**MINISTERUL EDUСAŢIEI ŞI СERСETĂRII AL REPUBLIСII MOLDOVA**

**UNIVERSITATEA DE STAT „ALEСU RUSSO” DIN BĂLŢI**

**FAСULTATEA DE ŞTIINŢE REALE, EСONOMIСE ȘI ALE MEDIULUI**

**СATEDRA DE MATEMATIСĂ ȘI INFORMATIСĂ**

**LUCRARE DE LABORATOR NR. 6**

**„Bibliotecă”**

**Curs „Platforma JAVA Enterprise II”**

**Coordonator:**

**Olga CERBU**

dr.,conf.univ

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**Autor:**

Studentul grupei AW21M

**Dumitru BAZAOСHI**

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**BĂLȚI, 2023**

**CUPRINS**

[Scop. Condiția problemei 2](#_Toc149204577)

[Cod Java – Clasa Book 2](#_Toc149204578)

[Cod Java – Clasa Library 7](#_Toc149204579)

[Cod Java – Clasa AdvancedLibrary 11](#_Toc149204580)

[Cod Java – Clasa ToCSV 13](#_Toc149204581)

[Cod Java – Clasa Main 17](#_Toc149204582)

[Concluzii 35](#_Toc149204583)

[Referințe bibliografice 35](#_Toc149204584)

Scop. Condiția problemei

Dezvoltarea în IDE a proiectului bibliotecă care va gestiona datele colecției de cărți. Fiecare bibliotecă se bazează pe clasa Bibliotecă, care conține ca caracteristici adresa, telefonul, cărțile și zona. O carte se bazează pe clasa Book, care se caracterizează prin gen și autori. Clasa Library va fi derivată din clasa AdvancedLibrary, la care se vor adăuga următoarele funcții: numărul de etaje și numele. Creați o clasă Main în care veți seta o listă de elemente AdvancedLibrary ale căror elemente vor fi citite de la tastatură.

Cod Java – Clasa Book

package Library;/\*

\* Book descrie cartea prin urmatoarele caracteristici:

\* ISBN, titlu, autor, gen, data publicarii, limba, nr de pagini, pretul, disponibilitatea

\*/

import java.time.LocalDate;

/\*\*

\* data 12.10.2023

\*

\* @author Bazaochi Dumitru, AW21M

\*/

public class Book {

private int isbn; // Variabila pentru ISBN-ul cărții

private String title; // Variabila pentru titlul cărții

private String author; // Variabila pentru autorul cărții

private String genre; // Variabila pentru genul cărții

private LocalDate dateOfRelease; // Variabila pentru data de publicare a cărții

private String lang; // Variabila pentru limba cărții

private int numberOfPages; // Variabila pentru numărul de pagini al cărții

private double price; // Variabila pentru prețul cărții

private String avalabil; // Variabila pentru disponibilitatea cărții

// Constructorul fără argumente al clasei

public Book() {

}

// Constructorul cu argumente al clasei

public Book(int isbn, String title, String author, String genre, LocalDate dateOfRelease, String lang, int numberOfPages, double price, String avalabil) {

this.isbn = isbn;

this.title = title;

this.author = author;

this.genre = genre;

this.dateOfRelease = dateOfRelease;

this.lang = lang;

this.numberOfPages = numberOfPages;

this.price = price;

this.avalabil = avalabil;

}

// Metodele getter și setter pentru parametrii cărții

// (pentru accesul și actualizarea variabilelor private)

public int getIsbn() {

return isbn;

}

public void setIsbn(int isbn) {

this.isbn = isbn;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getGenre() {

return genre;

}

public void setGenre(String genre) {

this.genre = genre;

}

public LocalDate getDateOfRelease() {

return dateOfRelease;

}

public void setDateOfRelease(LocalDate dateOfRelease) {

this.dateOfRelease = dateOfRelease;

}

public String getLang() {

return lang;

}

public void setLang(String lang) {

this.lang = lang;

}

public int getNumberOfPages() {

return numberOfPages;

}

public void setNumberOfPages(int numberOfPages) {

this.numberOfPages = numberOfPages;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public String getAvalabil() {

return avalabil;

}

public void setAvalabil(String avalabil) {

this.avalabil = avalabil;

}

@Override

// Metoda pentru generarea codului hash al obiectului

public int hashCode() {

int hash = 5; // declararea și inițializarea variabilei cod hash

hash = 79 \* hash + this.isbn; // algoritm pentru obținerea unui cod hash unic pentru fiecare obiect

return hash; // returnează valoarea codului hash

}

@Override

// Metoda pentru compararea a doi obiecte din clasa Library.Book pe baza ISBN-ului

public boolean equals(Object obj) {

if (this == obj) { // este verificat dacă obiectul este însuși.

return true; // returnează True dacă declarația if este adevărată

}

if (obj == null) {// verificat dacă obiectul este nul

return false; // returnează False dacă declarația if este adevărată

}

if (getClass() != obj.getClass()) {// verifica dacă obiectul nu este un obiect al clasei date

return false;// returnează False dacă declarația if este adevărată

}

final Book other = (Book) obj; // atribuirea unui obiect obj cu conversie explicită unui obiect din această clasă

// returnează True dacă niciuna dintre afirmațiile de mai sus nu este adevărată.

// declararea si initializarea obiectului final al clasei other by

return this.isbn == other.isbn;

}

@Override

// Metoda pentru a obține o reprezentare sub formă de string a obiectului

// utilizat de obicei în dezvoltare și testare prin imprimarea informațiilor pe consolă

public String toString() {

return "Library.Book{" + "isbn=" + isbn + ", title=" + title + ", author=" + author + ", genre=" + genre + ", dateOfRelease=" + dateOfRelease + ", lang=" + lang + ", numberOfPages=" + numberOfPages + ", price=" + price + ", avalabil=" + avalabil + '}';

}

}

//Inchiderea clasei Library.Book

Cod Java – Clasa Library

/\*\*

\* Clasa Library, descrie caracteristicile:

\* numărul de identificare al bibliotecii,

\* adresa, numărul de telefon, raionul în care se află biblioteca,

\* lista de cărți pe care o conține biblioteca.

\*/

package Library;

import java.util.ArrayList;

/\*\*

\* @author Bazaochi Dumitru, AW21M

\*/

// Clasa Library

public class Library {

private int idLibrary;

private String adres;

private String telephone;

private int numberOfBooks;

private ArrayList<Integer> books;

private String district;

// Constructor gol al clasei

public Library() {

}

// Constructor al clasei cu parametri

public Library(int idLibrary, String adres, String telephone, int numberOfBooks, ArrayList<Integer> books, String district) {

this.idLibrary = idLibrary; // Accesarea variabilei id bibliotecă folosind this

this.adres = adres;

this.telephone = telephone;

this.numberOfBooks = numberOfBooks;

this.books = books;

this.district = district;

}

// Metodele getter și setter

public String getDistrict() {

return district;

}

public void setDistrict(String district) {

this.district = district;

}

public int getIdLibrary() {

return idLibrary;

}

public void setIdLibrary(int idLibrary) {

this.idLibrary = idLibrary;

}

public String getAdres() {

return adres;

}

public void setAdres(String adres) {

this.adres = adres;

}

public String getTelephone() {

return telephone;

}

public void setTelephone(String telephone) {

this.telephone = telephone;

}

public int getNumberOfBooks() {

return numberOfBooks;

}

public void setNumberOfBooks(int numberOfBooks) {

this.numberOfBooks = numberOfBooks;

}

public ArrayList<Integer> getBooks() {

return books;

}

public void setBooks(ArrayList<Integer> books) {

this.books = books;

}

// Suprascriere a metodei hashCode

@Override

// Metodă pentru obținerea codului hash al obiectului

public int hashCode() {

int hash = 7; // Declarare și inițialize

hash = 79 \* hash + this.idLibrary; // Declarare și inițializare variabilă hashcode

return hash; // Returnează valoarea codului hash

}

// Suprascriere a metodei equals

@Override

// Metodă pentru compararea a două obiecte ale clasei pentru identitate

public boolean equals(Object obj) {

if (this == obj) { // Verifică dacă obiectul este identic cu el însuși.

return true; // Returnează adevărat dacă afirmația din if este adevărată

}

if (obj == null) { // Verifică dacă obiectul are valoarea null

return false; // Returnează fals dacă afirmația din if este adevărată

}

if (getClass() != obj.getClass()) { // Verifică dacă obiectul nu este o instanță a acestei clase

return false; // Returnează fals dacă afirmația din if este adevărată

}

final Library other = (Library) obj;

// Atribuirea obiectului obj cu conversie explicită la un obiect din această clasă

// Returnează adevărat dacă niciuna dintre afirmațiile de mai sus nu este adevărată

// Declarare și inițializare a unui obiect final other

return this.idLibrary == other.idLibrary;

}

// Suprascriere a metodei toString

@Override

// Metodă pentru acces rapid la valorile obiectelor clasei;

// de obicei utilizată în dezvoltare și testare prin afișarea informațiilor în consolă

public String toString() {

return "Library{" + "idLibrary=" + idLibrary + ", adres=" + adres + ", telephone=" +

telephone + ", numberOfBooks=" + numberOfBooks + ", books=" + books +

", district=" + district + '}';

}

}

//Inchiderea clasei Library

Cod Java – Clasa AdvancedLibrary

/\*\*

\* Clasa AdvancedLibrary, extinde clasa Library

\* descrie caracteristicile suplimentare

\* numărul etajelor din bibliotecă și numele bibliotecii

\*/

package Library;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* @author Bazaochi Dumitru, AW21M

\*/

public class AdvancedLibrary extends Library { //extinde clasa Library

private int id;

private int numberOfFloors;

private String libraryName;

//constructorul gol al clasei

public AdvancedLibrary() {

}

//constructor al clasei cu parametri

public AdvancedLibrary(int id, int numberOfFloors, String libraryName) {

this.id = id; //accesarea variabilei clasei id a bibliotecii prin this

this.numberOfFloors = numberOfFloors;

this.libraryName = libraryName;

}

// Metodele getter și setter

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getLibraryName() {

return libraryName;

}

public void setLibraryName(String libraryName) {

this.libraryName = libraryName;

}

public int getNumberOfFloors() {

return numberOfFloors;

}

public void setNumberOfFloors(int numberOfFloors) {

this.numberOfFloors = numberOfFloors;

}

// Anotație pentru suprascrierea metodei

@Override

// Metoda pentru acces rapid la valorile obiectelor clasei;

public String toString() {

return "AdvancedLibrary{" + "id=" + id + ", numberOfFloors=" + numberOfFloors +

", libraryName=" + libraryName + '}';

}

} //se închide clasa AdvancedLibrary

Cod Java – Clasa ToCSV

/\*\*

\* Clasa adițională care conține metode pentru export și import de informații în fișiere text.

\*/

package Library;

import java.time.LocalDate;

import java.util.ArrayList;

/\*\*

\* @author Numele și prenumele studentului

\*/

public class ToCsv {

// constante

private final static String ENTER = " "; // constanta pentru spațiu

private final static String ENDL = "\n"; // constanta pentru trecere la linie nouă

// Metoda pentru transformarea informațiilor despre carte într-un șir cu un parametru de tip Book

public static String toLineBook(Book book) {

StringBuilder strB = new StringBuilder();

// Inițializarea unui obiect din clasa StringBuilder pentru a scrie toate informațiile într-o singură linie

strB.append(book.getIsbn()); // Adaugarea ISBN

strB.append(ENTER); // Separarea informațiilor cu un spațiu

strB.append(book.getTitle());

strB.append(ENTER);

strB.append(book.getAuthor());

strB.append(ENTER);

strB.append(book.getGenre());

strB.append(ENTER);

strB.append(book.getDateOfRelease());

strB.append(ENTER);

strB.append(book.getLang());

strB.append(ENTER);

strB.append(book.getNumberOfPages());

strB.append(ENTER);

strB.append(book.getPrice());

strB.append(ENTER);

strB.append(book.getAvalabil());

strB.append(ENDL);

return strB.toString(); // Returnarea șirului rezultat

}

// Metoda pentru transformarea informațiilor despre bibliotecă într-un șir cu un parametru de tip Library

static String toLineLib(Library lib) {

ArrayList<Integer> l = lib.getBooks(); // Scrierea în matrice valoarea Isbn a cărților de bibliotecă

StringBuilder strB = new StringBuilder();

// Inițializarea unui obiect din clasa StringBuilder pentru a scrie toate informațiile într-o singură linie

strB.append(lib.getIdLibrary());

strB.append(ENTER); // Separarea informațiilor cu un spațiu

strB.append(lib.getAdres());

strB.append(ENTER);

strB.append(lib.getTelephone());

strB.append(ENTER);

strB.append(lib.getNumberOfBooks());

strB.append(ENTER);

for (int i = 0; i < l.size(); i++) { // o buclă for pentru a înregistra cărțile isbn ale unei biblioteci după numărul lor

strB.append(l.get(i));

strB.append(ENTER);

}

strB.append(ENTER);

strB.append(lib.getDistrict());

strB.append(ENDL); // Separarea informațiilor prin rupere la o nouă linie

return strB.toString();

} // Inchiderea metodei

// Metoda pentru transformarea informațiilor despre bibliotecă avansată într-un șir cu un parametru de tip AdvancedLibrary

static String toLineAdLib(AdvancedLibrary lib) {

StringBuilder strB = new StringBuilder();

// Inițializarea unui obiect din clasa StringBuilder pentru a scrie toate informațiile într-o singură linie

strB.append(lib.getIdLibrary());

strB.append(ENTER); // Separarea informațiilor cu un spațiu

strB.append(lib.getNumberOfFloors());

strB.append(ENTER);

strB.append(lib.getLibraryName());

strB.append(ENDL); // Separarea informațiilor prin rupere la o nouă linie

return strB.toString();

}

// Metoda pentru transformarea informațiilor despre carte de la client la obiect de tip Book

public static Book toClientBook(String line) {

String data[] = line.split(ENTER); // Crearea unei matrice de șiruri de caractere folosind delimitarea spațiului

int isbn = Integer.parseInt(data[0]);

String title = data[1]; // Crearea unui parametru de titlu de carte din al doilea element al unui tablou de șiruri

String author = data[2]; // Crearea parametrului autor al cărții din al treilea element al matricei de șiruri

String genre = data[3]; // Crearea parametrului gen al cărții din al patrulea element al matricei de șiruri

String dateStr = data[4]; // Crearea parametrului data lansarii al cărții din al cincilea element al matricei de șiruri

LocalDate dateOfRelease = LocalDate.parse(dateStr);

String lang = data[5]; // Crearea parametrului limba al cărții din al șaselea element al matricei de șiruri

int numOfPages = Integer.parseInt(data[6]); // Parametru întreg numărul de pagini de carte

double price = Double.parseDouble(data[7]); // Parametrul de flotare a prețului cărții

String avail = data[8];

// crearea unui obiect din clasa Book cu parametrii dați

Book client = new Book(isbn, title, author, genre, dateOfRelease, lang, numOfPages, price, avail);

return client; // returnarea obiectului

}

// Metoda pentru transformarea informațiilor despre bibliotecă de la client la obiect de tip Library

static Library toClientLib(String line) {

String data[] = line.split(ENTER); // Crearea unei matrice de șiruri de caractere folosind delimitarea spațiului

int id = Integer.parseInt(data[0]);

String adres = data[1];

String telephone = data[2];

int n = Integer.parseInt(data[3]);

ArrayList<Integer> l = new ArrayList<>();

// Crearea unei liste destinate stocării cărților bibliotecii isbn

int i = 4;

for (int j = 0; j < n; i++, j++) {

// o buclă for pentru a scrie cărți de bibliotecă ISbn într-o listă de până la un număr de cărți de bibliotecă

l.add(Integer.parseInt(data[i]));

}

String district = data[i + 1];

// Crearea unui parametru de regiune de bibliotecă din elementul (i+1) --lea al matricei de șiruri

// Crearea unui obiect de clasă Library cu parametrii dați

Library lib = new Library(id, adres, telephone, n, l, district);

return lib; // Returnarea obiectului

}

// Metoda pentru transformarea informațiilor despre bibliotecă avansată de la client la obiect de tip AdvancedLibrary

static AdvancedLibrary toClientAdvLib(String line) {

String data[] = line.split(ENTER);

int id = Integer.parseInt(data[0]);

int floors = Integer.parseInt(data[1]);

String name = data[2];

// Crearea unui obiect de clasă Advanced Library cu parametri specificați

AdvancedLibrary client = new AdvancedLibrary(id, floors, name);

return client; // Returnarea obiectului

} // Inchiderea metodei

} // Inchiderea clasei

Cod Java – Clasa Main

/\*

\* Clasa principală Main, în care se execută funcțiile descrise în sarcină,

\* precum și metodele și funcțiile necesare pentru implementare și utilizare ușoară.

\*/

package Library;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\* @author Bazaochi Dumitru, AW21M

\*/

public class Main {

// Inițializarea logger-ului de erori și excepții pentru a fi afișat în consolă

private static final Logger LOG = Logger.getLogger(Main.class.getName());

static Scanner sc = new Scanner(System.in); // Inițializarea scanner-ului pentru datele introduse de la tastatură

static ArrayList<Library> libList = new ArrayList<>(); // Inițializarea listei de biblioteci

static ArrayList<Book> bookList = new ArrayList<>(); // Inițializarea listei de cărți

static ArrayList<AdvancedLibrary> advList = new ArrayList<>(); // Inițializarea listei de biblioteci avansate

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

// Metoda main, care definește clasa ca "executabilă" pentru compilator

menu(); // Apelarea funcției meniu

}

public static void numberOfLibsThatStartWith45() {

// Metodă pentru determinarea numărului de biblioteci, ale căror numere de telefon încep cu 45

int numOfLibs = 0; // Inițializarea numărului de biblioteci cu 0

for (Library lib1 : libList) {

// Buclă îmbunătățită pentru lista de biblioteci

if (lib1.getTelephone().startsWith("45")) {

// Verificare pentru îndeplinirea condiției

numOfLibs++; // Dacă condiția este adevărată, numărul de biblioteci crește cu 1

}

}

System.out.println("Number of Libraries which telephone number start with 45 = " + numOfLibs);

// Afișarea rezultatului în consolă

}

public static void booksWithMaxPages() {

// Metodă pentru determinarea numărului de cărți cu numărul maxim de pagini

int maxNumOfPages = 0; // Inițializarea variabilei cu numărul maxim de pagini într-o carte

int counter = 0; // Inițializarea contorului de cărți

for (Book book : bookList) {

// Buclă îmbunătățită pentru lista de cărți

if (book.getNumberOfPages() > maxNumOfPages) {

// Dacă numărul de pagini al cărții este mai mare decât valoarea maximă

maxNumOfPages = book.getNumberOfPages();

// Valoarea maximă se schimbă cu numărul de pagini al cărții curente

}

}

System.out.println("The books with maximum number of pages:\n");

// Afișarea informațiilor în consolă

for (Book book : bookList) {

// Buclă îmbunătățită pentru lista de cărți

if (book.getNumberOfPages() == maxNumOfPages) {

// Dacă numărul de pagini al cărții curente este egal cu numărul maxim de pagini

System.out.println(book.toString());

// Informații despre această carte sunt afișate în consolă

counter++; // Contorul crește cu 1

}

}

System.out.println("Number of books with maximum number of pages = " + counter);

// Afișarea informațiilor despre numărul de cărți cu numărul maxim de pagini

}

public static void libsWithFloors() {

// Metodă pentru numărarea bibliotecilor cu un număr de etaje introdus de la tastatură dintr-un anumit interval

int counter = 0; // Inițializarea contorului pentru numărul de biblioteci

System.out.println("Please enter number of floors (1 to 7) in the library :\t");

// Afișarea informației și cererea de a introduce o anumită valoare

int floors = sc.nextInt(); // Inițializarea variabilei pentru numărul de etaje necesar și determinarea valorii acestuia folosind Scanner din consolă

for (AdvancedLibrary lib : advList) {

// Buclă îmbunătățită pentru lista bazată pe clasa Advanced Library

if (lib.getNumberOfFloors() == floors) {

// Dacă numărul de etaje al bibliotecii este egal cu cel dat

counter++; // Contorul crește cu 1

}

}

System.out.println("Number of libraries with " + floors + " floors = " + counter);

// Afișarea informației și rezultatului în consolă

}

public static void insertBooks() {

// Metodă pentru introducerea informațiilor despre cărți în listă

System.out.println("Please enter number of books:\t");

// Sugestie pentru utilizator

int n = sc.nextInt();

// Inițializarea numărului de cărți introduse de la tastatură

System.out.println("Please enter the information about " + n + " books\n");

// Sugestie pentru utilizator

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

// Buclă for

Book b = new Book();

// Inițializarea unui nou obiect de tip carte

System.out.println("Book #" + (i + 1));

// Sugestie pentru utilizator

System.out.println("ISBN:\t");

// Sugestie pentru utilizator

b.setIsbn(sc.nextInt());

// Atribuirea valorii ISBN-ului prin intermediul consolei

System.out.println("Title:\t");

// Sugestie pentru utilizator

b.setTitle(sc.next());

// Atribuirea valorii titlului cărții prin intermediul consolei

System.out.println("Author:\t");

// Sugestie pentru utilizator

b.setAuthor(sc.next());

// Atribuirea valorii autorului prin intermediul consolei

System.out.println("Genre:\t");

// Sugestie pentru utilizator

b.setGenre(sc.next());

// Atribuirea valorii genului prin intermediul consolei

System.out.println("Date of Release yyyy-mm-dd:\t");

// Sugestie pentru utilizator

b.setDateOfRelease(LocalDate.parse(sc.next()));

// Atribuirea valorii datei de lansare prin intermediul consolei

System.out.println("Language:\t");

// Sugestie pentru utilizator

b.setLang(sc.next());

// Atribuirea valorii limbii prin intermediul consolei

System.out.println("Number of Pages:\t");

// Sugestie pentru utilizator

b.setNumberOfPages(sc.nextInt());

// Atribuirea valorii numărului de pagini prin intermediul consolei

System.out.println("Price:\t");

// Sugestie pentru utilizator

b.setPrice(sc.nextDouble());

// Atribuirea valorii prețului prin intermediul consolei

System.out.println("Availability:\t");

// Sugestie pentru utilizator

b.setAvalabil(sc.next());

// Atribuirea valorii disponibilității prin intermediul consolei

bookList.add(b);

// Adăugarea obiectului în listă

}

}

public static void insertLibraries() {

// Metodă pentru introducerea informațiilor despre biblioteci în listă

System.out.println("Please enter number of libraries:\t");

// Sugestie pentru utilizator

int n = sc.nextInt();

// Inițializarea numărului de biblioteci introduse de la tastatură

System.out.println("Please enter the information about " + n + " libraries\n");

// Sugestie pentru utilizator

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

// Buclă for

Library l = new Library();

// Inițializarea unui nou obiect de tip bibliotecă

System.out.println("Library #" + (i + 1));

// Sugestie pentru utilizator

System.out.println("ID:\t");

// Sugestie pentru utilizator

l.setIdLibrary(sc.nextInt());

// Atribuirea valorii ID-ului bibliotecii prin intermediul consolei

System.out.println("Adres:\t");

// Sugestie pentru utilizator

l.setAdres(sc.next());

// Atribuirea valorii adresei prin intermediul consolei

System.out.println("Telephone number:\t");

// Sugestie pentru utilizator

l.setTelephone(sc.next());

// Atribuirea valorii numărului de telefon prin intermediul consolei

System.out.println("Number of books:\t");

// Sugestie pentru utilizator

l.setNumberOfBooks(sc.nextInt());

// Atribuirea valorii numărului de cărți prin intermediul consolei

ArrayList<Integer> lb = new ArrayList<>();

// Inițializarea unui nou obiect de tip listă pentru ISBN-uri

for (int j = 0; j < l.getNumberOfBooks(); j++) {

// Buclă for

System.out.println("ISBN of book #" + (j + 1) + ":\t");

// Sugestie pentru utilizator

lb.add(sc.nextInt());

// Atribuirea valorii ISBN-ului cărții prin intermediul consolei

l.setBooks(lb);

// Adăugarea ISBN-ului cărții în listă

}

System.out.println("District:\t");

// Sugestie pentru utilizator

l.setDistrict(sc.next());

// Atribuirea valorii cartierului prin intermediul consolei

libList.add(l);

// Adăugarea obiectului în listă

System.out.println("\tDo you want to enter and advanced information about this library?");

// Sugestie pentru utilizator

System.out.println("\t'y' - yes");

// Sugestie pentru utilizator

System.out.println("\t'n' - no");

// Sugestie pentru utilizator

System.out.println("\tYour choise:\t");

// Sugestie pentru utilizator

String ch = sc.next();

// Citirea opțiunii de la utilizator

if ("y".equals(ch)) {

// Dacă utilizatorul alege să adauge informații

insertAdvancedLib(l.getIdLibrary());

// Apelarea metodei pentru completarea informațiilor la clasa AdvancedLibrary

exportAdLib();

// Apelarea metodei pentru exportul datelor din lista de tip AdvancedLibrary

}

}

}

private static void insertAdvancedLib(int id) {

// Metodă pentru introducerea informațiilor despre biblioteci cu parametri avansați în listă

AdvancedLibrary al = new AdvancedLibrary();

// Inițializarea unui nou obiect de tip bibliotecă cu parametri avansați

al.setId(id);

// Atribuirea valorii ID-ului din clasa Library

System.out.println("Number of Floors:\t");

// Sugestie pentru utilizator

al.setNumberOfFloors(sc.nextInt());

// Atribuirea valorii numărului de etaje prin intermediul consolei

System.out.println("Name of Library:\t");

// Sugestie pentru utilizator

al.setLibraryName(sc.next());

// Atribuirea valorii numelui bibliotecii prin intermediul consolei

advList.add(al);

// Adăugarea obiectului în listă

}

private static void exportBook() {

// Metodă pentru exportul informațiilor despre cărți într-un fișier text

try (FileWriter fileW = new FileWriter("bookList.csv")) {

// Bloc try-with-resources, se inițializează și se deschide un flux FileWriter cu parametrul fișierului destinație

for (Book user : bookList) {

// Buclă for pentru lista de cărți

String line = ToCsv.toLineBook(user);

// Inițializarea liniei de intrare în fișier cu ajutorul metodei returnate de apelul metodei ToCsv.toLineBook()

fileW.write(line);

// Scrierea liniei în fișier

}

} catch (IOException ex) {

// Bloc catch pentru excepțiile care pot apărea în timpul execuției codului din blocul try, se așteaptă erori de intrare/ieșire

LOG.log(Level.SEVERE, null, ex);

// Afișarea erorii la consolă folosind logger-ul

}

}

private static void exportLib() {

// Metodă pentru exportul informațiilor despre biblioteci într-un fișier text

try (FileWriter fileW = new FileWriter("libraryList.csv")) {

// Bloc try-with-resources, se inițializează și se deschide un flux FileWriter cu parametrul fișierului destinație

for (Library user : libList) {

// Buclă for pentru lista de biblioteci

String line = ToCsv.toLineLib(user);

// Inițializarea liniei de intrare în fișier cu ajutorul metodei returnate de apelul metodei ToCsv.toLineLib()

fileW.write(line);

// Scrierea liniei în fișier

}

} catch (IOException ex) {

// Bloc catch pentru excepțiile care pot apărea în timpul execuției codului din blocul try, se așteaptă erori de intrare/ieșire

LOG.log(Level.SEVERE, null, ex);

// Afișarea erorii la consolă folosind logger-ul

}

}

private static void exportAdLib() {

// Metodă pentru exportul informațiilor despre biblioteci cu parametri avansați într-un fișier text

try (FileWriter fileW = new FileWriter("libraryAdvancedList.csv")) {

// Bloc try-with-resources, se inițializează și se deschide un flux FileWriter cu parametrul fișierului destinație

for (AdvancedLibrary user : advList) {

// Buclă for pentru lista de biblioteci cu parametri avansați

String line = ToCsv.toLineAdLib(user);

// Inițializarea liniei de intrare în fișier cu ajutorul metodei returnate de apelul metodei ToCsv.toLineAdLib()

fileW.write(line);

// Scrierea liniei în fișier

}

} catch (IOException ex) {

// Bloc catch pentru excepțiile care pot apărea în timpul execuției codului din blocul try, se așteaptă erori de intrare/ieșire

LOG.log(Level.SEVERE, null, ex);

// Afișarea erorii la consolă folosind logger-ul

}

}

private static void importBook() {

// Metodă pentru importul informațiilor despre cărți dintr-un fișier text

List<Book> list = new ArrayList();

// Inițializarea unei noi liste de cărți

try (FileReader fileR = new FileReader("bookList.csv")) {

// Bloc try-with-resources, se inițializează și se deschide un flux FileReader cu parametrul fișierului sursă

BufferedReader buffR = new BufferedReader(fileR);

// Inițializarea unui obiect de tip BufferedReader din fluxul FileReader pentru citirea eficientă a fișierului

String line = null;

// Inițializarea liniei de citit

while ((line = buffR.readLine()) != null) {

// Buclă while pentru citirea linie cu linie a fișierului, cât timp există linii

Book cl = ToCsv.toClientBook(line);

// Crearea unui nou obiect de tip carte cu ajutorul metodei returnate de apelul metodei ToCsv.toClientBook()

list.add(cl);

// Adăugarea cărții în lista de cărți

}

} catch (IOException ex) {

// Bloc catch pentru excepțiile care pot apărea în timpul execuției codului din blocul try, se așteaptă erori de intrare/ieșire și lipsa fișierului

LOG.log(Level.SEVERE, null, ex);

// Afișarea erorii la consolă folosind logger-ul

}

bookList = (ArrayList<Book>) list;

// Atribuirea unei noi valori listei de cărți

for (Book b : bookList) {

// Buclă for pentru lista de cărți

System.out.println(b.toString());

// Afișarea informațiilor la consolă folosind metoda toString()

}

}

private static void importLib() {

// Metodă pentru importul informațiilor despre biblioteci dintr-un fișier text

List<Library> list = new ArrayList();

try (FileReader fileR = new FileReader("libraryList.csv")) {

// Bloc try-with-resources, se inițializează și se deschide un flux FileReader cu parametrul fișierului sursă

BufferedReader buffR = new BufferedReader(fileR);

// Inițializarea unui obiect de tip BufferedReader din fluxul FileReader pentru citirea eficientă a fișierului

String line = null; // Inițializarea liniei de citit

while ((line = buffR.readLine()) != null) {

// Buclă while pentru citirea linie cu linie a fișierului, cât timp există linii

Library cl = ToCsv.toClientLib(line);

// Crearea unui nou obiect de tip bibliotecă cu ajutorul metodei returnate de apelul metodei ToCsv.toClientLib()

list.add(cl); // Adăugarea bibliotecii în lista de biblioteci

}

} catch (IOException ex) {

// Bloc catch pentru excepțiile care pot apărea în timpul execuției codului din blocul try, se așteaptă erori de intrare/ieșire și lipsa fișierului

LOG.log(Level.SEVERE, null, ex); // Afișarea erorii la consolă folosind logger-ul

}

libList = (ArrayList<Library>) list; // Atribuirea unei noi valori listei de biblioteci

for (Library l : libList) { // Buclă for pentru lista de biblioteci

System.out.println(l.toString()); // Afișarea informațiilor la consolă folosind metoda toString()

}

}

private static void importAdvLib() {

// Metodă pentru importul informațiilor despre biblioteci cu parametri avansați dintr-un fișier text

List<AdvancedLibrary> list = new ArrayList();

try (FileReader fileR = new FileReader("libraryAdvancedList.csv")) {

// Bloc try-with-resources, se inițializează și se deschide un flux FileReader cu parametrul fișierului sursă

BufferedReader buffR = new BufferedReader(fileR);

// Inițializarea unui obiect de tip BufferedReader din fluxul FileReader pentru citirea eficientă a fișierului

String line = null; // Inițializarea liniei de citit

while ((line = buffR.readLine()) != null) {

// Buclă while pentru citirea linie cu linie a fișierului, cât timp există linii

AdvancedLibrary cl = ToCsv.toClientAdvLib(line);

// Crearea unui nou obiect de tip bibliotecă avansată cu ajutorul metodei returnate de apelul metodei ToCsv.toClientAdvLib()

list.add(cl); // Adăugarea bibliotecii avansate în lista de biblioteci

}

} catch (IOException ex) {

// Bloc catch pentru excepțiile care pot apărea în timpul execuției codului din blocul try, se așteaptă erori de intrare/ieșire și lipsa fișierului

LOG.log(Level.SEVERE, null, ex); // Afișarea erorii la consolă folosind logger-ul

}

advList = (ArrayList<AdvancedLibrary>) list; // Atribuirea unei noi valori listei de biblioteci avansate

for (AdvancedLibrary al : advList) { // Buclă for pentru lista de biblioteci avansate

System.out.println(al.toString()); // Afișarea informațiilor la consolă folosind metoda toString()

}

}

private static void menu() {

// Metodă pentru meniul principal pentru utilizator în consolă

int ch = -1; // Inițializarea variabilei responsabile pentru alegerea utilizatorului

do { // Metodă do-while

System.out.println("\t\t\tTHE MENU"); // Indicație pentru utilizator

System.out.println("\t\tPLEASE CHOOSE ONE OF THE FOLLOWING"); // Indicație pentru utilizator

System.out.println("\tInsert the information (it will be exported automatically) --- 1"); // Indicație pentru utilizator

System.out.println("\tImport information (it will be shown automatically) --- 2"); // Indicație pentru utilizator

System.out.println("\tView the information --- 3"); // Indicație pentru utilizator

System.out.println("\tDo the tasks --- 4"); // Indicație pentru utilizator

System.out.println("\tExit --- 0"); // Indicație pentru utilizator

System.out.print("\tYour choice\t"); // Indicație pentru utilizator

ch = sc.nextInt(); // Citirea valorii variabilei din consolă

choice(ch); // Apelarea metodei de procesare a alegerii utilizatorului

} while (ch != 0); // Ciclul se încheie dacă utilizatorul alege "0"

}

private static void choice(int ch) {

// Metodă pentru procesarea alegerii utilizatorului cu un parametru întreg

switch (ch) { // Instrucțiune switch pentru procesarea diferitelor opțiuni ale utilizatorului în funcție de parametru

case 1: { // În cazul alegerii "1", se va afișa meniul pentru introducerea informațiilor

int c = -1; // Inițializarea variabilei de alegere

do { // Metodă do-while

System.out.println("\t\tINSERT INFORMATION"); // Indicație pentru utilizator

System.out.println("\t\tPLEASE CHOOSE ONE OF THE FOLLOWING"); // Indicație pentru utilizator

System.out.println("\tBooks ---1"); // Indicație pentru utilizator

System.out.println("\tLibraries ---2"); // Indicație pentru utilizator

System.out.println("\tExit ---0"); // Indicație pentru utilizator

System.out.print("\tYour choice\t"); // Indicație pentru utilizator

c = sc.nextInt(); // Citirea valorii variabilei din consolă

insert(c); // Apelarea metodei de procesare a alegerii utilizatorului pentru introducerea informațiilor

} while (c != 0); // Ciclul se încheie dacă utilizatorul alege "0"

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 2: { // În cazul alegerii "2", se va afișa meniul pentru importul și afișarea informațiilor

int c = -1; // Inițializarea variabilei de alegere

do { // Metodă do-while

System.out.println("\t\tIMPORT & SHOW INFORMATION"); // Indicație pentru utilizator

System.out.println("\t\tPLEASE CHOOSE ONE OF THE FOLLOWING"); // Indicație pentru utilizator

System.out.println("\tBooks ---1"); // Indicație pentru utilizator

System.out.println("\tLibraries ---2"); // Indicație pentru utilizator

System.out.println("\tExit ---0"); // Indicație pentru utilizator

System.out.print("\tYour choice\t"); // Indicație pentru utilizator

c = sc.nextInt(); // Citirea valorii variabilei din consolă

importShow(c); // Apelarea metodei de procesare a alegerii utilizatorului pentru importul și afișarea informațiilor

} while (c != 0); // Ciclul se încheie dacă utilizatorul alege "0"

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 3: { // În cazul alegerii "3", se va afișa meniul pentru afișarea informațiilor

int c = -1; // Inițializarea variabilei de alegere

do { // Metodă do-while

System.out.println("\t\tSHOW INFORMATION"); // Indicație pentru utilizator

System.out.println("\t\tPLEASE CHOOSE ONE OF THE FOLLOWING"); // Indicație pentru utilizator

System.out.println("\tBooks ---1"); // Indicație pentru utilizator

System.out.println("\tLibraries ---2"); // Indicație pentru utilizator

System.out.println("\tExit ---0"); // Indicație pentru utilizator

System.out.print("\tYour choice\t"); // Indicație pentru utilizator

c = sc.nextInt(); // Citirea valorii variabilei din consolă

show(c); // Apelarea metodei de procesare a alegerii utilizatorului pentru afișarea informațiilor

} while (c != 0); // Ciclul se încheie dacă utilizatorul alege "0"

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 4: { // În cazul alegerii "4", se vor efectua sarcinile descrise în cerință

numberOfLibsThatStartWith45(); // Apelarea metodei pentru numărul de biblioteci ale căror numere de telefon încep cu 45

booksWithMaxPages(); // Apelarea metodei pentru numărul de cărți cu cele mai multe pagini

libsWithFloors(); // Apelarea metodei pentru numărul de biblioteci cu numărul de etaje specificat de utilizator

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

}

}

private static void insert(int c) {

// Metodă pentru procesarea alegerii utilizatorului pentru introducerea informațiilor

switch (c) {

// Instrucțiune switch pentru procesarea diferitelor opțiuni ale utilizatorului în funcție de parametru pentru introducerea informațiilor

case 1: {

insertBooks(); // Apelarea metodei pentru introducerea listei de cărți din fișier

exportBook(); // Apelarea metodei pentru exportul listei de cărți din fișier

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 2: {

insertLibraries(); // Apelarea metodei pentru introducerea listei de biblioteci din fișier

exportLib(); // Apelarea metodei pentru exportul listei de biblioteci din fișier

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 0: {

menu(); // Apelarea metodei pentru afișarea meniului principal

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

}

}

private static void importShow(int c) {

// Metodă pentru procesarea alegerii utilizatorului pentru importul și afișarea informațiilor

switch (c) {

// Instrucțiune switch pentru procesarea diferitelor opțiuni ale utilizatorului în funcție de parametru pentru importul și afișarea informațiilor

case 1: {

importBook(); // Apelarea metodei pentru importul listei de cărți din fișier

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 2: {

importLib(); // Apelarea metodei pentru importul listei de biblioteci din fișier

importAdvLib(); // Apelarea metodei pentru importul listei de biblioteci avansate din fișier

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 0: {

menu(); // Apelarea metodei pentru afișarea meniului principal

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

}

}

private static void show(int c) {

// Metodă pentru procesarea alegerii utilizatorului pentru afișarea informațiilor pe consolă

switch (c) {

// Instrucțiune switch pentru procesarea diferitelor opțiuni ale utilizatorului în funcție de parametru pentru afișarea informațiilor pe consolă

case 1: {

for (Book b : bookList) { // It. for pentru parcurgerea listei de cărți

System.out.println("\tBook #");

System.out.println("Isbn\t" + b.getIsbn());

System.out.println("Title\t" + b.getTitle());

System.out.println("Author\t" + b.getAuthor());

System.out.println("Genre\t" + b.getGenre());

System.out.println("Release Date\t" + b.getDateOfRelease());

System.out.println("Language\t" + b.getLang());

System.out.println("Pages\t" + b.getNumberOfPages());

System.out.println("Price\t" + b.getPrice());

System.out.println("Availability\t" + b.getAvalabil());

}

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 2: {

for (Library l : libList) { // It. for pentru parcurgerea listei de biblioteci

System.out.println("\tLibrary");

System.out.println("ID\t" + l.getIdLibrary());

System.out.println("Address\t" + l.getAdres());

System.out.println("Telephone number\t" + l.getTelephone());

System.out.println("Books' IDs\t" + l.getBooks());

System.out.println("District\t" + l.getDistrict());

System.out.println("\tAdvanced");

for (AdvancedLibrary al : advList) {

if (al.getId() == l.getIdLibrary()) {

System.out.println("Number of Floors\t" + al.getNumberOfFloors());

System.out.println("Name of Library\t" + al.getLibraryName());

}

}

}

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

case 0: {

menu(); // Apelarea metodei pentru afișarea meniului principal

break; // Se va executa doar acest caz, celelalte vor fi ignorate

}

}

}

}

Concluzii

În concluzie, această lucrare de laborator a avut ca scop înțelegerea conceptelor esențiale ale programării orientate pe obiecte, cum ar fi moștenirea, crearea și manipularea obiectelor, citirea și scrierea în fișiere. A fost o experiență practică valoroasă în dezvoltarea unui proiect Java orientat pe gestionarea datelor colecțiilor de cărți și biblioteci. Am înțeles mai bine conceptele OOP, dezvoltarea unui proiect in OOP, manipularea datelor și a fișierelor.

Referințe bibliografice

1. *Java Tutorial* [online] [vizitat 14.10.2023]. Disponibil: <https://www.w3schools.com/java/>
2. *Concepte Programare Orientata de obiect* [online] [vizitat 14.10.2023]. Disponibil: <https://medium.com/@lucian.ritan/programare-orientat%C4%83-pe-obiect-java-83dbfd222c63>
3. *Introduction to Javadoc* [online] [vizitat 14.10.2023]. Disponibil: <https://www.baeldung.com/javadoc>