GDAPS1 – Practice Exercise

Binary File IO

# Objective

Practice reading and writing binary files and exploring the differences between working with binary files and text files.

# Details

You’re going to alter the text file exercise from last class so that it reads and writes a binary file instead. You'll then get information about the file and display it to the user. Start by making a **copy of your files** from the last exercise.

## Altering PlayerManager

Make the following changes to saving and loading within the PlayerManager class.

**public void Save( )**

Save the player data to a binary file named “players.data”. Binary files don’t really have lines, so the first data written to the file should be the total number of players you’ll be writing to the file. Then write each individual piece of data from each player. Be sure to catch any potential exceptions and report errors to the user.

**Note:** Do **NOT** just write out one long string to a binary file. Yes, it’s possible. But that’s not the point of this exercise.

**public void Load( )**

Load the player data from the players.data file. Read the number of players in the file, then loop that many times. Read each player’s data and add that data to a new Player object, then add the player to the dictionary. Catch any exceptions and report them to the user.

Since you’re no lo**n**ger writing one long string here, you won’t need to split or parse anything. Instead, you must use the specific read methods that correspond to the data you expect from the file.

## Main Method

At this point, ensure the old commands work with your new saving and loading.

## File Information

Add one more command for the user: “Info”. If the user types “Info”, print information about the “players.data” file itself, such as the creation date/time and the last date/time the file was accessed. If the file doesn’t exist yet (which you can check programmatically), tell the user they’ll need to save it first.

You’ll find appropriate methods for these tasks inside the static **File** class that comes with C#. The methods returns DateTime objects, which can be printed themselves or concatenated with strings.

# Sample Run

The bulk of the output should be identical as last exercise, with the exception of the new "info" command. Test the program like last time: create several players and save the data using the previous commands. Then test the “info” command. Restart the program and load the file to ensure all of the data remains intact.

Choose one of the following options:

Create. Print. Save. Load. Info. Quit. >> **Info**

File created: 10/23/2019 8:32:07 AM

Last modified: 11/01/2019 9:12:18 AM

# Submission

All of your work must be commented and follow this course’s coding standards. **Read through the Coding Standards document (located in MyCourses) to check over your code before you complete your program. Make sure you follow the coding standards for all code you create.**

1) Submit: Submit your program to the appropriate Assignments dropbox in MyCourses.

2) Check-off: Show your working program to the instructor or TA. If you do not finish before class ends, complete the exercise for homework and show one of us in-class on the next class period. If your program works as expected, you will be “checked off” to earn credit for the exercise.