Sistem de Management al unei Biblioteci

Proiect realizat pentru

Universitatea Transilvania din Brașov

Facultatea de Matematică și Informatică

Autor:

Vicaș Antonio-Samir

Anul de studiu: II

Specializarea: Informatică aplicată

Grupa: 10LF333

Email: antoniox2004.av@gmail.com

Data: Decembrie 2024

Rezumat

Proiectul Library Management System este o aplicație dezvoltată în Java care permite gestionarea articolelor unei biblioteci, cum ar fi cărți, reviste și benzi desenate. Aplicația oferă funcționalități pentru adăugarea, editarea, listarea și ștergerea articolelor, utilizând o interfață simplă bazată pe linia de comandă (CLI). Datele sunt stocate și gestionate în fișiere JSON, asigurând persistența între sesiunile de utilizare.

Acest proiect ilustrează utilizarea conceptelor de programare orientată pe obiecte, precum moștenirea și polimorfismul, împreună cu manipularea datelor în format JSON. Proiectul este structurat modular, facilitând extinderea și întreținerea ulterioară.

Cuvinte-cheie: Java, JSON, Programare Orientată pe Obiecte, Sistem de Gestionare a Bibliotecilor, CLI.

Cuprins

Listă de figuri

1 Introducere

Proiectul Library Management System este o aplicație dezvoltată în Java care oferă funcționalități pentru gestionarea articolelor dintr-o bibliotecă. Aplicația permite utilizatorilor să creeze, să editeze, să listeze și să gestioneze articole precum cărți, reviste și benzi desenate. Sistemul utilizează fișiere JSON pentru stocarea și încărcarea datelor, asigurând o interfață simplă și eficientă în linia de comandă (CLI).

1.1 Scopul proiectului

Acest proiect este conceput pentru:

- A oferi o soluție de gestionare a datelor pentru o bibliotecă simplă.
- A demonstra utilizarea conceptelor de programare orientată pe obiecte, precum moștenirea, polimorfismul și utilizarea interfețelor.
- A explora stocarea și manipularea datelor în format JSON.

Funcționalități principale:

1. **Adăugare de articole:** Utilizatorii pot adăuga cărți, reviste sau benzi desenate cu detalii specifice.
2. **Editarea articolelor:** Permite modificarea detaliilor articolelor existente.
3. **Listarea articolelor:** Afișează toate articolele din bibliotecă în format text.
4. **Salvarea și încărcarea datelor:** Suport pentru stocarea datelor în fișiere JSON și încărcarea lor ulterioară.
5. **Gestionarea fișierelor:** Posibilitatea de a crea și șterge fișiere JSON.

Beneficii:

- **Simplicitate:** Aplicația oferă o interfață intuitivă, bazată pe linia de comandă.
- **Extensibilitate:** Structura modulară permite extinderea funcționalităților prin adăugarea de noi tipuri de articole.
- **Persistență a datelor:** Stocarea articolelor în fișiere JSON asigură disponibilitatea datelor între sesiunile aplicației.

2 Cerințe

2.1 Hardware

- Procesor: Min. 2 GHz Dual-Core.
- Memorie RAM: Min. 4 GB.
- Spațiu de stocare: Min. x MB.

2.2 Software

- Java Development Kit (JDK): Versiunea 17 sau mai recentă
- IDE recomandat: IntelliJ IDEA / Eclipse / VS Code (cu plugin Java)
- Maven: Pentru gestionarea dependențelor și build-ul proiectului

2.3 Dependete

- Biblioteca JSON: org.json:json (versiunea 20240303)

3 Sistem de operare

Sistem de operare compatibil:

- Windows 10 sau mai recent
- macOS 11 sau mai recent
- Linux (distribuții recente, cu suport Java)

3.1 Environment Configuration

- Asigură-te că variabila de mediu `JAVA_HOME` este configurată corect.
- Adaugă directorul `bin` al JDK în `PATH`.

4 Design

4.1 Interfaces

- Storable: Interfață pentru obiectele care pot fi salvate și încărcate din fișiere JSON.

4.2 Modules

- Item: Clasă abstractă pentru articolele din bibliotecă.
- Book: Clasă pentru cărți.
- Magazine: Clasă pentru reviste.
- Comic: Clasă pentru benzi desenate.
- LibraryManager: Gestioneaza operatiile asupra bibliotecii.

4.3 Main Application

- LibraryApp: Clasa principală a aplicației.

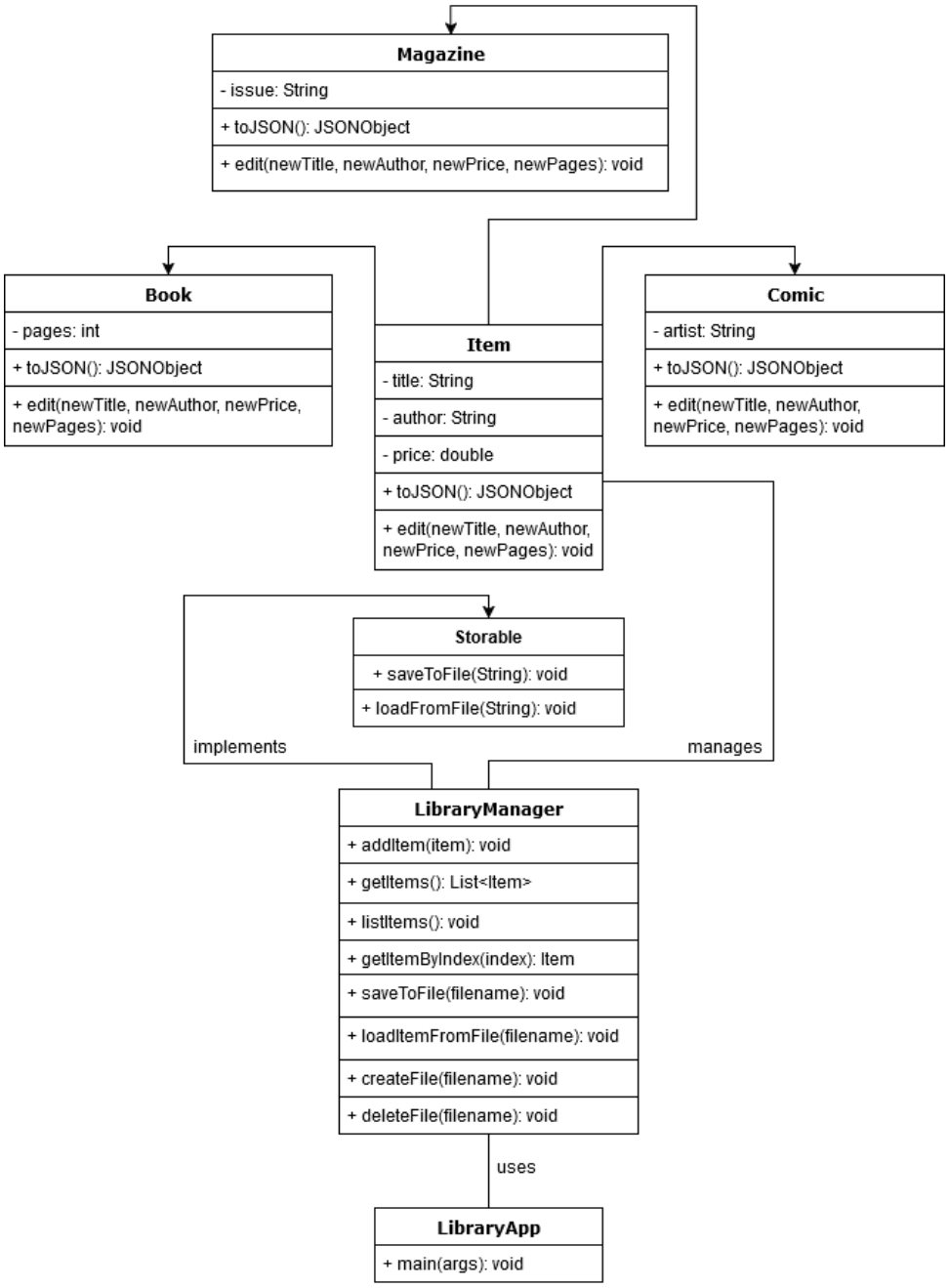

Figura 1: Diagrama UML a proiectului

5 Implementare. Descrierea detaliată a claselor

5.1 Clasa Item

- Este o clasă abstractă care definește structura de bază pentru articolele din bibliotecă.
- Atribute:
  - `title` (String): Titlul articolului.
  - `author` (String): Autorul articolului.
  - `price` (double): Prețul articolului.
- Metode:
  - `Item(String title, String author, double price)`: Constructor pentru inițializarea atributelor de bază.
  - `abstract JSONObject toJSON()`: Metodă abstractă pentru convertirea obiectului într-un format JSON.
  - `void edit(String newTitle, String newAuthor, double newPrice)`: Permite modificarea detaliilor comune.
  - `String toString()`: Returnează o reprezentare textuală a articolului.

5.2 Clasa Book

- Extinde clasa `Item`.
- Atribute suplimentare:
  - `pages` (int): Numărul de pagini al cărții.
- Metode:
  - `Book(String title, String author, double price, int pages)`: Constructor.
  - `JSONObject toJSON()`: Convertește obiectul într-un format JSON.
  - `void edit(String newTitle, String newAuthor, double newPrice, int newPages)`: Permite modificarea detaliilor cărții.

5.3 Clasa Magazine

- Extinde clasa `Item`.
- Atribute suplimentare:
  - `issue` (String): Ediția revistei.
- Metode:
  - `Magazine(String title, String author, double price, String issue)`: Constructor.

- `JSONObject toJSON()`: Convertește obiectul într-un format JSON.
- `void edit(String newTitle, String newAuthor, double newPrice, String newIssue)`: Permite modificarea detaliilor revistei.

5.4 Clasa Comic

- Extinde clasa `Item`.
- Atribute suplimentare:
  - `artist` (String): Artistul care a realizat banda desenată.
- Metode:
  - `Comic(String title, String author, double price, String artist)`: Constructor.
  - `JSONObject toJSON()`: Convertește obiectul într-un format JSON.
  - `void edit(String newTitle, String newAuthor, double newPrice, String newArtist)`: Permite modificarea detaliilor benzii desenate.

5.5 Clasa LibraryManager

- Implementarea interfeței `Storable`.
- Responsabilă cu gestionarea articolelor din bibliotecă și interacțiunea cu fișierele JSON.
- Atribute:
  - `items` (List¡Item¿): Lista articolelor gestionate.
- Metode:
  - `void addItem(Item item)`: Adaugă un articol în bibliotecă.
  - `List<Item> getItems()`: Returnează lista articolelor gestionate.
  - `void listItems()`: Listează toate articolele.
  - `Item getItemByIndex(int index)`: Returnează un articol pe baza indexului.
  - `void saveToFile(String filename)`: Salvează articolele în format JSON.
  - `void loadFromFile(String filename)`: Încarcă articolele dintr-un fișier JSON.
  - `void createFile(String filename)`: Creează un fișier JSON gol.
  - `void deleteFile(String filename)`: Șterge un fișier JSON.

5.6 Clasa LibraryApp

- Este clasa principală a aplicației, care implementează interfața în linia de comandă (CLI).
- Metodă principală:
  - `main(String[] args)`: Oferă utilizatorului funcționalități pentru gestionarea bibliotecii printr-un meniu.
- Funcționalități:
  - Adăugare articole: Permite utilizatorului să adauge cărți, reviste și benzi desenate.
  - Editare articole: Permite editarea atributelor unui articol selectat.
  - Salvare și încărcare: Oferă opțiuni pentru salvarea și încărcarea datelor în/din fișiere JSON.
  - Gestionare fișiere: Creează și șterge fișiere JSON.
  - Ieșire: Închide aplicația.

6 Testare

6.1 Testare funcționalități principale

6.1.1 Clasa `LibraryManagerTest`

- **Test: Adăugarea unui articol** (`testAddItem`)
  Testul validează că un articol este adăugat corect în listă:
  - **Așteptat:** Lista să conțină un singur articol după apelul metodei `addItem`.
  - **Rezultat:** Testul a trecut cu succes.
- **Test: Salvarea datelor în fișier** (`testSaveToFile`)
  Testul verifică dacă fișierele sunt generate corect:
  - **Așteptat:** Fișierul să fie creat cu conținut valid JSON.
  - **Rezultat:** Fișierul a fost creat și verificarea a trecut.
- **Test: Încărcarea datelor din fișier** (`testLoadFromFile`)
  Testul asigură că datele sunt încărcate corect dintr-un fișier JSON:
  - **Așteptat:** Lista să conțină exact elementele salvate anterior.
  - **Rezultat:** Testul a trecut, iar datele au fost încărcate corect.
- **Test: Ștergerea unui fișier** (`testDeleteFile`)
  Testul validează funcționalitatea de ștergere a fișierelor:
  - **Așteptat:** Fișierul să nu mai existe după apelul metodei.
  - **Rezultat:** Fișierul a fost șters cu succes.

6.1.2 Clasele specifice articolelor

Testele pentru clasele `Book`, `Magazine` și `Comic` au verificat următoarele:

- **Constructori:** Validarea inițializării atributelor.
- **Metoda `edit`:** Confirmarea actualizării atributelor cu valori noi.
- **Conversia JSON (`toJSON`):** Verificarea structurii JSON generate.
- **Reprezentarea textuală (`toString`):** Confirmarea formatului corect.

6.2 Rezultate

Toate testele au fost rulate cu succes, iar funcționalitățile principale ale aplicației au fost validate.

6.3 Instrumente utilizate

Testele au fost rulate folosind:

- **JUnit 5:** Pentru scrierea și rularea testelor unitare.
- **IntelliJ IDEA:** IDE-ul utilizat pentru integrarea testelor.

7 Argumentarea Laboratoarelor

Proiectul *Sistem de Management al unei Biblioteci* a fost realizat utilizând conceptele și tehnologiile studiate pe parcursul laboratoarelor. Fiecare laborator a contribuit la înțelegerea și implementarea unor funcționalități esențiale.

7.1 Lab 1: Introducere în Java (Output, Tipuri de Valori, Funcții)

Acest laborator a fost fundamental pentru înțelegerea variabilelor de bază utilizate în proiect.

- Tipuri de date precum `String`, `double`, `int` au fost utilizate pentru a stoca atributele fiecărui articol (titlu, autor, preț, etc.).
- Output-ul a fost implementat prin utilizarea metodei `System.out.print()` pentru a afișa datele în interfața liniei de comandă.
- Funcțiile utilizate în proiect au permis structurarea logicii programului prin metode precum `edit()`, `toJSON()` și altele.

7.2 Lab 2: Introducere în Java (Input, For, While, Switch, If)

Acest laborator a fost esențial pentru implementarea interacțiunii utilizatorului cu aplicația:

- **Input și Output**:
  - Utilizarea clasei `Scanner` pentru a prelua date de la utilizator (ex: titlu, autor, preț).
- **While și Switch**:
  - Meniul principal al aplicației este implementat folosind un ciclu `while` combinat cu o structură `switch` pentru gestionarea opțiunilor utilizatorului.
- **If și For**:
  - Structurile `if` au fost folosite pentru validări și luarea deciziilor.
  - Bucla `for` a fost utilizată pentru listarea articolelor din bibliotecă.

7.3 Lab 3: Colecții Java (Array, List, Map)

În acest laborator, am utilizat colecții Java pentru gestionarea articolelor din bibliotecă:

- **Lista `items` (List¡Item¿)**:
  - Toate articolele sunt stocate într-o colecție de tip `List`.
  - Metodele `addItem()` și `getItems()` gestionează această listă.
- **Sortare**:
  - Articolele pot fi sortate după atribute precum titlu sau autor (potențială extindere).

7.4 Lab 4: Clase Java (Clasă cu Atribute și Metode)

Clasa abstractă `Item` și clasele derivate (`Book`, `Comic`, `Magazine`) sunt exemple practice din acest laborator:

- Atributele comune (titlu, autor, preț) au fost definite în clasa `Item`.
- Fiecare clasă derivată extinde funcționalitățile prin atribute specifice (`pages`, `artist`, `issue`).

7.5 Lab 5: Moștenire în Java, Clase Abstracte

- Clasa abstractă `Item` definește comportamentul comun al articolelor.
- Relația dintre clase este evidențiată prin moștenirea implementată: `Book`, `Magazine` și `Comic` extind clasa abstractă.

7.6 Lab 6: Interfețe în Java

Interfața `Storable` definește funcționalitățile de persistență:

- Metodele `saveToFile()` și `loadFromFile()` sunt implementate în `LibraryManager`, asigurând stocarea și încărcarea articolelor.

7.7 Lab 7: Teste pentru Fiecare Metodă

Testele unitare pentru fiecare clasă au fost implementate folosind JUnit 5:

- Exemple: `LibraryManagerTest`, `BookTest`, `MagazineTest`.
- Metodele testate includ `addItem()`, `edit()`, `toJSON()`.

7.8 Lab 8: Persistența Datelor

- Datele articolelor sunt salvate și încărcate în/din fișiere JSON.
- Metodele `saveToFile()` și `loadFromFile()` gestionează acest proces.

7.9 Lab 9: Diagrame UML

Au fost realizate diagrame UML pentru structura claselor și relațiile dintre acestea:

- Diagrama claselor evidențiază moștenirea și implementarea interfeței.
- Diagrame suplimentare detaliază fluxurile funcționale.

7.10 Concluzie

Proiectul a fost implementat cu succes folosind conceptele și tehnologiile prezentate în laboratoare. Fiecare laborator a contribuit esențial la realizarea funcționalităților cerute.

8 Concluzie

Proiectul oferă o soluție simplă pentru gestionarea articolelor de bibliotecă. În viitor, se pot adăuga:

- Interfață grafică.
- Funcționalitati de căutare avansată.
- Integrare cu o bază de date.

9 Erori și Limitări

- **Erori posibile:**
  - Datele incorecte introduse de utilizator pot duce la comportamente neașteptate.
  - Ștergerea fișierelor fără verificare suplimentară poate cauza pierderea datelor.
- **Limitări:**
  - Aplicația nu suportă căutarea avansată sau filtrarea articolelor.
  - Sistemul de persistență este limitat la fișiere JSON; nu există integrare cu baze de date.
  - Nu există suport pentru gestionarea utilizatorilor sau a drepturilor de acces.