Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

ЗВІТ

З ЛАБОРАТОРНОЇ РОБОТИ №3

# ДИСЦИПЛІНА: «КРОС-ПЛАТФОРМНЕ ПРОГРАМУВАННЯ»

Виконав: студентка групи КС22

Мазуренко Анжеліки

Перевірив: Споров Олександр

Євгенович

Харків

2024

**Лабораторна робота No3 Java & XML**

**Git Repository:**

**Основні задання**

Завдання No1

Розробіть програму, що веде облік читачів та книг у бібліотеці. Для цього потрібно створити набір класів, що представляють бібліотеку (Автор, Книга, Книжкова шафа, Читач, Прокат, ...). У кожному класі мають бути конструктор за замовчуванням, геттери та сеттери, перевизначений метод toString(), а також методи, що реалізують базову функціональність.

За допомогою механізму серіалізації збережіть у файлі поточний стан системи та відновіть систему з цього файлу. Реалізуйте три версії програми, кожна з яких реалізує свою власну стратегію серіалізації:

1. всі класи, що входять до системи, є серіалізованими (реалізують інтерфейс java.io.Serializable);

2. серіалізованими (такими, що реалізують інтерфейс java.io.Serializable) є не всі класи системи: класи Автор, Читач, Книга є НЕ серіалізованими;

3. класи, що входять до системи, реалізують інтерфейс java.io.Externalizable.

Перша версія програми має бути реалізована у вигляді консольної програми, у функції main якої показана робота системи: створена «бібліотека» з книгами та списком читачів; видано кілька книг; виведено інформацію про поточний стан бібліотеки; проведена серіалізація та десеріалізація; показаний стан десеріалізованої системи.

***Перша версія програми:***

Усі класи у нас емплементують інтерфейс Serializable.

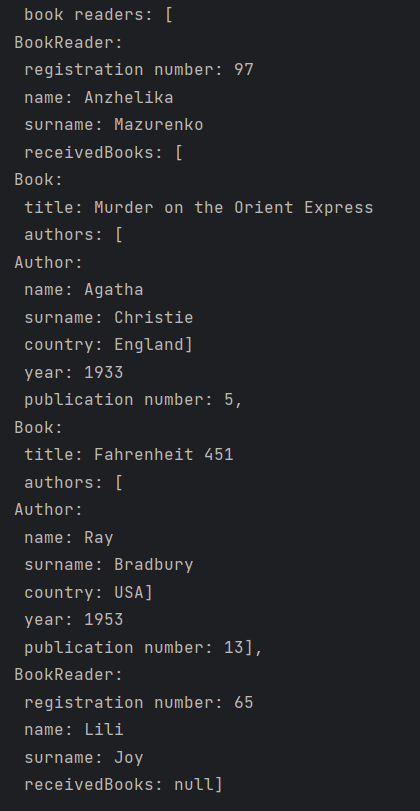
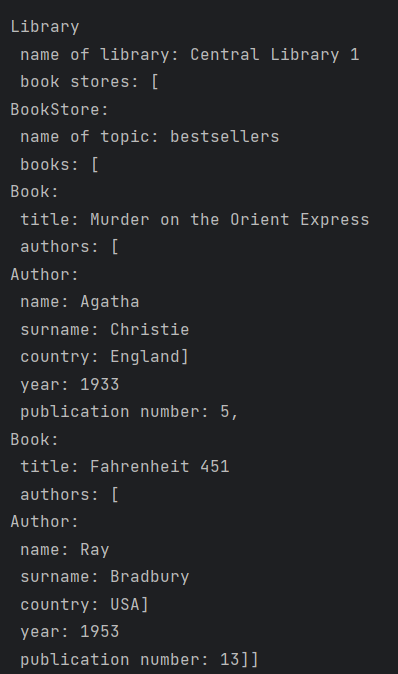
Наприклад:

Лістинг 1. Частина коду

public class Author extends Human implements Serializable {

//code }

У main створюємо та заповнюємо дані про бібліотеку(детальніше можна подивитись у [github](https://github.com/miorezu/LB2_JavaSerialization)). Й викликаємо метод serializeObject, передаючи файл serializationVer1.ser та наш об’єкт бібліотеки. Десереалізуємо наш файл та виводимо інформацію про об’єкт.



Скріншот 1- 2 – результати виконання першої версії програми

*Друга версія програми:*

* класи Автор, Читач, Книга є НЕ серіалізованими.

Так як ці 3 класи не є серіалізованими, поля зв’язані з ними нам потрібно позначити модіфікатором transient.

Наприклад, у класі BookStore у нас є

private transient ArrayList<Book> books;

а так як клас Book не є серіалізованим ми позначаємо поле модифікатором.

У класах BookStore та Library реалізовуємо ручне управління серіалізацією.

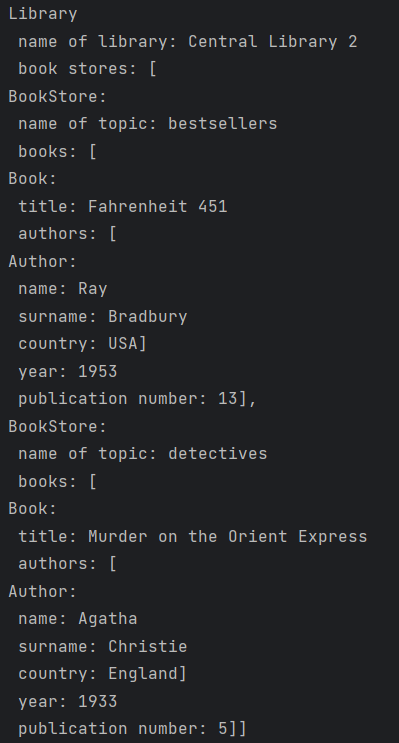
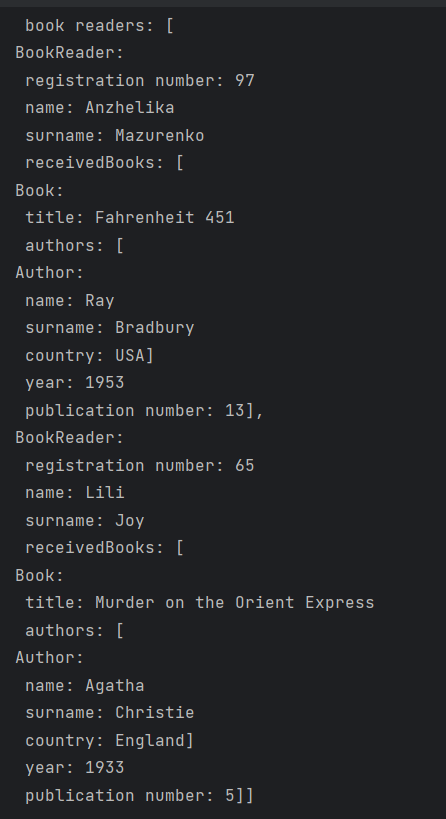
Лістинг 2. Ручне управління серіалізацією та десереалізацією у класі BookStore

@Serial  
private void writeObject(ObjectOutputStream out) throws IOException {  
 out.defaultWriteObject();  
 out.writeInt(books.size());  
 for (Book book : books) {  
 out.writeObject(book.getTitle());  
 out.writeInt(book.getAuthors().size());  
 for (Author a : book.getAuthors()) {  
 out.writeObject(a.getName());  
 out.writeObject(a.getSurname());  
 out.writeObject(a.getCountry());  
 }  
 out.writeInt(book.getYear());  
 out.writeInt(book.getPublicationNumber());  
 }  
}  
  
@Serial  
private void readObject(ObjectInputStream in) throws IOException,  
 ClassNotFoundException {  
 in.defaultReadObject();  
 books = new ArrayList<Book>();  
 int size = in.readInt();  
 for (int i = 0; i < size; i++) {  
 String title = (String) in.readObject();  
 int authorSize = in.readInt();  
 ArrayList<Author> authors = new ArrayList<>();  
 for (int j = 0; j < authorSize; j++) {  
 String name = (String) in.readObject();  
 String surname = (String) in.readObject();  
 String country = (String) in.readObject();  
 Author author = new Author(name, surname, country);  
 authors.add(author);  
 }  
 int year = in.readInt();  
 int publicationNumber = in.readInt();  
 Book book = new Book(title, year, publicationNumber);  
 book.setAuthors(authors);  
 books.add(book);  
 }  
}

Лістинг 3. Ручне управління серіалізацією та десереалізацією у класі Library

@Serial  
private void writeObject(ObjectOutputStream out) throws IOException {  
 out.defaultWriteObject();  
 out.writeInt(bookReaders.size());  
 for (BookReader reader : bookReaders) {  
 out.writeInt(reader.getRegistrationNumber());  
 out.writeObject(reader.getName());  
 out.writeObject(reader.getSurname());  
 out.writeInt(reader.getReceivedBooks().size());  
  
 for (Book book : reader.getReceivedBooks()) {  
 out.writeObject(book.getTitle());  
 out.writeInt(book.getAuthors().size());  
 for (Author a : book.getAuthors()) {  
 out.writeObject(a.getName());  
 out.writeObject(a.getSurname());  
 out.writeObject(a.getCountry());  
 }  
 out.writeInt(book.getYear());  
 out.writeInt(book.getPublicationNumber());  
 }  
 }  
}  
  
@Serial  
private void readObject(ObjectInputStream in) throws IOException,  
 ClassNotFoundException {  
 in.defaultReadObject();  
 bookReaders = new ArrayList<BookReader>();  
 int size = in.readInt();  
 for (int i = 0; i < size; i++) {  
 BookReader bookReader = new BookReader(in.readInt(), (String) in.readObject(), (String) in.readObject());  
 int bookSize = in.readInt();  
 ArrayList<Book> receivedBooks = new ArrayList<>();  
 for (int j = 0; j < bookSize; j++) {  
 String title = (String) in.readObject();  
 int authorSize = in.readInt();  
 ArrayList<Author> authors = new ArrayList<>();  
 for (int k = 0; k < authorSize; k++) {  
 String name = (String) in.readObject();  
 String surname = (String) in.readObject();  
 String country = (String) in.readObject();  
 Author author = new Author(name, surname, country);  
 authors.add(author);  
 }  
 int year = in.readInt();  
 int publicationNumber = in.readInt();  
 Book book = new Book(title, year, publicationNumber);  
 book.setAuthors(authors);  
 receivedBooks.add(book);  
 }  
 bookReader.setReceivedBooks(receivedBooks);  
 bookReaders.add(bookReader);  
 }  
}

У класі Main виконуються ті самі дії, що і у першій версії.

Скріншот 3 – 4 - результати виконання другої версії програми

***Третя версія програми:***

Усі класи у нас емплементують інтерфейс Externalizable.

Наприклад:

public class BookReader extends Human implements Externalizable {

//code }

Й реалізують перевизначенні методи readExternal та writeExternal.

Лістинг 4.Перевизначенні методи readExternal та writeExternal у класі Author

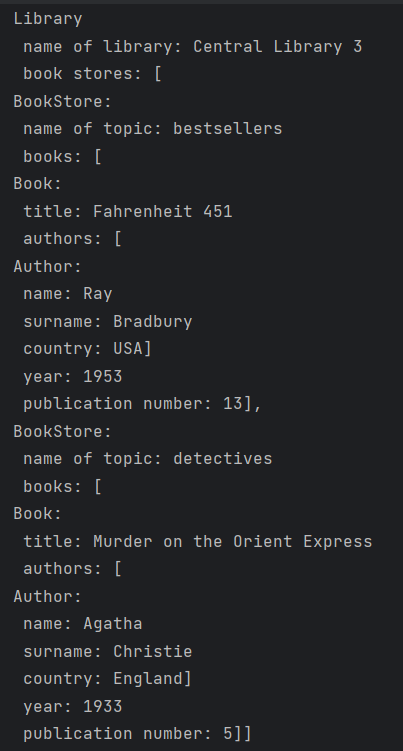
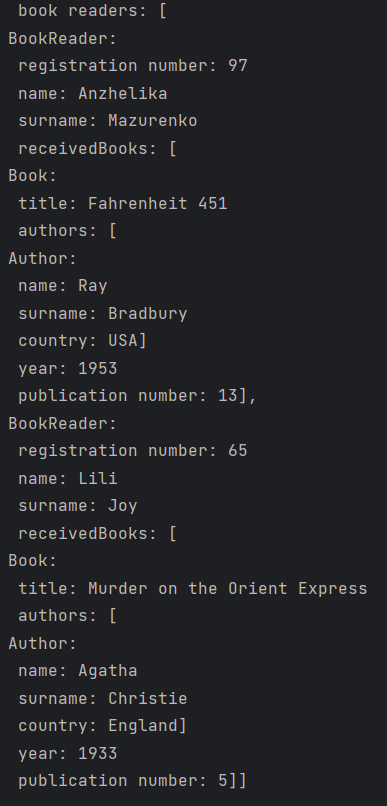
@Override  
public void writeExternal(ObjectOutput out) throws IOException {  
 out.writeObject(getName());  
 out.writeObject(getSurname());  
 out.writeObject(getCountry());  
}  
  
@Override  
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {  
 setName((String) in.readObject());  
 setSurname((String) in.readObject());  
 setCountry((String) in.readObject());  
}

Лістинг 5.Перевизначенні методи readExternal та writeExternal у класі Library

@Override  
public void writeExternal(ObjectOutput out) throws IOException {  
 out.writeObject(getName());  
 out.writeInt(bookStores.size());  
 for (Externalizable bookstore : bookStores) {  
 bookstore.writeExternal(out);  
 }  
 out.writeInt(bookReaders.size());  
 for (Externalizable bookReader : bookReaders) {  
 bookReader.writeExternal(out);  
 }  
}  
  
@Override  
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {  
 setName((String) in.readObject());  
 int bookStoresSize = in.readInt();  
 bookStores = new ArrayList<>();  
 for (int i = 0; i < bookStoresSize; i++) {  
 BookStore bookStore = new BookStore();  
 bookStore.readExternal(in);  
 bookStores.add(bookStore);  
 }  
 int readersSize = in.readInt();  
 bookReaders = new ArrayList<>();  
 for (int i = 0; i < readersSize; i++) {  
 BookReader reader = new BookReader();  
 reader.readExternal(in);  
 bookReaders.add(reader);  
 }  
}

Перевизначенні методи у інших класах можете подивитись на [github](https://github.com/miorezu/LB2_JavaSerialization).

У класі Main виконуються ті самі дії, що і у першій та другій версіях програми.

Скріншот 5-6 - результати виконання третьої версії програми