA
- numarul de telefon:
  - trebuie sa aiba 10 cifre
  - numerele mobile incep cu 07
  - numerele fixe incep cu 02 sau 03

Nu vor fi permise contacte duplicat; doua contacte se considera identice atunci cand valorile tuturor atributelor lor sunt egale. In cazul numelui si prenumelui compararea trebuie efectuata case-insensitive.

Operatii

Utilizatorul va putea adauga, sterge, edita, ordona, filtra, salva si incarca contacte.

La adaugarea unui nou contact:

- se va verifica daca contactul este duplicat si, in caz afirmativ, se va genera o eroare
- in cazul in care data curenta coincide cu ziua de nastere a contactului, se va afisa o fereastra de dialog de notificare

Filtrarea listei de contacte se poate realiza in doua moduri:

- pe baza urmatoarelor filtre predefinite:
  - contactele care au numar de fix
  - contactele care au numar de mobil
  - contactele ce au data de nastere in ziua curenta
  - contactele care au data de nastere in luna curenta, insa dupa data curenta
- personalizat, pe baza unui sir introdus de utilizator, care va fi cautat in cadrul valorilor atributelor nume, prenume si telefon ale fiecarui contact (nu si data nasterii!)

Filtrarea va avea ca efect afisarea in interfata grafica numai a contactelor care corespund filtrului.

La anularea filtrului va fi afisata lista completa.

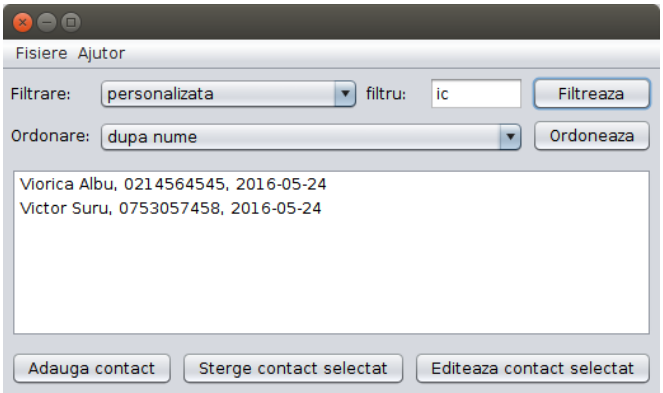
Lista de contacte poate fi salvata si incarcata de catre utilizator folosind optiunile din meniul *Fisiere* (vezi mai jos). In plus, odata ce utilizatorul a efectuat o salvare manuala, lista va fi salvata automat in fisierul ales odata pe minut, in background.

La incarcarea din fisier a unei liste se va afisa o fereastra popup ce listeaza persoanele care au data de nastere in ziua curenta.

Interfata grafica

Cu exceptia cerintelor prezentate in continuare, design-ul ei este la latitudinea implementatorului. Imaginea de mai jos este pur orientativa.

La pornirea aplicatiei va fi afisat timp de 1 secunda un splash screen (ecran de intampinare) - o fereastra nedecorata care va contine o poza aleasa de implementator dedesubtul careia este scris numele acestuia.



Fereastra principala va contine lista de contacte si componentele grafice necesare pentru asigurarea functionalitatii aplicatiei (adaugare, stergere, modificare, ordonare, filtrare de contacte etc).

Fereastra va avea o bara de meniu, cu meniurile:

- **Fisiere**, cu urmatoarele elemente:
  - **Deschidere** – deschiderea unei baze de date cu contacte; la click pe acest element se deschide o fereastra de alegere de fisier (*JFileChooser*)
  - **Salvare** – salvarea bazei de date curente. Va fi afisat de asemenea un *JFileChooser* pentru a da utilizatorului posibilitatea de a alege locatia si numele fisierului. Atat acest element de meniu cat si precedentul vor fi dezactivate cat timp aplicatia se afla in modul de functionare shareware
  - **Iesire** – afiseaza un dialog de confirmare creat folosind *JOptionPane* ("Doriti sa parasiti aplicatia?" Butoane: Da/Nu) si actioneaza in functie de decizia utilizatorului
- **Ajutor**
  - **Inregistrare** – deschide o fereastra de dialog care solicita codul de inregistrare; in cazul in care codul introdus este corect, se dezactiveaza acest element de meniu, se activeaza *Deschidere* si *Salvare* din meniul *Fisiere* si se elimina reclamele
  - **Despre** – deschide o fereastra cu scurte informatii despre autor si aplicatie

Fereastra de adaugare/modificare a unui contact va contine, in dreptul campului pentru numar de telefon, o modalitate prin care utilizatorul sa indice daca acesta este de fix sau de mobil (ex: un checkbox, doua radio button-uri, un combo box etc).

La stergerea unui contact va fi afisat un dialog de confirmare ("Doriti sa stergeti contactul X?", butoane Da/Nu).

Ordonarea contactelor va fi realizata folosind un *JComboBox* sau un grup de radio buttons ce listeaza criteriile posibile de ordonare, plus un buton "Filtreaza" la apasarea caruia va fi actualizata lista de contacte.

Filtrarea cu input de la utilizator se realizeaza cu ajutorul unui text field in care utilizatorul scrie un string ce constituie filtrul. Pentru filtrele predefinite se va utiliza fie un grup de radio buttons, fie un *JComboBox*.

Modul shareware

Aplicatia porneste initial in mod "shareware", urmand ca, dupa introducerea codului corect de inregistrare, functionalitatea sa sa fie completa. In modul shareware:

- salvarea si incarcarea bazei de date sunt dezactivate (se dezactiveaza elementele *Deschidere* si *Salvare* din meniul *Fisiere*)
- in meniul *Ajutor* este activ elementul *Inregistrare*
- in zona inferioara a ferestrei principale va exista un spatiu in care vor rula reclame. Se vor alege cateva poze de aceeasi marime care vor fi afisate intr-un *JLabel*; reclama curenta se schimba la fiecare 1-2 secunde

Codul de inregistrare va fi ales de catre implementator si definit sub forma unei constante.

Documentare

Unde credeti ca este cazul, folositi comentarii pentru a preciza care v-au fost intentiile (decizii de design particulare, procedee sau algoritmi mai dificil de urmarit etc).

Facilitati suplimentare, optionale

Puteti incerca afisarea contactelor intr-un *JTable* si editarea lor in-place (o rezolvare eleganta), cautare incrementala (filtrarea listei in timp real, pe masura ce utilizatorul scrie in campul de filtrare), memorarea listei de contacte pe un server de baze de date, sau orice altceva v-ar placea sa vedeti iesit din mainile voastre.

SUGESTII DE IMPLEMENTARE

Cele ce urmeaza se intentioneaza a fi o simpla mana de ajutor in organizarea implementarii proiectului. Studentul este liber sa aleaga orice alta varianta daca o considera potrivita.

Clasa Contact

- fiecare obiect *Contact* corespunde uneia dintre persoanele memorate in agenda
- campuri: **nume**, **prenume**, **data nasterii** si **numar de telefon** (ultimul fiind de tip *NrTel*)
- metode:

- constructor cu argumente protejat cu exceptii impotriva datelor de intrare invalide
- getteri, setteri in functie de necesitati
- **equals()**: doua contacte sunt egale atunci cand valorile tuturor atributelor lor sunt egale. Atentie! Comparatia de nume si prenume trebuie efectuata case insensitive!
- metoda **toString()** rescrisa, in scopul afisarii corecte a contactelor in *JList* atunci cand s-a ales aceasta varianta; utila de asemenea pentru diagnostic rapid folosind *System.out.println()*

Clasele NrTel, NrFix si NrMobil

- **NrTel** - clasa abstracta corespunzatoare functionalitatii comune celor doua tipuri de numere de telefon. Clasa implementeaza interfata *Comparable*. Metode:
  - camp de tip *String* pentru memorarea numarului de telefon
  - constructor cu parametru, protejat cu exceptii impotriva datelor de intrare invalide
  - metoda abstracta **validareNumar()** apelata din constructor si care va fi rescrisa in subclase
  - **equals()** - folosita pentru a verifica egalitatea a doua numere de telefon, necesara la compararea a doua contacte
  - **compareTo()** - metoda impusa de interfata *Comparable*
  - **toString()** - intoarce numarul de telefon memorat in cadrul clasei
- clasele **NrFix** si **NrMobil** vor extinde *NrTel* si isi vor implementa fiecare propriile reguli de validare

Enum-ul CriteriuOrdonare

Elementele acestui enum sunt cele 4 criterii de ordonare posibile pentru contacte (DUPA_NUME, DUPA_PRENUME etc). Acest tip de date va fi folosit pentru a preciza ordinea dorita la apelarea metodei de ordonare a clasei *Agenda*.

Clasa Agenda

Este clasa care memoreaza contactele si contine metodele de manipulare a acestora. Contactele vor fi stocate sub forma unei colectii. Vor fi prevazute campuri suplimentare pentru criteriul de ordonare ales si pentru filtrul curent.

Membri ai clasei:

- campuri:
  - colectia de contacte
  - o colectie de tip *Map* de forma criteriuordonare-->comparator, folosita ulterior la generarea listei filtrate si ordonate in metoda *contacte()*. Colectia va fi populata in cadrul constructorului
  - un camp de tip *Predicate<Contact>* ce constituie filtrul curent. Valoarea sa initiala va fi o expresie lambda care intoarce *true* pentru toate elementele
  - un camp de tip *CriteriuOrdonare* ce memoreaza criteriul de ordonare curent (implicit nume)
- metode:

- metode pentru adaugare/stergere/modificare de contact
  - la adaugare se va verifica daca contactul in cauza exista deja in agenda si, in caz afirmativ, se va genera o exceptie
- filtrare
  - **filtreazaNrFix()**, **filtreazaNrMobil()**, **filtreazaNascutiAstazi()**, **filtreazaNascutiLunaCurenta()** - filtrele predefinite; metodele actioneaza ca setteri pentru campul clasei ce constituie filtrul curent
  - **filtreazaPersonalizat(String)** - filtrul personalizat; metoda primeste ca parametru un *Predicate<Contact>* si va seta filtrul din cadrul clasei
- **ordoneaza(CriteriuOrdonare)** - ordonarea contactelor dupa oricare dintre cele 4 atribute ale acestora. Metoda actioneaza ca un setter pentru campul clasei ce memoreaza criteriul de ordonare curent; de acesta se va tine cont ulterior in metoda *contacte()*
- **contacte()** - foloseste stream-uri Java 8 pentru a filtra si ordona lista de contacte (folosind filtrul si criteriul de ordonare deja prezente sub forma de campuri ale clasei). Intoarce o colectie ce contine contactele rezultante in ordinea corecta
- salvarea si incarcarea listei de contacte: **salveaza(Path)** si **incarca(Path)**. Se vor folosi stream-uri din *java.io* pentru serializare si deserializare

Clasa corespunzatoare ferestrei de baza a interfetei grafice

Campuri:
- **agenda** - are ca valoare un obiect de tip *Agenda*
- **modelLista** - are ca valoare un obiect de tip *DefaultListModel* sau *DefaultTableModel* in functie de forma de afisare aleasa pentru contacte

Metode:
- **actualizareModel()** - goleste modelul si il repopuleaza pe baza continutului curent al agendei
- metodele corespunzatoare tratarii evenimentelor generate de interactiunea cu utilizatorul. Ori de cate ori datele din agenda se modifica va fi apelata metoda *actualizareModel()*

Management ferestre aditionale

Ferestre de dialog:
- selectie fisier pentru incarcare/salvare: se foloseste un obiect de tip **JFileChooser**
- mesaje popup: **JOptionPane.showMessageDialog()**
- fereastra de confirmare la parasirea aplicatiei sau la stergerea unui contact: **JOptionPane.showConfirmDialog()**
- fereastra de introducere a codului de inregistrare: **JOptionPane.showInputDialog()**
- ferestre de dialog pentru adaugare/modificare contact: in Netbeans, din designerul de interfete grafice pentru fereastra principala, se adauga din paleta o fereastra de dialog, avand grija sa nu o plasam in fereastra de baza ci undeva in afara ei. Ca urmare, noua componenta fi adaugata automat la sectiunea *Other Components* din Navigator; de acolo poate fi selectata, i se pot schimba proprietatile, si i se poate face design-ul individual (click dreapta pe ea si *Design this container*). In codul sursa vom constata ca ea figureaza tot pe post de camp al clasei, asadar este accesibila in toate metodele si deci poate fi folosita liber oriunde este nevoie. Toate componentele adaugate in aceasta fereastra secundara vor figura de

asemenea pe post de campuri ale clasei de baza.

Thread-uri

- pentru rularea reclamelor se poate folosi *javax.swingTimer*, *java.util.Timer* sau un obiect de tip *Thread*
- pentru salvarea automata se poate utiliza *java.util.Timer* sau un obiect de tip *Thread*

Elemente suplimentare

Informatiile de mai sus, desi relativ detaliate, sunt in principal pentru ghidaj si nu constituie solutia completa de implementare (si nici singura posibila). Studentul va adauga elementele pe care le considera necesare (campuri, metode, clase etc) pentru realizarea cerintei proiectului.

Succes!