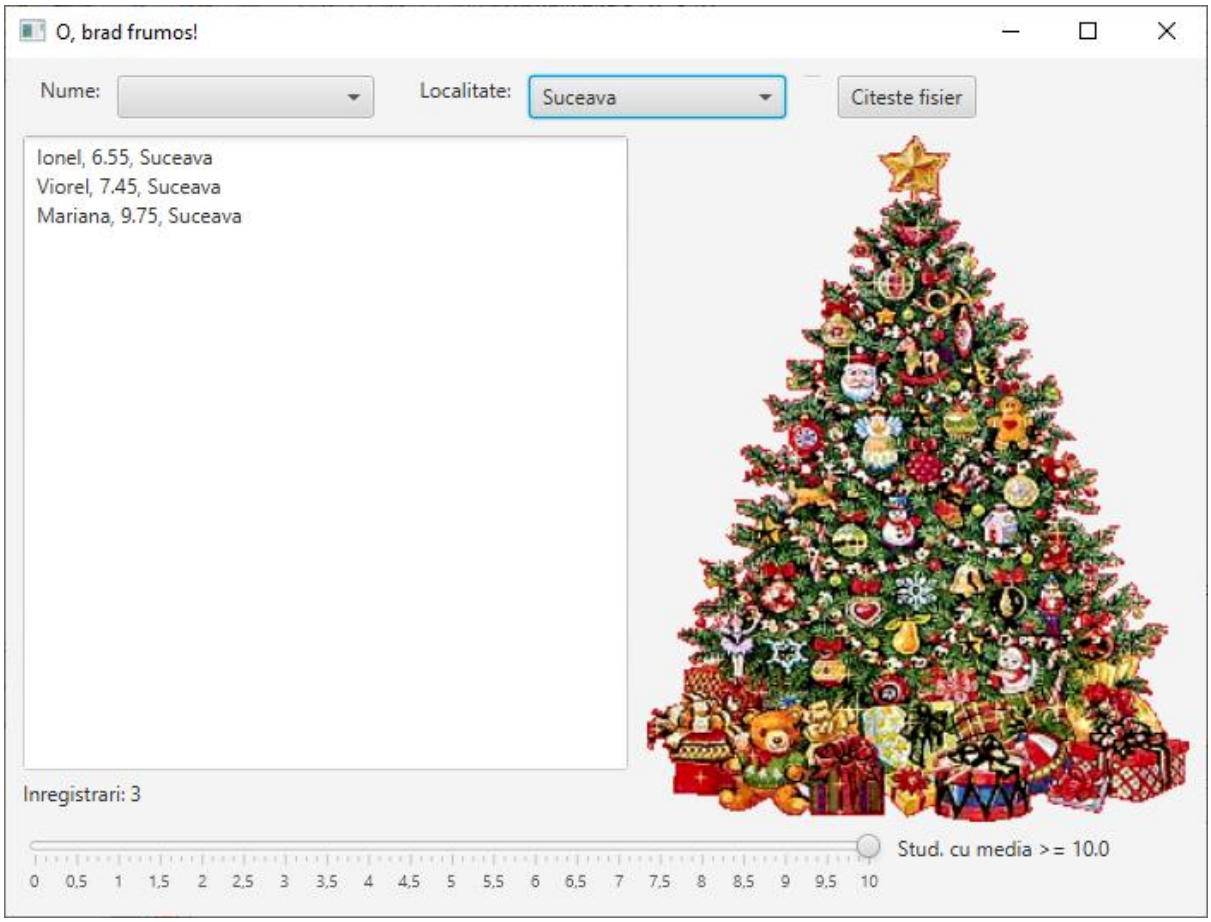
Laborator 12

Sunteti rugati sa scrieti o aplicatie Java care sa-l ajute pe Mosu' in distribuirea cadourilor pentru studenti. Aplicatia va afisa un panou ca cel de mai jos.



(imagine brad preluata de la http://www.fanpop.com/clubs/christmas/images/12140793/title/christmas-photo)

Mosu' are lista sa cu studenti si, date fiind capacitatile sale supranaturale, pentru fiecare student a pastrat doar numele, media si localitatea. Restul informatiilor le stie el foarte bine …

Lista de studenti a fost organizata pe aceste 3 informatii: numele, media si localitatea.

In aplicatie daca utilizatorul selecteaza un nume, de exemplu *Ionel*, se vor afisa toti studentii pe care ii cheama *Ionel*, indiferent de medie sau localitate..

Daca se va selecta o localitate (in figura Suceava) se vor afisa toti stdentii din acea localitate, indiferent ce nume sau medie au.

Daca utilizatorul muta slider-ul din partea inferioara pe o noua valoare, de ex. 9.50, in zona TextArea se vor afisa toti studentii care au media cel putin egala cu 9.50, indiferent de numele, sau localitatea selectata.

Aplicatia porneste cu o date de test, insa daca se va apasa pe butonul “Citeste fisier” se va citi un fisier **.txt** avand inregistrari cu formatul in care sunt afisate in figura de mai sus informatiile despre student (informatiile despre un student pe o singura linie cu campurile separate prin virgula).

Citirea fisierului presupune crearea obiectelor **Student**, cate unul pentru fiecare linie din fisier, cat si inregistrarea lor in structurile de cautare care sa permita regasirea rapida a studentilor dupa nume, medie ori localitate.

Aveti la dispozitie

Pachetul se va numi, in mod mormal, **`obradfrumos`**.

Clasa Student

```
package obradfrumos;

public class Student {
    private String nume;
    private double medie;
    private String localitate;

    public Student(String nume, double varsta, String localitate) {
        this.nume = nume;
        this.medie = varsta;
        this.localitate = localitate;
    }
    public String getNume() {
        return nume;
    }
    public double getMedie() {
        return medie;
    }
    public String getLocalitate() {
        return localitate;
    }
    @Override
    public String toString() {
        return nume + ", " + medie + ", " + localitate;
    }
}
```

Clasa ListaMosului

Aceasta clasa permite functionarea aplicatiei cu cele cateva date de test. Pentru ca sa functioneze cu datele Mosului va trebui completata cu un constructor ca cel indicat mai jos.

```
package obradfrumos;
import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
```

```
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

public class ListaMosului {

   private Map<String, List<Student>> mapNume = new TreeMap<>();
   private Map<Double, List<Student>> mapMedie = new TreeMap<>();
   private Map<String, List<Student>> mapLocalitate = new TreeMap<>();

   public ListaMosului ( File f ) {

      // DE COMPLETAT

  }
   public ListaMosului() {
      for (Student s : exempluDeTest()) {
         inscrieInListaMosului(s);
      }
   }
   public void inscrieInListaMosului(Student s) {
      adaugaInMap(mapNume, s.getNume(), s);
      adaugaInMap(mapMedie, s.getMedie(), s);
      adaugaInMap(mapLocalitate, s.getLocalitate(), s);

   }
   private void adaugaInMap(Map<String, List<Student>> map, String k, Student s) {
      List<Student> lst = map.get(k);
      if (lst == null) {
         lst = new ArrayList<>();
         map.put(k, lst);
      }
      lst.add(s);
   }

   private void adaugaInMap(Map<Double, List<Student>> map, Double k, Student s) {
      List<Student> lst = map.get(k);
      if (lst == null) {
         lst = new ArrayList<>();
         map.put(k, lst);
      }
      lst.add(s);
   }

   private List<Student> exempluDeTest() {
      Student[] lstStud = new Student[]{new Student("Ionel", 6.55, "Suceava"),
         new Student("Viorel", 7.45, "Suceava"),
         new Student("Violeta", 6.55, "Radauti"),
         new Student("Elena", 8.25, "Cajvana"),
         new Student("Mariana", 9.75, "Suceava"),
         new Student("Costel", 6.55, "Radauti"),
```

```
            new Student("Ionel", 8.25, "Cajvana"),
            new Student("Violeta", 8.55, "Campulung Moldovenesc"),
            new Student("Ioana", 8.35, "Vatra-Dornei")};
        return Arrays.asList(lstStud);

    }
    public Map<String, List<Student>> getMapNume() {
        return mapNume;
    }
    public Map<Double, List<Student>> getMapMedie() {
        return mapMedie;
    }
    public Map<String, List<Student>> getMapLocalitate() {
        return mapLocalitate;
    }
}
```

Clasa principala: OBradFrumos

Realizeaza aplicatia grafica si are cod pentru a reactiona la schimbarea item-ului selectat su la modificari ale valorii Slider-ului.

In aceasta clasa veti avea mai mult cod de completat.

```
package obradfrumos;
import java.io.FileNotFoundException;
import java.util.List;
import java.util.Map;
import javafx.application.Application;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.control.TextArea;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class NewClass extends Application {
    private static TextArea rezultatCautare;
    private static Label nrInregistrari;
    private ImageView imageView = new ImageView();
    private Text topText;
    private Label eticValSelectata;
    private ListaMosului listaMosului;
//
// puteti COMPLETA cu campuri si metode
//

    private static TextArea getTextArea(String textInitial) {
```

```
    TextArea ta = new TextArea();
    ta.appendText(textInitial);
    ta.setPrefWidth(400);
    ta.setPrefHeight(370);
    ta.setWrapText(true);
    return ta;
}
private ChoiceBox getChoiceBox(Map<String, List<Student>> map) {
    ObservableList<String> listaItemi;
   //
   //   DE COMPLETAT  listaItemi cu valorile cheilor din map
   //
   // sugestie clasa ConsultareDictionar din aplicatia Dictionar ilustrat prezentata
   // la cursul Controale grafice
   //
    ChoiceBox<String> list = new ChoiceBox<>(listaItemi);

    list.getSelectionModel().selectedItemProperty()
          .addListener(new SelectareItem(map));
  // este necesar sa scrieti clasa SelectareItem care sa implementeze ChangeListener
  //  scheletul clasei SelectareItem este prezentat in continuare
    return list;
}

Slider getSlider() {
    Slider valMedie;
   //
   //    DE COMPLETAT
   //  sugestie pentru Slider: cursul Scurt tutorial JavaFX,
   // aplicatia FORMULARE CU CONTROALE GRAFICE DIVERSE
   //

  valMedie.valueProperty().addListener ( new ChangeListener<Number>()  {
      @Override
      public void changed(ObservableValue<? extends Number> ov,
                                                    Number old_val, Number new_val) {
          double valSelectata = new_val.doubleValue();
          eticValSelectata.setText(String.format("Stud. cu media >= %.2f", new_val));
          List <Student> lstDupaMedie;
          //
         //  DE COMPLETAT cu instructiunile pentru obtinerea listei studentilor
         //  lstDupaMedie care au media cel putin egala cu new_val
         //
          afisareRezultat(lstDupaMedie);
      }
    });
    return valMedie;
}

// package protected
static void afisareRezultat(List<Student> lstud) {
```

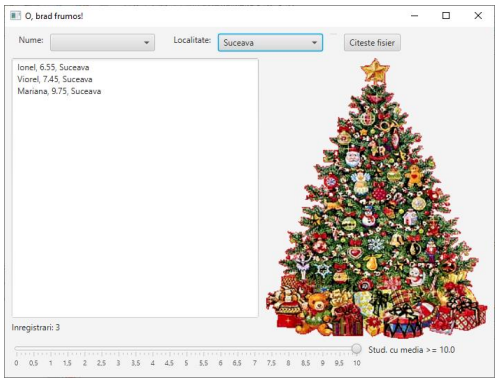
```
      //
      // DE COMPLETAT
      //
   }

   private BorderPane getPanou() {
      BorderPane panou = new BorderPane();
      //
      //    DE COMPLETAT
      //
      // sugestie: aplicatia Dictionar ilustrat prezentata
      // la cursul Controale grafice
      //
      Slider valMedie = getSlider();
      // sugestie pentru Slider: cursul Scurt tutorial JavaFX,
      // aplicatia FORMULARE CU CONTROALE GRAFICE DIVERSE
      eticValSelectata = new Label(" . . .  DE COMPLETAT . . . " );

      panou.setBottom(new HBox(10, valMedie, eticValSelectata));
      return panou;
   }

   @Override
   public void start(Stage primaryStage) throws FileNotFoundException {
      listaMosului = new ListaMosului();
      Scene scena = new Scene(getPanou(), 700, 500);
      primaryStage.setScene(scena);
      primaryStage.setTitle("O, brad frumos!");
      primaryStage.show();
   }

   public static void main(String[] args) {
      launch(args);
   }

}
```

Clasa SelectareItem

```
package obradfrumos;
import java.util.List;
import java.util.Map;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
public class SelectareItem implements ChangeListener<String>{
   Map<?, List<Student>> map;
            //
            // DE COMPLETAT
            //
  }
```

TEMA DE EFECTUAT

1. (**4p**) Scrieti codul necesar pentru a afisa panoul cu toate controalele grafice



Punctaj: **1p** pentru cele 2 ChoiceBox + buton, **1p** slider, **1p** TextArea si imaginea, **1p** dispunerea se mentine si la maximizarea ferestrei).

2. **(3p)** Controalele grafice reactioneaza si afiseaza datele de test (1p – nume, 1p – localitate, 1p slider medie).
3. **(3p)** Controalele grafice reactioneaza si afiseaza datele din Fisier – campurile fiind separate prin virgula). Atentie. **Fereastra pentru citirea fisierului va apare inainte de afisarea panoului grafic. Daca utilizatorul actioneaza butonul Anulare (Cancel) se vor incarca datele de test.** Dupa ce s-a stabilit setul de date urmeaza afisarea panoului graphic. **Procesarea** actionarii butonului *"Citeste fisier"* este ultimul punct al Temei pentru acasa.

IMPORTANT

Dupa cum banuiti, nimeni nu cunoaste lista cu care lucreaza Mosul.

Fisierul pus la dispozitie contine 1001 de inregistrari fictive, fara nici o legatura cu realitatea si are doar scopul de a va putea testa aplicatia cu un astfel de volum de date.

Coincidenta dintre unele informatii din fisier si persoane reale este cu totul si cu totul intamplatoare!

Punctaj: 1p citire fisier si afisare in TextArea, 0.5p – nume, 0.5p – localitate, 1p slider medie).

TEMA ACASA

1. **(3p)** Adaugati un camp text inainte de ChoiceBox-ul cu nume, in care sa puteti introduce un sir. Acest camp va fi sensibil la modificarile de continut si va popula ChoiceBox-ul doar cu acele nume din lista care contin subsirul din TextField.
2. **(3p)** Introduceti pe aceeasi linie cu eticheta cu nr. de inegistrari un control CheckBox cu textul "*Se ia in considerare SI media*" Daca este selectat acest CheckBox in zona TextArea vor fi afisati studentii
    - care au un anumit nume si au media>= valoarea selectata cu slider, sau

- cei care sunt dintr-o anumita localitate si au media>= valoarea selectata cu slider

functie de care ChoiceBox (Nume sau Localitate) a fost utilizat. Afisarea in TextArea se modifica ori de cate ori se modifica una din cele 3 selectii.

3. **(3p)** Introduceti pe aceeasi linie un nou control CheckBox cu textul “Toate”. In acest caz vor fi afisati studentii care indeplinesc ***simultan toate cel 3 conditii***: sa aiba numele identic cu cel selectat, localitatea identica cu cea selectata si media>= valoarea selectata cu slider. Afisarea in TextArea se modifica ori de cate ori se modifica una din cele 3 selectii.
4. **(3p)** Modificarea datelor cu care lucreaza aplicatia prin citirea unui nou fisier in timpul executiei la actionarea butonului *“Citeste fisier”*.
   ***Indicatie**. Pentru a schimba dinamic item-ii dintr-o lista de selectie (ChoiceBox) puteti utiliza listaSelectie.getItems().**setAll(colectie)** care va face reinitializarea listei (clear()) si va adauga elementele din colectie.*
   *Probabil veti lucra cu aceleasi 3 Map-uri (nume, localitate, medie) pe tot timpul executiei programului. Atunci reincarcarea lor cu datele din fisier trebuie sa fie precedata de reinitializarea lor cu map**.clear()** pentru a le goli de vechiul continut*.