Game Library Management System

Course: Data Structures and Object-Oriented Programming

By: Alexander Nikopoulos

Table Of Contents

Project Description

The Game Library Management System (GLMS) is an application whose purpose is to keep a collection of games that can be organized and managed for regular users to view, purchase, and add to their library. Managers can add or remove available games, view all available games, and sort them based on genre or platform. Regular users can browse for the available games to purchase based on how the manager presented them. After purchasing a game, it is added to the user's game library, where they can view all their games, rate them, or remove games. The system keeps separate information for the available games and each user's owned library.

Program Features

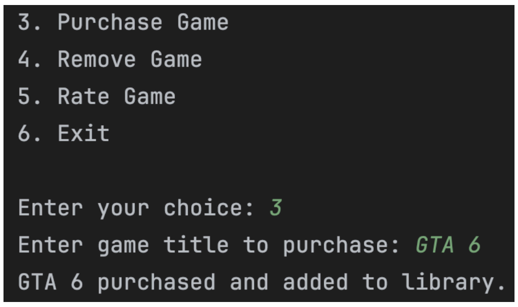
Common appropriate data structures:

- List<OwnedGame> and List<Game> to easily manage games and sorting using Comparators/Comparable.
- Map<String, List<OwnedGame>> to associate each user with their game library.

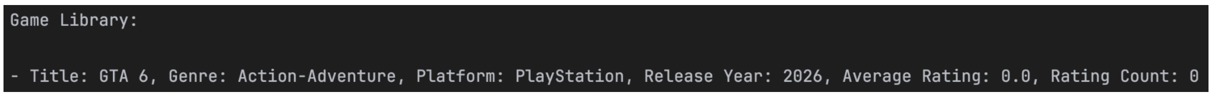
Unit testing: All necessary user-defined methods that needed testing were tested.

```
Hierarchies: User -> Manager              Game -> OwnedGame
                  -> RegularUser
```

Regular User purchasing an available game:

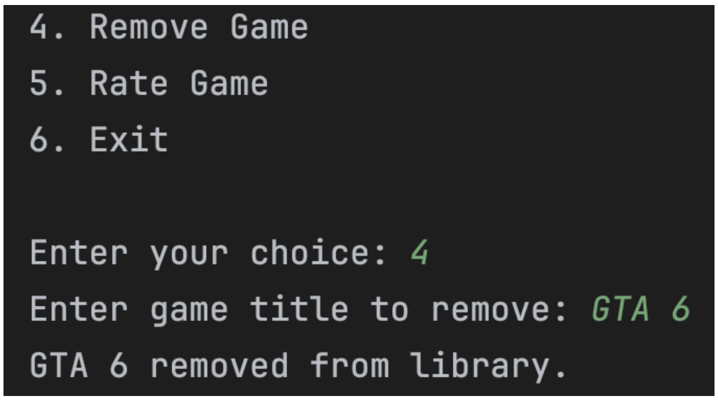


```
3. Purchase Game
4. Remove Game
5. Rate Game
6. Exit

Enter your choice: 3
Enter game title to purchase: GTA 6
GTA 6 purchased and added to library.
```

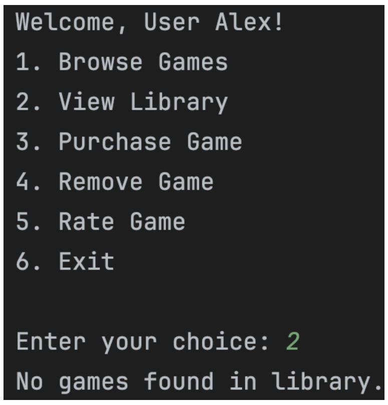


```
Game Library:

- Title: GTA 6, Genre: Action-Adventure, Platform: PlayStation, Release Year: 2026, Average Rating: 0.0, Rating Count: 0
```

Regular User removing a game from their personal library:

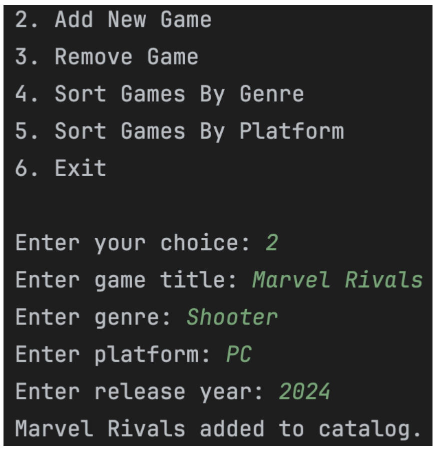


```
4. Remove Game
5. Rate Game
6. Exit

Enter your choice: 4
Enter game title to remove: GTA 6
GTA 6 removed from library.
```

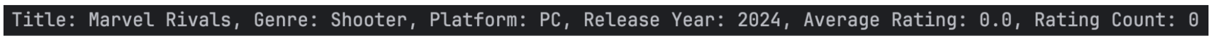


```
Welcome, User Alex!
1. Browse Games
2. View Library
3. Purchase Game
4. Remove Game
5. Rate Game
6. Exit

Enter your choice: 2
No games found in library.
```

Manager adding a game to the catalog:

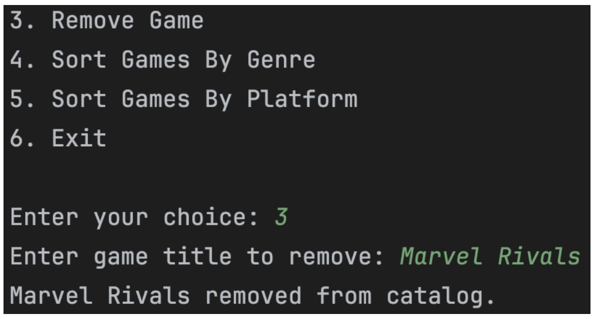


```
2. Add New Game
3. Remove Game
4. Sort Games By Genre
5. Sort Games By Platform
6. Exit

Enter your choice: 2
Enter game title: Marvel Rivals
Enter genre: Shooter
Enter platform: PC
Enter release year: 2024
Marvel Rivals added to catalog.
```

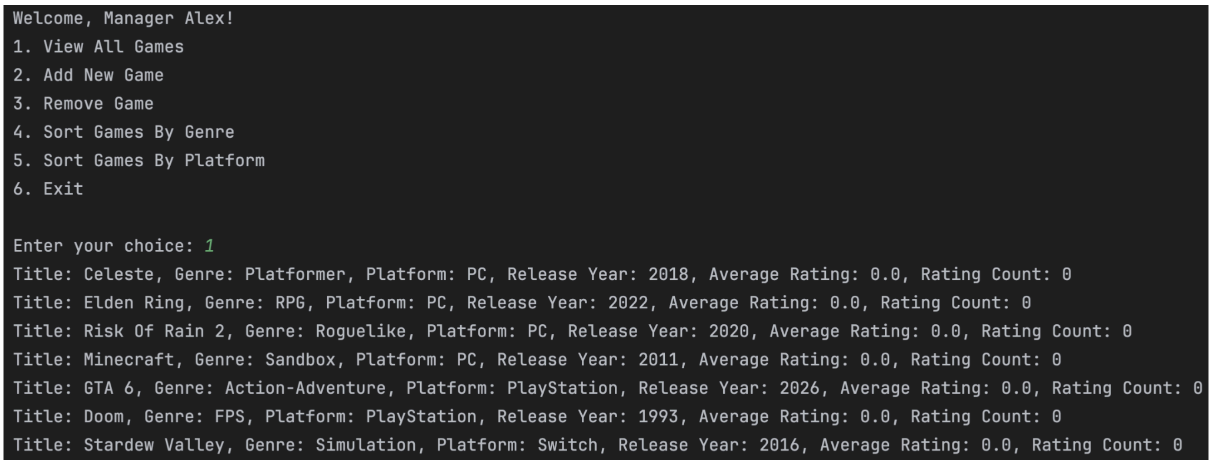


```
Title: Marvel Rivals, Genre: Shooter, Platform: PC, Release Year: 2024, Average Rating: 0.0, Rating Count: 0
```

Manager removing a game from the catalog:



```
3. Remove Game
4. Sort Games By Genre
5. Sort Games By Platform
6. Exit

Enter your choice: 3
Enter game title to remove: Marvel Rivals
Marvel Rivals removed from catalog.
```

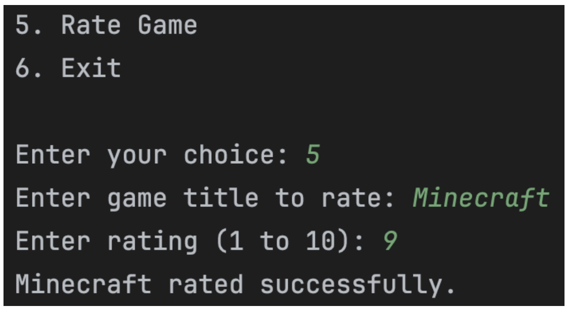


```
Welcome, Manager Alex!
1. View All Games
2. Add New Game
3. Remove Game
4. Sort Games By Genre
5. Sort Games By Platform
6. Exit

Enter your choice: 1
Title: Celeste, Genre: Platformer, Platform: PC, Release Year: 2018, Average Rating: 0.0, Rating Count: 0
Title: Elden Ring, Genre: RPG, Platform: PC, Release Year: 2022, Average Rating: 0.0, Rating Count: 0
Title: Risk Of Rain 2, Genre: Roguelike, Platform: PC, Release Year: 2020, Average Rating: 0.0, Rating Count: 0
Title: Minecraft, Genre: Sandbox, Platform: PC, Release Year: 2011, Average Rating: 0.0, Rating Count: 0
Title: GTA 6, Genre: Action-Adventure, Platform: PlayStation, Release Year: 2026, Average Rating: 0.0, Rating Count: 0
Title: Doom, Genre: FPS, Platform: PlayStation, Release Year: 1993, Average Rating: 0.0, Rating Count: 0
Title: Stardew Valley, Genre: Simulation, Platform: Switch, Release Year: 2016, Average Rating: 0.0, Rating Count: 0
```

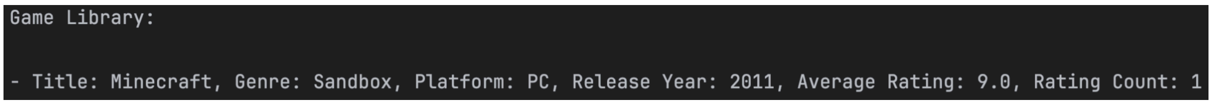
(Note that this is to show that the game is no longer available).

Interface: Rateable used in OwnedGame.

Regular User using rateGame():



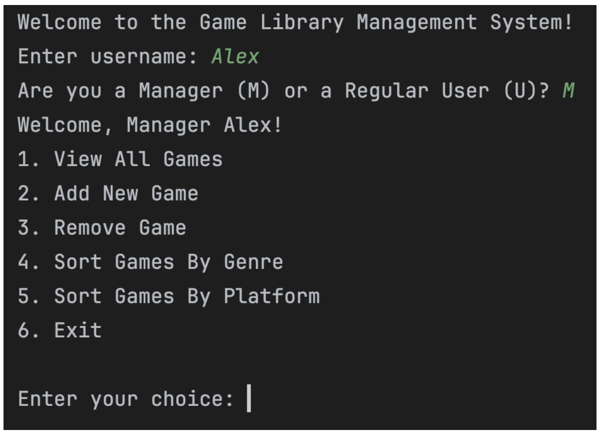
```
5. Rate Game
6. Exit

Enter your choice: 5
Enter game title to rate: Minecraft
Enter rating (1 to 10): 9
Minecraft rated successfully.
```



```
Game Library:

- Title: Minecraft, Genre: Sandbox, Platform: PC, Release Year: 2011, Average Rating: 9.0, Rating Count: 1
```

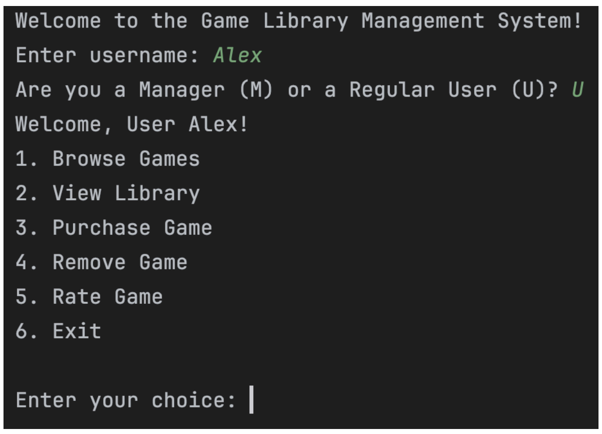
Runtime-polymorphism: displayMenu() method depending on the user.

Manager displayMenu():



```
Welcome to the Game Library Management System!
Enter username: Alex
Are you a Manager (M) or a Regular User (U)? M
Welcome, Manager Alex!
1. View All Games
2. Add New Game
3. Remove Game
4. Sort Games By Genre
5. Sort Games By Platform
6. Exit

Enter your choice: 
```

RegularUser displayMenu():



```
Welcome to the Game Library Management System!
Enter username: Alex
Are you a Manager (M) or a Regular User (U)? U
Welcome, User Alex!
1. Browse Games
2. View Library
3. Purchase Game
4. Remove Game
5. Rate Game
6. Exit

Enter your choice: 
```

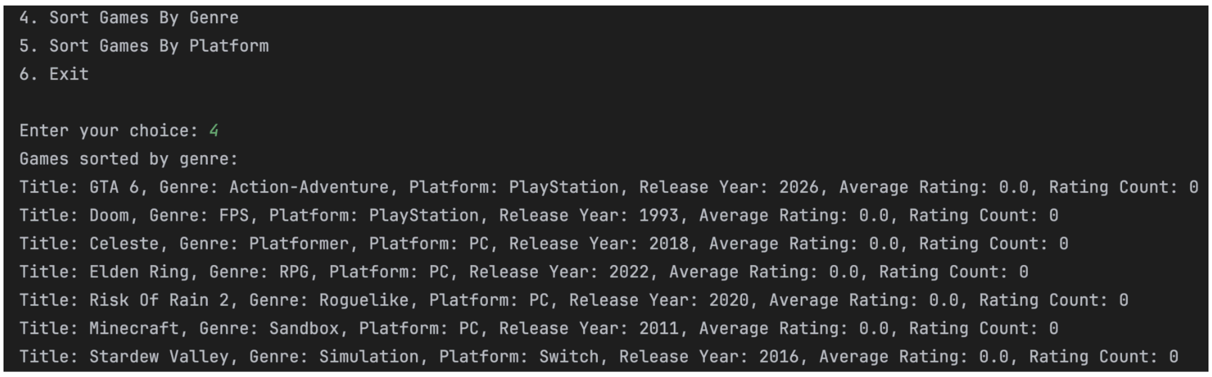
TextIO: GameDataController class to read/write to/from .csv files.

- saveAvailableGames(List<Game> games) and loadAvailableGames() used to keep track of all available games for the Regular User and update it by the Manager in availableGames.csv.
- saveUserLibraries(Map<String, List<OwnedGame>> userLibraries) and loadUserLibraries() used to keep track of each Regular User's own game library in userLibraries.csv.

Comparable: Game class -> sort games by title by default.

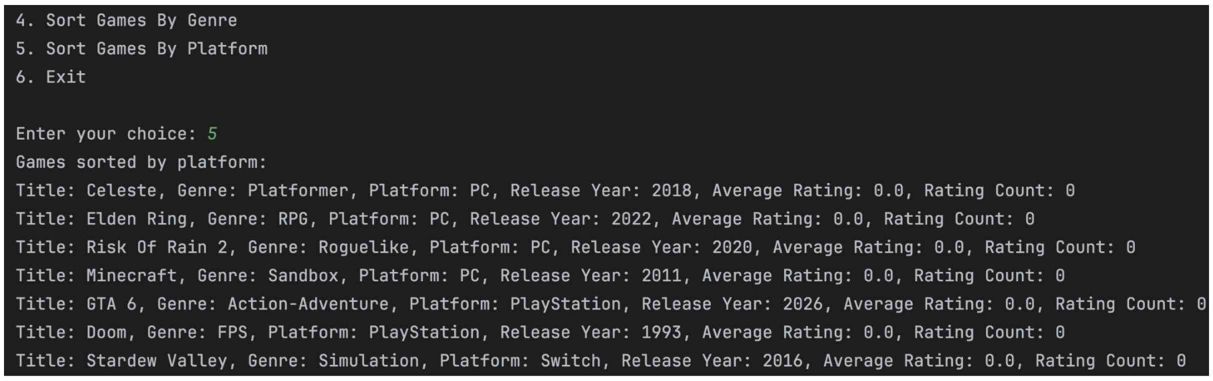
Comparators: Manager class -> sort games by genre/platform.

Manager sorts by genre:



```
4. Sort Games By Genre
5. Sort Games By Platform
6. Exit

Enter your choice: 4
Games sorted by genre:
Title: GTA 6, Genre: Action-Adventure, Platform: PlayStation, Release Year: 2026, Average Rating: 0.0, Rating Count: 0
Title: Doom, Genre: FPS, Platform: PlayStation, Release Year: 1993, Average Rating: 0.0, Rating Count: 0
Title: Celeste, Genre: Platformer, Platform: PC, Release Year: 2018, Average Rating: 0.0, Rating Count: 0
Title: Elden Ring, Genre: RPG, Platform: PC, Release Year: 2022, Average Rating: 0.0, Rating Count: 0
Title: Risk Of Rain 2, Genre: Roguelike, Platform: PC, Release Year: 2020, Average Rating: 0.0, Rating Count: 0
Title: Minecraft, Genre: Sandbox, Platform: PC, Release Year: 2011, Average Rating: 0.0, Rating Count: 0
Title: Stardew Valley, Genre: Simulation, Platform: Switch, Release Year: 2016, Average Rating: 0.0, Rating Count: 0
```

Manager sorts by platform:



```
4. Sort Games By Genre
5. Sort Games By Platform
6. Exit

Enter your choice: 5
Games sorted by platform:
Title: Celeste, Genre: Platformer, Platform: PC, Release Year: 2018, Average Rating: 0.0, Rating Count: 0
Title: Elden Ring, Genre: RPG, Platform: PC, Release Year: 2022, Average Rating: 0.0, Rating Count: 0
Title: Risk Of Rain 2, Genre: Roguelike, Platform: PC, Release Year: 2020, Average Rating: 0.0, Rating Count: 0
Title: Minecraft, Genre: Sandbox, Platform: PC, Release Year: 2011, Average Rating: 0.0, Rating Count: 0
Title: GTA 6, Genre: Action-Adventure, Platform: PlayStation, Release Year: 2026, Average Rating: 0.0, Rating Count: 0
Title: Doom, Genre: FPS, Platform: PlayStation, Release Year: 1993, Average Rating: 0.0, Rating Count: 0
Title: Stardew Valley, Genre: Simulation, Platform: Switch, Release Year: 2016, Average Rating: 0.0, Rating Count: 0
```

Challenges

While developing this project, some minor unimplemented features that were simply left out include the user being able to search for a game based on specific parameters. This was because the manager could already sort the available games in the way they wanted them to be presented. Another feature that was left out was the manager being able to directly edit the game details of an added game. However, this seemed unnecessary as there wasn't anything that needed to be regularly updated by the manager about any game's details once it was added to the system.

An issue that was faced leading to an unimplemented feature was the ability to see the rating of a game while browsing the catalog. The ability of the user to rate a game and have its average rating presented in their library is perfectly fine, but this data isn't stored for it to be saved across each specific game. As a result, the ratings aren't visible in the catalog, only in the user's library.

Learning Outcomes

After the development of this project, I have learned many very important principles. I learned how to properly apply inheritance and polymorphism when working with related classes and methods. I also learned about the importance of JUnit testing when trying to determine if specific methods will function as intended. Most importantly, I learned a lot about how to deal with Text IO when it comes to reading and writing data, and how I can apply it to various other scenarios in the future.