Battleship sprint 5

1 Outline

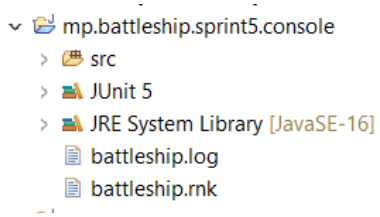
This sprint will add functions related to the management of the ranking and file input/output.

- There will be a persistent copy of the ranking in a file with a predefined name and location.
- When menu options 2 or 3 are selected, the ranking will be displayed sorted by some criteria.
- Finally, a log will be implemented so that some events will be recorded in a file.

2 Persistent copy of the Ranking

Up to the previous sprint, the ***ranking*** is transient data. That is, the list of scores is created within an application session in main memory. At the end of the session, it is discarded and not stored in a persistent media.

This sprint requires to implement a **persistent ranking** so, **when user wins and decides to save their score, it will exist from session to session**. To achieve this, there will be a **copy of the ranking in-memory in a file on secondary storage**. This file has a default name (**battleship.rnk**) and location (the **root of the project**) and it is unique for all sessions.



A new Main file is provided for the Console project. It contains constants defining the names for the new file.

It is essential that memory and disk rankings are always **consistent**, that is, their content must be the same at any moment. To that end:

- When the **session is run**, the content of the file *battleship.rnk* is loaded in the application memory.
- Whenever the user decides to save their score, a new Score object is added to the ranking and *battleship.rnk* will also be updated to maintain consistency.

Running these operations might cause several errors.

Error	How to handle
When launching the application, if *battleship.rnk* is not found in the expected location.	* Ranking in memory is **created anew**.
When launching the application. Other IO error while trying to read battleship.rnk	* The stack trace is sent to the log, and a message is printed out. The application exits.
When saving a new score. Any exception trying to save the ranking in the battleship.rnk.	

*** NOTICE:** Use **SessionInteractor::showFatalErrorMessage(msg)** to warn the user before the application exits.

2.1 Implementing a persistent version of the ranking

This time, we are using **Java serialization**, do not implement your own parsers/serializers. The following changes on the previous project are needed:

1. A **new parameter must be added to GameRanking constructor**: **String rankingFilename**. It refers to the path to the ranking file.
2. The file ranking **should be loaded** in memory when the application starts, and it **should be overwritten** each time ranking in memory changes. These operations involve **serializing or de-serializing the score list in GameRanking from/to the file** received in the constructor's argument.
3. The class **Score** must implement Java **Serializable** interface so Java serialization mechanism can use it.

A new class `ObjectPersister` is provided and must be used to load and save ranking from/to the file.

3 Sorting the scores in the ranking

Depending on the option selected, the scores will be printed out differently.

When user selects option 2 (**display my scores**), they appear sorted by date where newest games will appear before (**natural order**).

When user selects option 3 (**display all scores**), they appear sorted by the following criteria:

1. **Level**. Games with higher levels of difficulty (SEA is the easiest, while PLANET is the hardest) will appear before. Java compareTo() method for enums sorts according to the order they are written in the type.
2. **Elapsed time**. If a tie in level, games with a shorter duration will appear before.
3. **Date.** If a tie in elapsed time, oldest games will appear before.

Sort Score objects by **implementing a ScoreComparator that implements interface Comparator<Score>** as well as whatever classes and methods you think are necessary.

4 Write logs to a text file

Use **class FileLogger** to **write log messages in text file named battleship.log**.

Add a public **constructor** to the class FileLogger.

`constructor public FileLogger (String filename)`
It initializes the filename to the argument so the logger will write in this file.

Sample line: `Fri Apr 19 13:12:41 CEST 2024 java.lang.NumberFormatException: For input string: "s"`

5 Class GameSession

Add the following public methods to class GameSession:

`public void setLoggerFile (String filename)`
Where filename is the name of the file to act as log file.

`public void setGameRankingFile (String filename)`
Where filename is the name of the file to act as ranking file.

Apart from these new methods, **method run must also change** so, when it is executed, new GameRanking and FileLogger objects are created with the filenames set with the previous methods.

6 Class GameRanking

Add the following public methods to class GameRanking or change the actual ones:

`constructor public GameRanking (String filename)`
It loads from the file with this name the objects to populate the list of scores.

`public void append (Score score)`
It adds the score to the ranking and saves the ranking in the backup file.

IMPORTANT: The final project must be submitted with a file `battleship.rnk` in the default location. It must contain, at least, these four scores: for a player, three games, two Sea, another Ocean. For a second player, one game, any level.