CODING ACTIVITY: SKI RUN RATER S1 (THE REPOSITORY PATTERN)

OVERVIEW

The focus of this activity is to provide the student experience with the Repository Pattern. The student will modify an application that writes to a text file in comma separated value format (csv) to complete the required CRUD and query functionality.

OUTCOMES

- The student will demonstrate implementing the MVC pattern in an application.
- The student will demonstrate implementing the Repository pattern.
- The student will demonstrate managing CRUD operations with a CSV text file.

NOTES AND RESOURCES

- Students may use all resources available.

INSTRUCTIONS

1. Download the ***SkiRunRater.Sprint1.Starter*** code from **Course > Resources**.
2. Run the application to confirm its functionality.
3. Explore the application. Be sure to note the following items.
    a. Models
        i. AppEnum.cs – general class to contain all public enumerables
        ii. SkiRun.cs – entity class for the domain
    b. Utilities
        i. ConsoleUtil.cs – helper class to improve console output
    c. Data
        i. Data.txt – data file
        ii. DataSettings.cs – stores path to data file
        iii. InitializeDataFile.cs – creates initial data objects and writes them to the data file
    d. DAL (Data Access Layer)
        i. SkiRunRepository.cs – repository class to manage CRUD operations using a CSV text file
    e. View (Presentation Layer)
        i. ViewSettings.cs – class to hold the default console settings
        ii. ConsoleView.cs – MVC view class
            1. GetUserActionChoice method
            2. DisplayAllSkiRuns method
    f. Controller (Logic Layer)
        i. Controller.cs
            1. Constructor
            2. ApplicationControl method

4. Create a ***ReadMe.txt*** file and include the following information.
    a. Title: Ski Run Rater
    b. Description: (add your description)
    c. Application Type: Console
    d. Author: (add your name)
    e. Dated Created: (current date)
    f. Last Modified:
5. Add functionality to the application.
    a. Add the following items to the ***GetUserActionChoice*** method in the ***ConsoleView*** class. Note, the numbers in parenthesis represents a suggested order to complete the assignment.
        i. **List All Ski Runs** (1)
        ii. **Display a Ski Run Detail** (5)
        iii. **Delete a Ski Run** (3)
        iv. **Add a Ski Run** (4)
        v. **Update a Ski Run** (6)
        vi. **Query Ski Runs by Vertical** (7)
        vii. **Quit** (2)
    b. For each menu item (See Example in Step iv.)
        i. Write a corresponding method in the ***ConsoleView*** class to implement the functionality.
        ii. Note that some ***ConsoleView*** class methods will need to return values to the controller for the controller to perform the correct action on the repository object.
        iii. Complete all case blocks in the ***ApplicationControl*** method in the ***Controller*** class.
        iv. **Example: Delete a Ski Run**
            1. Add the **GetSkiRunID** method to the ***ConsoleView*** class. Note and explore the **ValidateIntegerResponse** method.

```
/// <summary>
/// method to get the user's choice of ski run id
/// </summary>
/// <param name="skiRuns">list of all ski runs</param>
/// <returns>id of user selected ski run</returns>
public static int GetSkiRunID(List<SkiRun> skiRuns)
{
    int skiRunID = -1;

    DisplayAllSkiRuns(skiRuns);

    DisplayMessage("");
    DisplayPromptMessage("Enter the ski run ID: ");

    skiRunID = ConsoleUtil.ValidateIntegerResponse("Please enter the ski run ID: ",
      Console.ReadLine());

    return skiRunID;
}
```

2. Add the following variables to the **ApplicationControl** method in the ***Controller*** class.

```
while (active)
{
    AppEnum.ManagerAction userActionChoice;

    int skiRunID;
    SkiRun skiRun;
    string message;

    userActionChoice = ConsoleView.GetUserActionChoice();
```

3. Modify the menu of the **GetUserActionChoice** method in the ***ConsoleView*** class.

```
//
// display the menu
//
DisplayMessage("Ski Manager Menu");
DisplayMessage("");
Console.WriteLine(
    leftTab + "1. Display All Ski Runs Information" + Environment.NewLine +
    leftTab + "2. Delete a Ski Run" + Environment.NewLine +
    leftTab + "E. Exit" + Environment.NewLine);
```

4. Test the application to confirm the functionality.

SUBMIT FOR GRADE.

1. Submit to Moodle.
    a. Zip the solution folder into a single file.
    b. Click the ***Coding Activity – Ski Run Rater S1 (The Repository Pattern)*** assignment link.
    c. Upload the zipped file and add any additional files
    d. Click Save Changes.
2. After receiving a grade, refer to Moodle to review the graded rubric and additional comments.

MARKING GUIDE – SKI RUN RATER SPRINT 1

Author _____ **Reviewer(s)** _____

Functionality	Points	Score
List All Ski Runs **Quit**	**5 Points Each**	
Display a Ski Run Detail **Delete a Ski Run** **Add a Ski Run**	**10 Points Each**	
Update a Ski Run **Query Ski Runs by Vertical**	**5 Points Each**	
Total Points	**50**	