E-Church Agenda

Paltenie Adrian Valentin,

Peligrad Cristian,

Grupa 4

**Descrierea problemei**:

Aplicația software E-ChurchAgenda este un instrument util oricărui preot slujitor la o parohie.

Grupurile de functionalități sunt prezentate în continuare:

* Planificarea unui serviciu religios se face de către preot și constă în specificarea datei, a orei, numele enoriașului, numărului de telefon și adresa.
* La cererea utilizatorului, aplicația afișează lista slujbelor/serviciilor religioase din ziua respectivă. Aceasta listă este listată la imprimantă de catre aplicație.
* Editor de predici/anunțuri. User-ul poate crea o predică/anunț nouă/nou, o/îl poate modifica, vizualiza sau șterge.
* La cererea utilizatorului, aplicația afișează numele sfinților pomeniți în ziua curentă.

**Cerințele proiectului:**

1. Să se creeze diagrama UML de clase a aplicației E-Church Agenda.

2. Să se implementeze in Java aplicația E-Church Agenda.

3. Aplicația trebuie să îndeplinească toate funcționalitățile specificate în descrierea problemei.

4. Aplicația memorează pe suport extern (fișiere) toate datele necesare funcționării sale astfel încât existența datelor nu depinde de oprirea temporară (accidentală sau nu) a calculatoarelor pe care rulează.

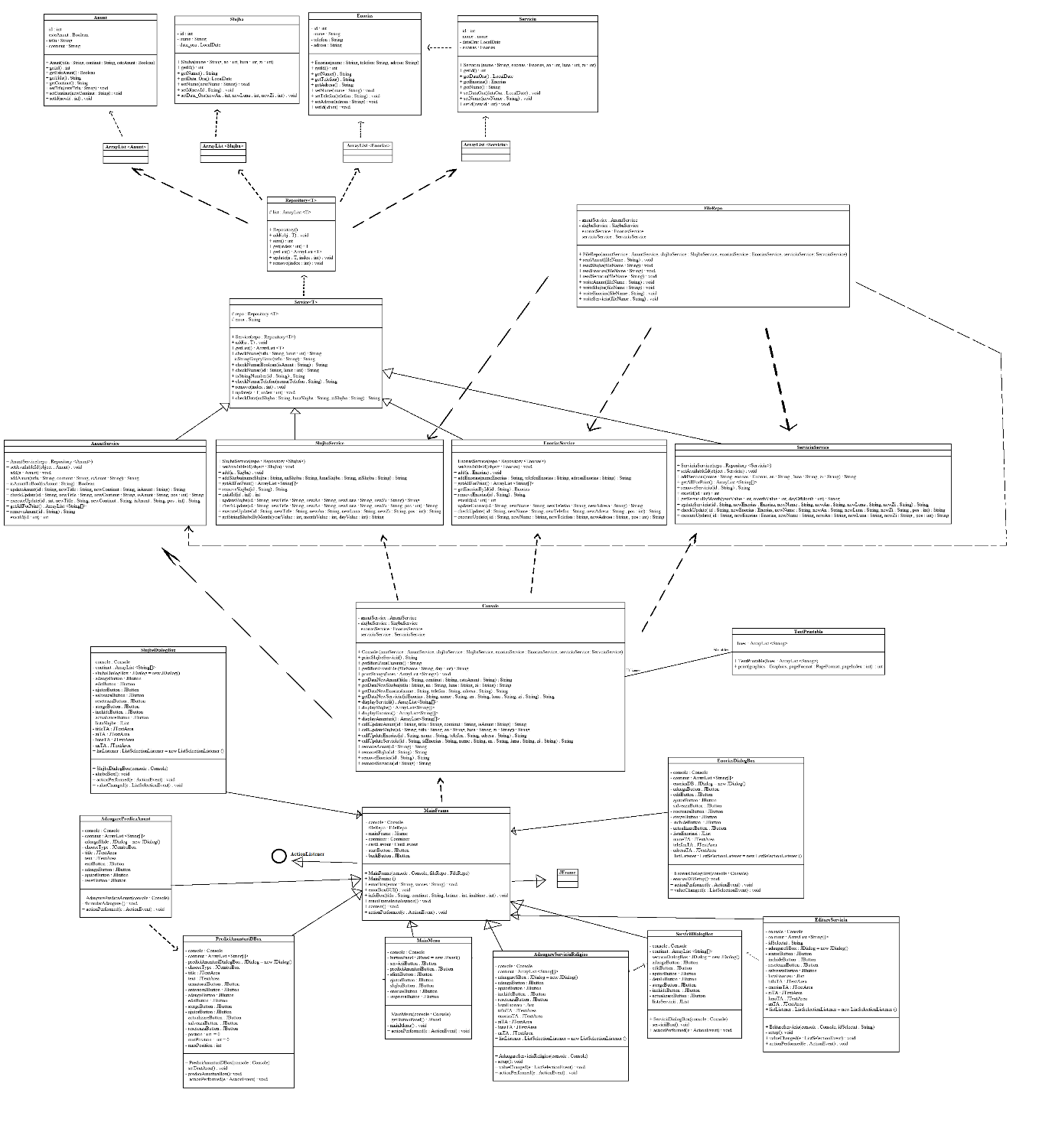
5. Pentru realizarea interfețelor grafice ale aplicației va fi folosită tehnologia Swing.

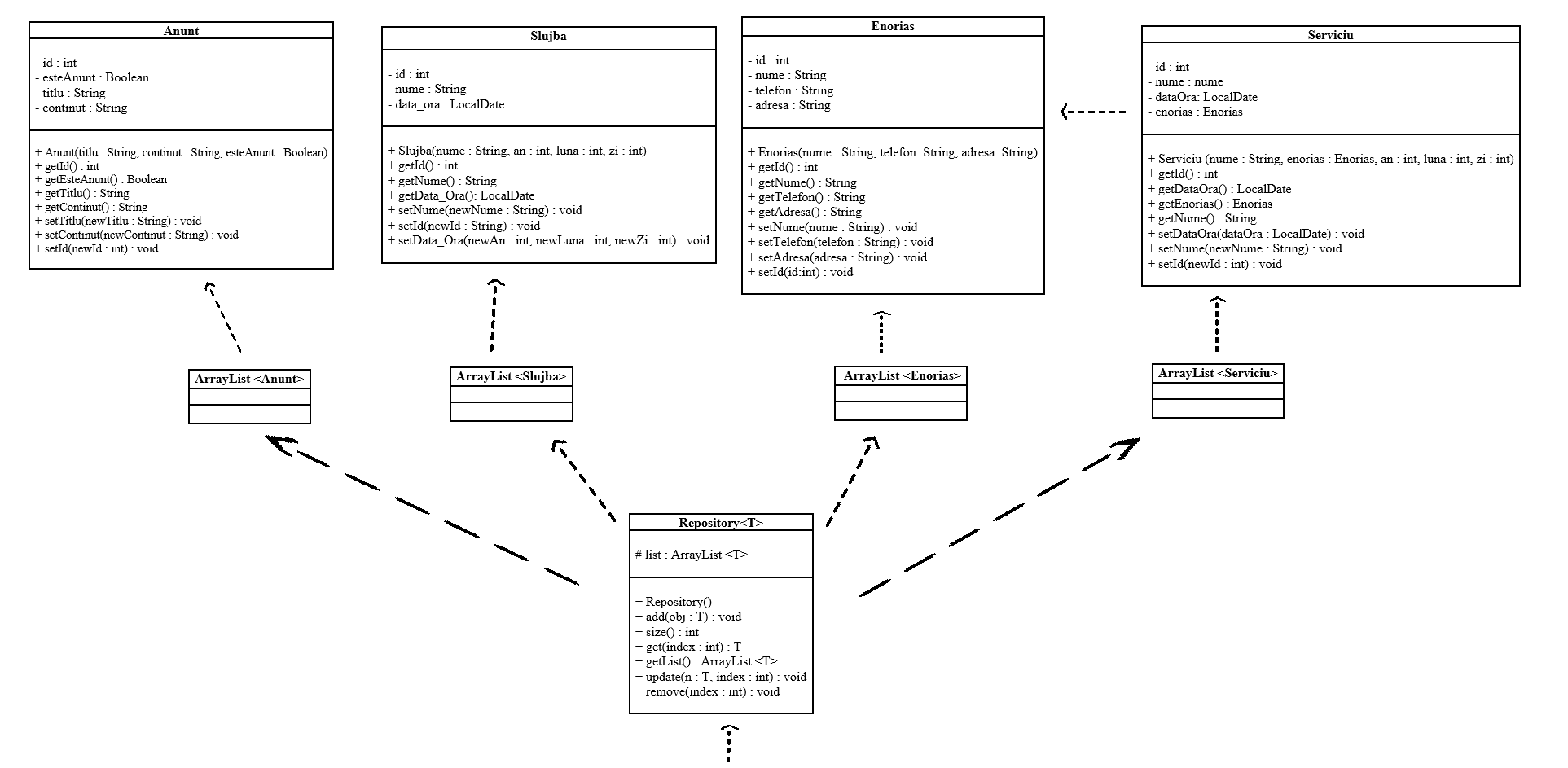
6. Pentru realizarea diagramei UML de clase și a interfețelor grafice ale aplicației nu vor fi folosite programe software care facilitează acest lucru.

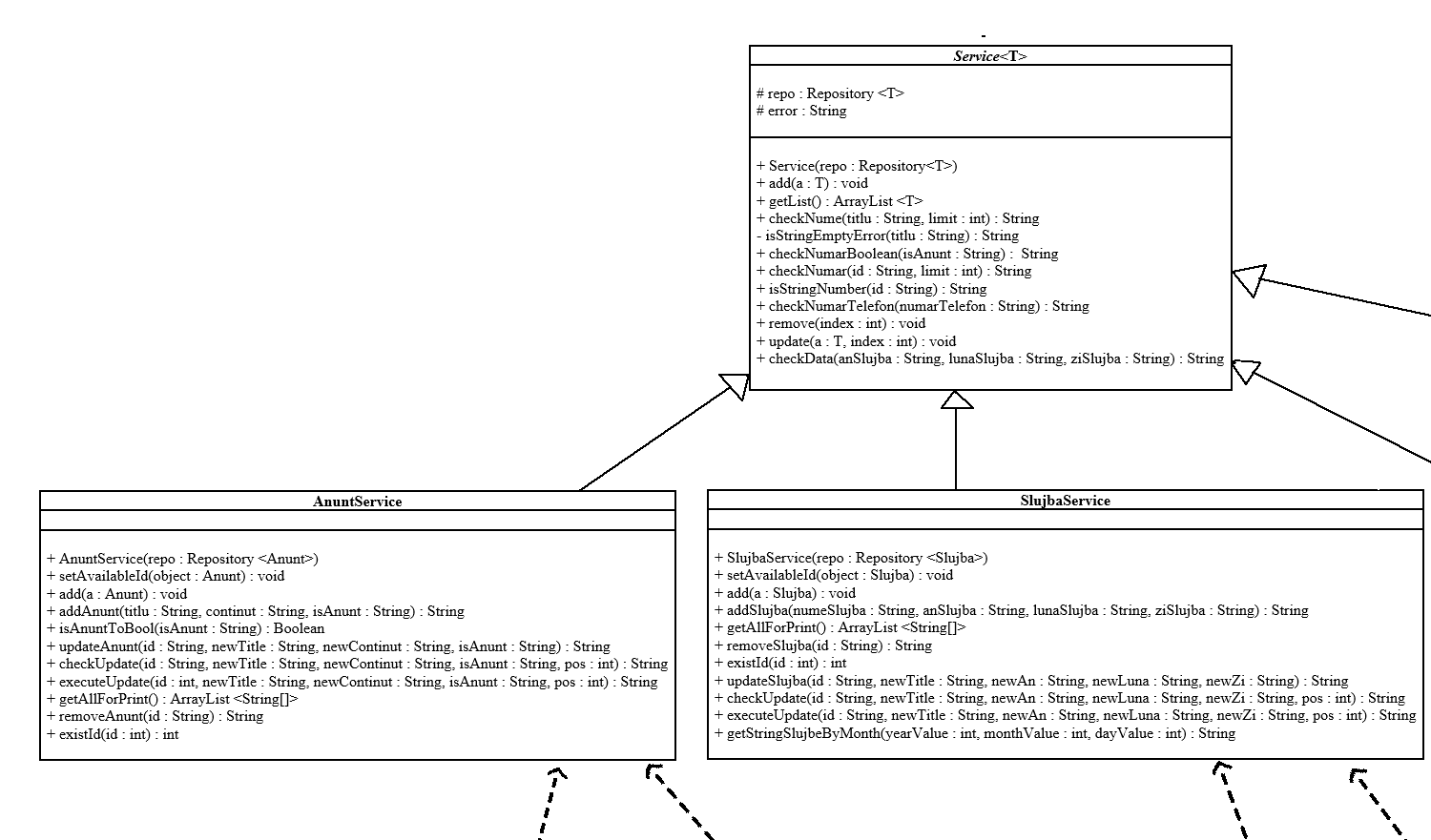
7. Pentru gestiunea evenimentelor generate de utilizatorii aplicației vor fi folosite clase interne.

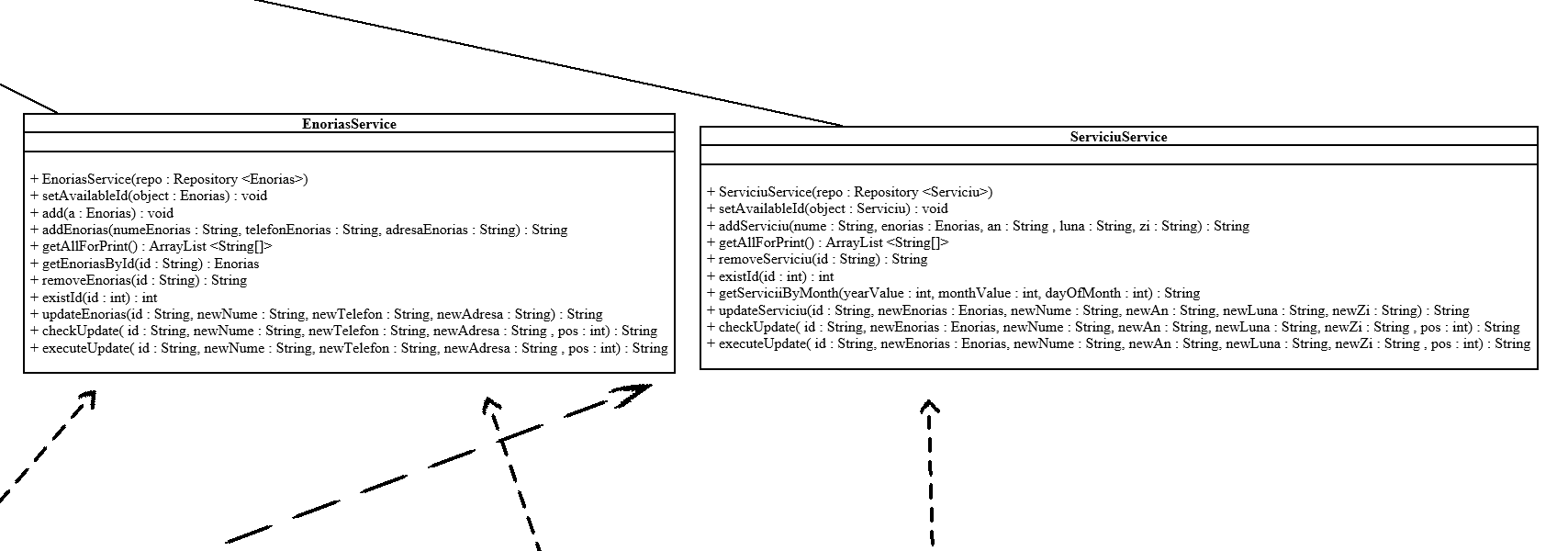
8. Pentru gestiunea predicilor, etc. vor fi folosite colecții dinamice de obiecte.

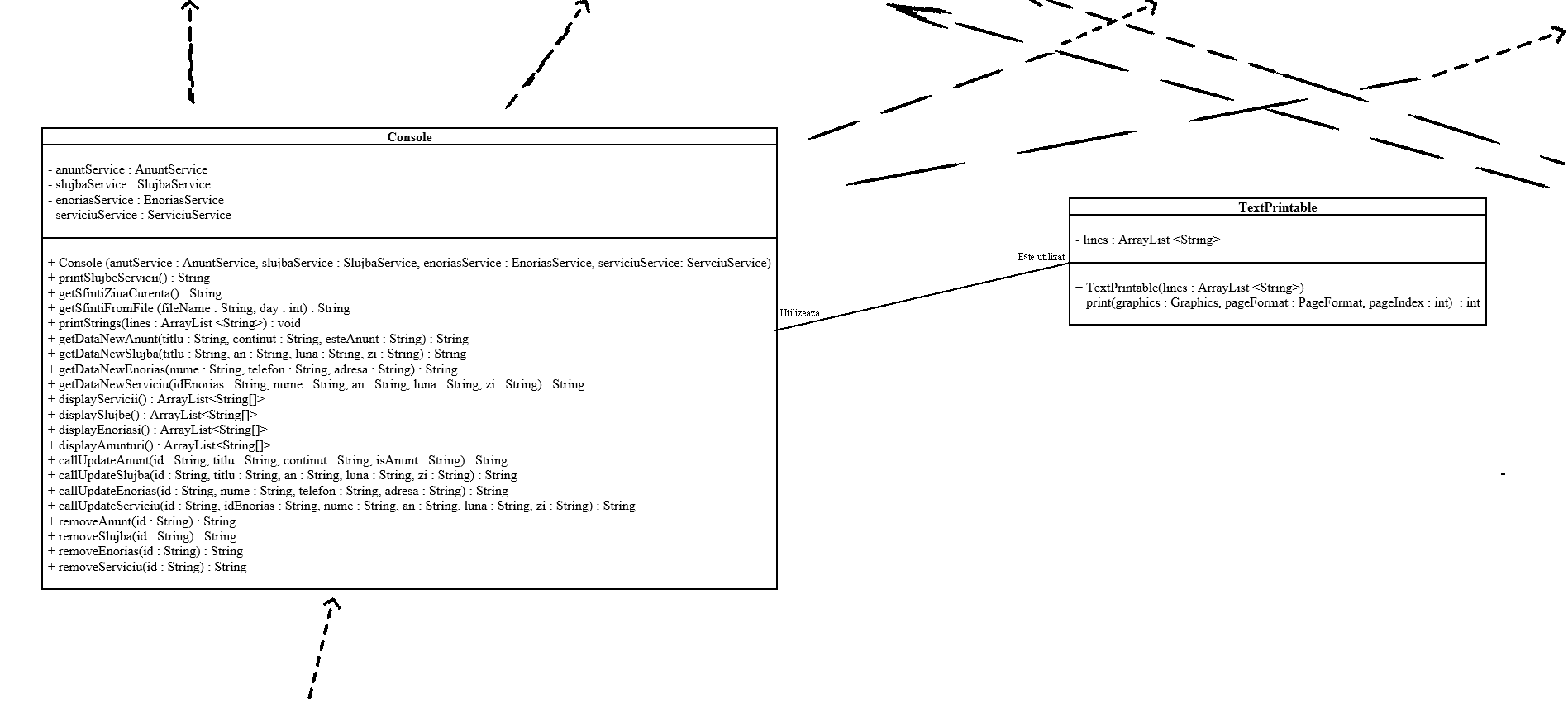
UML

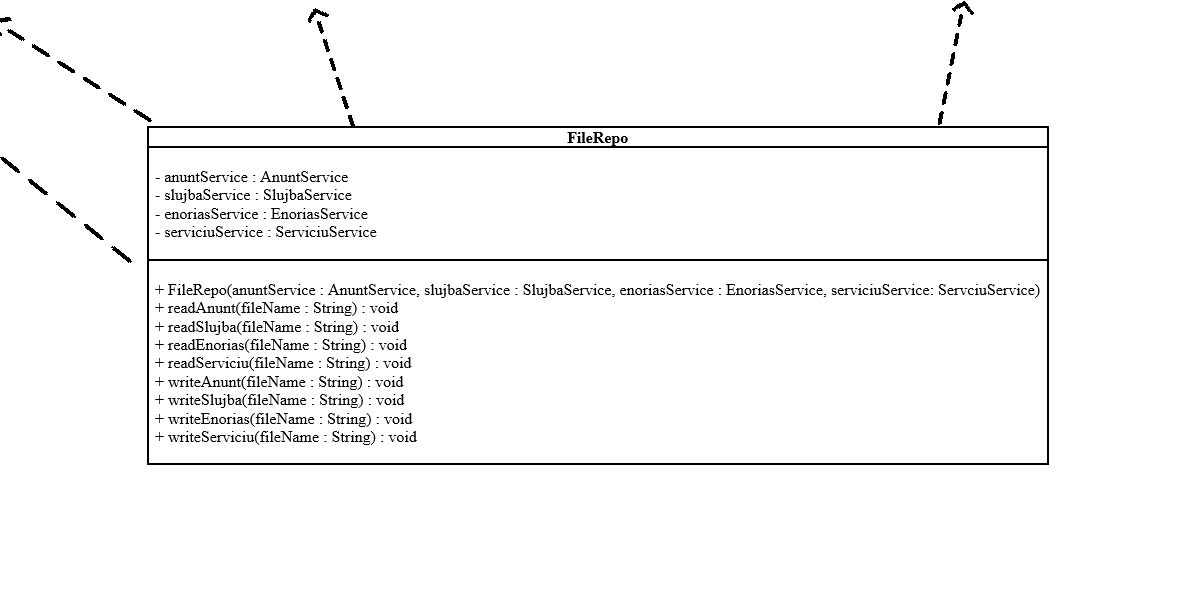


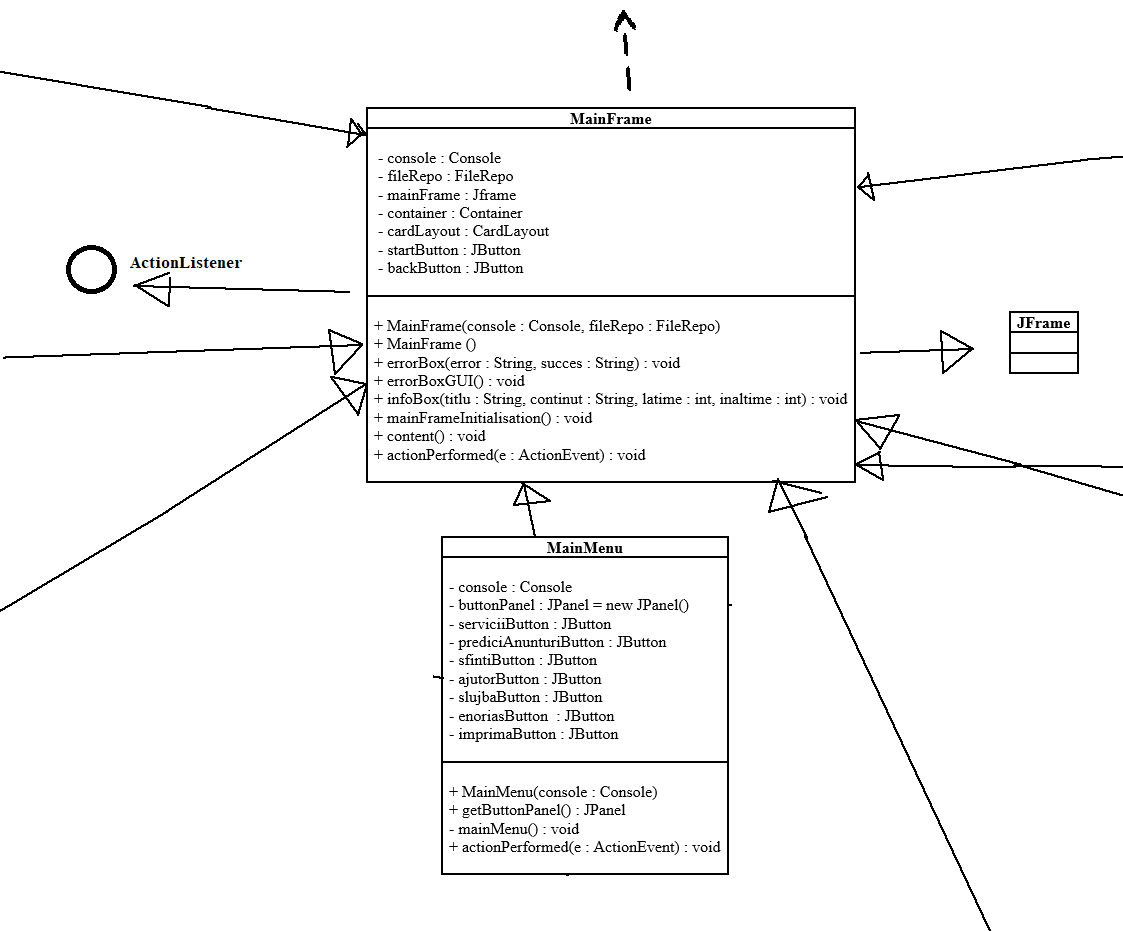


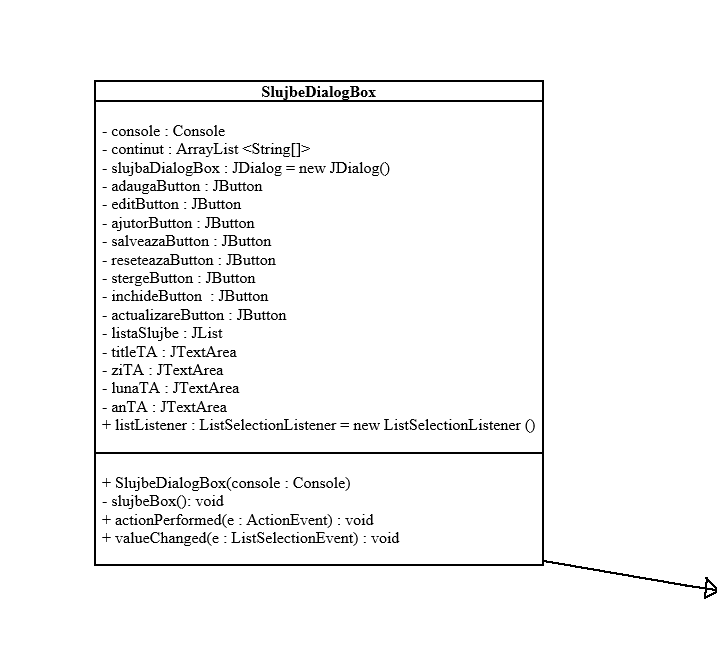


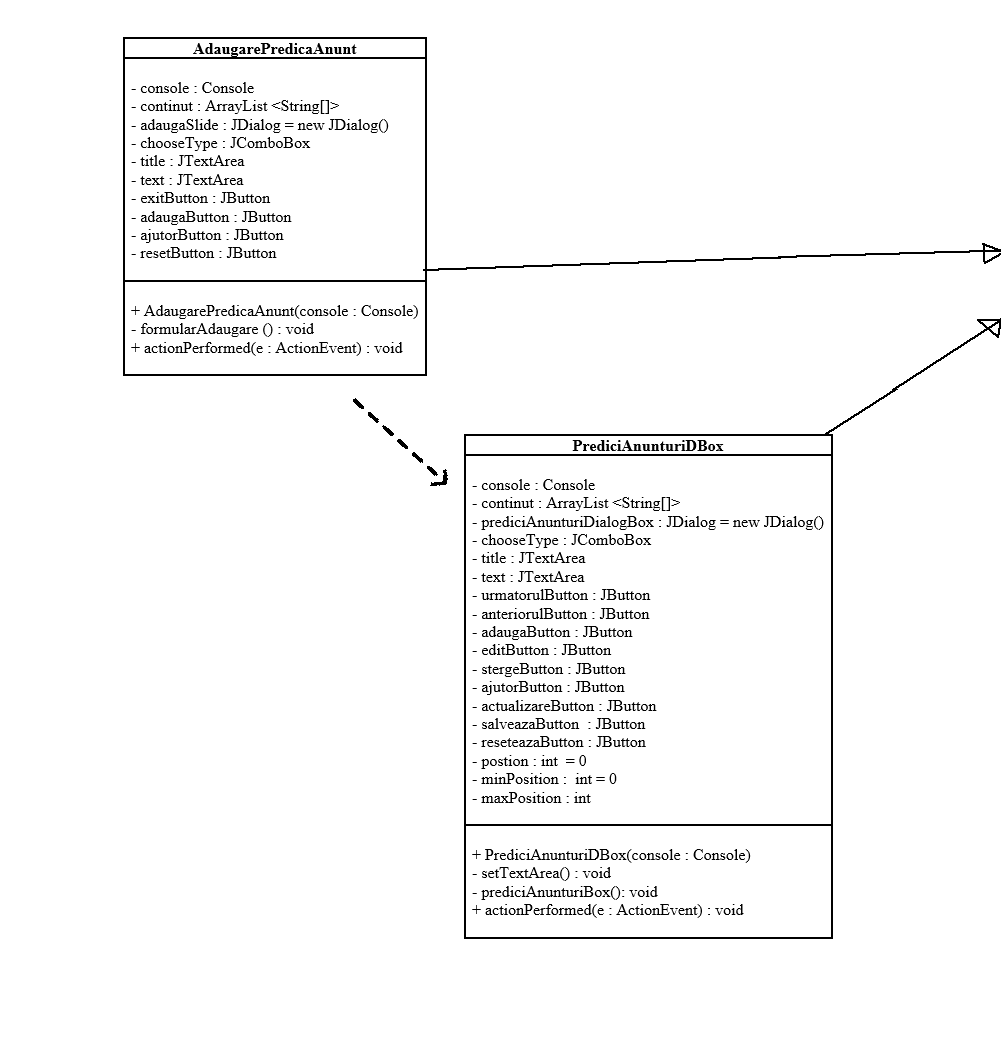


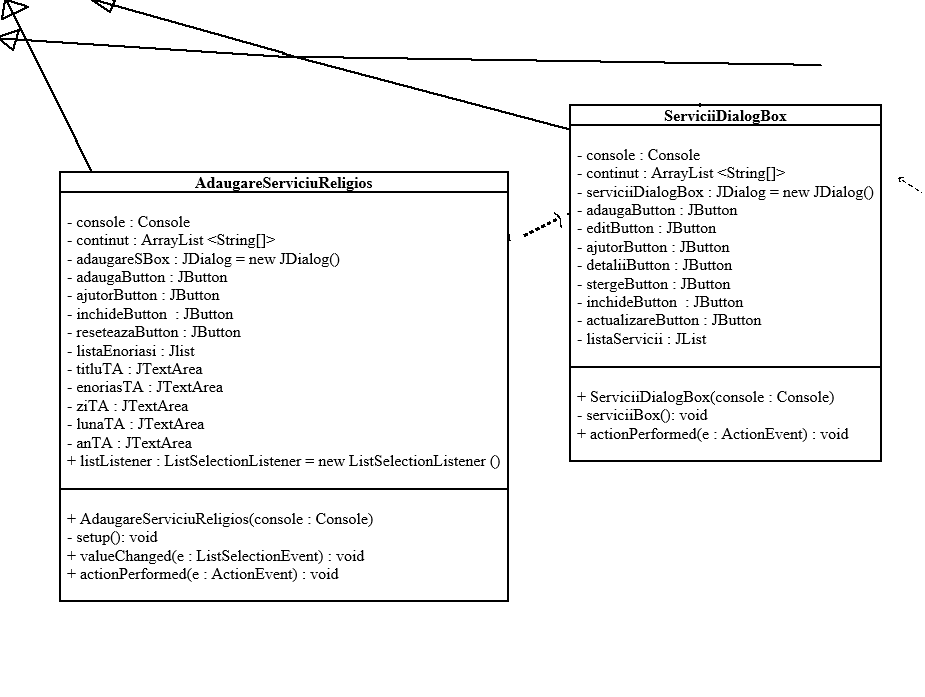


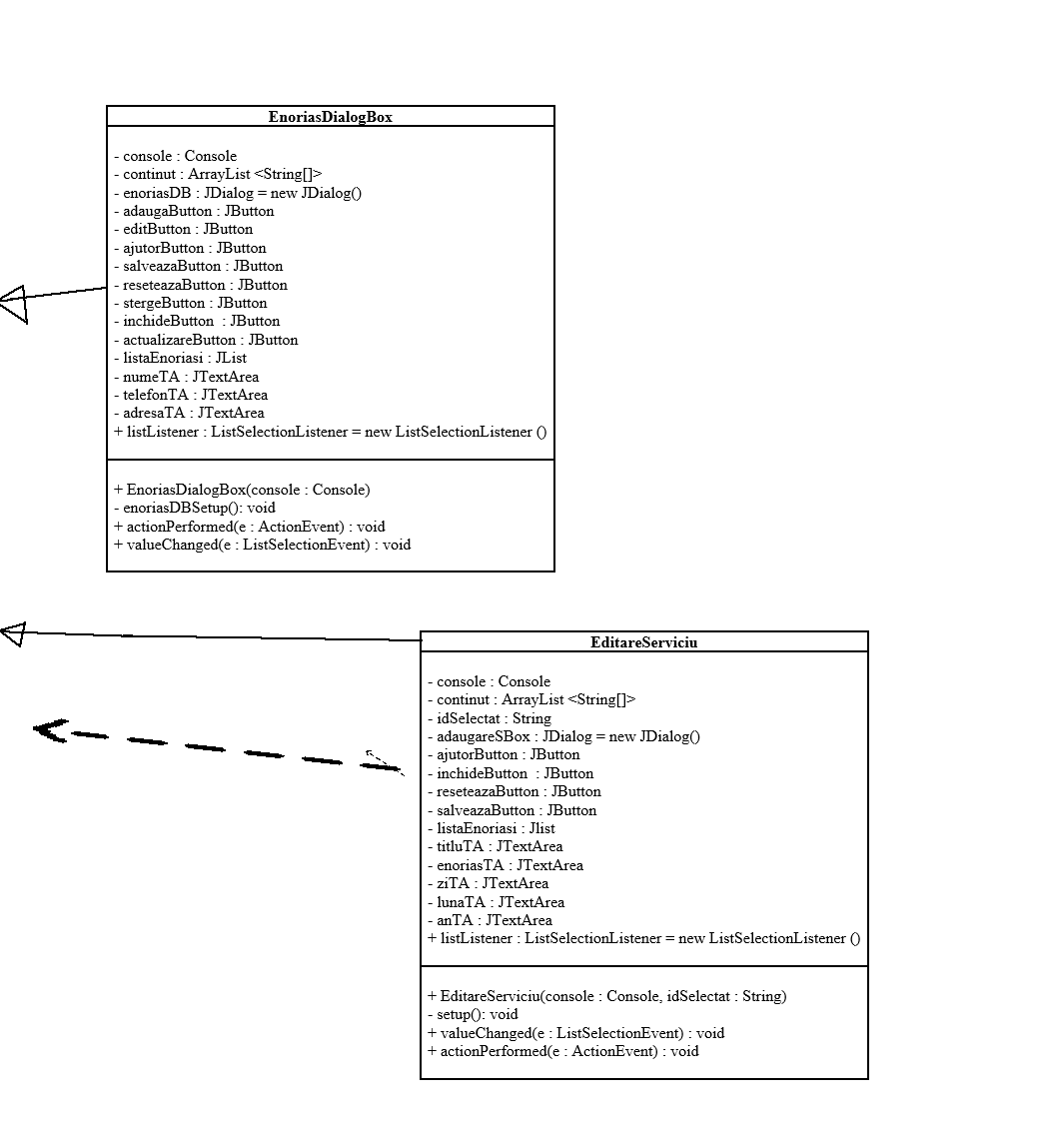












Clasele și conținutul lor

**Enorias :**

public class Enorias {

private int id;

private String nume,telefon,adresa;

public Enorias(String nume, String telefon, String adresa)

{

id = 1;

this.nume = nume;

this.telefon = telefon;

this.adresa = adresa;

}

public int getId()

{

return id;

}

public String getNume()

{

return nume;

}

public String getTelefon()

{

return telefon;

}

public String getAdresa()

{

return adresa;

}

public void setNume( String nume)

{

this.nume = nume;

}

public void setTelefon(String telefon)

{

this.telefon = telefon;

}

public void setAdresa(String adresa)

{

this.adresa = adresa;

}

public void setId(int id) {

this.id=id;

}

}

**Serviciu :**

import java.time.LocalDate;

public class Serviciu {

private int id;

private String nume;

private LocalDate dataOra;

private Enorias enorias;

public Serviciu(String nume,Enorias enorias,int an, int luna, int zi)

{

id =1;

this.nume = nume;

this.dataOra=LocalDate.of(an, luna, zi);

this.enorias = enorias;

}

public int getId()

{

return id;

}

public LocalDate getDataOra()

{

return dataOra;

}

public Enorias getEnorias()

{

return enorias;

}

public String getNume()

{

return nume;

}

public void setNume(String newNume)

{

this.nume=newNume;

}

public void setDataOra(LocalDate dataOra)

{

this.dataOra = dataOra;

}

public void setId(int newId)

{

this.id = newId;

}

}

**Slujba:**

import java.time.LocalDate;

public class Slujba {

private int id;

private String nume;

private LocalDate data\_ora;

public Slujba(String nume, int an, int luna, int zi)

{

this.nume=nume;

this.data\_ora=LocalDate.of(an, luna, zi);

this.id=1;

}

public void setId(int newId)

{

this.id= newId;

}

public void setNume(String newNume)

{

this.nume=newNume;

}

public void setData\_Ora(int newAn,int newLuna,int newZi)

{

this.data\_ora=LocalDate.of(newAn, newLuna, newZi);

}

public int getId()

{

return id;

}

public String getNume()

{

return nume;

}

public LocalDate getData\_Ora()

{

return data\_ora;

}

}

**Anunt:**

import java.time.LocalDate;

public class Anunt {

private int id;

private String titlu;

private String continut;

private boolean esteAnunt;

public Anunt(String titlu, String continut, boolean esteAnunt)

{

this.titlu = titlu;

this.continut = continut;

this.esteAnunt = esteAnunt;

this.id=1;

}

public void setTitlu(String newTitlu)

{

this.titlu = newTitlu;

}

public void setContinut(String newContinut)

{

this.continut = newContinut;

}

public void setId(int newId)

{

this.id = newId;

}

public void setEsteAnunt(boolean x)

{

this.esteAnunt = x;

}

public int getId()

{

return id;

}

public String getTitlu()

{

return titlu;

}

public String getContinut()

{

return continut;

}

public boolean getEsteAnunt()

{

return esteAnunt;

}

}

**Repository:**

import java.util.ArrayList;

public class Repository<T> {

protected ArrayList<T> list;

public Repository()

{

list = new ArrayList<T>();

}

public void add(T obj) //am schimbat numele la functie din ,,addObj" in add ca e mai usor

{

this.list.add(obj);

}

public int size()

{

return this.list.size();

}

public T get(int index) {

return list.get(index);

}

public ArrayList<T> getList()

{

return this.list;

}

public void update(T n , int index)

{

list.set(index, n);

}

public void remove(int index) {

list.remove(index);

}

}

**Service:**

import java.util.ArrayList;

import java.time.LocalDate;

public abstract class Service<T> {

// Elementul principal din layer 3 , primeste la fiecare intializare un Repository generic

// cu obiectul necesar, si contine functii care se folosesc de un obiect generic precum si

// functii de verificare al input ului care sunt folosite de toate Service urile specifice ce

// extind aceasta clasa

protected Repository<T> repo;

protected String error;

public Service(Repository<T> repo)

{

this.repo = repo;

}

public void add(T a) {

repo.add(a);

}

public ArrayList<T> getList()

{

return this.repo.getList();

}

public String checkNume(String titlu, int limit) {

// Cheama toate functiile necesare verificarii unui String ce trebuie sa

// inceapa cu litera mare , sa nu fie gol , iar primul caracter sa fie o litera

// Returneaza un string util pentru GUI

try {

if (titlu.isEmpty()) {

throw new IllegalArgumentException("Input must not be empty");

}

if (Character.isDigit(titlu.charAt(0))) {

throw new IllegalArgumentException("First character must be a letter");

}

if (!Character.isUpperCase(titlu.charAt(0))) {

throw new IllegalArgumentException("First letter must be uppercase");

}

if (titlu.length() > limit) {

throw new IllegalArgumentException("Input must be less than " + (limit + 1) + " characters");

}

return "valid!";

} catch (IllegalArgumentException ex) {

return ex.getMessage();

}

}

private String isStringEmptyError(String titlu) {

// Verifica daca input ul este un string gol ("")

try {

if (titlu.isEmpty()) {

throw new IllegalArgumentException("Input must not be empty");

}

return "valid!";

} catch (IllegalArgumentException ex) {

return ex.getMessage();

}

}

public String checkNumarBoolean(String isAnunt) {

// Aceasta functie verifica daca input ul este 1 sau 2 , o conventie formata in consola in care 1 reprezinta false

// iar 2 reprezinta true , functia a fost utila la implementarea aplicatiei fara GUI , in baza unei console

try {

if (!isAnunt.equals("1") && !isAnunt.equals("2")) {

throw new IllegalArgumentException("Value should be 1 or 2");

}

return "valid!";

} catch (IllegalArgumentException ex) {

return ex.getMessage();

}

}

public String checkNumar(String id,int limit) {

// Functia primeste input ul specific de la consola ca si String si limita numarului (necesara pentru zii luna an)

String error = new String();

error= this.isStringEmptyError(id);

if(error!="valid!") return error;

error = this.isStringNumber(id);

if(error!="valid!") return error;

return error;

}

public String isStringNumber(String id) {

//Verifica daca input ul de la GUI este un numar si returneaza String uri utile

try {

Integer.parseInt(id);

return "valid!";

} catch (NumberFormatException e) {

return "Input must be integer";

}

}

public String checkNumarTelefon(String numarTelefon) {

// Folosita doar in cazul enoriasului , dar este creata in Service ul generic pentru

// a putea fii apelata de toate celelalte service uri

// Un numar de telefon trebuie sa inceapa cu 07 si sa aiba 10 caractere

try {

if (numarTelefon.length() == 10 && numarTelefon.startsWith("07")) {

for (int i = 0; i < 10; i++) {

if (!Character.isDigit(numarTelefon.charAt(i))) {

return "invalid";

}

}

return "valid!";

} else {

return "invalid";

}

} catch (Exception e) {

return "invalid";

}

}

// Funtia de remove apeleaza repo ul generic si se foloseste de functiile integrate in ArrayList

public void remove(int index)

{

repo.remove(index);

}

// Funtia de update apeleaza repo ul generic si se foloseste de functiile integrate in ArrayList

public void update(T a , int index)

{

repo.update(a, index);

}

public String checkData(String anSlujba, String lunaSlujba, String ziSlujba) {

//Verifica daca input ul de la GUI este o data valabila calendaristic

//returneaza erori utile

try

{

LocalDate data = LocalDate.of(Integer.parseInt(anSlujba),Integer.parseInt(lunaSlujba),Integer.parseInt(ziSlujba));

return "valid!";

}

catch (Exception e) {

return "Date does not exist";

}

}

}

**EnoriasService:**

import java.time.LocalDate;

import java.util.ArrayList;

public class EnoriasService extends Service {

public void setAvailableId(Enorias object) {

ArrayList<Enorias> enoriasList = super.getList();

int maxId = 0;

for (int i=0;i<enoriasList.size();i++) {

maxId = Math.max(maxId, enoriasList.get(i).getId());

}

int[] existingIds = new int[maxId + 1];

for (int i=0;i<enoriasList.size();i++) {

existingIds[enoriasList.get(i).getId() - 1] = 1;

}

for (int i = 0; i <= maxId; i++) {

if (existingIds[i] == 0) {

object.setId(i + 1);

break;

}

}

}

public EnoriasService(Repository<Enorias> repo)

{

super(repo);

}

public void add(Enorias a) {

super.repo.add(a);

}

public String addEnorias(String numeEnorias,String telefonEnorias , String adresaEnorias)

{

String checkError = super.checkNume(numeEnorias, 25);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(adresaEnorias, 50);

if (checkError!="valid!") return checkError;

checkError = super.checkNumarTelefon(telefonEnorias);

if (checkError!="valid!") return checkError;

Enorias a = new Enorias(numeEnorias,telefonEnorias,adresaEnorias);

this.add(a);

this.setAvailableId(a);

return "valid!";

}

public int existId(int id) {

try {

ArrayList<Enorias> lista = this.repo.getList();

for (int i = 0; i < lista.size(); i++) {

if (id == lista.get(i).getId()) {

return i;

}

}

throw new IllegalArgumentException("Id does not exist");

} catch (IllegalArgumentException ex) {

return -1;

}

}

public ArrayList<String[]> getAllForPrint() {

ArrayList<Enorias> listaPrint = super.getList();

ArrayList<String[]> result = new ArrayList<String[]>();

for(int i=0;i<listaPrint.size();i++)

{

String[] object = new String[5];

object[0] = String.valueOf(listaPrint.get(i).getId());

object[1] = listaPrint.get(i).getNume();

object[2] = listaPrint.get(i).getTelefon();

object[3] = listaPrint.get(i).getAdresa();

result.add(object);

}

return result;

}

public Enorias getEnoriasById(String id)

{

String checkError = super.checkNumar(id, 5000);

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

ArrayList<Enorias> lista = super.getList();

return lista.get(pos);

}

}

return null;

}

public String removeEnorias(String id) {

String checkError = super.checkNumar(id,5000);

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

super.remove(pos);

return "valid!";

}

return checkError;

}

return checkError;

}

public String updateEnorias(String id, String newNume, String newTelefon, String newAdresa) {

String checkError = super.checkNumar(id,5000);

int pos = this.existId(Integer.parseInt(id));

if(checkError=="valid!")

{

if(pos!=-1)

{

checkError = this.checkupdate(id,newNume,newTelefon,newAdresa,pos);

return checkError;

}

return "id does not exist";

}

return checkError;

}

private String checkupdate(String id, String newNume, String newTelefon, String newAdresa, int pos) {

String checkError = super.checkNume(newNume,25);

if (checkError!="valid!") return checkError;

checkError = super.checkNumarTelefon(newTelefon);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(newAdresa, 50);

return executeUpdate(id,newNume,newTelefon,newAdresa,pos);

}

private String executeUpdate(String id, String newNume, String newTelefon, String newAdresa, int pos) {

ArrayList<Enorias> lista = super.getList();

Enorias a = new Enorias(newNume,newTelefon,newAdresa);

a.setId(Integer.parseInt(id));

//carbonara(a);

super.update(a, pos);

return "valid!";

}

}

**ServiciuService:**

import java.time.LocalDate;

import java.util.ArrayList;

public class ServiciuService extends Service {

public ServiciuService(Repository repo) {

super(repo);

}

public void setAvailableId(Serviciu object) {

ArrayList<Serviciu> serviciuList = super.getList();

int maxId = 0;

for (int i=0;i<serviciuList.size();i++) {

maxId = Math.max(maxId, serviciuList.get(i).getId());

}

int[] existingIds = new int[maxId + 1];

for (int i=0;i<serviciuList.size();i++) {

existingIds[serviciuList.get(i).getId() - 1] = 1;

}

for (int i = 0; i <= maxId; i++) {

if (existingIds[i] == 0) {

object.setId(i + 1);

break;

}

}

}

public String addServiciu(String nume,Enorias enorias, String an, String luna, String zi) {

if(enorias==null) return "id of enorias does not exist";

else

{

String error = super.checkNumar(an, 5000);

if(error!="valid!") return error;

error = super.checkNumar(luna, 13);

if(error!="valid!") return error;

error = super.checkNumar(zi, 32);

if(error!="valid!") return error;

error = super.checkData(an, luna, zi);

if(error!="valid!") return error;

Serviciu a = new Serviciu(nume,enorias,Integer.parseInt(an), Integer.parseInt(luna), Integer.parseInt(zi));

super.add(a);

setAvailableId(a);

return "valid!";

}

}

public String removeServiciu(String id) {

String checkError = super.checkNumar(id,5000);

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

super.remove(pos);

return "valid!";

}

return checkError;

}

return checkError;

}

public String updateServiciu(String id, Enorias newEnorias , String newNume, String newAn,String newLuna,String newZi){

String checkError = super.checkNumar(id,5000);

if(checkError!="valid!") return checkError;

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

checkError = this.checkUpdate(id, newEnorias, newNume, newAn,newLuna,newZi,pos);

return checkError;

}

return "Id of serviciu does not exist";

}

return checkError;

}

public String checkUpdate(String id, Enorias newEnorias, String newNume, String newAn, String newLuna, String newZii , int pos)

{

String checkError = super.checkNumar(id,5000);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(newNume,25);

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(newAn, 5000);

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(newLuna, 13);

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(newZii, 32);

if (checkError!="valid!") return checkError;

return executeUpdate(id, newEnorias, newNume, newAn,newLuna,newZii,pos) ;

}

public String executeUpdate(String id, Enorias newEnorias, String newNume, String newAn, String newLuna, String newZii , int pos)

{

Serviciu a = new Serviciu(newNume,newEnorias,Integer.parseInt(newAn),Integer.parseInt(newLuna),Integer.parseInt(newZii));

a.setId(Integer.parseInt(id));

super.update(a, pos);

return "valid!";

}

public int existId(int id) {

try {

ArrayList<Serviciu> lista = this.repo.getList();

for (int i = 0; i < lista.size(); i++) {

if (id == lista.get(i).getId()) {

return i;

}

}

throw new IllegalArgumentException("Id does not exist");

} catch (IllegalArgumentException ex) {

return -1;

}

}

public ArrayList<String[]> getAllForPrint() {

ArrayList<Serviciu> listaPrint = super.getList();

ArrayList<String[]> result = new ArrayList<String[]>();

for(int i=0;i<listaPrint.size();i++)

{

String[] object = new String[6];

object[0] = String.valueOf(listaPrint.get(i).getId());

object[1] = listaPrint.get(i).getNume();

object[2] =String.valueOf(listaPrint.get(i).getDataOra().getYear()) ;

object[3] =String.valueOf(listaPrint.get(i).getDataOra().getMonthValue()) ;

object[4] =String.valueOf(listaPrint.get(i).getDataOra().getDayOfMonth()) ;

object[5] =listaPrint.get(i).getEnorias().getNume();

result.add(object);

}

return result;

}

public String getServiciiByMonth(int yearValue, int monthValue, int dayOfMonth) {

String result = new String();

ArrayList<Serviciu> lista = super.getList();

boolean flag = false;

for(int i=0;i<lista.size();i++)

{

if(lista.get(i).getDataOra().getYear()==yearValue && lista.get(i).getDataOra().getMonthValue()==monthValue && lista.get(i).getDataOra().getDayOfMonth()==dayOfMonth)

{

result= result+" "+ lista.get(i).getNume() + " - " + lista.get(i).getEnorias().getNume() + " | ";

flag = true;

}

}

if(flag) return result;

else return "invalid";

}

}

**SlujbaService:**

import java.lang.reflect.Array;

import java.time.LocalDate;

import java.util.ArrayList;

public class SlujbaService extends Service {

public SlujbaService(Repository<Slujba> repo)

{

super(repo);

}

public void setAvailableId(Slujba object) {

ArrayList<Slujba> slujbaList = super.getList();

int maxId = 0;

for (int i=0;i<slujbaList.size();i++) {

maxId = Math.max(maxId, slujbaList.get(i).getId());

}

int[] existingIds = new int[maxId + 1];

for (int i=0;i<slujbaList.size();i++) {

existingIds[slujbaList.get(i).getId() - 1] = 1;

}

for (int i = 0; i <= maxId; i++) {

if (existingIds[i] == 0) {

object.setId(i + 1);

break;

}

}

}

public void add(Slujba a) {

super.repo.add(a);

}

public String addSlujba(String numeSlujba, String anSlujba, String lunaSlujba , String ziSlujba)

{

String checkError = super.checkNumar(anSlujba,4000);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(numeSlujba,25);

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(lunaSlujba,13 );

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(ziSlujba,32 );

if (checkError!="valid!") return checkError;

checkError = super.checkData(anSlujba, lunaSlujba, ziSlujba);

if (checkError!="valid!") return checkError;

Slujba a = new Slujba(numeSlujba,Integer.parseInt(anSlujba),Integer.parseInt(lunaSlujba),Integer.parseInt(ziSlujba));

this.add(a);

setAvailableId(a);

return "valid!";

}

public ArrayList<String[]> getAllForPrint() {

ArrayList<Slujba> listaPrint = super.getList();

ArrayList<String[]> result = new ArrayList<String[]>();

for(int i=0;i<listaPrint.size();i++)

{

String[] object = new String[5];

object[0] = String.valueOf(listaPrint.get(i).getId());

object[1] = listaPrint.get(i).getNume();

LocalDate printedDate = listaPrint.get(i).getData\_Ora();

object[2] = String.valueOf(printedDate.getYear());

object[3] = String.valueOf(printedDate.getMonthValue());

object[4] = String.valueOf(printedDate.getDayOfMonth());

result.add(object);

}

return result;

}

public String removeSlujba(String id) {

String checkError = super.checkNumar(id,5000);

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

super.remove(pos);

return "valid!";

}

return checkError;

}

return checkError;

}

public int existId(int id) {

try {

ArrayList<Slujba> lista = this.repo.getList();

for (int i = 0; i < this.repo.size(); i++) {

if (id == lista.get(i).getId()) {

return i;

}

}

throw new IllegalArgumentException("Id does not exist");

} catch (IllegalArgumentException ex) {

return -1;

}

}

public String updateSlujba(String id, String newTitle, String newAn, String newLuna,String newZi){

String checkError = super.checkNumar(id,5000);

if(checkError!="valid!") return checkError;

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

checkError = this.checkUpdate(id, newTitle, newAn, newLuna,newZi,pos);

return checkError;

}

return checkError;

}

return checkError;

}

public String checkUpdate(String id, String newTitle, String newYear , String newMonth, String newDay, int pos)

{

String checkError = super.checkNumar(id,5000);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(newTitle,25);

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(newYear, 4000);

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(newMonth, 13);

if (checkError!="valid!") return checkError;

checkError = super.checkNumar(newDay, 32);

if (checkError!="valid!") return checkError;

return executeUpdate(id, newTitle, newYear, newMonth, newDay, pos) ;

}

public String executeUpdate(String id , String newTitle, String newYear, String newMonth, String newDay, int pos)

{

Slujba a = new Slujba(newTitle,Integer.parseInt(newYear),Integer.parseInt(newMonth),Integer.parseInt(newDay));

a.setId(Integer.parseInt(id));

super.update(a, pos);

return "valid!";

}

public String getStringSlujbeByMonth(int yearValue, int monthValue, int dayValue) {

ArrayList<Slujba> lista = super.getList();

String result= new String();

boolean flag=false;

for(int i=0;i<lista.size();i++)

{

if(lista.get(i).getData\_Ora().getYear()==yearValue && lista.get(i).getData\_Ora().getMonthValue()==monthValue && lista.get(i).getData\_Ora().getDayOfMonth()==dayValue)

{

result= result+" "+ lista.get(i).getNume() + "|";

flag = true;

}

}

System.out.println(result);

if (flag) return result;

else return "invalid";

}

}

**AnuntService:**

import java.util.ArrayList;

public class AnuntService extends Service<Anunt> {

// Contine functiile specifice manipularii Anunturilor

// Reprezinta legatura dintre consola si Service generic

// Genereaza un id valabil pentru obiectul adaugat

public AnuntService(Repository<Anunt> repo)

{

super(repo);

}

public void setAvailableId(Anunt object) {

// Se foloseste de un array de flag uri (0 , 1)

ArrayList<Anunt> anuntList = super.getList();

int maxId = 0;

for (int i=0;i<anuntList.size();i++) {

maxId = Math.max(maxId, anuntList.get(i).getId());

}

int[] existingIds = new int[maxId + 1];

for (int i=0;i<anuntList.size();i++) {

existingIds[anuntList.get(i).getId() - 1] = 1;

}

for (int i = 0; i <= maxId; i++) {

if (existingIds[i] == 0) {

object.setId(i + 1);

break;

}

}

}

@Override

public void add(Anunt a) {

super.repo.add(a);

}

public String addAnunt(String titlu , String continut , String isAnunt)

{

String checkError = super.checkNumar(isAnunt,3);

if (checkError!="valid!") return checkError;

checkError = super.checkNumarBoolean(isAnunt);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(titlu,25);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(continut, 100);

if (checkError!="valid!") return checkError;

Anunt a = new Anunt(titlu,continut,isAnuntToBool(isAnunt));

this.add(a);

setAvailableId(a);

return "valid!";

}

public boolean isAnuntToBool(String isAnunt)

{

if(isAnunt.equals("1"))

return false;

else

return true;

}

public String updateAnunt(String id, String newTitle, String newContinut, String isAnunt){

String checkError = super.checkNumar(id,5000);

if(checkError!="valid!") return checkError;

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

checkError = this.checkUpdate(id, newTitle, newContinut, isAnunt,pos);

return checkError;

}

return checkError;

}

return checkError;

}

public String checkUpdate(String id, String newTitle, String newContinut, String isAnunt,int pos)

{

String checkError = super.checkNume(newTitle,25);

if (checkError!="valid!") return checkError;

checkError = super.checkNume(newContinut,50);

if (checkError!="valid!") return checkError;

return executeUpdate(Integer.parseInt(id), newTitle, newContinut,isAnunt,pos) ;

}

public String executeUpdate(int id , String newTitle, String newContinut, String isAnunt,int pos)

{

ArrayList<Anunt> listaPrint = super.getList();

Anunt a = new Anunt(newTitle,newContinut,false);

if(isAnunt.equals("true"))

{

a.setEsteAnunt(true);

}

a.setId(id);

super.update(a, pos);

return "valid!";

}

public ArrayList<String[]> getAllForPrint() {

ArrayList<Anunt> listaPrint = super.getList();

ArrayList<String[]> result = new ArrayList<String[]>();

for(int i=0;i<listaPrint.size();i++)

{

String[] object = new String[4];

object[0] = String.valueOf(listaPrint.get(i).getId());

object[1] = listaPrint.get(i).getTitlu();

object[2] = listaPrint.get(i).getContinut();

object[3] = String.valueOf(listaPrint.get(i).getEsteAnunt());

result.add(object);

}

return result;

}

public String removeAnunt(String id) {

String checkError = super.checkNumar(id,5000);

if(checkError=="valid!")

{

int pos = this.existId(Integer.parseInt(id));

if(pos!=-1)

{

super.remove(pos);

return "valid!";

}

return "id does not exist";

}

return checkError;

}

public int existId(int id) {

ArrayList<Anunt> lista = this.repo.getList();

for (int i = 0; i < this.repo.size(); i++) {

if (id == repo.get(i).getId()) {

return i;

}

}

return -1;

}

}

**TextPrintable:**

import java.awt.print.PrinterException;

import java.awt.print.PrinterJob;

import java.util.ArrayList;

import java.awt.Graphics;

import java.awt.print.PageFormat;

import java.awt.print.Printable;

class TextPrintable implements Printable {

private final ArrayList<String> lines;

// Al doilea element din layer 2 , se ocupa cu accesarea default printer ului de la user si formatarea paginii in functie

//de input ul dat de printStrings()

TextPrintable(ArrayList<String> lines) {

this.lines = lines;

}

@Override

public int print(Graphics graphics, PageFormat pageFormat, int pageIndex)

throws PrinterException {

int linesPerPage = (int) ((pageFormat.getImageableHeight() - (2 \* pageFormat.getImageableY())) / graphics.getFontMetrics().getHeight());

if (pageIndex \* linesPerPage >= this.lines.size()) {

return NO\_SUCH\_PAGE;

}

int x = (int) pageFormat.getImageableX();

int y = (int) pageFormat.getImageableY() + graphics.getFontMetrics().getAscent();

for (int i = pageIndex \* linesPerPage; i < Math.min((pageIndex + 1) \* linesPerPage, this.lines.size()); i++) {

graphics.drawString(this.lines.get(i), x, y);

y += graphics.getFontMetrics().getHeight();

}

return PAGE\_EXISTS;

}

}

**FileRepo:**

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.nio.file.StandardOpenOption;

import java.util.ArrayList;

import java.util.Scanner;

public class FileRepo {

private AnuntService anuntService;

private SlujbaService slujbaService;

private EnoriasService enoriasService;

private ServiciuService serviciuService;

public FileRepo(AnuntService anuntService, SlujbaService slujbaService, EnoriasService enoriasService, ServiciuService serviciuService) {

this.anuntService = anuntService;

this.enoriasService = enoriasService;

this.serviciuService = serviciuService;

this.slujbaService = slujbaService;

}

public void readAnunt(String fileName) throws FileNotFoundException {

try {

Scanner scanner = new Scanner(new File(fileName));

while (scanner.hasNext()) {

int id = Integer.parseInt(scanner.nextLine());

String titlu = scanner.nextLine();

String continut = scanner.nextLine();

boolean isAnunt = Boolean.parseBoolean(scanner.nextLine());

Anunt a = new Anunt(titlu, continut, isAnunt);

a.setId(id);

this.anuntService.add(a);

}

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

public void readSlujba(String fileName) throws FileNotFoundException {

try {

Scanner scanner = new Scanner(new File(fileName));

while (scanner.hasNext()) {

int id = Integer.parseInt(scanner.nextLine());

String nume = scanner.nextLine();

int an = Integer.parseInt(scanner.nextLine());

int luna = Integer.parseInt(scanner.nextLine());

int zi = Integer.parseInt(scanner.nextLine());

Slujba a = new Slujba(nume, an, luna, zi);

a.setId(id);

this.slujbaService.add(a);

}

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

public void readEnorias(String fileName) throws FileNotFoundException {

try {

Scanner scanner = new Scanner(new File(fileName));

while (scanner.hasNext()) {

int id = Integer.parseInt(scanner.nextLine());

String nume = scanner.nextLine();

String telefon = scanner.nextLine();

String adresa = scanner.nextLine();

Enorias a = new Enorias(nume, telefon, adresa);

a.setId(id);

this.enoriasService.add(a);

}

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

public void readServiciu(String fileName) throws FileNotFoundException {

try {

Scanner scanner = new Scanner(new File(fileName));

while (scanner.hasNext()) {

int id = Integer.parseInt(scanner.nextLine());

String nume = scanner.nextLine();

int idEnorias = Integer.parseInt(scanner.nextLine());

int an = Integer.parseInt(scanner.nextLine());

int luna = Integer.parseInt(scanner.nextLine());

int zi = Integer.parseInt(scanner.nextLine());

Enorias a = this.enoriasService.getEnoriasById(String.valueOf(idEnorias));

Serviciu b = new Serviciu(nume, a, an, luna, zi);

b.setId(id);

this.serviciuService.add(b);

}

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

public void writeAnunt(String fileName) throws IOException {

try {

new PrintWriter(fileName).close();

File file = new File(fileName);

FileOutputStream stream = new FileOutputStream(file);

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stream));

ArrayList<Anunt> list = this.anuntService.getList();

for (int i = 0; i < list.size(); i++) {

Anunt a = list.get(i);

writer.write(String.valueOf(a.getId()));

writer.newLine();

writer.write(a.getTitlu());

writer.newLine();

writer.write(a.getContinut());

writer.newLine();

writer.write(String.valueOf(a.getEsteAnunt()));

writer.newLine();

}

writer.close();

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

public void writeSlujba(String fileName) throws IOException {

try {

new PrintWriter(fileName).close();

File file = new File(fileName);

FileOutputStream stream = new FileOutputStream(file);

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stream));

ArrayList<Slujba> list = this.slujbaService.getList();

for (int i = 0; i < list.size(); i++) {

Slujba a = list.get(i);

writer.write(String.valueOf(a.getId()));

writer.newLine();

writer.write(a.getNume());

writer.newLine();

writer.write(String.valueOf(a.getData\_Ora().getYear()));

writer.newLine();

writer.write(String.valueOf(a.getData\_Ora().getMonthValue()));

writer.newLine();

writer.write(String.valueOf(a.getData\_Ora().getDayOfMonth()));

writer.newLine();

}

writer.close();

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

public void writeEnorias(String fileName) throws IOException {

try {

new PrintWriter(fileName).close();

File file = new File(fileName);

FileOutputStream stream = new FileOutputStream(file);

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stream));

ArrayList<Enorias> list = this.enoriasService.getList();

for (int i = 0; i < list.size(); i++) {

Enorias a = list.get(i);

writer.write(String.valueOf(a.getId()));

writer.newLine();

writer.write(a.getNume());

writer.newLine();

writer.write(a.getTelefon());

writer.newLine();

writer.write(a.getAdresa());

writer.newLine();

}

writer.close();

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

public void writeServiciu(String fileName) throws IOException {

try {

new PrintWriter(fileName).close();

File file = new File(fileName);

FileOutputStream stream = new FileOutputStream(file);

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stream));

ArrayList<Serviciu> list = this.serviciuService.getList();

for (int i = 0; i < list.size(); i++) {

Serviciu a = list.get(i);

writer.write(String.valueOf(a.getId()));

writer.newLine();

writer.write(a.getNume());

writer.newLine();

writer.write(String.valueOf(a.getEnorias().getId()));

writer.newLine();

writer.write(String.valueOf(a.getDataOra().getYear()));

writer.newLine();

writer.write(String.valueOf(a.getDataOra().getMonthValue()));

writer.newLine();

writer.write(String.valueOf(a.getDataOra().getDayOfMonth()));

writer.newLine();

}

writer.close();

} catch (FileNotFoundException e) {

throw new FileNotFoundException();

}

}

}

**Console:**

import java.time.LocalDate;

import java.util.ArrayList;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.awt.print.PrinterException;

import java.awt.print.PrinterJob;

public class Console {

private AnuntService anuntService;

private SlujbaService slujbaService;

private EnoriasService enoriasService;

private ServiciuService serviciuService;

//

// Programul este alcatuit din 4 layer uri distincte :

// 1: GUI ul , primeste input de la utilizator , il transmite consolei , si primeste raspunsuri utile ( string uri) , si valori

// printarii

// 2: Console + TextPrintable , prima are ca functie procesarea raspunului de la GUI si apelarea functiilor necesare din service

// precum si convertirea unor input uri si outpuri si formatarea lor, iar TextPrintable organizeaza listarea la imprimanta a input-ului

// 3: Service , primeste input uri specifice de la consola si le verifica , in cazul in care sunt eronate returneaza string uri cu erori

// utile la consola care sunt apoi returnate la GUI . Iar in cazul in care input ul este corect, apeleaza functiile clasei generice Repository

// 4: Repository , formata din clasa generica Repository ce contine functiile pentru CRUD si clasele de instantiere

public Console(AnuntService anuntService, SlujbaService slujbaService, EnoriasService enoriasService,ServiciuService serviciuService) {

// Connstructor consola , intocmeste layer-ul 2 impreuna cu clasa de Print de la structura programului , primeste ca si parametrii toate elementele din layer 3 (servicii)

this.anuntService = anuntService;

this.slujbaService = slujbaService;

this.enoriasService = enoriasService;

this.serviciuService = serviciuService;

}

public String printSlujbeServicii() {

// Functie care este chemata de catre GUI (layer 1), primeste ca si string de la service urile celor doua obiecte

// verifica daca sunt goale ("") si le adauga in rezultat care este pasat spre functia de printare a layer-ului 2

// returneaza string uri utile pentru GUI

LocalDate currentDate = LocalDate.now();

ArrayList<String> result = new ArrayList<String>();

if(slujbaService.getStringSlujbeByMonth(currentDate.getYear(),currentDate.getMonthValue(),currentDate.getDayOfMonth())!="invalid")

{

result.add(slujbaService.getStringSlujbeByMonth(currentDate.getYear(),currentDate.getMonthValue(),currentDate.getDayOfMonth()));

}

if(serviciuService.getServiciiByMonth(currentDate.getYear(),currentDate.getMonthValue(),currentDate.getDayOfMonth())!="invalid")

{

result.add(serviciuService.getServiciiByMonth(currentDate.getYear(),currentDate.getMonthValue(),currentDate.getDayOfMonth()));

}

if(result.size()==0)

{

return "no such serviciu/sluja exist";

}

else {

printStrings(result);

return "serviciu/slujba are being printed";

}

}

public String getSfintiZiuaCurenta() {

// Obtine datele zilei in care se afla utilizatorul

// Si bazat pe valoare lunii decide care dintre file uri o sa fie folosite

// Apeleaza functia de citire in file , getSfintiFromFile cu parametrii numele file ul si ziua in care se afla utilizatorul

LocalDate currentDate = LocalDate.now();

int month = currentDate.getMonthValue();

switch (month) {

case 1:

return getSfintiFromFile("Ianuarie.txt", currentDate.getDayOfMonth());

case 2:

return getSfintiFromFile("Februarie.txt", currentDate.getDayOfMonth());

case 3:

return getSfintiFromFile("Martie.txt", currentDate.getDayOfMonth());

case 4:

return getSfintiFromFile("Aprilie.txt", currentDate.getDayOfMonth());

case 5:

return getSfintiFromFile("Mai.txt", currentDate.getDayOfMonth());

case 6:

return getSfintiFromFile("Iunie.txt", currentDate.getDayOfMonth());

case 7:

return getSfintiFromFile("Iulie.txt", currentDate.getDayOfMonth());

default:

return "Luna indisponibila momentan";

}

}

public String getSfintiFromFile(String fileName, int day)

{

// Parcurge file ul linie cu linie de day-1 ori ( adica pana ajunge la linia necesara print ului)

// la linia respectiva o returneaza ca si String GUI ului

try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

for (int i = 1; i < day; i++) {

reader.readLine();

}

return reader.readLine();

} catch (IOException e) {

return null;

}

}

public static void printStrings(ArrayList<String> lines) {

// Pregateste input ul pentru clasa TextPrintable in baza listei de String uri ( obiectele care au fost gasite pe data curenta)

// In cazul neasteptat in care utilizatorul nu are un default printer , atunci se va afisa eroarea specifica in consola cu Error printing:

PrinterJob printJob = PrinterJob.getPrinterJob();

printJob.setPrintable(new TextPrintable(lines));

if (printJob.printDialog()) {

try {

printJob.print();

} catch (PrinterException e) {

System.out.println("Error printing: " + e.getMessage());

}

}

}

public String getDataNewAnunt(String titlu, String continut, String esteAnunt) {

//Apeleaza functia speficica din service si returneaza eroarea obtinuta GUI ului ( care poate fi doar "valid!")

String error = new String();

error = this.anuntService.addAnunt(titlu, continut, esteAnunt);

return error;

}

public String getDataNewSlujba(String titlu, String an, String luna, String zi) {

//Apeleaza functia speficica din service si returneaza eroarea obtinuta GUI ului ( care poate fi doar "valid!")

String error = new String();

error = this.slujbaService.addSlujba(titlu,an,luna,zi);

return error;

}

public String getDataNewEnorias(String nume, String telefon, String adresa) {

//Apeleaza functia speficica din service si returneaza eroarea obtinuta GUI ului ( care poate fi doar "valid!")

String error = new String();

error = this.enoriasService.addEnorias(nume,telefon,adresa);

return error;

}

public String getDataNewServiciu(String idEnorias, String nume, String an, String luna, String zi) {

//Apeleaza functia speficica din service si returneaza eroarea obtinuta GUI ului ( care poate fi doar "valid!")

String error = new String();

error = this.serviciuService.addServiciu(nume,this.enoriasService.getEnoriasById(idEnorias) , an, luna, zi);

return error;

}

public ArrayList<String[]> displayServicii() {

// Obtine de la service un ArrayList<String[]> cu toate propietatiile obiectelor din Repository la momentul chemarii

ArrayList<String[]> displayResult = new ArrayList<String[]>();

displayResult = this.serviciuService.getAllForPrint();

return displayResult;

}

public ArrayList<String[]> displayAnunturi() {

// Obtine de la service un ArrayList<String[]> cu toate propietatiile obiectelor din Repository la momentul chemarii

ArrayList<String[]> displayResult = new ArrayList<String[]>();

displayResult = anuntService.getAllForPrint();

return displayResult;

}

public ArrayList<String[]> displaySlujbe() {

// Obtine de la service un ArrayList<String[]> cu toate propietatiile obiectelor din Repository la momentul chemarii

ArrayList<String[]> displayResult = new ArrayList<String[]>();

displayResult = slujbaService.getAllForPrint();

return displayResult;

}

public ArrayList<String[]> displayEnoriasi() {

// Obtine de la service un ArrayList<String[]> cu toate propietatiile obiectelor din Repository la momentul chemarii

ArrayList<String[]> displayResult = new ArrayList<String[]>();

displayResult = enoriasService.getAllForPrint();

return displayResult;

}

public String callUpdateAnunt(String id, String titlu, String continut, String isAnunt) {

// Obtine de la GUI input ul necesar in string uri si apeleaza functia respectiva din Service

String error = new String();

error = this.anuntService.updateAnunt(id, titlu, continut, isAnunt);

return error;

}

public String callUpdateSlujba(String id, String titlu, String an, String luna,String zi) {

// Obtine de la GUI input ul necesar in string uri si apeleaza functia respectiva din Service

String error = new String();

error = this.slujbaService.updateSlujba(id, titlu, an, luna, zi);

return error;

}

public String callUpdateEnorias(String id, String nume, String telefon, String adresa) {

// Obtine de la GUI input ul necesar in string uri si apeleaza functia respectiva din Service

String error = new String();

error = this.enoriasService.updateEnorias(id,nume,telefon,adresa);

return error;

}

public String callUpdateServiciu(String id, String idEnorias, String nume, String an,String luna,String zi) {

// Obtine de la GUI input ul necesar in string uri si apeleaza functia respectiva din Service

String error = new String();

if(this.enoriasService.getEnoriasById(idEnorias)!=null)

{

error = this.serviciuService.updateServiciu(id,this.enoriasService.getEnoriasById(idEnorias),nume,an,luna,zi);

}

else

{

return "id of Enorias does not exist";

}

return error;

}

public String removeAnunt(String id) {

// Obtine de la GUI input ul necesar in string si apeleaza functia respectiva din Service

String error = new String();

error = this.anuntService.removeAnunt(id);

return error;

}

public String removeSlujba(String id) {

// Obtine de la GUI input ul necesar in string uri si apeleaza functia respectiva din Service

String error = new String();

error = this.slujbaService.removeSlujba(id);

return error;

}

public String removeEnorias(String id) {

// Obtine de la GUI input ul necesar in string uri si apeleaza functia respectiva din Service

String error = new String();

error = this.enoriasService.removeEnorias(id);

return error;

}

public String removeServiciu(String id) {

// Obtine de la GUI input ul necesar in string uri si apeleaza functia respectiva din Service

String error = new String();

error = this.serviciuService.removeServiciu(id);

return error;

}

}

**MainFrame:**

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import java.awt.\*;

import java.awt.event.\*;

import java.io.FileNotFoundException;

//import java.nio.file.Path;

//import java.nio.file.Paths;

import java.io.IOException;

import java.util.Objects;

public class MainFrame extends JFrame implements ActionListener {

Console console;

FileRepo fileRepo;

private JFrame mainFrame;

private Container container;

private CardLayout cardLayout;

private JButton startButton, backButton;

MainFrame(Console console, FileRepo fileRepo) throws FileNotFoundException {

this.console = console;

this.fileRepo = fileRepo;

// Path path = Paths.get("anuntRepo.txt");

fileRepo.readAnunt("./anuntRepo.txt");

fileRepo.readSlujba("./slujbaRepo.txt");

fileRepo.readEnorias("./enoriasRepo.txt");

fileRepo.readServiciu("./serviciuRepo.txt");

cardLayout = new CardLayout();

mainFrameInitialisation();

content();

}

MainFrame(){

}

public void errorBox(String error, String succes)

{

JDialog eroare = new JDialog();

JPanel eroareArea = new JPanel();

eroare.setLayout(new GridBagLayout());

eroareArea.setLayout(new BorderLayout());

eroare.setSize(300,100);

eroare.setResizable(false);

if(error.equals("valid!"))

eroare.setTitle(succes);

else

eroare.setTitle("Eroare!");

eroare.setLocationRelativeTo(null);

eroare.setVisible(true);

JTextArea eroareText = new JTextArea();

eroareText.setText(error);

eroareText.setEditable(false);

eroareText.setOpaque(false);

eroareArea.add(eroareText,BorderLayout.CENTER);

eroare.add(eroareArea);

}

public void errorBoxGUI()

{

JDialog eroare = new JDialog();

JPanel eroareArea = new JPanel();

eroare.setLayout(new GridBagLayout());

eroareArea.setLayout(new BorderLayout());

eroare.setSize(300,100);

eroare.setResizable(false);

eroare.setTitle("Eroare!");

eroare.setLocationRelativeTo(null);

eroare.setVisible(true);

JTextArea eroareText = new JTextArea();

eroareText.setText("Lista este goală!");

eroareText.setEditable(false);

eroareText.setOpaque(false);

eroareArea.add(eroareText,BorderLayout.CENTER);

eroare.add(eroareArea);

}

public void infoBox(String titlu, String continut, int latime, int inaltime)

{

JDialog ajutorText = new JDialog();

ajutorText.setLayout(new BorderLayout());

JTextArea text = new JTextArea(continut);

text.setWrapStyleWord(true);

text.setLineWrap(true);

text.setEditable(false);

text.setBorder(BorderFactory.createLineBorder(Color.black));

ajutorText.setTitle(titlu);

ajutorText.setSize(latime,inaltime);

ajutorText.setLocationRelativeTo(null);

ajutorText.setResizable(false);

ajutorText.setVisible(true);

ajutorText.add(text, BorderLayout.CENTER);

}

private void mainFrameInitialisation()

{

mainFrame = new JFrame("E-Church Agenda");

mainFrame.setSize(800,600);

mainFrame.setLocationRelativeTo(null);

mainFrame.setVisible(true);

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.setResizable(false);

mainFrame.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

super.windowClosing(e);

try {

fileRepo.writeAnunt("./anuntRepo.txt");

fileRepo.writeSlujba("./slujbaRepo.txt");

fileRepo.writeEnorias("./enoriasRepo.txt");

fileRepo.writeServiciu("./serviciuRepo.txt");

} catch (IOException ex) {

throw new RuntimeException(ex);

}

}

});

}

private void content()

{

container = mainFrame.getContentPane();

container.setLayout(cardLayout);

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new BoxLayout(mainPanel,BoxLayout.Y\_AXIS));

mainPanel.setBorder(new EmptyBorder(new Insets(25,200,200,200)));

JLabel image = new JLabel();

image.setIcon(new ImageIcon(Objects.requireNonNull(getClass().getResource("./testCross.png"))));

startButton = new JButton("Start");

startButton.addActionListener(this);

JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new GridBagLayout());

buttonPanel.add(startButton);

//buttonPanel.setBorder(BorderFactory.createLineBorder(Color.black));

mainPanel.add(image);

mainPanel.add(Box.createRigidArea(new Dimension(0,50)));

mainPanel.add(buttonPanel);

//mainPanel.add(Box.createRigidArea(new Dimension(0,60)));

MainMenu menu = new MainMenu(this.console);

JPanel mainMenu = new JPanel();

JPanel menuButtonPanel = menu.getButtonPanel();

mainMenu.setLayout(new GridBagLayout());

mainMenu.setBorder(BorderFactory.createEmptyBorder(150,150,150,150));

backButton = new JButton("Înapoi");

backButton.addActionListener(this);

menuButtonPanel.add(backButton);

mainMenu.add(menuButtonPanel);

container.add(mainPanel);

container.add(mainMenu);

}

@Override

public void actionPerformed(ActionEvent e)

{ if(e.getSource()==startButton)

{

cardLayout.next(container);

} else if (e.getSource()==backButton)

{

cardLayout.previous(container);

}

}

}

**MainMenu:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class MainMenu extends MainFrame implements ActionListener {

Console console;

JPanel buttonPanel = new JPanel();

JButton serviciiButton,prediciAnunturiButton,sfintiiButton,ajutorButton,slujbaButton,enoriasButton, imprimaButton;

MainMenu(Console console){

this.console = console;

mainMenu();

}

public JPanel getButtonPanel(){

return buttonPanel;

}

private void mainMenu()

{

buttonPanel.setLayout(new GridLayout(8,1,10,30));

buttonPanel.setSize(new Dimension(100,65));

serviciiButton = new JButton("Servicii Religioase");

prediciAnunturiButton = new JButton("Predici/Anunțuri");

slujbaButton = new JButton("Slujbe");

sfintiiButton = new JButton("Sfinții pomeniți azi și/sau sărbătorile!");

ajutorButton = new JButton("Ajutor");

enoriasButton = new JButton("Enoriasi");

imprimaButton = new JButton("Imprimă servicii/slujbe din ziua curentă");

serviciiButton.addActionListener(this);

prediciAnunturiButton.addActionListener(this);

slujbaButton.addActionListener(this);

sfintiiButton.addActionListener(this);

ajutorButton.addActionListener(this);

enoriasButton.addActionListener(this);

imprimaButton.addActionListener(this);

buttonPanel.add(enoriasButton);

buttonPanel.add(serviciiButton);

buttonPanel.add(slujbaButton);

buttonPanel.add(imprimaButton);

buttonPanel.add(prediciAnunturiButton);

buttonPanel.add(sfintiiButton);

buttonPanel.add(ajutorButton);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource() == serviciiButton)

{

ServiciiDialogBox sdb = new ServiciiDialogBox(this.console);

} else if (e.getSource() == prediciAnunturiButton)

{

PrediciAnunturiDBox padb = new PrediciAnunturiDBox(this.console);

} else if (e.getSource() == slujbaButton)

{

SlujbeDialogBox sljdb = new SlujbeDialogBox(this.console);

} else if (e.getSource() == enoriasButton)

{

EnoriasiDialogBox edb = new EnoriasiDialogBox(this.console);

}

else if(e.getSource() == imprimaButton)

{

infoBox("Status printare",console.printSlujbeServicii(),150,150);

}

else if(e.getSource() == sfintiiButton)

{

infoBox("Sfinții și/sau sărbătorile de azi!",console.getSfintiZiuaCurenta(),350,350);

}

else if(e.getSource() == ajutorButton)

{

String continutAjutor;

continutAjutor = "În centrul ecranului se află meniul principal.\n" +

"Butonul „Enoriași” deschide „Manager enoriași”.\n" +

"Butonul „Servicii Religioase” deschide „Manager Servicii Religioase”.\n" +

"Butonul „Slujbe” deschide „Manager slujbe”.\n"+

"Butonul „Imprimă servicii/slujbe din ziua curentă” face ce sugerează numele.\n"+

"Butonul „Predici/Anunțuri” deschide „Manager predici/anunțuri”.\n" +

"Butonul „Sfinții pomeniți azi și/sau sărbătorile!” face ce sugerează numele.\n" +

"Butonul „Ajutor” afișează acest mesaj.\n" +

"Butonul „Înapoi” întoarce utilizatorul la pagina inițială.\n" +

"Butonul „Start” din pagina inițială face trecerea la acest meniu.";

infoBox("Ajutor!",continutAjutor,350,255);

}

}

}

**ServiciiDialogBox:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

public class ServiciiDialogBox extends MainFrame implements ActionListener {

Console console;

ArrayList<String[]> continut;

private JDialog serviciiDialogBox = new JDialog();

private JButton adaugaButton, editButton, ajutorButton, detaliiButton, stergeButton, inchideButton, actualizareButton;

JList listaServicii;

public ServiciiDialogBox(Console console)

{

this.console = console;

continut = console.displayServicii();

serviciiBox();

}

private void serviciiBox()

{

serviciiDialogBox.setSize(550,550);

serviciiDialogBox.setLocationRelativeTo(null);

serviciiDialogBox.setResizable(false);

serviciiDialogBox.setTitle("Manager Servicii Religioase");

serviciiDialogBox.setVisible(true);

JPanel srvDialogBox = new JPanel();

JPanel listArea = new JPanel();

JPanel buttonsArea = new JPanel();

JPanel buttonsPanel = new JPanel();

JPanel listPanel = new JPanel();

srvDialogBox.setLayout(new GridLayout(1,2));

buttonsArea.setLayout(new GridBagLayout());

listArea.setLayout(new BorderLayout());

buttonsPanel.setLayout(new GridLayout(7,1,5,30));

listPanel.setLayout(new BorderLayout());

ArrayList<String> nume = new ArrayList<String>();

for (int i = 0; i < continut.size(); i++) {

nume.add(continut.get(i)[1] + "-" + continut.get(i)[5]);

}

listaServicii = new JList(nume.toArray());

JScrollPane scrollPanelList = new JScrollPane();

//scrollPanelList.setBorder(BorderFactory.createLineBorder(Color.red));

scrollPanelList.setViewportView(listaServicii);

listaServicii.setLayoutOrientation(JList.VERTICAL);

listaServicii.setVisible(true);

listaServicii.setBorder(BorderFactory.createLineBorder(Color.black));

//listPanel.add(listaServicii);

listPanel.add(scrollPanelList);

listArea.add(listPanel,BorderLayout.CENTER);

srvDialogBox.add(listArea);

//buttonsArea.setBorder(BorderFactory.createLineBorder(Color.black));

//buttonsPanel.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));

adaugaButton = new JButton("Adaugă serviciu religios");

detaliiButton = new JButton("Detalii despre serviciul religios selectat");

editButton = new JButton("Modifică serviciul religios selectat");

stergeButton = new JButton("Șterge serviciul religios selectat");

ajutorButton = new JButton("Ajutor");

inchideButton = new JButton("Închide fereastra");

actualizareButton = new JButton("Actualizează lista");

buttonsPanel.add(adaugaButton);

buttonsPanel.add(actualizareButton);

buttonsPanel.add(detaliiButton);

buttonsPanel.add(editButton);

buttonsPanel.add(stergeButton);

buttonsPanel.add(ajutorButton);

buttonsPanel.add(inchideButton);

adaugaButton.addActionListener(this);

inchideButton.addActionListener(this);

actualizareButton.addActionListener(this);

stergeButton.addActionListener(this);

editButton.addActionListener(this);

detaliiButton.addActionListener(this);

ajutorButton.addActionListener(this);

buttonsArea.add(buttonsPanel);

srvDialogBox.add(listArea);

srvDialogBox.add(buttonsArea);

serviciiDialogBox.add(srvDialogBox);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource() == adaugaButton)

{

AdaugareServiciuReligios nou = new AdaugareServiciuReligios(this.console);

}

else if(e.getSource() == inchideButton)

{

serviciiDialogBox.dispose();

}

else if (e.getSource() == actualizareButton)

{

serviciiDialogBox.dispose();

ServiciiDialogBox reset = new ServiciiDialogBox(this.console);

}

else if(e.getSource() == stergeButton)

{

String error;

if(!continut.isEmpty()) {

error = console.removeServiciu(continut.get(listaServicii.getAnchorSelectionIndex())[0]);

actualizareButton.doClick();

}

else{

errorBoxGUI();

}

} else if (e.getSource() == editButton) {

if(!continut.isEmpty())

{

EditareServiciu edit = new EditareServiciu(this.console,continut.get(listaServicii.getAnchorSelectionIndex())[0]);

}

else

{

errorBoxGUI();

}

}

else if(e.getSource() == detaliiButton)

{

if(!continut.isEmpty())

{

String detalii = "Titlu: " + continut.get(listaServicii.getAnchorSelectionIndex())[1] +

"\nEnoriaș:" + continut.get(listaServicii.getAnchorSelectionIndex())[5] +

"\nData: " + continut.get(listaServicii.getAnchorSelectionIndex())[4] + "/" + continut.get(listaServicii.getAnchorSelectionIndex())[3] + "/" + continut.get(listaServicii.getAnchorSelectionIndex())[2];

infoBox("Detalii serviciu selectat",detalii,125,125);

}

else

{

errorBoxGUI();

}

}

else if(e.getSource() == ajutorButton)

{

String continutAjutor;

continutAjutor = "În partea stângă a ferestrei „Manager Servicii Religioase” se află lista de servicii religioase create și programate\n" +

"În partea dreaptă, se află butoanele.\n" +

"Butonul „Adaugă serviciu religios” deschide o fereastră corespunzătoare operației.\n" +

"Butonul „Actualizează lista” face ce sugerează numele.\n" +

"Butonul „Detalii despre serviciul religios selectat” face ce sugerează numele.\n"+

"Butonul „Modifică serviciul religios selectat” deschide o fereastră corespunzătoare operației.\n"+

"Butonul „Șterge serviciul religios selectat” face ce sugerează numele.\n" +

"Butonul „Ajutor” afișează acest mesaj.\n" +

"Butonul „Închide fereastra” face ce sugerează numele.";

infoBox("Ajutor!",continutAjutor,350,265);

}

}

}

**AdăugareServiciuReligios:**

import com.sun.tools.javac.Main;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

public class AdaugareServiciuReligios extends MainFrame implements ActionListener {

Console console;

JDialog adaugareSBox = new JDialog();

ArrayList<String[]> continut;

JButton adaugaButton, ajutorButton, inchideButton, reseteazaButton;

JList listaEnoriasi;

JTextArea titluTA,enoriasTA, ziTA, lunaTA, anTA;

AdaugareServiciuReligios(Console console)

{

this.console = console;

continut = console.displayEnoriasi();

setup();

}

private void setup()

{

adaugareSBox.setSize(750,500);

adaugareSBox.setLocationRelativeTo(null);

adaugareSBox.setResizable(false);

adaugareSBox.setTitle("Adăugare serviciu religios");

adaugareSBox.setVisible(true);

JPanel addSRBoxPanel = new JPanel();

JPanel listArea = new JPanel();

JPanel list = new JPanel();

JPanel infoButtonsArea = new JPanel();

JPanel infoPanel = new JPanel();

JPanel gridBPTitlu = new JPanel();

JPanel gridBPEnorias = new JPanel();

JPanel gridBPZi = new JPanel();

JPanel gridBPLuna = new JPanel();

JPanel gridBPAn = new JPanel();

JPanel gridBPTitluLabel = new JPanel();

JPanel gridBPEnoriasLabel = new JPanel();

JPanel gridBPZiLabel = new JPanel();

JPanel gridBPLunaLabel = new JPanel();

JPanel gridBPAnLabel = new JPanel();

JPanel buttonsArea = new JPanel();

JPanel buttonsPanel = new JPanel();

addSRBoxPanel.setLayout(new GridLayout(1,2,5,0));

listArea.setLayout(new BorderLayout());

list.setLayout(new BorderLayout());

infoButtonsArea.setLayout(new GridLayout(2,1,0,5));

infoPanel.setLayout(new GridLayout(5,2,0,5));

buttonsArea.setLayout(new GridBagLayout());

buttonsPanel.setLayout(new GridLayout(2,2,10,15));

gridBPTitlu.setLayout(new GridBagLayout());

gridBPEnorias.setLayout(new GridBagLayout());

gridBPZi.setLayout(new GridBagLayout());

gridBPLuna.setLayout(new GridBagLayout());

gridBPAn.setLayout(new GridBagLayout());

gridBPTitluLabel.setLayout(new GridBagLayout());

gridBPEnoriasLabel.setLayout(new GridBagLayout());

gridBPZiLabel.setLayout(new GridBagLayout());

gridBPLunaLabel.setLayout(new GridBagLayout());

gridBPAnLabel.setLayout(new GridBagLayout());

JLabel titluLabel = new JLabel("Titlu:");

JLabel enoriasLabel = new JLabel("Enoriaș:");

JLabel ziLabel = new JLabel("Zi:");

JLabel lunaLabel = new JLabel("Lună:");

JLabel anLabel = new JLabel("An:");

titluTA = new JTextArea(1,15);

enoriasTA = new JTextArea(1,15);

ziTA = new JTextArea(1,15);

lunaTA = new JTextArea(1,15);

anTA = new JTextArea(1,15);

titluTA.setBorder(BorderFactory.createLineBorder(Color.black));

enoriasTA.setBorder(BorderFactory.createLineBorder(Color.black));

ziTA.setBorder(BorderFactory.createLineBorder(Color.black));

lunaTA.setBorder(BorderFactory.createLineBorder(Color.black));

anTA.setBorder(BorderFactory.createLineBorder(Color.black));

enoriasTA.setEditable(false);

ArrayList<String> nume = new ArrayList<String>();

for (int i = 0; i < continut.size(); i++) {

nume.add(continut.get(i)[1]);

}

listaEnoriasi = new JList(nume.toArray());

listaEnoriasi.addListSelectionListener(listListener);

JScrollPane listaScroll = new JScrollPane();

listaScroll.setViewportView(listaEnoriasi);

adaugaButton = new JButton("Adaugă serviciu");

ajutorButton = new JButton("Ajutor");

inchideButton = new JButton("Închide fereastra");

reseteazaButton = new JButton("Șterge conținut formular");

adaugaButton.addActionListener(this);

ajutorButton.addActionListener(this);

inchideButton.addActionListener(this);

reseteazaButton.addActionListener(this);

list.add(listaScroll,BorderLayout.CENTER);

listArea.add(list, BorderLayout.CENTER);

gridBPTitluLabel.add(titluLabel);

gridBPTitlu.add(titluTA);

gridBPEnoriasLabel.add(enoriasLabel);

gridBPEnorias.add(enoriasTA);

gridBPZiLabel.add(ziLabel);

gridBPZi.add(ziTA);

gridBPLunaLabel.add(lunaLabel);

gridBPLuna.add(lunaTA);

gridBPAnLabel.add(anLabel);

gridBPAn.add(anTA);

infoPanel.add(gridBPTitluLabel);

infoPanel.add(gridBPTitlu);

infoPanel.add(gridBPEnoriasLabel);

infoPanel.add(gridBPEnorias);

infoPanel.add(gridBPZiLabel);

infoPanel.add(gridBPZi);

infoPanel.add(gridBPLunaLabel);

infoPanel.add(gridBPLuna);

infoPanel.add(gridBPAnLabel);

infoPanel.add(gridBPAn);

buttonsPanel.add(adaugaButton);

buttonsPanel.add(ajutorButton);

buttonsPanel.add(inchideButton);

buttonsPanel.add(reseteazaButton);

buttonsArea.add(buttonsPanel);

infoButtonsArea.add(infoPanel);

infoButtonsArea.add(buttonsArea);

addSRBoxPanel.add(listArea);

addSRBoxPanel.add(infoButtonsArea);

adaugareSBox.add(addSRBoxPanel);

}

ListSelectionListener listListener = new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if(!e.getValueIsAdjusting()){

enoriasTA.setText(continut.get(listaEnoriasi.getAnchorSelectionIndex())[1]);

}

}

};

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource() == inchideButton)

{

adaugareSBox.dispose();

} else if (e.getSource() == reseteazaButton)

{

titluTA.setText(null);

listaEnoriasi.clearSelection();

enoriasTA.setText(null);

ziTA.setText(null);

lunaTA.setText(null);

anTA.setText(null);

} else if (e.getSource() == ajutorButton) {

String continutAjutor = "În partea stângă a ferestrei „Adăugare serviciu religios” se află lista cu enoriașii parohiei din care se selectează enoriașul pentru care se programează serviciul religios.\n" +

"În partea dreaptă, în prima jumătate a secțiunii se află câmpurile „Titlu”,„Enoriaș”, „Zi”, „Lună”, „An”.\n" +

"În aceste câmpuri se introduc datele dorite.\n" +

"În a doua jumatate a secțiunii se află butoanele.\n" +

"Butonul „Adaugă serviciu” face exact ce spune numele.\n" +

"Butonul „Ajutor” afișează acest mesaj.\n" +

"Butonul „Închide fereastra” face exact ce sugerează numele." +

"Butonul „Șterge conținut formular” face ce sugerează numele.";

infoBox("Ajutor!",continutAjutor,300,270);

}

else if(e.getSource() == adaugaButton)

{

String error;

error = console.getDataNewServiciu(String.valueOf(continut.get(listaEnoriasi.getAnchorSelectionIndex())[0]),titluTA.getText(), anTA.getText(), lunaTA.getText(), ziTA.getText());

errorBox(error,"Serviciul a fost adăugat cu succes!");

}

}

}

**EditareServiciu:**

import com.sun.tools.javac.Main;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

public class EditareServiciu extends MainFrame implements ActionListener {

Console console;

String idSelectat;

JDialog adaugareSBox = new JDialog();

ArrayList<String[]> continut;

JButton salveazaButton, ajutorButton, inchideButton, reseteazaButton;

JList listaEnoriasi;

JTextArea titluTA,enoriasTA, ziTA, lunaTA, anTA;

EditareServiciu(Console console , String idSelectat)

{

this.console = console;

this.idSelectat = idSelectat;

continut = console.displayEnoriasi();

setup();

}

private void setup()

{

adaugareSBox.setSize(750,500);

adaugareSBox.setLocationRelativeTo(null);

adaugareSBox.setResizable(false);

adaugareSBox.setTitle("Editare serviciu religios");

adaugareSBox.setVisible(true);

JPanel addSRBoxPanel = new JPanel();

JPanel listArea = new JPanel();

JPanel list = new JPanel();

JPanel infoButtonsArea = new JPanel();

JPanel infoPanel = new JPanel();

JPanel gridBPTitlu = new JPanel();

JPanel gridBPEnorias = new JPanel();

JPanel gridBPZi = new JPanel();

JPanel gridBPLuna = new JPanel();

JPanel gridBPAn = new JPanel();

JPanel gridBPTitluLabel = new JPanel();

JPanel gridBPEnoriasLabel = new JPanel();

JPanel gridBPZiLabel = new JPanel();

JPanel gridBPLunaLabel = new JPanel();

JPanel gridBPAnLabel = new JPanel();

JPanel buttonsArea = new JPanel();

JPanel buttonsPanel = new JPanel();

addSRBoxPanel.setLayout(new GridLayout(1,2,5,0));

listArea.setLayout(new BorderLayout());

list.setLayout(new BorderLayout());

infoButtonsArea.setLayout(new GridLayout(2,1,0,5));

infoPanel.setLayout(new GridLayout(5,2,0,5));

buttonsArea.setLayout(new GridBagLayout());

buttonsPanel.setLayout(new GridLayout(2,2,10,15));

gridBPTitlu.setLayout(new GridBagLayout());

gridBPEnorias.setLayout(new GridBagLayout());

gridBPZi.setLayout(new GridBagLayout());

gridBPLuna.setLayout(new GridBagLayout());

gridBPAn.setLayout(new GridBagLayout());

gridBPTitluLabel.setLayout(new GridBagLayout());

gridBPEnoriasLabel.setLayout(new GridBagLayout());

gridBPZiLabel.setLayout(new GridBagLayout());

gridBPLunaLabel.setLayout(new GridBagLayout());

gridBPAnLabel.setLayout(new GridBagLayout());

JLabel titluLabel = new JLabel("Titlu:");

JLabel enoriasLabel = new JLabel("Enoriaș:");

JLabel ziLabel = new JLabel("Zi:");

JLabel lunaLabel = new JLabel("Lună:");

JLabel anLabel = new JLabel("An:");

titluTA = new JTextArea(1,15);

enoriasTA = new JTextArea(1,15);

ziTA = new JTextArea(1,15);

lunaTA = new JTextArea(1,15);

anTA = new JTextArea(1,15);

titluTA.setBorder(BorderFactory.createLineBorder(Color.black));

enoriasTA.setBorder(BorderFactory.createLineBorder(Color.black));

ziTA.setBorder(BorderFactory.createLineBorder(Color.black));

lunaTA.setBorder(BorderFactory.createLineBorder(Color.black));

anTA.setBorder(BorderFactory.createLineBorder(Color.black));

enoriasTA.setEditable(false);

ArrayList<String> nume = new ArrayList<String>();

for (int i = 0; i < continut.size(); i++) {

nume.add(continut.get(i)[1]);

}

listaEnoriasi = new JList(nume.toArray());

listaEnoriasi.addListSelectionListener(listListener);

JScrollPane listaScroll = new JScrollPane();

listaScroll.setViewportView(listaEnoriasi);

salveazaButton = new JButton("Salvează modficările");

ajutorButton = new JButton("Ajutor");

inchideButton = new JButton("Închide fereastra");

reseteazaButton = new JButton("Șterge conținut formular");

salveazaButton.addActionListener(this);

ajutorButton.addActionListener(this);

inchideButton.addActionListener(this);

reseteazaButton.addActionListener(this);

list.add(listaScroll,BorderLayout.CENTER);

listArea.add(list, BorderLayout.CENTER);

gridBPTitluLabel.add(titluLabel);

gridBPTitlu.add(titluTA);

gridBPEnoriasLabel.add(enoriasLabel);

gridBPEnorias.add(enoriasTA);

gridBPZiLabel.add(ziLabel);

gridBPZi.add(ziTA);

gridBPLunaLabel.add(lunaLabel);

gridBPLuna.add(lunaTA);

gridBPAnLabel.add(anLabel);

gridBPAn.add(anTA);

infoPanel.add(gridBPTitluLabel);

infoPanel.add(gridBPTitlu);

infoPanel.add(gridBPEnoriasLabel);

infoPanel.add(gridBPEnorias);

infoPanel.add(gridBPZiLabel);

infoPanel.add(gridBPZi);

infoPanel.add(gridBPLunaLabel);

infoPanel.add(gridBPLuna);

infoPanel.add(gridBPAnLabel);

infoPanel.add(gridBPAn);

buttonsPanel.add(salveazaButton);

buttonsPanel.add(ajutorButton);

buttonsPanel.add(inchideButton);

buttonsPanel.add(reseteazaButton);

buttonsArea.add(buttonsPanel);

infoButtonsArea.add(infoPanel);

infoButtonsArea.add(buttonsArea);

addSRBoxPanel.add(listArea);

addSRBoxPanel.add(infoButtonsArea);

adaugareSBox.add(addSRBoxPanel);

}

ListSelectionListener listListener = new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if(!e.getValueIsAdjusting()){

enoriasTA.setText(continut.get(listaEnoriasi.getAnchorSelectionIndex())[1]);

}

}

};

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource() == inchideButton)

{

adaugareSBox.dispose();

} else if (e.getSource() == reseteazaButton)

{

titluTA.setText(null);

listaEnoriasi.clearSelection();

enoriasTA.setText(null);

ziTA.setText(null);

lunaTA.setText(null);

anTA.setText(null);

} else if (e.getSource() == ajutorButton) {

String continutAjutor = "În partea stângă a ferestrei „Editare serviciu religios” se află lista cu enoriașii parohiei din care se selectează enoriașul pentru care se programează serviciul religios.\n" +

"În partea dreaptă, în prima jumătate a secțiunii se află câmpurile „Titlu”,„Enoriaș”, „Zi”, „Lună”, „An”.\n" +

"În aceste câmpuri se modifică datele dorite.\n" +

"În a doua jumatate a secțiunii se află butoanele.\n" +

"Butonul „Salvează modificările” face exact ce spune numele.\n" +

"Butonul „Ajutor” afișează acest mesaj.\n" +

"Butonul „Închide fereastra” face exact ce sugerează numele." +

"Butonul „Șterge conținut formular” face ce sugerează numele.";

infoBox("Ajutor!",continutAjutor,300,270);

}

else if(e.getSource() == salveazaButton)

{

String error;

error = console.callUpdateServiciu(idSelectat,continut.get(listaEnoriasi.getAnchorSelectionIndex())[0],titluTA.getText(),anTA.getText(),lunaTA.getText(),ziTA.getText());

errorBox(error,"Serviciul a fost modificat cu succes!");

}

}

}

**PrediciAnunturiDBox:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

public class PrediciAnunturiDBox extends MainFrame implements ActionListener {

Console console;

private JDialog prediciAnunturiDialogBox = new JDialog();

private JComboBox chooseType;

private JTextArea title,text;

private JButton urmatorulButton, anteriorulButton, adaugaButton, editButton, stergeButton,ajutorButton,salveazaButton,resetButton,actualizareButton;

private ArrayList<String[]> continut;

private int position = 0, minPosition = 0, maxPosition;

public PrediciAnunturiDBox(Console console)

{

this.console = console;

continut = this.console.displayAnunturi();

if(continut.isEmpty())

{

maxPosition = 0;

}

else

{

maxPosition = continut.size()-1;

}

prediciAnunturiBox();

}

private void setTextArea()

{

title.setText(continut.get(position)[1]);

text.setText(continut.get(position)[2]);

if(continut.get(position)[3].equals("false"))

{

chooseType.setSelectedIndex(0);

}

else

{

chooseType.setSelectedIndex(1);

}

}

private void prediciAnunturiBox()

{

prediciAnunturiDialogBox.setSize(715,715);

prediciAnunturiDialogBox.setLocationRelativeTo(null);

prediciAnunturiDialogBox.setResizable(false);

prediciAnunturiDialogBox.setTitle("Manager predici/anunțuri");

prediciAnunturiDialogBox.setVisible(true);

JPanel paDialogBox = new JPanel();

JPanel textPanel = new JPanel();

JPanel textArea = new JPanel();

JPanel titlePanel = new JPanel();

JPanel buttonsArea = new JPanel();

JPanel buttonsPanel = new JPanel();

JLabel typeLabel = new JLabel("Tip Slide: ");

JLabel titleLabel = new JLabel("Title:");

chooseType = new JComboBox();

chooseType.addItem("Predică");

chooseType.addItem("Anunț");

chooseType.setEnabled(false);

title = new JTextArea(1,13);

title.setBorder(BorderFactory.createLineBorder(Color.black));

title.setLineWrap(true);

title.setEditable(false);

JScrollPane textScroll = new JScrollPane();

text = new JTextArea();

text.setLineWrap(true);

text.setWrapStyleWord(true);

text.setEditable(false);

textScroll.setViewportView(text);

if(!continut.isEmpty())

{

setTextArea();

}

paDialogBox.setLayout(new GridLayout(2,1));

textArea.setLayout(new BorderLayout());

textPanel.setLayout(new BorderLayout());

buttonsArea.setLayout(new GridBagLayout());

buttonsPanel.setLayout(new GridLayout(3,3,5,30));

urmatorulButton = new JButton("Slide-ul următor");

anteriorulButton = new JButton("Slide-ul anterior");

adaugaButton = new JButton("Adaugă un slide nou");

editButton = new JButton("Editează slide-ul curent");

stergeButton = new JButton("Șterge slide-ul curent");

ajutorButton = new JButton("Ajutor");

salveazaButton = new JButton("Salvează modificările efectuate");

resetButton = new JButton("Șterge conținutul slide-ului curent");

actualizareButton = new JButton("Actualizare listă");

salveazaButton.setEnabled(false);

resetButton.setEnabled(false);

adaugaButton.addActionListener(this);

ajutorButton.addActionListener(this);

urmatorulButton.addActionListener(this);

anteriorulButton.addActionListener(this);

actualizareButton.addActionListener(this);

stergeButton.addActionListener(this);

editButton.addActionListener(this);

salveazaButton.addActionListener(this);

resetButton.addActionListener(this);

titlePanel.add(typeLabel);

titlePanel.add(chooseType);

titlePanel.add(titleLabel);

titlePanel.add(title);

textPanel.add(textScroll,BorderLayout.CENTER);

textArea.add(titlePanel,BorderLayout.PAGE\_START);

textArea.add(textPanel,BorderLayout.CENTER);

buttonsPanel.add(urmatorulButton);

buttonsPanel.add(anteriorulButton);

buttonsPanel.add(adaugaButton);

buttonsPanel.add(editButton);

buttonsPanel.add(stergeButton);

buttonsPanel.add(ajutorButton);

buttonsPanel.add(salveazaButton);

buttonsPanel.add(actualizareButton);

buttonsPanel.add(resetButton);

buttonsArea.add(buttonsPanel);

paDialogBox.add(textArea);

paDialogBox.add(buttonsArea);

prediciAnunturiDialogBox.add(paDialogBox);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==adaugaButton)

{

AdaugarePredicaAnunt nou = new AdaugarePredicaAnunt(this.console);

continut = console.displayAnunturi();

maxPosition++;

}

else if (e.getSource()==urmatorulButton)

{ if(continut.isEmpty())

{

errorBoxGUI();

}

else{

if(position<maxPosition)

{

position++;

setTextArea();

} else if (position==maxPosition)

{

position = 0;

setTextArea();

}

}

}

else if (e.getSource()==anteriorulButton)

{ if(continut.isEmpty())

{

errorBoxGUI();

}

else{

if(position>minPosition)

{

position--;

setTextArea();

} else if (position==minPosition)

{

position = maxPosition;

setTextArea();

}

}

}

else if(e.getSource()==ajutorButton)

{ String continutAjutor;

int latime, inaltime;

continutAjutor = "În partea de sus a ferestrei „Manager predici/anunțuri”, în stânga, este afișat tipul slide-ului curent " +

"În partea dreaptă se află câmpul în care se află titlu slide-ului curent.\n" +

"În partea de mijloc a ferestrei se află zona în care se află conținutul slide-ului curent.\n" +

"În partea de jos a ferestrei se află butoanele.\nButonul „Ajutor” afișează acest mesaj.\nButonul „Actualizare listă” " +

"repornește fereastra „Manager predici/anunțuri” pentru a actualiza lista.\nButonul „Slide-ul următor” trece la slide-ul următor.\n" +

"Butonul „Slide-ul anterior” trece la slide-ul anterior.\nButonul „Șterge slide-ul curent” șterge slide-ul curent." +

"\nButonul „Editează slide-ul curent” activează editarea câmpurilor din slide-ul afișat cât și butoanele „Salvează modificările efectuate” și „Șterge conținutul slide-ului curent” care fac ce sugerează numele.";

infoBox("Ajutor!",continutAjutor,350,300);

}else if(e.getSource()==editButton)

{

if(continut.isEmpty())

{

errorBoxGUI();

}

else {

salveazaButton.setEnabled(true);

resetButton.setEnabled(true);

title.setEditable(true);

text.setEditable(true);

chooseType.setEnabled(true);

}

}

else if(e.getSource()==actualizareButton)

{

prediciAnunturiDialogBox.dispose();

PrediciAnunturiDBox restart = new PrediciAnunturiDBox(this.console);

}

else if(e.getSource()==stergeButton)

{

if(continut.isEmpty())

{

errorBoxGUI();

}

else {

String error;

error = console.removeAnunt(continut.get(position)[0]);

//errorBox(error, "Slide-ul a fost șters cu succes!");

actualizareButton.doClick();

}

}

else if(e.getSource()==resetButton)

{

title.setText(null);

text.setText(null);

chooseType.setSelectedIndex(0);

} else if (e.getSource()==salveazaButton)

{

String change;

if(chooseType.getSelectedIndex() == 0)

{

change = "false";

}

else

{

change = "true";

}

String error;

error = console.callUpdateAnunt(continut.get(position)[0],title.getText(),text.getText(),change);

errorBox(error,"Slide-ul a fost editat cu succes!");

if(error.equals("valid!"))

{salveazaButton.setEnabled(false);

resetButton.setEnabled(false);

title.setEditable(false);

text.setEditable(false);

chooseType.setEditable(false);}

actualizareButton.doClick();

}

}

}

**AdaugarePredicaAnunt:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

public class AdaugarePredicaAnunt extends MainFrame implements ActionListener {

Console console;

JDialog adaugaSlide = new JDialog();

JButton adaugaButton, resetButton, exitButton, ajutorButton;

JTextArea title,text;

JComboBox chooseType;

public AdaugarePredicaAnunt(Console console){

this.console = console;

formularAdaugare();

}

private void formularAdaugare()

{

adaugaSlide.setTitle("Adauga un slide nou!");

adaugaSlide.setSize(605,605);

adaugaSlide.setLocationRelativeTo(null);

adaugaSlide.setResizable(false);

adaugaSlide.setVisible(true);

JPanel adaugaPanel = new JPanel();

JPanel typeTextPanel = new JPanel();

JPanel typeTextArea = new JPanel();

JPanel buttonsArea = new JPanel();

JPanel buttonsPanel = new JPanel();

text = new JTextArea();

text.setLineWrap(true);

text.setWrapStyleWord(true);

text.setBorder(BorderFactory.createLineBorder(Color.black));

title = new JTextArea(1,13);

title.setWrapStyleWord(true);

title.setLineWrap(true);

title.setBorder(BorderFactory.createLineBorder(Color.black));

JScrollPane textScroll = new JScrollPane();

textScroll.setViewportView(text);

adaugaPanel.setLayout(new GridLayout(2,1));

typeTextArea.setLayout(new BorderLayout());

buttonsArea.setLayout(new GridBagLayout());

buttonsPanel.setLayout(new GridLayout(2,2,30,30));

chooseType = new JComboBox();

chooseType.addItem("Predică");

chooseType.addItem("Anunț");

JLabel typeLabel = new JLabel("Tip slide:");

JLabel titleLabel = new JLabel("Titlu:");

adaugaButton = new JButton("Adaugă slide-ul la listă");

resetButton = new JButton("Restează câmpurile");

exitButton = new JButton("Închide fereastra");

ajutorButton = new JButton("Ajutor");

adaugaButton.addActionListener(this);

resetButton.addActionListener(this);

exitButton.addActionListener(this);

ajutorButton.addActionListener(this);

typeTextPanel.add(typeLabel);

typeTextPanel.add(chooseType);

typeTextPanel.add(titleLabel);

typeTextPanel.add(title);

typeTextArea.add(typeTextPanel,BorderLayout.PAGE\_START);

typeTextArea.add(textScroll,BorderLayout.CENTER);

buttonsPanel.add(adaugaButton);

buttonsPanel.add(resetButton);

buttonsPanel.add(ajutorButton);

buttonsPanel.add(exitButton);

buttonsArea.add(buttonsPanel);

adaugaPanel.add(typeTextArea);

adaugaPanel.add(buttonsArea);

adaugaSlide.add(adaugaPanel);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==adaugaButton)

{ String error;

error = console.getDataNewAnunt(title.getText(),text.getText(),String.valueOf(chooseType.getSelectedIndex()+1));

errorBox(error,"Slide-ul a fost adăugat cu succes!");

}

else if (e.getSource()==resetButton) {

chooseType.setSelectedIndex(0);

title.setText(null);

text.setText(null);

}

else if (e.getSource()==exitButton) {

adaugaSlide.dispose();

}

else if (e.getSource()==ajutorButton) {

String continutAjutor = "În partea de sus a ferestrei „Adaugă un slide nou!”, în stânga, puteți selecta tipul dorit " +

"al slide-ului dacă apăsați pe săgeată. În partea dreaptă se află câmpul în care puteți introduce titlu.\n" +

"În partea de mijloc a ferestrei se află zona în care puteți introduce conținutul slide-ului.\n" +

"În partea de jos a ferestrei se află butoanele.\nButonul „Ajutor” afișează acest mesaj.\nButonul „Închide fereastra” " +

"închide fereastra „Adaugă un slide nou!”.\nButonul „Resetează câmpurile” șterge datele introduse și setează tipul slide-ului drept „Predică”.\n" +

"Butonul „Adaugă slide-ul la listă” adaugă formularul completat în lista de anunțuri/predici.";

infoBox("Ajutor!",continutAjutor,300,275);

}

}

}

**SlujbeDialogBox:**

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

public class SlujbeDialogBox extends MainFrame implements ActionListener {

JDialog slujbaDialogBox = new JDialog();

Console console;

ArrayList<String[]> continut;

JButton adaugaButton, stergeButton, editButton, actualizeazaButton, ajutorButton, inchideButton, salveazaButton, reseteazaButton;

JList listaSlujbe;

JTextArea titleTA, ziTA, lunaTA, anTA;

SlujbeDialogBox(Console console) {

continut = console.displaySlujbe();

this.console = console;

slujbeBox();

}

private void slujbeBox()

{

slujbaDialogBox.setSize(750,500);

slujbaDialogBox.setLocationRelativeTo(null);

slujbaDialogBox.setResizable(false);

slujbaDialogBox.setTitle("Manager slujbe");

slujbaDialogBox.setVisible(true);

JPanel slujbaBoxPanel = new JPanel();

JPanel listArea = new JPanel();

JPanel list = new JPanel();

JPanel infoButtonsArea = new JPanel();

JPanel infoPanel = new JPanel();

JPanel gridBPTitle = new JPanel();

JPanel gridBPZi = new JPanel();

JPanel gridBPLuna = new JPanel();

JPanel gridBPAn = new JPanel();

JPanel gridBPTitleLabel = new JPanel();

JPanel gridBPZiLabel = new JPanel();

JPanel gridBPLunaLabel = new JPanel();

JPanel gridBPAnLabel = new JPanel();

JPanel buttonsArea = new JPanel();

JPanel buttonsPanel = new JPanel();

slujbaBoxPanel.setLayout(new GridLayout(1,2,5,0));

listArea.setLayout(new BorderLayout());

list.setLayout(new BorderLayout());

infoButtonsArea.setLayout(new GridLayout(2,1,0,5));

infoPanel.setLayout(new GridLayout(4,2,0,5));

buttonsArea.setLayout(new GridBagLayout());

buttonsPanel.setLayout(new GridLayout(4,2,10,15));

gridBPTitle.setLayout(new GridBagLayout());

gridBPZi.setLayout(new GridBagLayout());

gridBPLuna.setLayout(new GridBagLayout());

gridBPAn.setLayout(new GridBagLayout());

gridBPTitleLabel.setLayout(new GridBagLayout());

gridBPZiLabel.setLayout(new GridBagLayout());

gridBPLunaLabel.setLayout(new GridBagLayout());

gridBPAnLabel.setLayout(new GridBagLayout());

JLabel titluLabel = new JLabel("Titlu:");

JLabel ziLabel = new JLabel("Zi:");

JLabel lunaLabel = new JLabel("Lună:");

JLabel anLabel = new JLabel("An:");

titleTA = new JTextArea(1,15);

ziTA = new JTextArea(1,15);

lunaTA = new JTextArea(1,15);

anTA = new JTextArea(1,15);

titleTA.setBorder(BorderFactory.createLineBorder(Color.black));

ziTA.setBorder(BorderFactory.createLineBorder(Color.black));

lunaTA.setBorder(BorderFactory.createLineBorder(Color.black));

anTA.setBorder(BorderFactory.createLineBorder(Color.black));

titleTA.setEditable(false);

ziTA.setEditable(false);

lunaTA.setEditable(false);

anTA.setEditable(false);

ArrayList<String> nume = new ArrayList<String>();

for (int i = 0; i < continut.size(); i++) {

nume.add(continut.get(i)[1]);

}

listaSlujbe = new JList(nume.toArray());

listaSlujbe.addListSelectionListener(listListener);

JScrollPane listaScroll = new JScrollPane();

listaScroll.setViewportView(listaSlujbe);

adaugaButton = new JButton("Adaugă slujbă");

stergeButton = new JButton("Șterge slujba selectată");

editButton = new JButton("Activare câmpuri");

actualizeazaButton = new JButton("Actualizează lista");

ajutorButton = new JButton("Ajutor");

inchideButton = new JButton("Închide fereastra");

salveazaButton = new JButton("Salvează");

reseteazaButton = new JButton("Șterge conținut formular");

adaugaButton.addActionListener(this);

stergeButton.addActionListener(this);

editButton.addActionListener(this);

actualizeazaButton.addActionListener(this);

ajutorButton.addActionListener(this);

inchideButton.addActionListener(this);

salveazaButton.addActionListener(this);

reseteazaButton.addActionListener(this);

salveazaButton.setEnabled(false);

reseteazaButton.setEnabled(false);

list.add(listaScroll,BorderLayout.CENTER);

listArea.add(list, BorderLayout.CENTER);

gridBPTitleLabel.add(titluLabel);

gridBPTitle.add(titleTA);

gridBPZiLabel.add(ziLabel);

gridBPZi.add(ziTA);

gridBPLunaLabel.add(lunaLabel);

gridBPLuna.add(lunaTA);

gridBPAnLabel.add(anLabel);

gridBPAn.add(anTA);

infoPanel.add(gridBPTitleLabel);

infoPanel.add(gridBPTitle);

infoPanel.add(gridBPZiLabel);

infoPanel.add(gridBPZi);

infoPanel.add(gridBPLunaLabel);

infoPanel.add(gridBPLuna);

infoPanel.add(gridBPAnLabel);

infoPanel.add(gridBPAn);

buttonsPanel.add(adaugaButton);

buttonsPanel.add(stergeButton);

buttonsPanel.add(editButton);

buttonsPanel.add(actualizeazaButton);

buttonsPanel.add(ajutorButton);

buttonsPanel.add(inchideButton);

buttonsPanel.add(salveazaButton);

buttonsPanel.add(reseteazaButton);

buttonsArea.add(buttonsPanel);

infoButtonsArea.add(infoPanel);

infoButtonsArea.add(buttonsArea);

slujbaBoxPanel.add(listArea);

slujbaBoxPanel.add(infoButtonsArea);

slujbaDialogBox.add(slujbaBoxPanel);

}

@Override

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==adaugaButton)

{

String error;

error = console.getDataNewSlujba(titleTA.getText(),anTA.getText(),lunaTA.getText(),ziTA.getText());

if(error.equals("valid!"))

{

actualizeazaButton.doClick();

errorBox(error,"Slujba a fost adăugată cu succes!");

}

else

{

errorBox(error,"Slujba a fost adăugată cu succes!");

}

} else if (e.getSource() == actualizeazaButton)

{

slujbaDialogBox.dispose();

SlujbeDialogBox reset = new SlujbeDialogBox(this.console);

}

else if(e.getSource() == stergeButton)

{ String error;

if(!continut.isEmpty()) {

error = console.removeSlujba(continut.get(listaSlujbe.getAnchorSelectionIndex())[0]);

actualizeazaButton.doClick();

}

else{

errorBoxGUI();

}

}

else if(e.getSource() == editButton)

{

titleTA.setEditable(true);

ziTA.setEditable(true);

lunaTA.setEditable(true);

anTA.setEditable(true);

salveazaButton.setEnabled(true);

reseteazaButton.setEnabled(true);

} else if (e.getSource() == reseteazaButton )

{

titleTA.setText(null);

ziTA.setText(null);

lunaTA.setText(null);

anTA.setText(null);

}

else if(e.getSource() == salveazaButton)

{ if(continut.isEmpty())

{

errorBoxGUI();

}

else {

String error;

error = console.callUpdateSlujba(continut.get(listaSlujbe.getAnchorSelectionIndex())[0], titleTA.getText(), anTA.getText(), lunaTA.getText(), ziTA.getText());

errorBox(error, "Modificările au fost salvate!");

if(error.equals("valid!"))

{

titleTA.setEditable(false);

ziTA.setEditable(false);

lunaTA.setEditable(false);

anTA.setEditable(false);

actualizeazaButton.doClick();

}

}

} else if (e.getSource()==inchideButton)

{

slujbaDialogBox.dispose();

}

else if(e.getSource()==ajutorButton)

{

String continutAjutor;

continutAjutor = "În partea stângă a ferestrei „Manager Slujbe” se află lista de slujbe create și programate\n" +

"În partea dreaptă, în prima jumătate a secțiunii se află câmpurile „Titlu”, „Zi”, „Lună”, „An”.\n" +

"Aceste câmpuri se actualizează cu informațiile slujbei selectate și sunt folosite pentru operațiile de adăugare și editare.\n" +

"În a doua jumatate a secțiunii se află butoanele.\n" +

"Butonul „Adaugă slujbă” face exact ce spune numele.\n" +

"Pentru a folosi butonul „Adaugă slujbă” trebuie întâi apăsat butonul „Activare câmpuri”.\n" +

"Butonul „Activare câmpuri” setează câmpurile drept editabile și activează butoanele „Salvează” și „Șterge conținut formular”, care fac exact ce sugerează numele lor.\n" +

"Butonul „Ajutor” afișează acest mesaj.\n" +

"Butonul „Șterge slujba selectată” face exact ce sugerează numele.\n" +

"Butonul „Actualizează lista” face exact ce sugerează numele.\n" +

"Butonul „Închide fereastra” face exact ce sugerează numele.";

infoBox("Ajutor!",continutAjutor,350,335);

}

}

ListSelectionListener listListener = new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if(!e.getValueIsAdjusting()){

titleTA.setText(continut.get(listaSlujbe.getAnchorSelectionIndex())[1]);

ziTA.setText(continut.get(listaSlujbe.getAnchorSelectionIndex())[4]);

lunaTA.setText(continut.get(listaSlujbe.getAnchorSelectionIndex())[3]);

anTA.setText(continut.get(listaSlujbe.getAnchorSelectionIndex())[2]);

}

}

};

}

**EnoriașiDialogBox:**

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

public class EnoriasiDialogBox extends MainFrame implements ActionListener {

Console console;

JDialog enoriasiDB = new JDialog();

ArrayList<String[]> continut;

JButton adaugaButton, stergeButton, editButton, actualizeazaButton, ajutorButton, inchideButton, salveazaButton, reseteazaButton;

JTextArea numeTA, telefonTA, adresaTA;

JList listaEnoriasi;

EnoriasiDialogBox(Console console)

{

this.console = console;

continut = console.displayEnoriasi();

enoriasDBSetup();

}

private void enoriasDBSetup()

{

enoriasiDB.setSize(750,500);

enoriasiDB.setLocationRelativeTo(null);

enoriasiDB.setResizable(false);

enoriasiDB.setTitle("Manager enoriași");

enoriasiDB.setVisible(true);

JPanel enoriasBoxPanel = new JPanel();

JPanel listArea = new JPanel();

JPanel list = new JPanel();

JPanel infoButtonsArea = new JPanel();

JPanel infoPanel = new JPanel();

JPanel gridBPNume = new JPanel();

JPanel gridBPTelefon = new JPanel();

JPanel gridBPAdresa = new JPanel();

JPanel gridBPNumeLabel = new JPanel();

JPanel gridBPTelefonLabel = new JPanel();

JPanel gridBPAdresaLabel = new JPanel();

JPanel buttonsArea = new JPanel();

JPanel buttonsPanel = new JPanel();

enoriasBoxPanel.setLayout(new GridLayout(1,2,5,0));

listArea.setLayout(new BorderLayout());

list.setLayout(new BorderLayout());

infoButtonsArea.setLayout(new GridLayout(2,1,0,5));

infoPanel.setLayout(new GridLayout(3,2,0,5));

buttonsArea.setLayout(new GridBagLayout());

buttonsPanel.setLayout(new GridLayout(4,2,10,15));

gridBPNume.setLayout(new GridBagLayout());

gridBPTelefon.setLayout(new GridBagLayout());

gridBPAdresa.setLayout(new GridBagLayout());

gridBPNumeLabel.setLayout(new GridBagLayout());

gridBPTelefonLabel.setLayout(new GridBagLayout());

gridBPAdresaLabel.setLayout(new GridBagLayout());

JLabel numeLabel = new JLabel("Nume:");

JLabel telefonLabel = new JLabel("Telefon:");

JLabel adresaLabel = new JLabel("Adresă:");

numeTA = new JTextArea(1,15);

telefonTA = new JTextArea(1,15);

adresaTA = new JTextArea(1,15);

numeTA.setBorder(BorderFactory.createLineBorder(Color.black));

telefonTA.setBorder(BorderFactory.createLineBorder(Color.black));

adresaTA.setBorder(BorderFactory.createLineBorder(Color.black));

numeTA.setEditable(false);

telefonTA.setEditable(false);

adresaTA.setEditable(false);

ArrayList<String> nume = new ArrayList<String>();

for (int i = 0; i < continut.size(); i++) {

nume.add(continut.get(i)[1]);

}

listaEnoriasi = new JList(nume.toArray());

listaEnoriasi.addListSelectionListener(listListener);

JScrollPane listaScroll = new JScrollPane();

listaScroll.setViewportView(listaEnoriasi);

adaugaButton = new JButton("Adaugă enoriaș");

stergeButton = new JButton("Șterge enoriașul selectat");

editButton = new JButton("Activare câmpuri");

actualizeazaButton = new JButton("Actualizează lista");

ajutorButton = new JButton("Ajutor");

inchideButton = new JButton("Închide fereastra");

salveazaButton = new JButton("Salvează");

reseteazaButton = new JButton("Șterge conținut formular");

adaugaButton.addActionListener(this);

stergeButton.addActionListener(this);

editButton.addActionListener(this);

actualizeazaButton.addActionListener(this);

ajutorButton.addActionListener(this);

inchideButton.addActionListener(this);

salveazaButton.addActionListener(this);

reseteazaButton.addActionListener(this);

salveazaButton.setEnabled(false);

reseteazaButton.setEnabled(false);

list.add(listaScroll,BorderLayout.CENTER);

listArea.add(list, BorderLayout.CENTER);

gridBPNumeLabel.add(numeLabel);

gridBPNume.add(numeTA);

gridBPTelefonLabel.add(telefonLabel);

gridBPTelefon.add(telefonTA);

gridBPAdresaLabel.add(adresaLabel);

gridBPAdresa.add(adresaTA);

infoPanel.add(gridBPNumeLabel);

infoPanel.add(gridBPNume);

infoPanel.add(gridBPTelefonLabel);

infoPanel.add(gridBPTelefon);

infoPanel.add(gridBPAdresaLabel);

infoPanel.add(gridBPAdresa);

buttonsPanel.add(adaugaButton);

buttonsPanel.add(stergeButton);

buttonsPanel.add(editButton);

buttonsPanel.add(actualizeazaButton);

buttonsPanel.add(ajutorButton);

buttonsPanel.add(inchideButton);

buttonsPanel.add(salveazaButton);

buttonsPanel.add(reseteazaButton);

buttonsArea.add(buttonsPanel);

infoButtonsArea.add(infoPanel);

infoButtonsArea.add(buttonsArea);

enoriasBoxPanel.add(listArea);

enoriasBoxPanel.add(infoButtonsArea);

enoriasiDB.add(enoriasBoxPanel);

}

ListSelectionListener listListener = new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if(!e.getValueIsAdjusting()){

numeTA.setText(continut.get(listaEnoriasi.getAnchorSelectionIndex())[1]);

telefonTA.setText(continut.get(listaEnoriasi.getAnchorSelectionIndex())[2]);

adresaTA.setText(continut.get(listaEnoriasi.getAnchorSelectionIndex())[3]);

}

}

};

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == adaugaButton) {

String error;

error = console.getDataNewEnorias(numeTA.getText(), telefonTA.getText(), adresaTA.getText());

if (error.equals("valid!")) {

actualizeazaButton.doClick();

errorBox(error, "Enoriaș a fost adăugată cu succes!");

} else {

errorBox(error, "Enoriaș a fost adăugată cu succes!");

}

} else if (e.getSource() == editButton) {

numeTA.setEditable(true);

telefonTA.setEditable(true);

adresaTA.setEditable(true);

salveazaButton.setEnabled(true);

reseteazaButton.setEnabled(true);

} else if (e.getSource() == actualizeazaButton) {

enoriasiDB.dispose();

EnoriasiDialogBox reset = new EnoriasiDialogBox(this.console);

} else if (e.getSource() == stergeButton) {

String error;

if (!continut.isEmpty()) {

error = console.removeEnorias(continut.get(listaEnoriasi.getAnchorSelectionIndex())[0]);

actualizeazaButton.doClick();

} else {

errorBoxGUI();

}

} else if (e.getSource() == inchideButton) {

enoriasiDB.dispose();

} else if (e.getSource() == reseteazaButton) {

numeTA.setText(null);

telefonTA.setText(null);

adresaTA.setText(null);

} else if (e.getSource() == salveazaButton) {

if (continut.isEmpty()) {

errorBoxGUI();

} else {

String error;

error = console.callUpdateEnorias(continut.get(listaEnoriasi.getAnchorSelectionIndex())[0], numeTA.getText(), telefonTA.getText(), adresaTA.getText());

errorBox(error, "Modificările au fost salvate!");

if (error.equals("valid!")) {

numeTA.setEditable(false);

telefonTA.setEditable(false);

adresaTA.setEditable(false);

actualizeazaButton.doClick();

}

}

} else if(e.getSource()== ajutorButton)

{

String continutAjutor;

continutAjutor = "În partea stângă a ferestrei „Manager enoriași” se află lista cu enoriașii care aparțin de parohie.\n" +

"În partea dreaptă, în prima jumătate a secțiunii se află câmpurile „Nume”, „Telefon”, „Adresă”.\n" +

"Aceste câmpuri se actualizează cu informațiile enoriașului selectat și sunt folosite pentru operațiile de adăugare și editare.\n" +

"În a doua jumatate a secțiunii se află butoanele.\n" +

"Butonul „Adaugă enoriaș” face exact ce sugerează numele.\n" +

"Pentru a folosi butonul „Adaugă enoriaș” trebuie întâi apăsat butonul „Activare câmpuri”.\n" +

"Butonul „Activare câmpuri” setează câmpurile drept editabile și activează butoanele „Salvează” și „Șterge conținut formular”, care fac exact ce sugerează numele lor.\n" +

"Butonul „Ajutor” afișează acest mesaj.\n" +

"Butonul „Șterge enoriașul selectat” face exact ce sugerează numele.\n" +

"Butonul „Actualizează lista” face exact ce sugerează numele.\n" +

"Butonul „Închide fereastra” face exact ce sugerează numele.";

infoBox("Ajutor!",continutAjutor,350,335);

}

}

}

**Test:**

import java.io.FileNotFoundException;

public class Test {

public static void main(String[] args) throws FileNotFoundException {

Repository<Anunt> anuntRepository = new Repository<Anunt>();

Repository<Slujba> slujbaRepository = new Repository<Slujba>();

Repository<Enorias> enoriasRepository = new Repository<Enorias>();

Repository<Serviciu> serviciuRepository = new Repository<Serviciu>();

AnuntService anuntService = new AnuntService(anuntRepository);

SlujbaService slujbaService = new SlujbaService(slujbaRepository);

EnoriasService enoriasService = new EnoriasService(enoriasRepository);

ServiciuService serviciuService = new ServiciuService(serviciuRepository);

Console console = new Console(anuntService,slujbaService,enoriasService, serviciuService);

FileRepo fileRepo = new FileRepo(anuntService,slujbaService,enoriasService,serviciuService);

MainFrame app = new MainFrame(console,fileRepo);

}

}