C-DAC Mumbai OOPJ Lab Assignment

Capstone Project – Game Hub; Game Collection Manager

ANSWERS

START - Developer Requirements

1) Project Overview

Project Name: Game Hub **Project Type:** Java Console Application **Duration:** 120 minutes **Objective:** Develop a **menu-driven game collection manager** using **Java OOP concepts** and **Collection Framework**.

2) Functional Requirements

2.1 Game Management

● Add Game

□ Input: Name, Genre, Rating (1-5), Platform (Console/PC)
□ Validation: Rating must be 1-5
□ Action: Add game to allGames list
□ Output: Success message with game ID

● Remove Game

□ Input: Game ID
□ Validation: Cannot remove if game is borrowed
□ Output: Success/Error message

● View Games

□ Options: Sort by ID, Name, Rating
□ Output: List of games with all details (platform-specific info included)

● Search Game

□ Input: Name or Genre
□ Output: Display matching games

2.2 User Management

● Register User

□ Input: Username, Email
□ Validation: Username must be unique
□ Output: Confirmation message

● View Users

□ Output: List of all registered users

2.3 Borrowing System
● Issue Game

□ Input: Game ID, Username
□ Validation: Game must be available, User must exist
□ Action: Move game to borrowedQueue, add game to user's borrowed list
□ Output: Success/Error message

● Return Game

□ Input: Game ID, Username
□ Validation: Only borrower can return the game
□ Action: Move game back to available list
□ Output: Success/Error message

● Borrowed Games Queue

□ Output: Display current borrowed games in queue order

2.4 Reporting & Statistics
● Collection Statistics

□ Total Games, Available Games, Borrowed Games

● User Statistics

□ Total Registered Users

● Borrowed Queue Display

□ Current games on loan

3) Technical Requirements

3.1 OOP Concepts to Implement
● **Classes & Objects:** Game, ConsoleGame, PCGame, User, GameHubManager
● **Encapsulation:** Private fields with public getters/setters
● **Inheritance:** ConsoleGame and PCGame extend Game
● **Polymorphism:** Method overriding (displayDetails())
● **Abstract Classes:** Game as base abstract class
● **Interfaces:** GameActions for managing operations
● **Constructor Chaining:** Default + parameterized constructors
● **Static Variables:** Track totalGames and totalUsers
● **toString() Override:** Clean display for objects

3.2 Collections & Advanced Features
● **ArrayList allGames** – Main game collection
● **LinkedList borrowedQueue** – Queue of borrowed games
● **HashSet users** – Maintain unique users
● **HashMap<Integer, Game> gameMap** – Quick lookup by game ID
● **Comparable Interface** – Sort games by ID
● **Comparator Interface** – Sort games by Name or Rating

3.3 Exception Handling
● Custom Exceptions:

□ GameNotFoundException
□ UserNotFoundException
□ InvalidRatingException

END – Developer Requirements
User Stories & Acceptance Criteria (User POV)
User Story 1 – Add Game
● **As a user**, I want to add a new game to my collection so that I can track it.
● Acceptance Criteria:

□ Game ID auto-generated
□ Rating validated (1–5)
□ Confirmation displayed

User Story 2 – Borrow Game
● **As a user**, I want to borrow a game so that I can play it.
● Acceptance Criteria:

□ Game must exist and be available
□ User must be registered
□ Borrowed queue updates automatically

User Story 3 – Return Game
● **As a user**, I want to return a borrowed game so it becomes available for others.
● Acceptance Criteria:

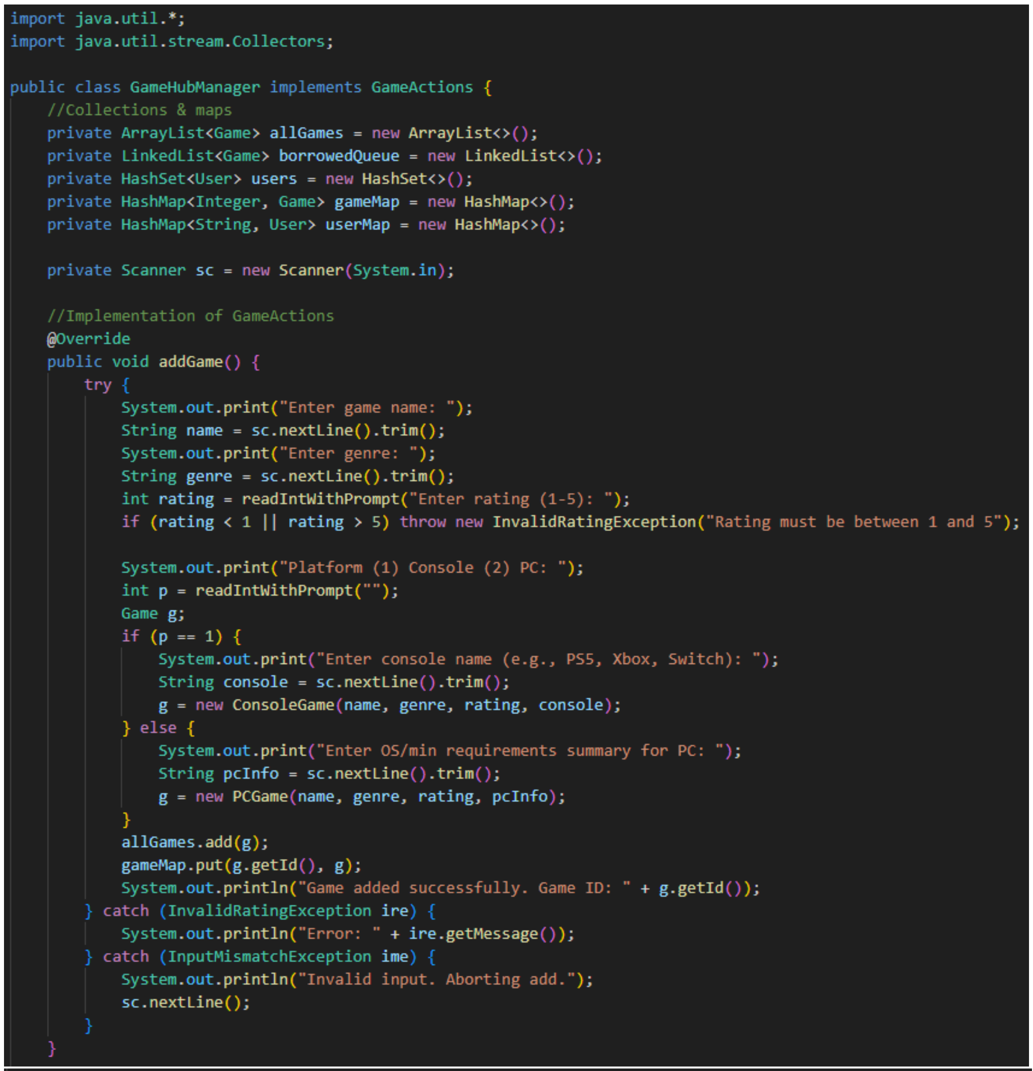
□ Only the original borrower can return
□ Borrowed queue updates automatically

User Story 4 – View & Sort Games
● **As a user**, I want to view my collection sorted by ID, Name, or Rating.
● Acceptance Criteria:

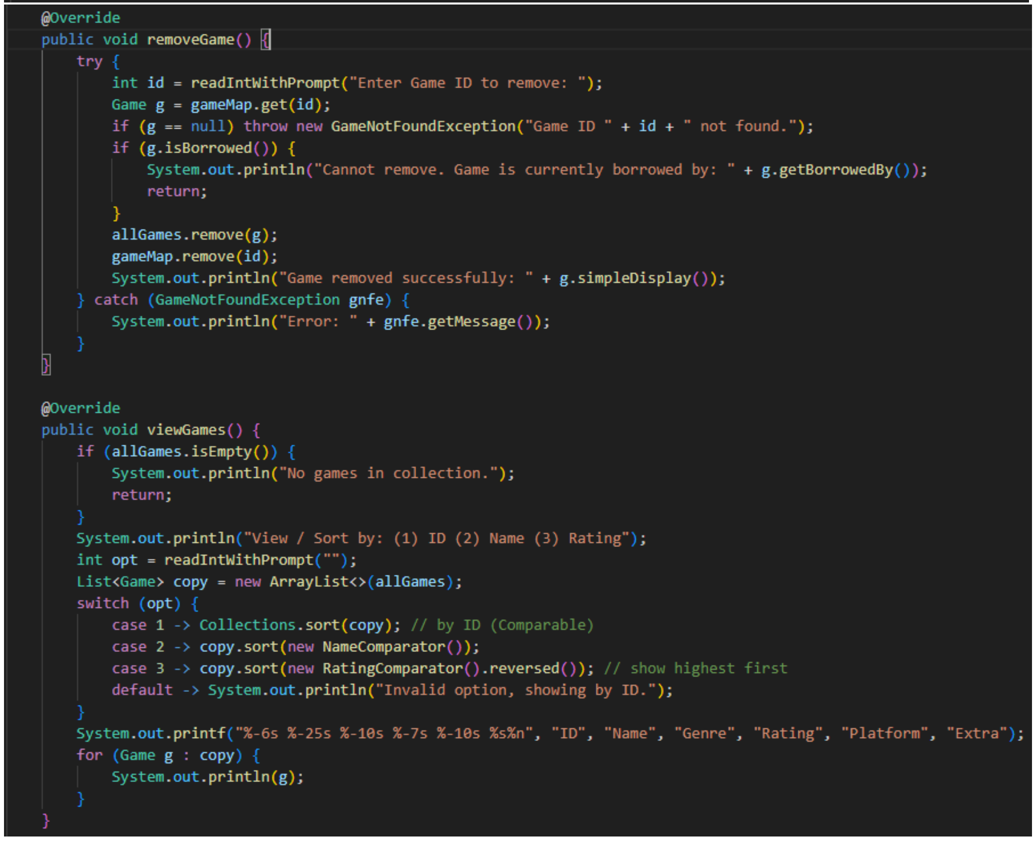
□ Sort options selectable
□ Output includes platform-specific info

User Story 5 – User Management
● **As an admin**, I want to register new users and prevent duplicates.
● Acceptance Criteria:

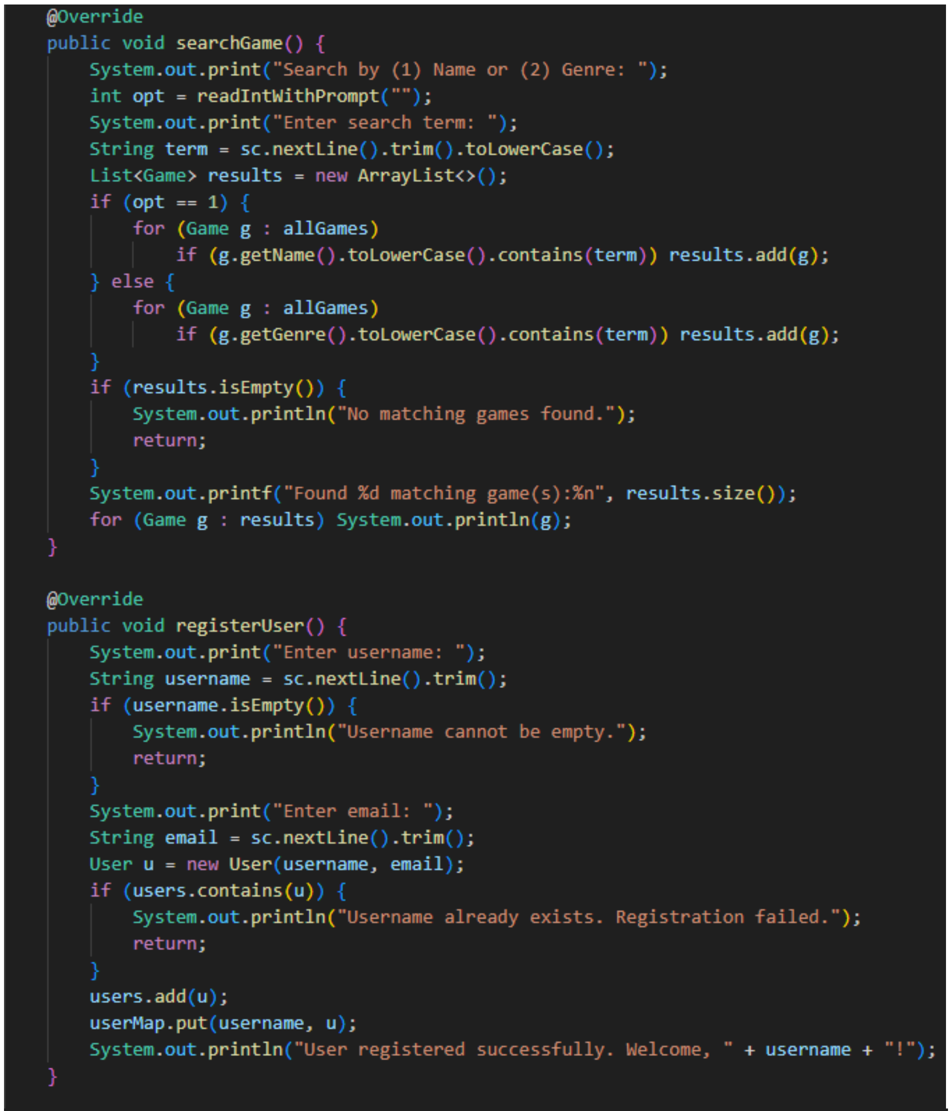
□ Duplicate usernames rejected
□ Success message on registration



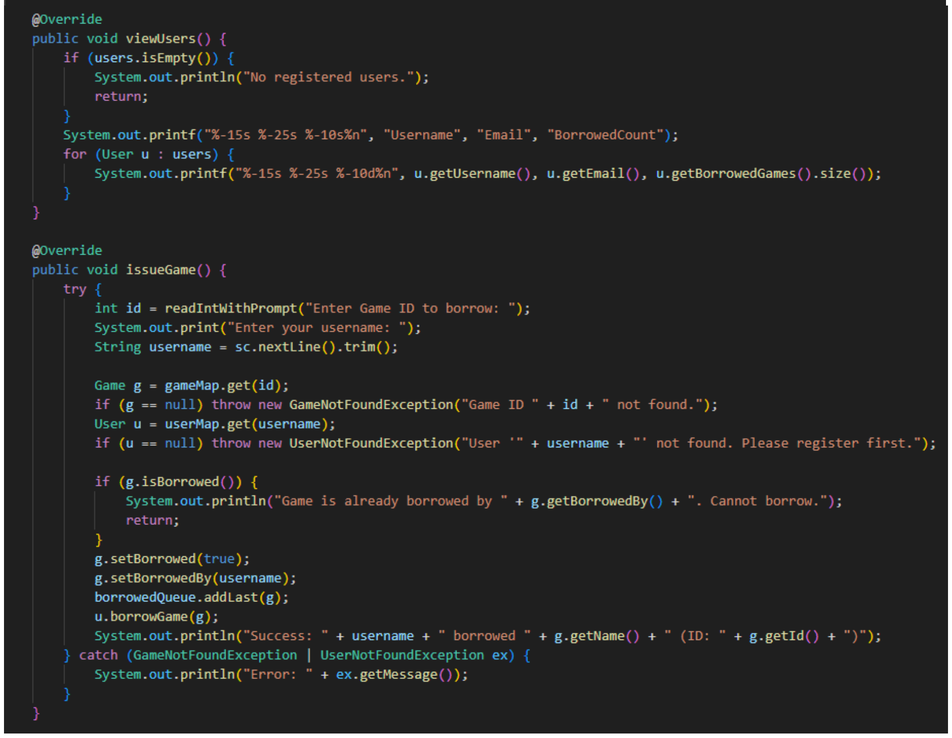
```java
import java.util.*;
import java.util.stream.Collectors;

public class GameHubManager implements GameActions {
    //Collections & maps
    private ArrayList<Game> allGames = new ArrayList<>();
    private LinkedList<Game> borrowedQueue = new LinkedList<>();
    private HashSet<User> users = new HashSet<>();
    private HashMap<Integer, Game> gameMap = new HashMap<>();
    private HashMap<String, User> userMap = new HashMap<>();

    private Scanner sc = new Scanner(System.in);

    //Implementation of GameActions
    @Override
    public void addGame() {
        try {
            System.out.print("Enter game name: ");
            String name = sc.nextLine().trim();
            System.out.print("Enter genre: ");
            String genre = sc.nextLine().trim();
            int rating = readIntWithPrompt("Enter rating (1-5): ");
            if (rating < 1 || rating > 5) throw new InvalidRatingException("Rating must be between 1 and 5");

            System.out.print("Platform (1) Console (2) PC: ");
            int p = readIntWithPrompt("");
            Game g;
            if (p == 1) {
                System.out.print("Enter console name (e.g., PS5, Xbox, Switch): ");
                String console = sc.nextLine().trim();
                g = new ConsoleGame(name, genre, rating, console);
            } else {
                System.out.print("Enter OS/min requirements summary for PC: ");
                String pcInfo = sc.nextLine().trim();
                g = new PCGame(name, genre, rating, pcInfo);
            }
            allGames.add(g);
            gameMap.put(g.getId(), g);
            System.out.println("Game added successfully. Game ID: " + g.getId());
        } catch (InvalidRatingException ire) {
            System.out.println("Error: " + ire.getMessage());
        } catch (InputMismatchException ime) {
            System.out.println("Invalid input. Aborting add.");
            sc.nextLine();
        }
    }
```



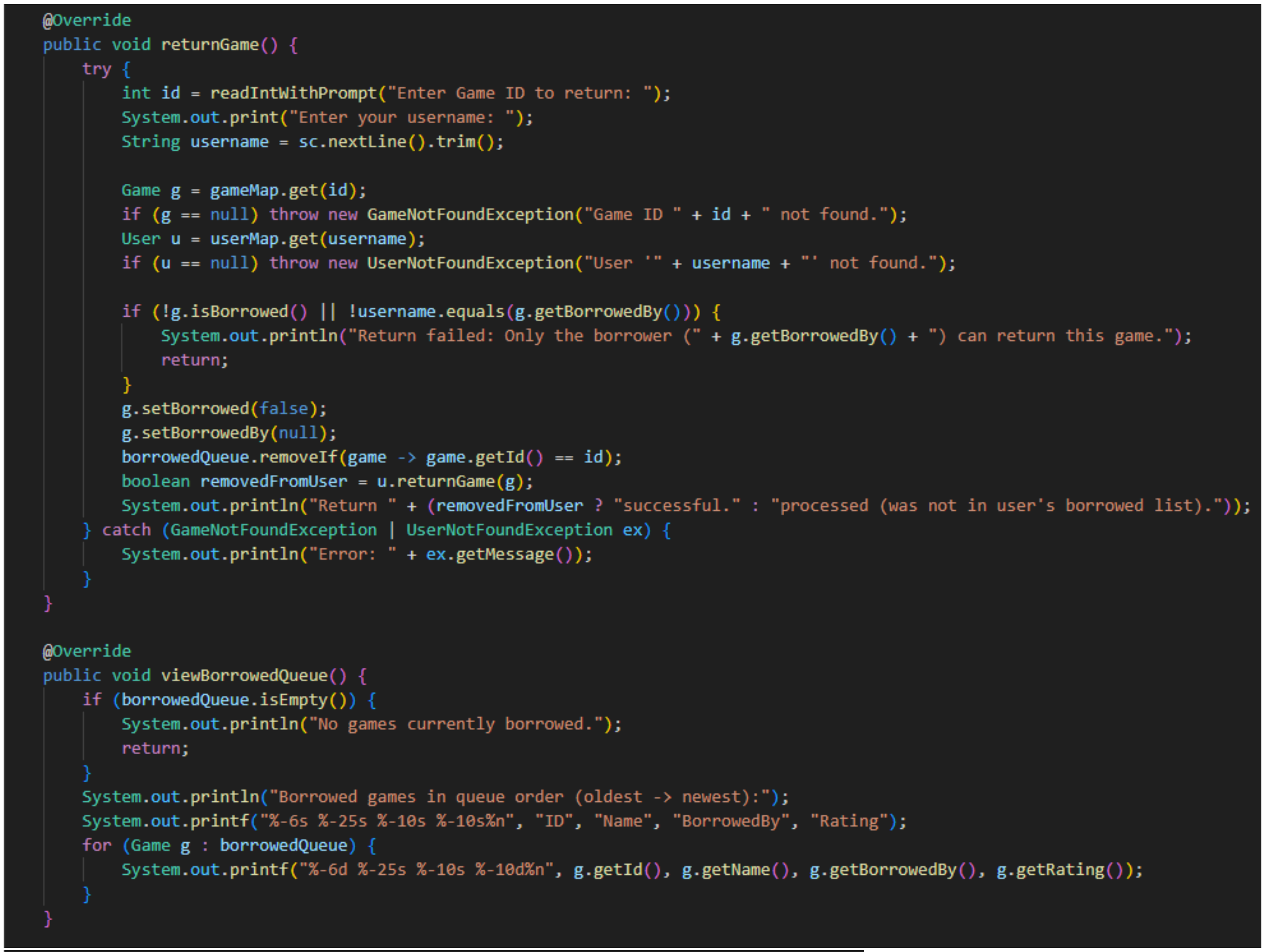
```java
    @Override
    public void removeGame() {
        try {
            int id = readIntWithPrompt("Enter Game ID to remove: ");
            Game g = gameMap.get(id);
            if (g == null) throw new GameNotFoundException("Game ID " + id + " not found.");
            if (g.isBorrowed()) {
                System.out.println("Cannot remove. Game is currently borrowed by: " + g.getBorrowedBy());
                return;
            }
            allGames.remove(g);
            gameMap.remove(id);
            System.out.println("Game removed successfully: " + g.simpleDisplay());
        } catch (GameNotFoundException gnfe) {
            System.out.println("Error: " + gnfe.getMessage());
        }
    }

    @Override
    public void viewGames() {
        if (allGames.isEmpty()) {
            System.out.println("No games in collection.");
            return;
        }
        System.out.println("View / Sort by: (1) ID (2) Name (3) Rating");
        int opt = readIntWithPrompt("");
        List<Game> copy = new ArrayList<>(allGames);
        switch (opt) {
            case 1 -> Collections.sort(copy); // by ID (Comparable)
            case 2 -> copy.sort(new NameComparator());
            case 3 -> copy.sort(new RatingComparator().reversed()); // show highest first
            default -> System.out.println("Invalid option, showing by ID.");
        }
        System.out.printf("%-6s %-25s %-10s %-7s %-10s %s%n", "ID", "Name", "Genre", "Rating", "Platform", "Extra");
        for (Game g : copy) {
            System.out.println(g);
        }
    }
```



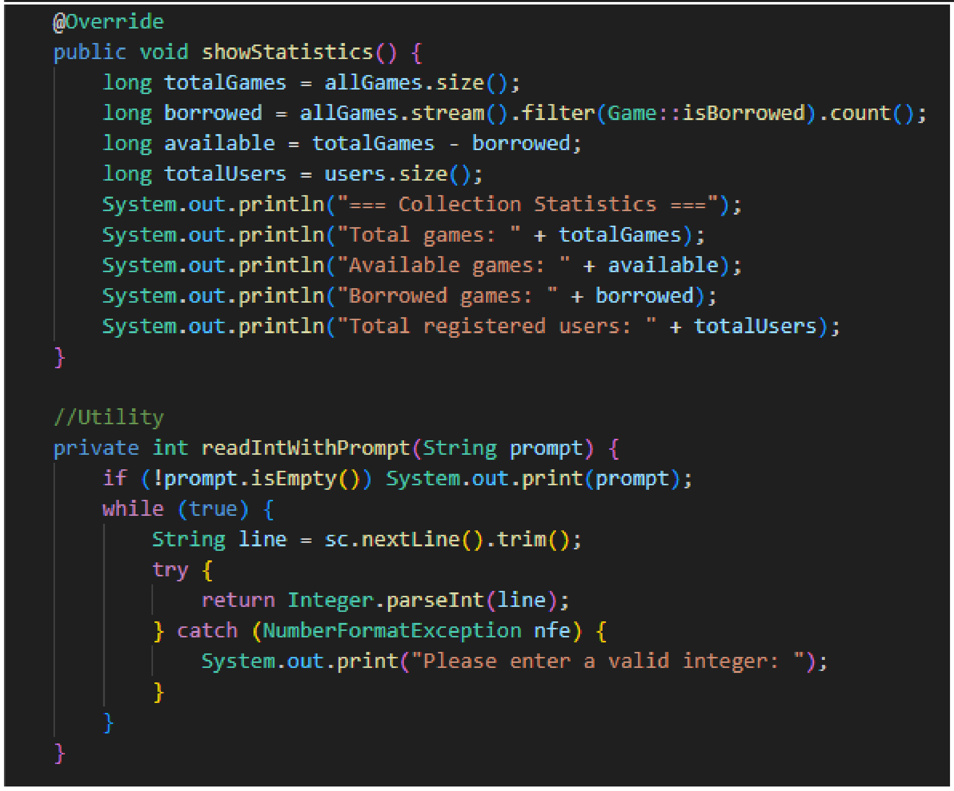
```java
@Override
public void searchGame() {
    System.out.print("Search by (1) Name or (2) Genre: ");
    int opt = readIntWithPrompt("");
    System.out.print("Enter search term: ");
    String term = sc.nextLine().trim().toLowerCase();
    List<Game> results = new ArrayList<>();
    if (opt == 1) {
        for (Game g : allGames)
            if (g.getName().toLowerCase().contains(term)) results.add(g);
    } else {
        for (Game g : allGames)
            if (g.getGenre().toLowerCase().contains(term)) results.add(g);
    }
    if (results.isEmpty()) {
        System.out.println("No matching games found.");
        return;
    }
    System.out.printf("Found %d matching game(s):%n", results.size());
    for (Game g : results) System.out.println(g);
}

@Override
public void registerUser() {
    System.out.print("Enter username: ");
    String username = sc.nextLine().trim();
    if (username.isEmpty()) {
        System.out.println("Username cannot be empty.");
        return;
    }
    System.out.print("Enter email: ");
    String email = sc.nextLine().trim();
    User u = new User(username, email);
    if (users.contains(u)) {
        System.out.println("Username already exists. Registration failed.");
        return;
    }
    users.add(u);
    userMap.put(username, u);
    System.out.println("User registered successfully. Welcome, " + username + "!");
}
```



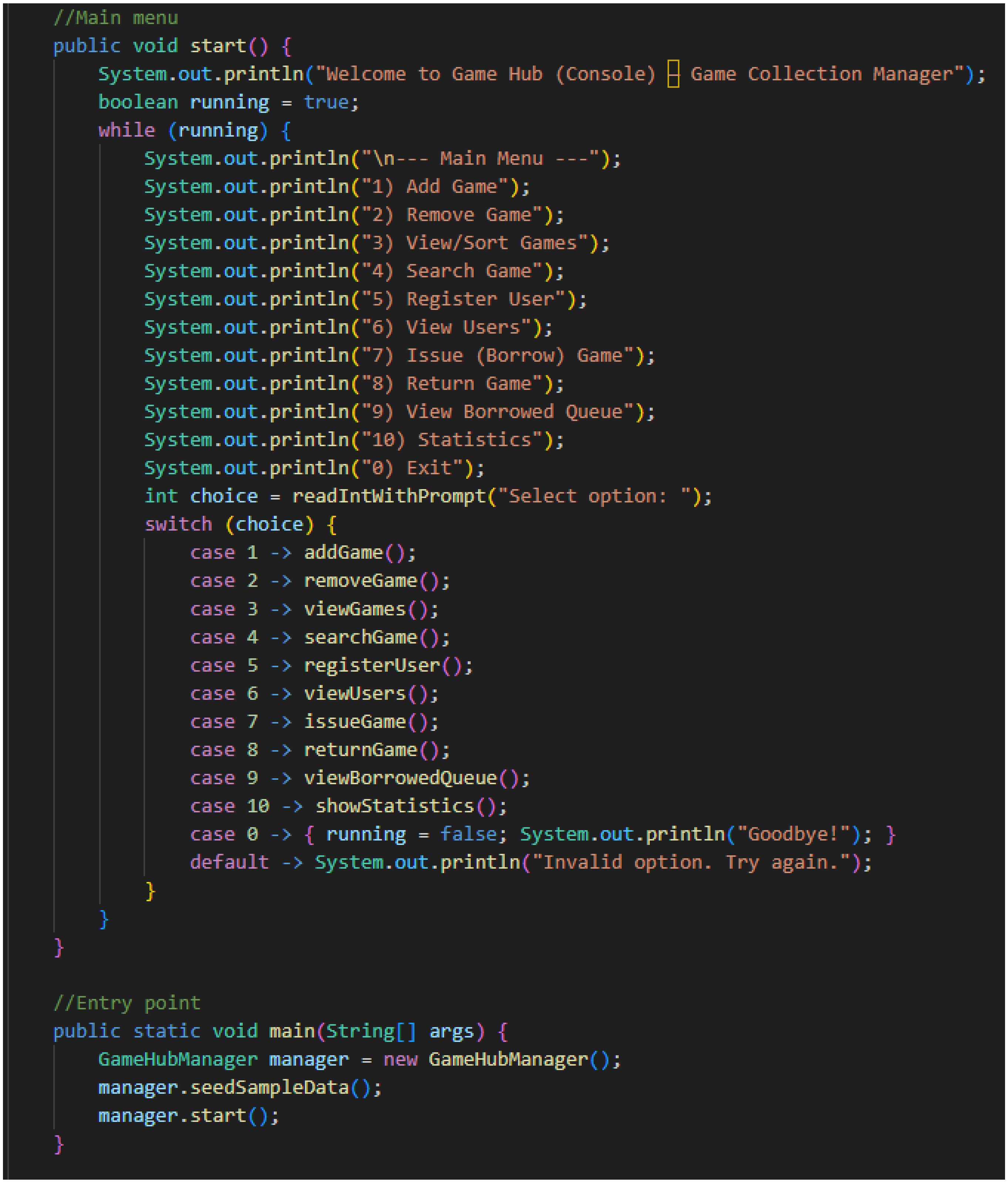
```java
@Override
public void viewUsers() {
    if (users.isEmpty()) {
        System.out.println("No registered users.");
        return;
    }
    System.out.printf("%-15s %-25s %-10s%n", "Username", "Email", "BorrowedCount");
    for (User u : users) {
        System.out.printf("%-15s %-25s %-10d%n", u.getUsername(), u.getEmail(), u.getBorrowedGames().size());
    }
}

@Override
public void issueGame() {
    try {
        int id = readIntWithPrompt("Enter Game ID to borrow: ");
        System.out.print("Enter your username: ");
        String username = sc.nextLine().trim();

        Game g = gameMap.get(id);
        if (g == null) throw new GameNotFoundException("Game ID " + id + " not found.");
        User u = userMap.get(username);
        if (u == null) throw new UserNotFoundException("User '" + username + "' not found. Please register first.");

        if (g.isBorrowed()) {
            System.out.println("Game is already borrowed by " + g.getBorrowedBy() + ". Cannot borrow.");
            return;
        }
        g.setBorrowed(true);
        g.setBorrowedBy(username);
        borrowedQueue.addLast(g);
        u.borrowGame(g);
        System.out.println("Success: " + username + " borrowed " + g.getName() + " (ID: " + g.getId() + ")");
    } catch (GameNotFoundException | UserNotFoundException ex) {
        System.out.println("Error: " + ex.getMessage());
    }
}
```



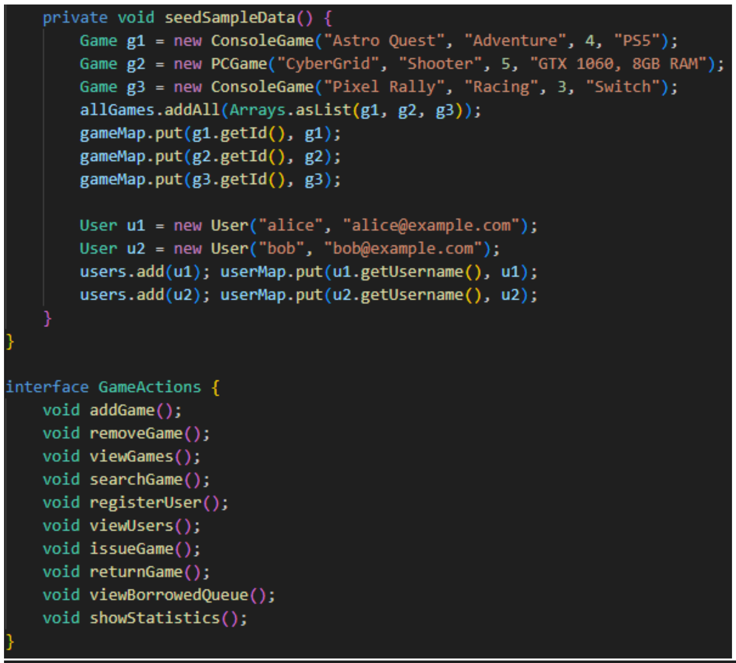
```java
    @Override
    public void returnGame() {
        try {
            int id = readIntWithPrompt("Enter Game ID to return: ");
            System.out.print("Enter your username: ");
            String username = sc.nextLine().trim();

            Game g = gameMap.get(id);
            if (g == null) throw new GameNotFoundException("Game ID " + id + " not found.");
            User u = userMap.get(username);
            if (u == null) throw new UserNotFoundException("User '" + username + "' not found.");

            if (!g.isBorrowed() || !username.equals(g.getBorrowedBy())) {
                System.out.println("Return failed: Only the borrower (" + g.getBorrowedBy() + ") can return this game.");
                return;
            }
            g.setBorrowed(false);
            g.setBorrowedBy(null);
            borrowedQueue.removeIf(game -> game.getId() == id);
            boolean removedFromUser = u.returnGame(g);
            System.out.println("Return " + (removedFromUser ? "successful." : "processed (was not in user's borrowed list)."));
        } catch (GameNotFoundException | UserNotFoundException ex) {
            System.out.println("Error: " + ex.getMessage());
        }
    }

    @Override
    public void viewBorrowedQueue() {
        if (borrowedQueue.isEmpty()) {
            System.out.println("No games currently borrowed.");
            return;
        }
        System.out.println("Borrowed games in queue order (oldest -> newest):");
        System.out.printf("%-6s %-25s %-10s %-10s%n", "ID", "Name", "BorrowedBy", "Rating");
        for (Game g : borrowedQueue) {
            System.out.printf("%-6d %-25s %-10s %-10d%n", g.getId(), g.getName(), g.getBorrowedBy(), g.getRating());
        }
    }
```



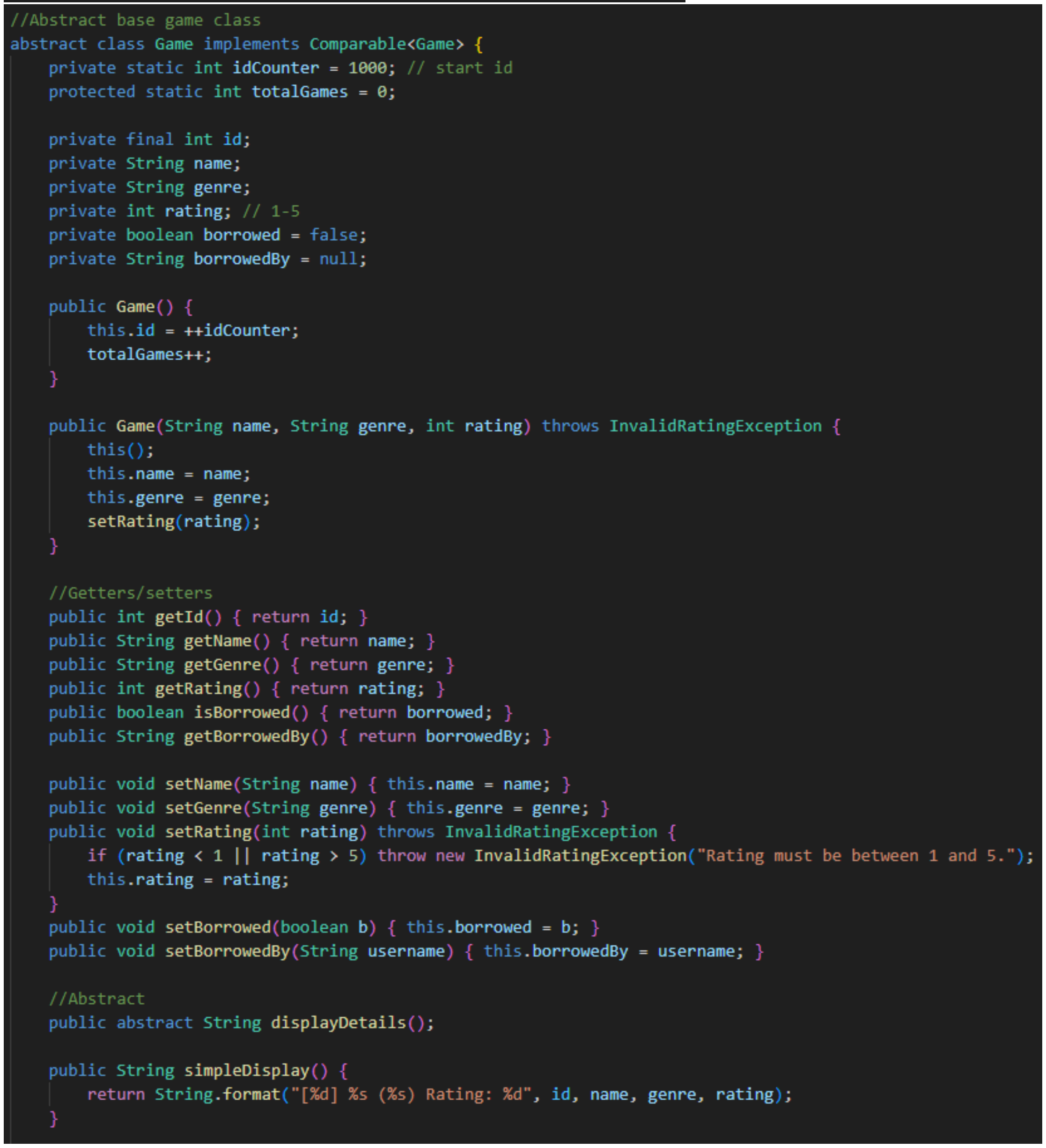
```java
    @Override
    public void showStatistics() {
        long totalGames = allGames.size();
        long borrowed = allGames.stream().filter(Game::isBorrowed).count();
        long available = totalGames - borrowed;
        long totalUsers = users.size();
        System.out.println("=== Collection Statistics ===");
        System.out.println("Total games: " + totalGames);
        System.out.println("Available games: " + available);
        System.out.println("Borrowed games: " + borrowed);
        System.out.println("Total registered users: " + totalUsers);
    }

    //Utility
    private int readIntWithPrompt(String prompt) {
        if (!prompt.isEmpty()) System.out.print(prompt);
        while (true) {
            String line = sc.nextLine().trim();
            try {
                return Integer.parseInt(line);
            } catch (NumberFormatException nfe) {
                System.out.print("Please enter a valid integer: ");
            }
        }
    }
```



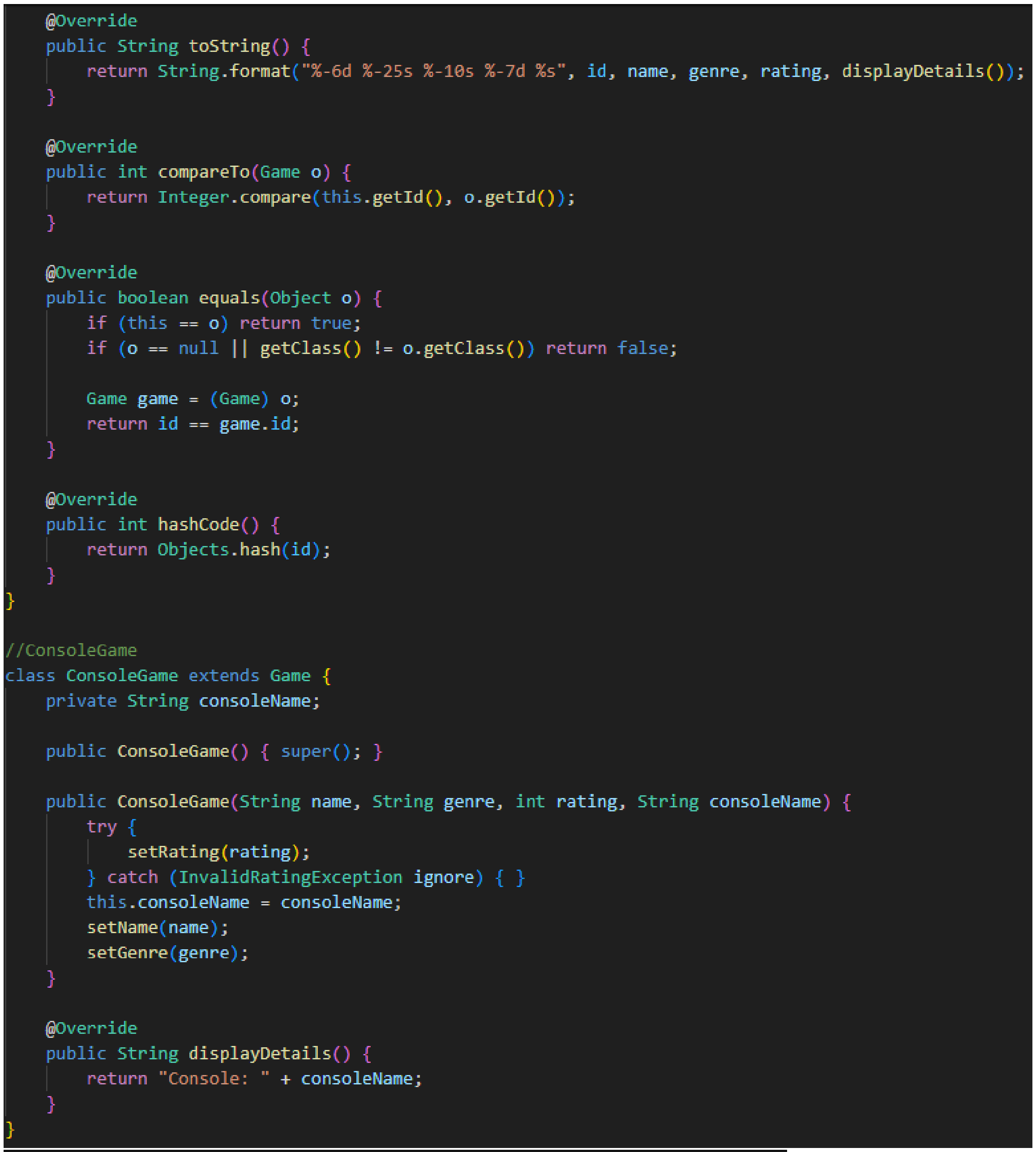
```java
    //Main menu
    public void start() {
        System.out.println("Welcome to Game Hub (Console) □ Game Collection Manager");
        boolean running = true;
        while (running) {
            System.out.println("\n--- Main Menu ---");
            System.out.println("1) Add Game");
            System.out.println("2) Remove Game");
            System.out.println("3) View/Sort Games");
            System.out.println("4) Search Game");
            System.out.println("5) Register User");
            System.out.println("6) View Users");
            System.out.println("7) Issue (Borrow) Game");
            System.out.println("8) Return Game");
            System.out.println("9) View Borrowed Queue");
            System.out.println("10) Statistics");
            System.out.println("0) Exit");
            int choice = readIntWithPrompt("Select option: ");
            switch (choice) {
                case 1 -> addGame();
                case 2 -> removeGame();
                case 3 -> viewGames();
                case 4 -> searchGame();
                case 5 -> registerUser();
                case 6 -> viewUsers();
                case 7 -> issueGame();
                case 8 -> returnGame();
                case 9 -> viewBorrowedQueue();
                case 10 -> showStatistics();
                case 0 -> { running = false; System.out.println("Goodbye!"); }
                default -> System.out.println("Invalid option. Try again.");
            }
        }
    }

    //Entry point
    public static void main(String[] args) {
        GameHubManager manager = new GameHubManager();
        manager.seedSampleData();
        manager.start();
    }
```



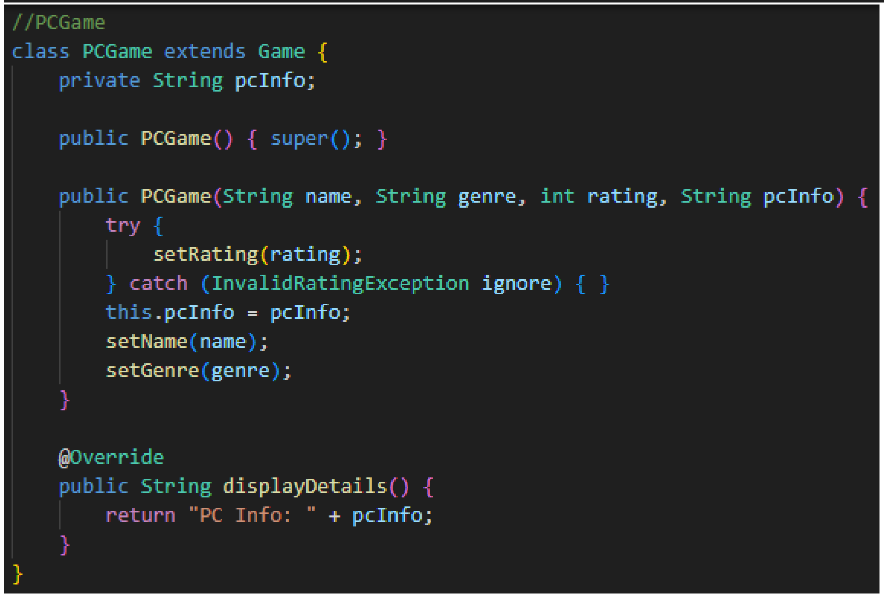
```java
    private void seedSampleData() {
        Game g1 = new ConsoleGame("Astro Quest", "Adventure", 4, "PS5");
        Game g2 = new PCGame("CyberGrid", "Shooter", 5, "GTX 1060, 8GB RAM");
        Game g3 = new ConsoleGame("Pixel Rally", "Racing", 3, "Switch");
        allGames.addAll(Arrays.asList(g1, g2, g3));
        gameMap.put(g1.getId(), g1);
        gameMap.put(g2.getId(), g2);
        gameMap.put(g3.getId(), g3);

        User u1 = new User("alice", "alice@example.com");
        User u2 = new User("bob", "bob@example.com");
        users.add(u1); userMap.put(u1.getUsername(), u1);
        users.add(u2); userMap.put(u2.getUsername(), u2);
    }
}

interface GameActions {
    void addGame();
    void removeGame();
    void viewGames();
    void searchGame();
    void registerUser();
    void viewUsers();
    void issueGame();
    void returnGame();
    void viewBorrowedQueue();
    void showStatistics();
}
```



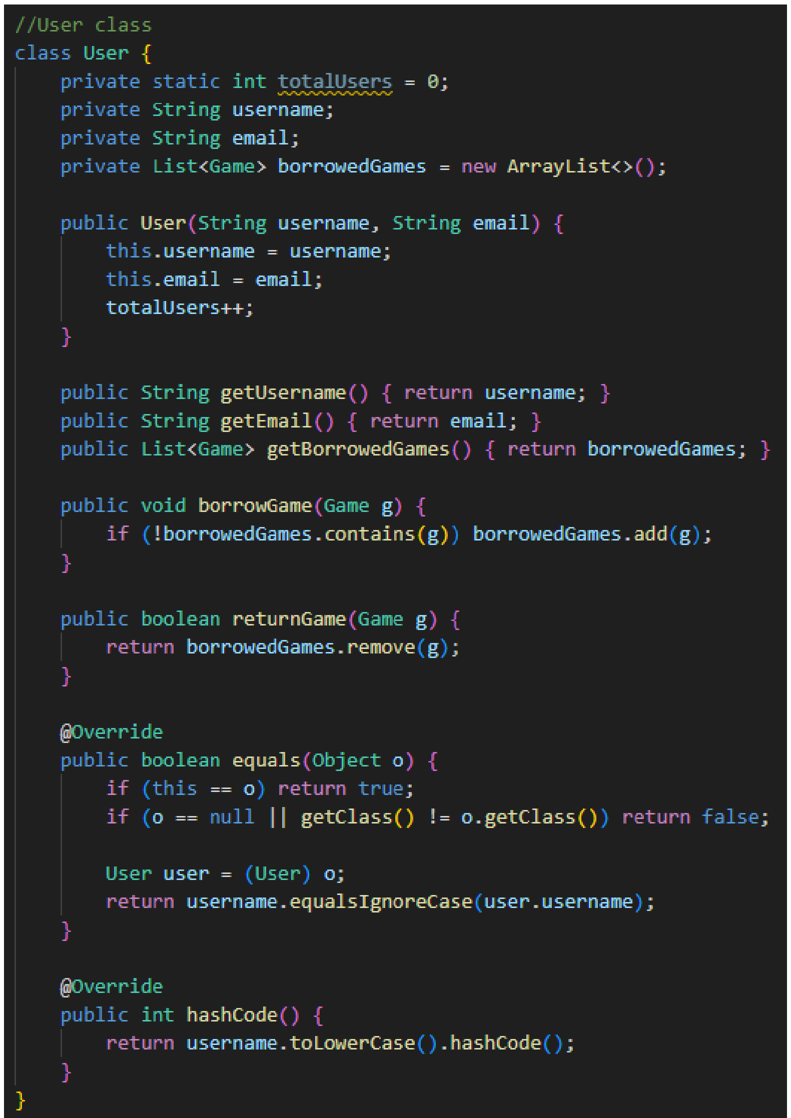
```java
//Abstract base game class
abstract class Game implements Comparable<Game> {
    private static int idCounter = 1000; // start id
    protected static int totalGames = 0;

    private final int id;
    private String name;
    private String genre;
    private int rating; // 1-5
    private boolean borrowed = false;
    private String borrowedBy = null;

    public Game() {
        this.id = ++idCounter;
        totalGames++;
    }

    public Game(String name, String genre, int rating) throws InvalidRatingException {
        this();
        this.name = name;
        this.genre = genre;
        setRating(rating);
    }

    //Getters/setters
    public int getId() { return id; }
    public String getName() { return name; }
    public String getGenre() { return genre; }
    public int getRating() { return rating; }
    public boolean isBorrowed() { return borrowed; }
    public String getBorrowedBy() { return borrowedBy; }

    public void setName(String name) { this.name = name; }
    public void setGenre(String genre) { this.genre = genre; }
    public void setRating(int rating) throws InvalidRatingException {
        if (rating < 1 || rating > 5) throw new InvalidRatingException("Rating must be between 1 and 5.");
        this.rating = rating;
    }
    public void setBorrowed(boolean b) { this.borrowed = b; }
    public void setBorrowedBy(String username) { this.borrowedBy = username; }

    //Abstract
    public abstract String displayDetails();

    public String simpleDisplay() {
        return String.format("[%d] %s (%s) Rating: %d", id, name, genre, rating);
    }
```



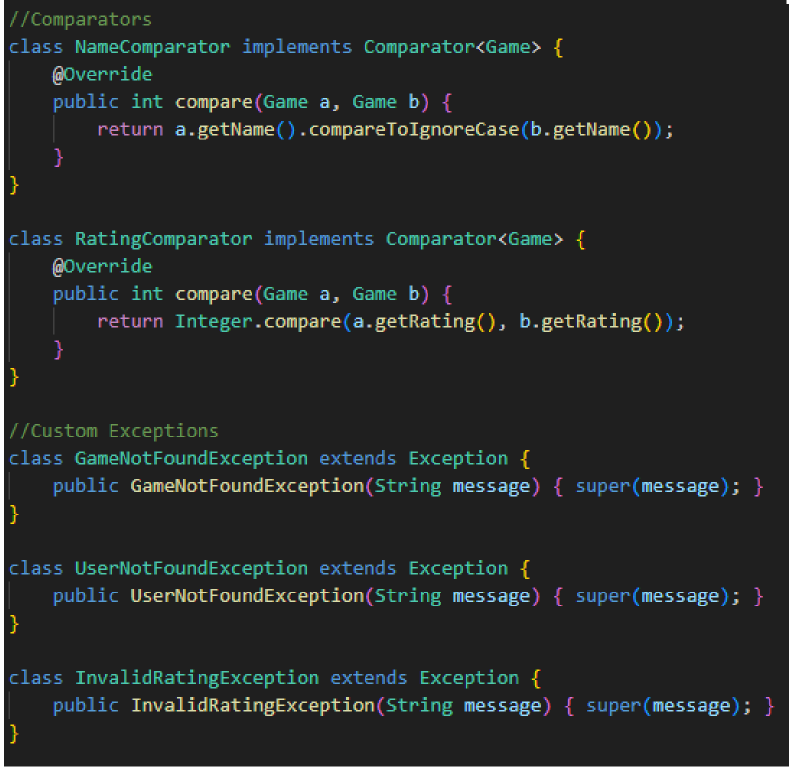
```java
    @Override
    public String toString() {
        return String.format("%-6d %-25s %-10s %-7d %s", id, name, genre, rating, displayDetails());
    }

    @Override
    public int compareTo(Game o) {
        return Integer.compare(this.getId(), o.getId());
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Game game = (Game) o;
        return id == game.id;
    }

    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
}

//ConsoleGame
class ConsoleGame extends Game {
    private String consoleName;

    public ConsoleGame() { super(); }

    public ConsoleGame(String name, String genre, int rating, String consoleName) {
        try {
            setRating(rating);
        } catch (InvalidRatingException ignore) { }
        this.consoleName = consoleName;
        setName(name);
        setGenre(genre);
    }

    @Override
    public String displayDetails() {
        return "Console: " + consoleName;
    }
}
```



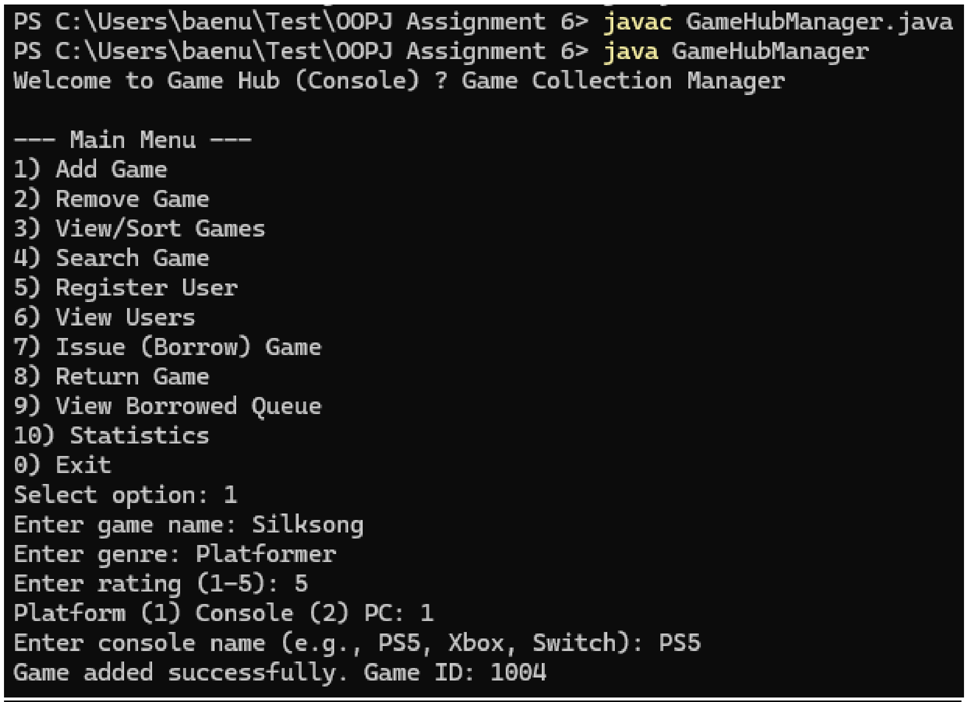
```java
//PCGame
class PCGame extends Game {
    private String pcInfo;

    public PCGame() { super(); }

    public PCGame(String name, String genre, int rating, String pcInfo) {
        try {
            setRating(rating);
        } catch (InvalidRatingException ignore) { }
        this.pcInfo = pcInfo;
        setName(name);
        setGenre(genre);
    }

    @Override
    public String displayDetails() {
        return "PC Info: " + pcInfo;
    }
}
```



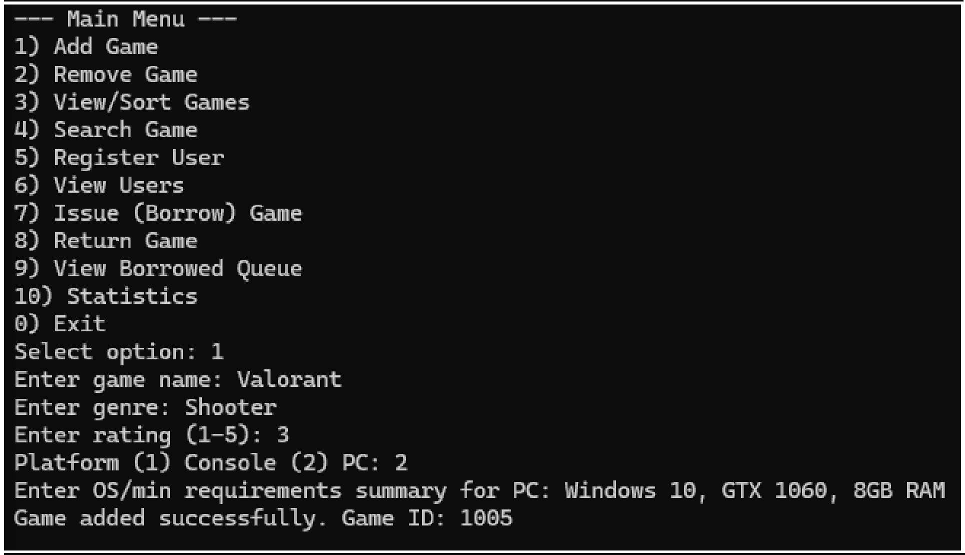
```java
//User class
class User {
    private static int totalUsers = 0;
    private String username;
    private String email;
    private List<Game> borrowedGames = new ArrayList<>();

    public User(String username, String email) {
        this.username = username;
        this.email = email;
        totalUsers++;
    }

    public String getUsername() { return username; }
    public String getEmail() { return email; }
    public List<Game> getBorrowedGames() { return borrowedGames; }

    public void borrowGame(Game g) {
        if (!borrowedGames.contains(g)) borrowedGames.add(g);
    }

    public boolean returnGame(Game g) {
        return borrowedGames.remove(g);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        User user = (User) o;
        return username.equalsIgnoreCase(user.username);
    }

    @Override
    public int hashCode() {
        return username.toLowerCase().hashCode();
    }
}
```



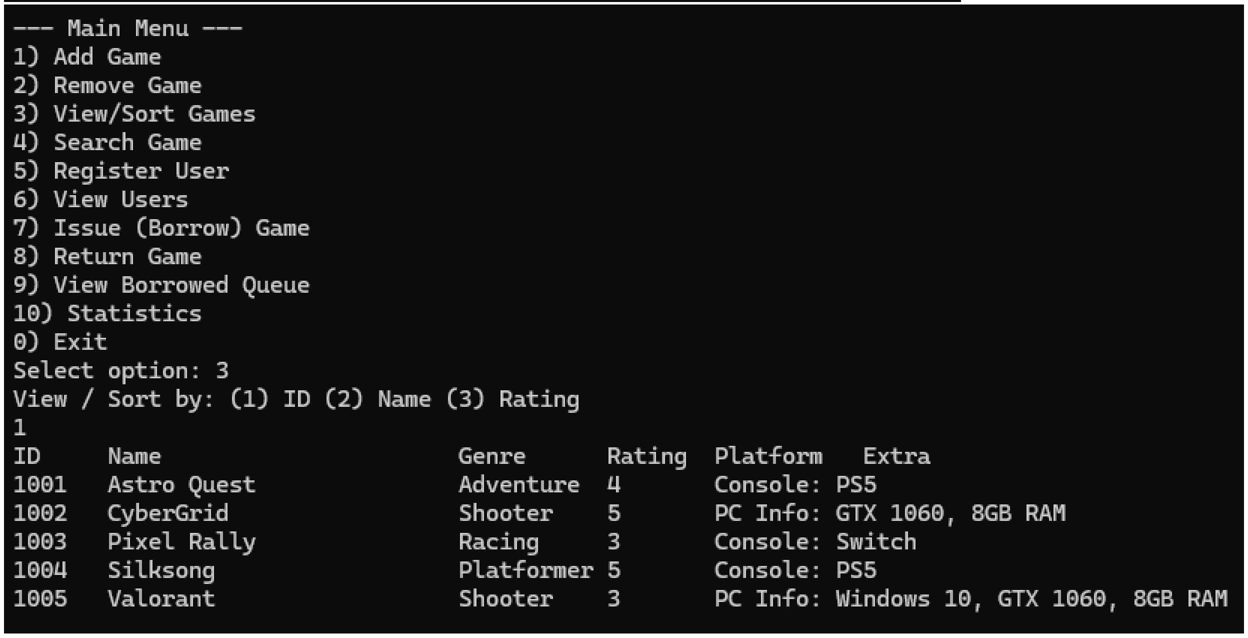
```java
//Comparators
class NameComparator implements Comparator<Game> {
    @Override
    public int compare(Game a, Game b) {
        return a.getName().compareToIgnoreCase(b.getName());
    }
}

class RatingComparator implements Comparator<Game> {
    @Override
    public int compare(Game a, Game b) {
        return Integer.compare(a.getRating(), b.getRating());
    }
}

//Custom Exceptions
class GameNotFoundException extends Exception {
    public GameNotFoundException(String message) { super(message); }
}

class UserNotFoundException extends Exception {
    public UserNotFoundException(String message) { super(message); }
}

class InvalidRatingException extends Exception {
    public InvalidRatingException(String message) { super(message); }
}
```



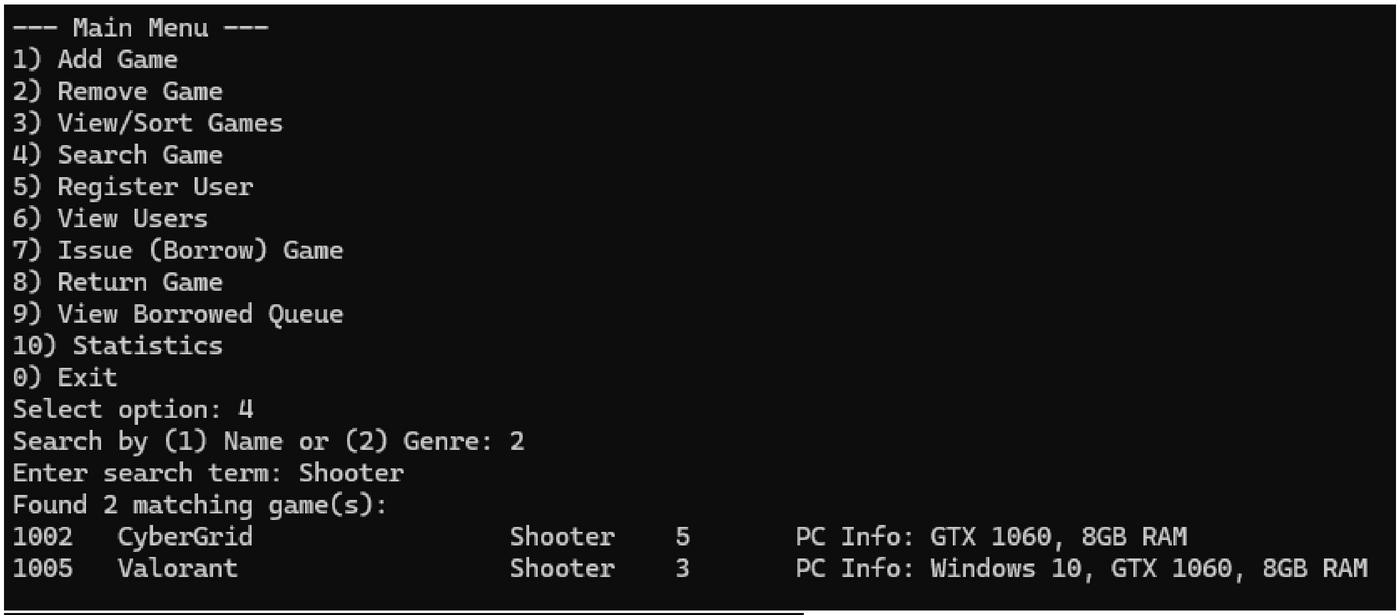
```
PS C:\Users\baenu\Test\OOPJ Assignment 6> javac GameHubManager.java
PS C:\Users\baenu\Test\OOPJ Assignment 6> java GameHubManager
Welcome to Game Hub (Console) ? Game Collection Manager

--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 1
Enter game name: Silksong
Enter genre: Platformer
Enter rating (1-5): 5
Platform (1) Console (2) PC: 1
Enter console name (e.g., PS5, Xbox, Switch): PS5
Game added successfully. Game ID: 1004
```



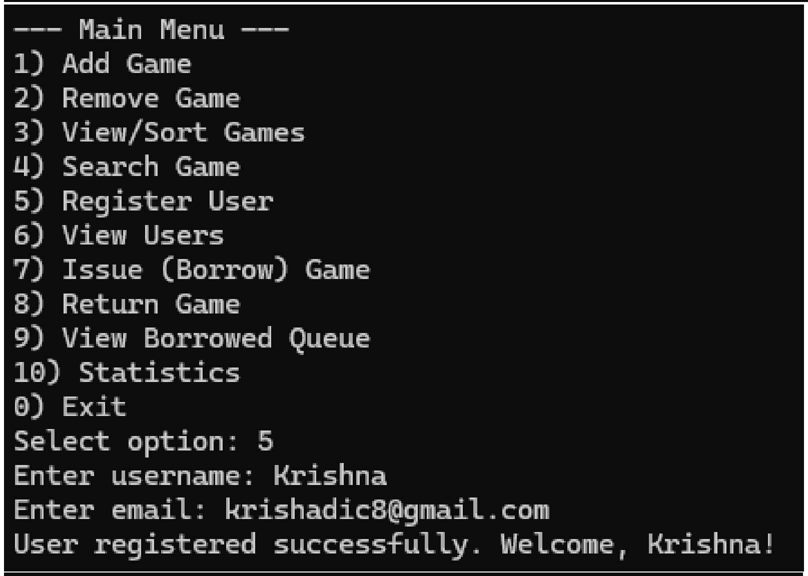
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 1
Enter game name: Valorant
Enter genre: Shooter
Enter rating (1-5): 3
Platform (1) Console (2) PC: 2
Enter OS/min requirements summary for PC: Windows 10, GTX 1060, 8GB RAM
Game added successfully. Game ID: 1005
```



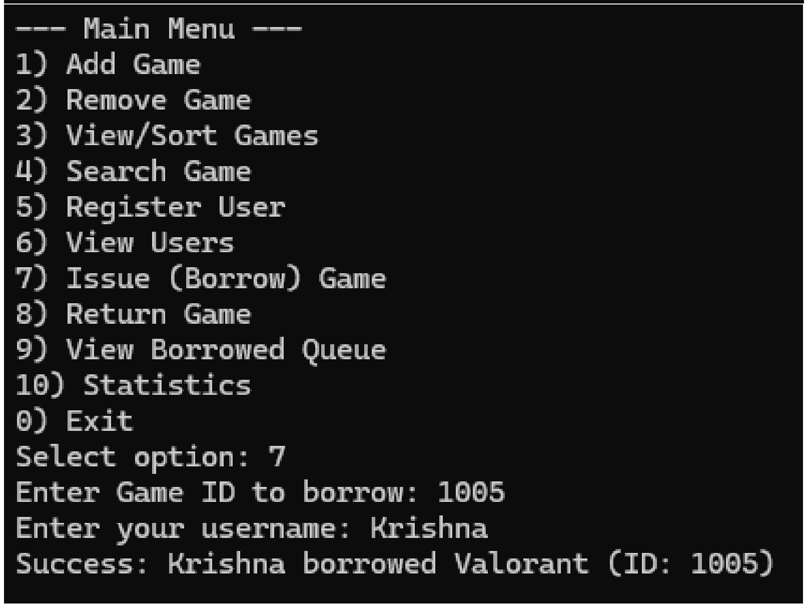
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 3
View / Sort by: (1) ID (2) Name (3) Rating
1
ID     Name                      Genre      Rating  Platform   Extra
1001   Astro Quest               Adventure  4       Console: PS5
1002   CyberGrid                 Shooter    5       PC Info: GTX 1060, 8GB RAM
1003   Pixel Rally               Racing     3       Console: Switch
1004   Silksong                  Platformer 5       Console: PS5
1005   Valorant                  Shooter    3       PC Info: Windows 10, GTX 1060, 8GB RAM
```



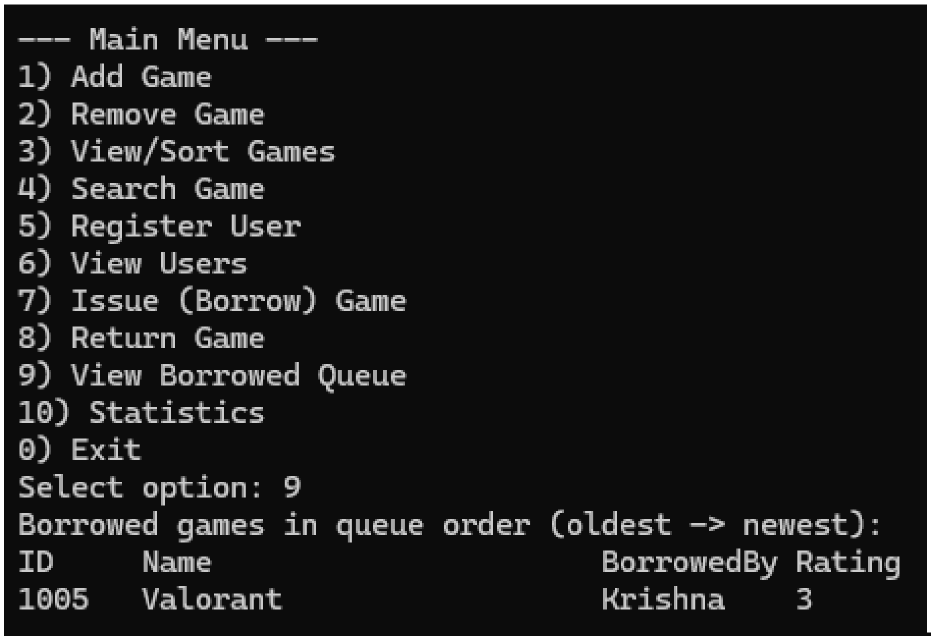
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 4
Search by (1) Name or (2) Genre: 2
Enter search term: Shooter
Found 2 matching game(s):
1002   CyberGrid                 Shooter    5      PC Info: GTX 1060, 8GB RAM
1005   Valorant                  Shooter    3      PC Info: Windows 10, GTX 1060, 8GB RAM
```



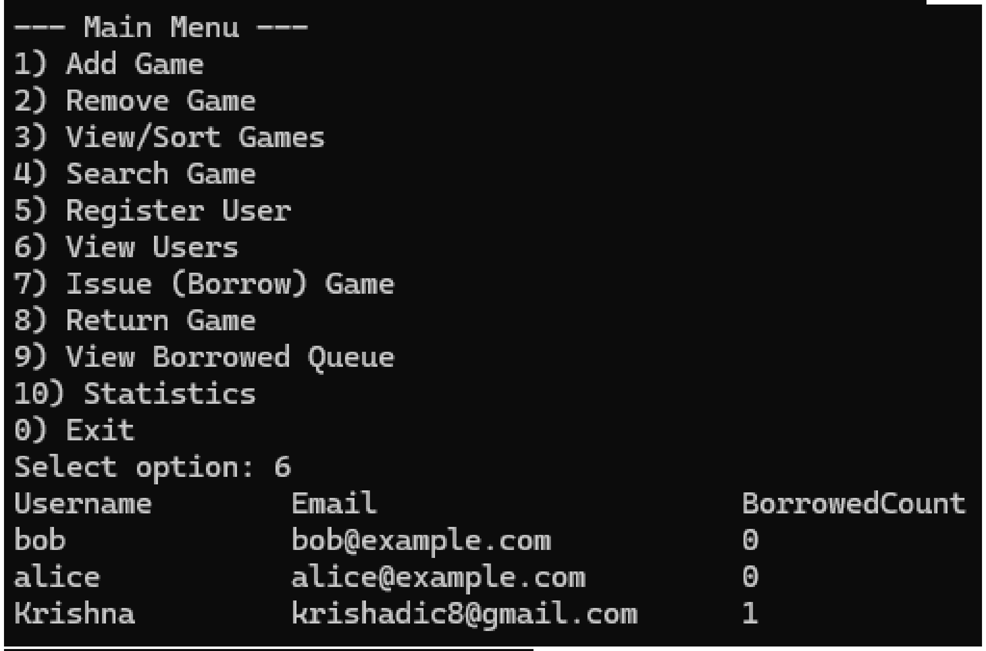
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 5
Enter username: Krishna
Enter email: krishadic8@gmail.com
User registered successfully. Welcome, Krishna!
```



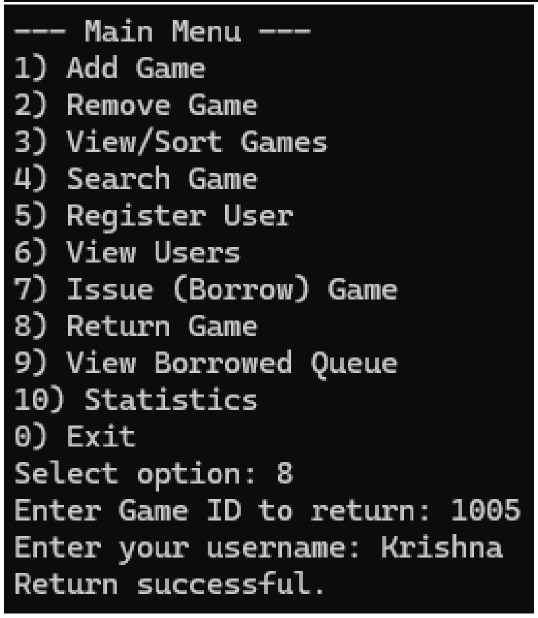
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 7
Enter Game ID to borrow: 1005
Enter your username: Krishna
Success: Krishna borrowed Valorant (ID: 1005)
```



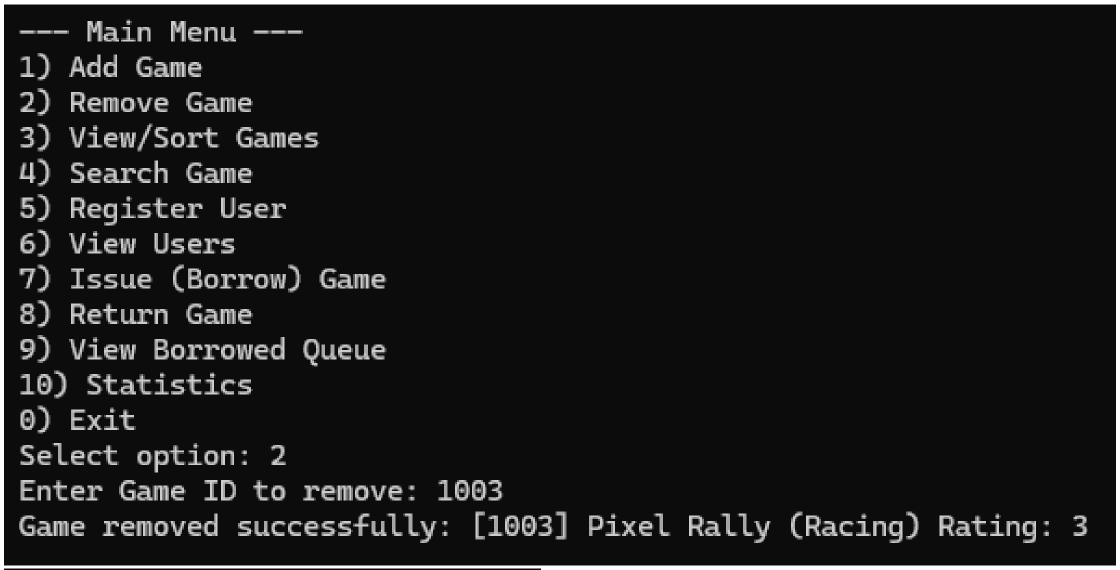
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 9
Borrowed games in queue order (oldest -> newest):
ID     Name                            BorrowedBy Rating
1005   Valorant                        Krishna    3
```



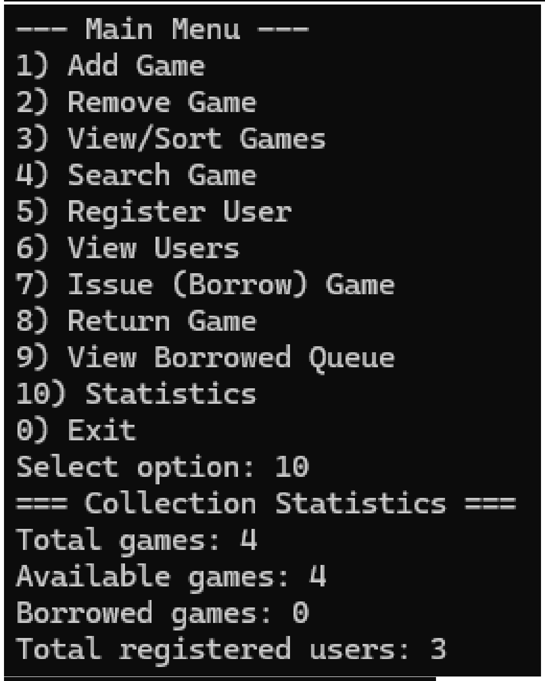
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 6
Username        Email                     BorrowedCount
bob             bob@example.com           0
alice           alice@example.com         0
Krishna         krishadic8@gmail.com      1
```



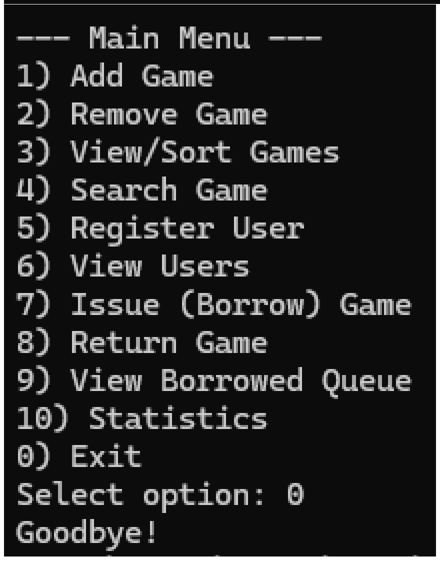
```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 8
Enter Game ID to return: 1005
Enter your username: Krishna
Return successful.
```



```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 2
Enter Game ID to remove: 1003
Game removed successfully: [1003] Pixel Rally (Racing) Rating: 3
```



```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 10
=== Collection Statistics ===
Total games: 4
Available games: 4
Borrowed games: 0
Total registered users: 3
```



```
--- Main Menu ---
1) Add Game
2) Remove Game
3) View/Sort Games
4) Search Game
5) Register User
6) View Users
7) Issue (Borrow) Game
8) Return Game
9) View Borrowed Queue
10) Statistics
0) Exit
Select option: 0
Goodbye!
```