MK Fowlkes

4-2 Assignment: Game Player Management Application

Paul.fowkes.snhu.edu

Working on creating a singleton pattern for `GameService` class in Java. The basic structure is there, but there are some corrections and improvements that can be made.

GameService.java

```java

package com.gamingroom;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class GameService {

/\*\*

\* Properties

\*/

private static GameService instance; // Renamed to instance

private List<Game> games = new ArrayList<>(); // Removed static keyword

private long nextGameId = 1;

private long nextPlayerId = 1;

private long nextTeamId = 1;

// Private constructor to prevent instantiation

private GameService() {}

// Return reference to Singleton Game Service

public static GameService getInstance() {

// Use double-check locking for thread safety

if (instance == null) {

synchronized (GameService.class) {

if (instance == null) {

instance = new GameService();

}

}

}

return instance;

}

// Other methods remain unchanged

// ...

/\*\*

\* Returns the next player id

\*

\* @return the next player id

\*/

public long getNextPlayerId() {

return nextPlayerId++;

}

/\*\*

\* Returns the next team id

\*

\* @return the next team id

\*/

public long getNextTeamId() {

return nextTeamId++;

}

}

```

### ProgramDriver.java

```java

package com.gamingroom;

public class ProgramDriver {

/\*\*

\* The one-and-only main() method

\*

\* @param args command line arguments

\*/

public static void main(String[] args) {

// Obtain reference to the singleton instance

GameService service = GameService.getInstance();

System.out.println("\nAbout to test initializing game data...");

// initialize with some game data

Game game1 = service.addGame("Game #1");

System.out.println(game1);

Game game2 = service.addGame("Game #2");

System.out.println(game2);

// use another class to prove there is only one instance

SingletonTester tester = new SingletonTester();

tester.testSingleton();

}

}

```

### SingletonTester.java (Assuming you have a class like this)

```java

package com.gamingroom;

public class SingletonTester {

public void testSingleton() {

// Obtain reference to the singleton instance

GameService service = GameService.getInstance();

// Additional testing logic if needed

}

}

```

Key changes:

1. Renamed `reference` to `instance` for clarity.

2. Made the `games` list non-static to allow for multiple instances of `GameService` if needed in the future.

3. Added a private constructor to prevent external instantiation.

4. Modified the `getInstance` method to use double-check locking for thread safety.

5. Updated the `ProgramDriver` to use the `getInstance` method instead of the `getReference`.

6. Removed unnecessary comments and fixed some formatting issues for readability.

These changes should help improve the design and implementation of the singleton pattern.