ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

The Game Library Manager

von

Yahya E. Selo Matrikel Nr. 5013655

Sherwan Diko Matrikel Nr. 5013656

Clarita Kawam Matrikel Nr. 5013484

Bayan Alshami Matrikel Nr. 5011108

Ein Projekt Bericht im Rahmen der Vorlesung

„Programmierung 3 (PIB-PR3)“

an der htw saar im Studiengang Informatik

Saarbrücken, February 3, 2025

Summary

The Game Library Manager is a software application designed for managing personal or shared game collections. It provides features such as adding, editing, and deleting games, tracking progress and statistics, and offering a user-friendly, text-based interface. The application employs a robust three-layer architecture to facilitate maintenance and extensibility. Persistent data storage is implemented using an SQLite database, abstracted with the help of jOOQ[1]. This document outlines the requirements, design, and implementation strategy of the application.

[1]Why You Should Use jOOQ With Code Generation [luk]

Contents

1 Project Overview

- **Project Name:** Game Library Manager
- **Objective:** To provide users with an organized system to track and manage their game collections, progress, and multiplayer statistics.
- **Target Audience:** Gamers and collectors.

2 Requirements

2.1 Must-Have Requirements

1. **Core Functionalities:**
    - Add, edit, and delete games from the library.
    - Search and filter games by title, genre, or platform.
    - Track gameplay progress and multiplayer statistics.
    - Allow users to rate and review games.
    - Provide recommendations based on user preferences and borrowing history.
2. **Persistence:**
    - Store data in a local SQLite database.
    - Use jOOQ to abstract database interactions.
3. **User Interface:**
    - Text-based user interface (CLI or TUI).
    - Platform-independent implementation.
4. **Three-Layer Architecture:**
    - **Presentation Layer:** Handles user interaction.
    - **Logic Layer:** Contains business logic for managing the library.
    - **Persistence Layer:** Manages data storage and retrieval.

2.2 Should-Have Requirements

1. Integration of gamification elements like achievements.
2. Detailed analytics and statistics about game usage.
3. Export/import functionalities for game data.
4. Integration with external APIs (e.g., Steam or Xbox Live) to automatically pull game data.
5. Advanced search functionalities (e.g., search by release date, developer, or rating).

2.3 Can-Have Requirements

1. Optional graphical user interface (GUI).
2. Integration with a game engine (e.g., Unity or Unreal) for virtual game previews.
3. Multiplayer statistics tracking (e.g., win/loss ratios, leaderboards).
4. Cloud-based synchronization for game libraries across multiple devices.

3 Design and Architecture

3.1 Three-Layer Architecture

- **Presentation Layer:** Handles user interaction through a text-based interface (CLI or TUI).
- **Logic Layer:** Contains the business logic for managing the game library, including adding, editing, and deleting games, as well as tracking progress and statistics.
- **Persistence Layer:** Manages data storage and retrieval using a SQLite database and jOOQ for database abstraction.

3.2 Model-View-Controller (MVC) Pattern

- **Model:** Manages the data and business logic of the application (e.g., game details, user ratings).
- **View:** Displays the data to the user (e.g., game list, search results).
- **Controller:** Mediates between the Model and View, processing user input and updating the Model and View accordingly.

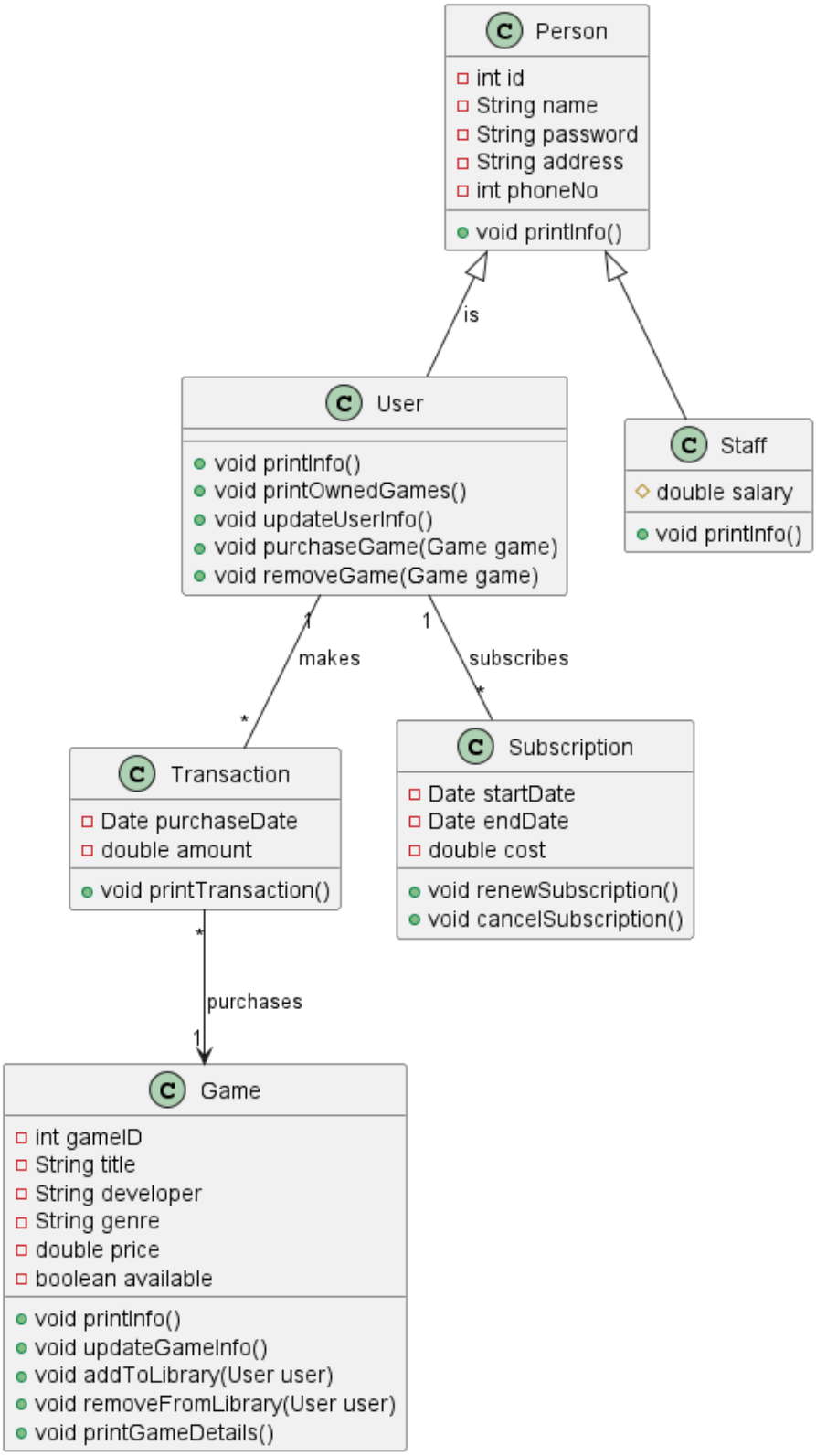

Figure 1: (Class Diagram)

3.3 Database Design

- **Entities:**
  - **Game:** Title, Genre, Platform, Availability, Rating, Review.
  - **User:** Gamer profile, Borrowing history, Preferences.
  - **Loan:** Game borrowed, User, Due date, Return status.
- **Relationships:**
  - A User can borrow multiple Games.
  - A Game can be borrowed by multiple Users.

4 Quick Start Guide

Installation

1. Clone the repository.
2. Install dependencies using Maven [Ap].
3. Install Java 17 or higher. [Ora]
4. Build the project.

4.1 Running the Application

1. Run the application using the command: `java -jar GameLibraryManager.jar`
2. Follow the on-screen instructions to navigate the menu.

Literaturverzeichnis

[Ap] Apache Software Foundation. Apache maven. https://maven.apache.org/.

[luk] lukaseder. Why you should use jooq with code generation. https://blog.jooq.org/why-you-should-use-jooq-with-code-generation/.

[Ora] Oracle. Java se 17 archive downloads. https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html.