GAME DATA PARSER

Ultimate C# Masterclass Assignment

Overview

This application reads and deserializes video game data from a JSON file. It is robust and should not crash under any circumstances. Any unhandled exceptions are written to the log file.	Console App

Sample input JSON:


```
[
  {
    "Title": "Stardew Valley",
    "ReleaseYear": 2016,
    "Rating": 4.9
  },
  {
    "Title": "Frostpunk",
    "ReleaseYear": 2017,
    "Rating": 4.7
  },
  {
    "Title": "Oxygen Not Included",
    "ReleaseYear": 2017,
    "Rating": 4.8
  },
  {
    "Title": "Red Dead Redemtpion II",
    "ReleaseYear": 2018,
    "Rating": 4.8
  },
  {
    "Title": "Portal 2",
    "ReleaseYear": 2011,
    "Rating": 4.8
  }
]
```

Main application workflow

<table>
<tr><td colspan="2">When the application starts, it shall print:</td></tr>
<tr><td colspan="2">Enter the name of the file you want to read:</td></tr>
<tr><td colspan="2">The user must enter the name of an existing file. The app should keep asking the user to enter the file until the valid name is entered. See Entering the file name by the user for more details.</td></tr>
<tr><td colspan="2">Next, the app tries to read the video game data from the given file. See Reading the data from the JSON file.</td></tr>
<tr><td>The file contains valid JSON</td><td>The file does not contain valid JSON</td></tr>
<tr><td>Sample file: https://drive.google.com/file/d/1xfSqAdaNglliZqgi-q1Tune8G500mVCo/view?usp=sharing</td><td>Sample file: https://drive.google.com/file/d/1UKdgCMSTcY8oduWGaDCxrZQ7AjmzNUgP/view?usp=sharing</td></tr>
<tr><td>Video games read from the file are printed to the screen. See Printing video games.</td><td>“JSON in the FILE_NAME was not in a valid format. JSON body:” is printed to the console in red, followed by the contents of the JSON file.</td></tr>
<tr><td></td><td>“Sorry! The application has experienced an unexpected error and will have to be closed.” is printed.</td></tr>
<tr><td></td><td>A new entry is added to the log file, containing the date and time of the exception, the exception message, and the stack trace. See Logging exceptions in a file for more information.</td></tr>
<tr><td colspan="2">Finally, the app prints “Press any key to close.” and after that, the window closes.</td></tr>
</table>

Entering the file name by the user

Scenario	User action	Result
Sunny day	User enters a file name that is not null and not empty, and file with such a name exists.	The app moves on to the next step - **Reading the data from the JSON file.**
Null input	The user presses CTRL+Z in the console.	“File name cannot be null.” is printed to the console. Then, the user is asked again to enter the file name. It is repeated until the user provides the correct input.
Empty input	The user enters an empty file path.	“File name cannot be empty.” is printed to the console. Then, the user is asked again to enter the file name. It is repeated until the user provides the correct input.
No file with such a name	The user enters a valid file name, but such a file does not exist.	“File not found.” is printed to the console. Then, the user is asked again to enter the file name. It is repeated until the user provides the correct input.

Reading the data from the JSON file

We, as the developers of this app, do not create the JSON files ourselves. We can imagine we read them from the web, or from some database. We must create our code in such a way, that it can handle the given files.

An example of a valid file can be found here: https://drive.google.com/file/d/1xfSqAdaNglliZqgi-q1Tune8G500mVCo/view?usp=sharing

Scenario	User action	Result
Sunny day	The file contains a valid array of video game data. You can see a **sample valid JSON** under this link**:** https://drive.google.com/file/d/1xfSqAdaNglliZqgi-q1Tune8G500mVCo/view?usp=sharing	The app reads the file and deserializes its contents to a collection of objects, which can then be printed to the console. See **Printing video games** for more detail.
The file is not valid	The file does not contain valid JSON. You can see a **sample invalid JSON** under this link**:** https://drive.google.com/file/d/1UKdgCMSTcY8oduWGaDCxrZQ7AjmzNUgP/view?usp=sharing	"JSON in the **FILE_NAME** was not in a valid format. JSON body:" is printed to the console in red, followed by the contents of the JSON file. (**FILE_NAME** should be replaced with the file name entered by the user.) "Sorry! The application has experienced an unexpected error and will have to be closed." is printed. This situation should trigger an exception, which will be logged as a new entry is added to the log file. See **Logging exceptions**.

Printing video games

Scenario	User action	Result
Sunny day	At least one game was loaded from the JSON file.	“Loaded games are:” is printed, and then, games are printed each in a separate line. The title, release day, and rating are included.
Example		Stardew Valley, released in 2016, rating: 4.9 Frostpunk, released in 2017, rating: 4.7 Oxygen Not Included, released in 2017, rating: 4.8 Red Dead Redemtpion II, released in 2018, rating: 4.8 Portal 2, released in 2011, rating: 4.8
Video games collection is empty	The file was valid, but it contained an empty collection of objects.	“No games are present in the input file.” is printed to the console.

Logging exceptions in a file

Once the application experiences an unhandled exception for the first time, it should create a **log.txt** file that will store information about all unhandled exceptions.

This should specifically include an exception thrown when we try to deserialize a file containing invalid JSON, but also all other exceptions that are not handled otherwise.

New entries should **never override** the old ones - they should be **appended** to the file.

Each entry should contain:

- Date and time when the exception happened
- The exception message
- The stack trace

The exact format of the log file is at the developer's discretion.

An **example** of the log file contents could be :

[12/1/2022 8:00:13 AM], Exception message:**EXCEPTION_MESSAGE_1**, Stack trace: **STACK_TRACE_1**

[12/1/2022 8:00:15 AM], Exception message:**EXCEPTION_MESSAGE_2**, Stack trace: **STACK_TRACE_2**

You can also find an example of a logged file under this link:
https://drive.google.com/file/d/16hpQik8gATemu13iVIypqaMn1h_2BCWK/view?usp=sharing