King developer test

Personal Statement

This program does not use any external frameworks as stipulated by the Non-functional requirements. As such it is designed as a single standalone program instead of it being run in an application container. It is not designed as a client/server application to reduce the need for plumbing code outside the requirements. This program may miss some requirements but I hope that this is sufficient in providing an insight into my coding ability.

Despite the risk of over-engineering as the program has few features, my approach to the design is to provide components and code structure that may be used in more complex applications.

Instructions

To run the Java program run the ./run.sh script in Linux/Unix environment, or ./run.bat in a Windows environment.

Score results are output in the console as well as in an external log file **./log/results.log** Total execution time: approximately 3 to 5 minutes

- Players can be added to the external **./data/players.properties** file in the format |.
    - 11 players configured
- Config Features **./config/config.properties**:
    - Each level configured to simulate 1000ms of game play to speed total execution time: simulateLevelPlayPause=1000
    - Game has 10 levels: maxGameLevelsAllowed = 10
    - Each level has 20 plays: maxLevelPlay =20

Code Features

- External files for Player input data and configuration properties to allow easy modification.
- Application configuration properties located in the **./config** directory:
    - config.properties Global configuration properties e.g. game levels etc.
    - logger.properties java.util.logging.Logger properties
- Lambda expressions used where suitable.

- Java Generics used where suitable to highlight extensibility for interfaces while maintaining strong type checks.
- Repository interface with specific MapRepository interface implemention to store the scores
- LifetimeThreadExcutorService wrapper for the concurrency ExecutorService created so its ThreadPool can be reused. This executor does not shutdown, much like a ThreadPool in an application server.
- newCachedThreadPool ThreadPool used instead of a fixed pool. newWorkStealingThreadPool starts with too few threads and does not provide the initial concurrency required for the demo. Limited testing done for this.
- ScoreEventListener to highlight event listener pattern for keeping scores. To be replaced with a 'messaging' system in the real-world.

Limitations

- Some requirements not completed (due to time spent)
- Test coverage is not sufficient.
- Not client/server.
- Not Rest or no server side services to handle requests (to avoid introducing external frameworks, and provide an executable .jar file)
- Logs used to show score results as there is no request client to pass score quieries to.
- Authentication - No facade used to intercept function calls to verify user session. Crude inline calls from each function only.