LaSalle College
Vancouver

VGP 232 Game Tools and Pipeline

Assignment 2b

Console Application with json and xml serialization

Objective:

Using your existing assignment 2 **C# Console Application**, extend the save and load functionality from the CSV file to also support XML file and JSON file.

Implementation:

Add the following interfaces
IXmlSerializable (IXmlSerializable.cs)
- bool LoadXML(string path)
- bool SaveAsXML(string path)

IJsonSerializable (IJsonSerializable.cs)
- bool LoadJSON(string path)
- bool SaveAsJSON(string path)

ICsvSerializable (ICsvSerializable.cs)
- bool LoadCSV(string path)
- bool SaveAsCSV(string path)

The WeaponCollection should now implemented IXmlSerializable, IJsonSerializable, and ICsvSerializable which will require you to implement the following functions to your WeaponCollection class

- LoadXML() - Deserialize the XML file to the WeaponCollection i.e. a collection of weapons
- SaveAsXML() - Serialize WeaponCollection to an XML file
- LoadJSON() - Deserialize the JSON file to the WeaponCollection
- SaveAsJSON() - Serialize WeaponCollection to a JSON file
- LoadCSV() - move your existing Load implementation from Assignment 2 to here
- SaveAsCSV() - move your existing Save implementation from Assignment 2 to here
- Save() - checks for the file extension to determine which Save<Extension> function to call
- Load() - checks for the file extension to determine which Load<Extension> function to call

Revised: 10/22/2020

VGP 232 Game Tools and Pipeline

The Load and Save method will determine the type based on the extension of the file using System.IO.Path.GetExtension(filename) and will throws an exception or an error if the extension type is not supported

Unit Test:

Update the nUnit tests with the new changes to WeaponCollection (**15 additional tests**)
Please note since Load and Save have return values, make sure each test you also validate that if it succeeds, it should return true and false if it fails.

Test LoadJson Valid

1. **WeaponCollection_Load_Save_Load_ValidJson** - Load the data2.csv and **Save**() it to *weapons.json* and call **Load**() output and validate that there's 95 entries
2. **WeaponCollection_Load_SaveAsJSON_Load_ValidJson** - Load the data2.csv and **SaveAsJSON**() it to weapons.json and call **Load**() output and validate that there's 95 entries
3. **WeaponCollection_Load_SaveAsJSON_LoadJSON_ValidJson** - Load the data2.csv and **SaveAsJSON**() it to weapons.json and call **LoadJSON**() on output and validate that there's 95 entries
4. **WeaponCollection_Load_Save_LoadJSON_ValidJson** - Load the data2.csv and **Save**() it to weapons.json and call **LoadJSON**() on output and validate that there's 95 entries

Test LoadCsv Valid

1. **WeaponCollection_Load_Save_Load_ValidCsv** - Load the data2.csv and **Save**() it to *weapons.csv* and **Load**() output and validate that there's 95 entries
2. **WeaponCollection_Load_SaveAsCSV_LoadCSV_ValidCsv** - Load the data2.csv and **SaveAsCSV**() it to weapons.csv and **LoadCsv**() output and validate that there's 95 entries

Test LoadXML Valid

1. **WeaponCollection_Load_Save_Load_ValidXml** - Load the data2.csv and **Save**() it to weapons.xml and **Load**() output and validate that there's 95 entries
2. **WeaponCollection_Load_SaveAsXML_LoadXML_ValidXml** - Load the data2.csv and **SaveAsXML**() it to weapons.xml and **LoadXML**() output and validate that there's 95 entries

Test SaveAsJSON Empty

VGP 232 Game Tools and Pipeline

1. **WeaponCollection_SaveEmpty_Load_ValidJson** - Create an **empty** WeaponCollection, call **SaveAsJSON**() to empty.json, and **Load**() the output and verify the WeaponCollection has a Count of 0

Test SaveAsCSV Empty

1. **WeaponCollection_SaveEmpty_Load_ValidCsv** - Create an **empty** WeaponCollection, call **SaveAsCSV**() to empty.csv, and **Load**() the output and verify the WeaponCollection has a Count of 0

Test SaveAsXML Empty

1. **WeaponCollection_SaveEmpty_Load_ValidXml** - Create an **empty** WeaponCollection, call **SaveAsXML**() to empty.xml, and **Load** and verify the WeaponCollection has a Count of 0

Test Load InvalidFormat

1. **WeaponCollection_Load_SaveJSON_LoadXML_InvalidXml** - Load the data2.csv and **SaveAsJSON**() it to weapons.json and call **LoadXML**() output and validate that it returns false, and there's 0 entries
2. **WeaponCollection_Load_SaveXML_LoadJSON_InvalidJson** - Load the data2.csv and **SaveAsXML**() it to weapons.xml and call **LoadJSON**() output and validate that it returns false, and there's 0 entries
3. **WeaponCollection_ValidCsv_LoadXML_InvalidXml** - **LoadXML()** on the data2.csv and validate that returns false, and there's 0 entries
4. **WeaponCollection_ValidCsv_LoadJSON_InvalidJson** - **LoadJSON()** on the data2.csv and validate that Load returns false, and there's 0 entries

Note: the **[TearDown]** should **delete** the **output files** i.e. weapons.csv, weapons.json, weapons.xml, empty.json, empty.csv, and empty.xml.

Error Handling:
- invalid path (path does not exist)
- invalid arguments
- invalid data from the csv, json, xml

C# Helpful links

https://support.microsoft.com/en-us/help/816149/how-to-read-from-and-write-to-a-text-file-by-using-visual-c

Revised: 10/22/2020

LaSalle College
Vancouver

VGP 232 Game Tools and Pipeline

Due date

Next class, week 4 (at the start of class 6:30pm)

Submission

Implement the answers in a new project with the name "**Assignment2b**". This project should be submitted through committing and pushing through GIT to your VGP232 repository which you shared with the instructor.

Grading

Marks: Out of 100 (10% of final grade)

(60) Functional: Does it compile? Does it meet the requirements and work? Does it give the correct results?

(25) Error Handling: Does you application handle bad input? If so, does it handle failures and exceptions gracefully or does it crash?

(15) Naming convention & Comments: Does it follow the coding standards? Are your variables and method names descriptive? Did you leave descriptive comments on methods that does not have obvious functionality?

Coding Standard

C#
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

Late Assignments

After the due date, the student will no longer be able to submit their assignment and will receive a 0 as I will go over the solution in the next class.

Revised: 10/22/2020