DEMO: THE TARDIS PROJECT (SPRINT 4 – INTERFACES)

1. Prepare the **Traveler** class and object for battle
    a. Add the **Health**, **Lives**, and **BattleIndex** properties to the **Traveler** class.

```
#region FIELDS

private List<Item> _travelersItems;
private List<Treasure> _travelersTreasures;
private int _health;
private int _lives;
private int _battleIndex;

#endregion
```

    b. Initialize these property values in the **InitializeMission** method of the **Controller** class.
    Note: The **BattleIndex** is a value ranging between 1 and 100 and represents their skill level. The higher the number, the more likely they are to win a battle over an opponent with a lower **BattleIndex** value. Given that battles are calculated in a weighted random fashion, it is possible for someone with a lower **BattleIndex** value to beat and opponent with a higher **BattleIndex** value, just not as often.

```
//
// set the traveler's initial status
//
_gameTraveler.BattleIndex = 75;
_gameTraveler.Health = 100;
_gameTraveler.Lives = 1;
```

    c. Add a **NoLives** method to the **Traveler** class.

```
73          /// <summary>
74          /// determine if all lives are gone
75          /// </summary>
76          /// <returns>true if no lives left</returns>
77          public bool NoLives()
78          {
79              return (_lives < 1);
80          }
```

2. Prepare the game for battle.
    a. Create a file ***BattleEnums.cs*** and add the **BattleAction** and **BattleResult** enums. (refer to printout)
    b. Add the **Battle** and **ProcessBattle** methods to the **Controller** class. (refer to printout)
    c. Add the **DisplayGetBattleActionChoice** method to the **ConsoleView** class. (refer to printout)
    d. Update the menu.
        i. Add a **DisplayGameStatus** method to the **ConsoleView** class.. (refer to printout)
        ii. Update the **DisplayGetTravelersActionChoice** method in the **ConsoleView** class to include "Battle" as a choice. (refer to printout)
    e. Update the game loop.
        i. Add **Battle** to the **TravelerAction** enum.
        ii. Modify the **ManageGameLoop** method in the **Controller** class to handle the **Battle TravelerAction** and call the **DisplayBattleResults** method from the **ConsoleView** class.

```
            break;
        case TravelerAction.Travel:
            _gameTraveler.SpaceTimeLocationID =
              _gameConsoleView.DisplayGetTravelersNewDestinat
            break;
        case TravelerAction.Battle:
            _gameConsoleView.DisplayBattleResults(Battle());
            break;
        case TravelerAction.TravelerInfo:
```

        iii. Add a **DisplayOutOfLives** method in the **ConsoleView** class**.**

```
/// <summary>
/// display message indicating the traveler is out of lives
/// </summary>
0 references | 0 changes | 0 authors, 0 changes
public void DisplayOutOfLives()
{
    ConsoleUtil.HeaderText = "Exit";
    ConsoleUtil.DisplayReset();

    Console.CursorVisible = false;

    Console.WriteLine();
    ConsoleUtil.DisplayMessage("It appears that you are out of
       lives in for this game. Pleaser return and play
      again.");

    DisplayContinuePrompt();
}
```

        iv. Modify the **UpdateGameStatus** method to monitor the traveler's current number of lives and, if lives are 0, call the **DisplayOutOfLives** and **DisplayExitPrompt** methods in the **ConsoleView** class.

```
/// <summary>
/// part of the game loop to facilitate game maintenance
/// </summary>
1 reference | John Velis, 15 hours ago | 1 author, 1 change
private void UpdateGameStatus()
{
    //
    // check for out of lives
    //
    if (_gameTraveler.NoLives())
    {
        _gameConsoleView.DisplayOutOfLives();
        _gameConsoleView.DisplayExitPrompt();
    }
}
```

3. Prepare the Daleks for battle.
    a. Create a file **IBattle.cs** and develop the **IBattle** interface. (refer to printout)
    b. Implement the **IBattle** interface with the **Dalek** class.
    c. Set the **IBattle** properties for each Dalek object.
    Note: The **AggressionIndex** is a percentage and represents how often a Dalek will choose to attack as opposed to retreat. The higher the percentage, the more aggressive they are and the more likely they are to attack. The **BattleIndex** is a value ranging between 1 and 100 and represents their skill level. The higher the number, the more likely they are to win a battle over an opponent with a lower **BattleIndex** value. Given that battles are calculated in a weighted random fashion, it is possible for someone with a lower **BattleIndex** value to beat and opponent with a higher **BattleIndex** value, just not as often.

```
/// <summary>
/// initialize the universe with all of the treasures
/// </summary>
1 reference | John Velis, 13 hours ago | 1 author, 1 change
private void IntializeUniverseDaleks()
{
    Daleks.Add(new Dalek
    {
        Name = "Thordan",
        CharacterID = 1,
        Race = Character.RaceType.Dalek,
        SpaceTimeLocationID = 2,
        HasMessage = true,
        Message = "You must leave the Lodestone in the Eelandrian Plains.",
        AggressionIndex = 100,
        BattleIndex = 50
    });

    Daleks.Add(new Dalek
    {
        Name = "Rolan",
        CharacterID = 2,
        Race = Character.RaceType.Dalek,
        SpaceTimeLocationID = 2,
        HasMessage = false,
        Message = "",
        AggressionIndex = 0,
        BattleIndex = 50
    });
}
```

4. Test the application to be sure that the battle functionality works.