Integration Documentation for Leaderboard

1. RankingAlgorithm.java — Core rating engine

Purpose: Implements the ELO rating algorithm to calculate and update player ratings.

How it works:

- Uses ELO: calculates expected scores, then updates ratings based on actual vs expected
- Supports multiple games: chess, go, tic-tac-toe (each with its own K-factor)
- Updates ratings, win/loss/tie counts, and historical ratings

Persists changes via WriterDatabase.save()

Key methods:

- updateRatings(player1, player2, game, score1, score2) — Main entry point
- getPlayerRating(player, game) — Get current rating
- setPlayerRating(player, game, rating) — Set rating

Integration: Call after each game ends: RankingAlgorithm.updateRatings(player1, player2, "chess", 1.0, 0.0);

2. Leaderboard.java — Display and query system

Purpose: Provides leaderboard views and rankings.Features:

- Top N players: getTopPlayers(game, topN)
- Global rankings: displayGlobalRankings(game)
- Friend rankings: displayFriendRankings(currentPlayer, game)
- Historical rankings: displayHistoricalRankings(player, game)

How it works:

- Uses Java Streams to sort players by rating
- Delegates rating retrieval to RankingAlgorithm
- Console output (can be adapted for GUI)

Integration: Call to display leaderboards: Leaderboard.displayLeaderboard("chess", 10); *// Top 10 chess players*

3. AdminControls.java — Administrative functions

Purpose: Admin operations for leaderboard management.Features:

- resetLeaderboard(game) — Resets all players' stats for a game to defaults (rating 1000, wins/losses/ties = 0)

How it works:

- Iterates through all players in PlayerData
- Resets ratings, stats, and historical data for the specified game

Integration: Admin-only function: AdminControls.resetLeaderboard("chess"); *// Reset chess leaderboard*

4. LeaderboardConfig.java — Configuration management

Purpose: Manages K-factors (rating volatility) per game.How it works:

- Static map stores K-factors (chess: 32, go: 24, tic-tac-toe: 16)
- Allows runtime configuration changes

Note: Currently not used by RankingAlgorithm (which has hardcoded K-factors). Could be refactored to use this.Integration: Configure K-factors: LeaderboardConfig.setKFactor("chess", 40); *// Increase volatility*

5. AchievementSharing.java — Social feature

Purpose: Simulates sharing achievements to social media.How it works:

- Simple utility that formats and prints achievement messages
- Placeholder for future social integration

Integration: Share achievements: AchievementSharing.shareAchievement(player, "Reached 1500 rating!");

6. Dependencies

- Player — player objects
- PlayerStats — stores ratings and stats
- PlayerData — global player database
- WriterDatabase — persistence