GDAPS1 – Practice Exercise

Text File IO

# *Objective*

Practice reading and writing to text files.

# *Details*

Writing files to the hard drive is one of the easiest ways to achieve data persistence. You’re going to practice reading and writing text files by making a player manager.

## Player Class

Create a simple Player class with a name and at least 2 randomly-generated numeric stats. Feel free to use whatever stats you want. Create a parameterized constructor, an overridden ToString() method, and properties where necessary.

## PlayerManager Class

This class will keep track of multiple players in a List<Player>. Implement the following methods:

**public void CreatePlayer(String name)**

Create a new Player object with the specified name and randomly generated stats. Add it to the manager’s list.

**public void Print( )**

Print all of the players using their ToString() method. If there are no players yet, print an error message suggesting that the user loads player data from a file.

**public void Save( )**

Save the player data to a text file named “players.txt”. There should be one player per line. The player’s individual data (name and stats) should all be separated by commas. Be sure to catch any potential file-writing exceptions and report errors to the user.

**public void Load( )**

Clear the list of players, then load new player data from the “players.txt” file. For each line of the file, you’ll need to use the String class’s Split() method to convert the line into an array of data. Parse and add that data to a new Player object, then add the player to the list. Catch any potential file-reading exceptions and report them to the user.

## Main Method

Create a PlayerManager object, then loop and get input from the user.

Handle the following commands:

* **Create** – Get a name from the user and add the player to the manager’s list
* **Print** – Print all the players in the manager
* **Save** – Saves all the players
* **Load** – Loads players from the save file
* **Quit** – Ends the program

To finish testing your program, create several players and save the data. Close your program, then open the text file you’ve generated in Notepad to ensure data was written correctly. (It can be found inside the bin\Debug folder, which is where the compiled executable program is placed.)

Now restart the program (re-run it from Visual Studio), then load the file to ensure you’re able to read it all back in.

# *Sample Run*

Choose one of the following options:

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

There are no players yet. You might want to load them from a file.

Choose one of the following options:

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

There is no player data yet.

Choose one of the following options:

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

Loading data from players.txt…

Added Robin to the list.

Added Lily to the list.

Added Marshall to the list.

Loaded all data from file. Players created.

Choose one of the following options:

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

Player: Robin. Level 100, Performance skill 31.

Player: Lily. Level 119, Performance skill 35.

Player: Marshall. Level 82, Performance skill 50.

Choose one of the following options:

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

What is the player’s name? **Bobby**

Added Bobby to the list.

Choose one of the following options:

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

Saved players to file players.txt

Choose one of the following options:

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

Loading data from players.txt…

Added Robin to the list.

Added Lily to the list.

Added Marshall to the list.

Added Bobby to the list.

Loaded all data from file. Players created.

Choose one of the following options:

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

Player: Robin. Level 100, Performance skill 31.

Player: Lily. Level 119, Performance skill 35.

Player: Marshall. Level 82, Performance skill 50.

Player: Bobby. Level 94, Performance skill 41.

Choose one of the following options:

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

Goodbye!

# *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.