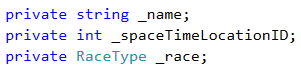
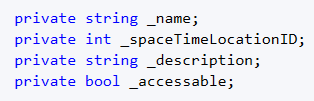
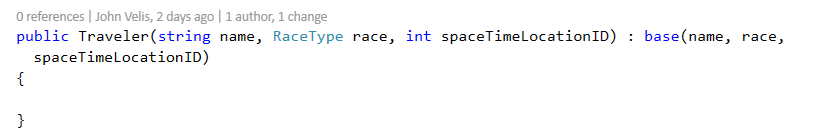
## The TARDIS Project Demonstration

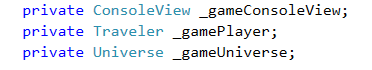
### Sprint 1

1. Discuss and explore the ConsoleUtil class.
2. Models
   1. Character.cs
      1. Enum RaceType
      2. 
      3. Add Constructor with 3 parameters
   2. SpaceTimeLocations.cs
      1. 
   3. Traveler.cs
      1. Add Constructor with 3 parameters and reference to base class

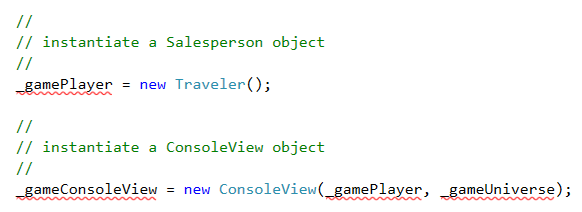


* 1. TravelerAction.cs
     1. Explore
  2. Universe.cs
     1. Add list of SpaceTimeLocations with auto implemented property
     2. Add constructor and initialize the list
     3. Note GetSpaceTimeLocationByID() method

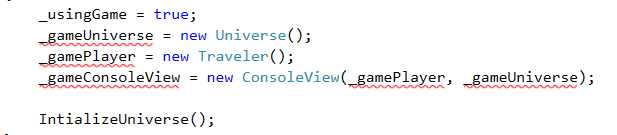
1. Program
   1. Instantiate a Controller object
2. Controller
   1. Note bool fields: \_usingGame and \_missionInitialized
   2. Add objects fields



* 1. Complete the constructor



* 1. Complete InitializeGame



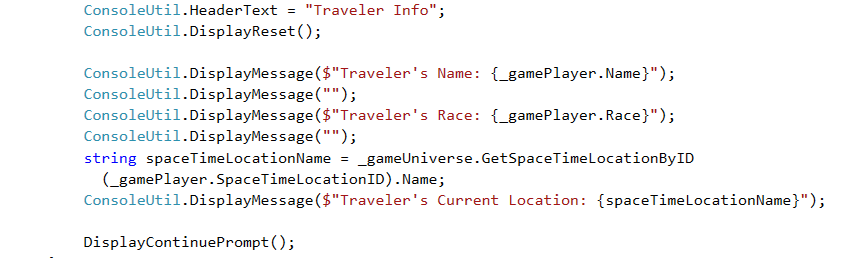
* 1. Explore InitializeUniverse() method
  2. Explore ManageGameLoop and DisplayGetTravelerActionChoice
  3. InitializeMission
     1. DisplayMissionSetupIntro()
     2. DisplayGetTravelersName()
     3. DisplayGetTravelersRace()
     4. DisplayGetTravelersNewDestination()
        1. Universe.GetSpaceTimeLocationByID
           1. Exception thrown

1. ConsoleView
   1. Add object fields



* 1. Initialize object fields in constructor
  2. Update InitializeConsole

1. Start coding out menu
   1. Mission Setup
      1. DisplayMissionSetupIntro()
      2. DisplayGetTravelersName()
      3. DisplayGetTravelersRace()
      4. DisplayGetTravelersNewDestination()
         1. Universe.GetSpaceTimeLocationByID
            1. Exception thrown
      5. DisplayMissionConfirmation()
   2. DisplayTravelerInfo()



* 1. DisplayListAllTARDISDestinations()
  2. DisplayLookAround()

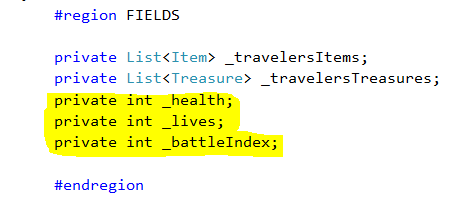


### Sprint 2

1. Create the Object abstract class.
2. Create the Jewel class and inherit from the Object class.
3. Create the Furniture class and inherit from the Object class.
4. Create the GallifrianMirror class and inherit form the Object class.
5. Ff

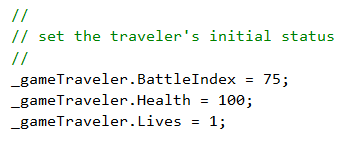
### Demo: The TARDIS Project (Sprint 4 – Interfaces)

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

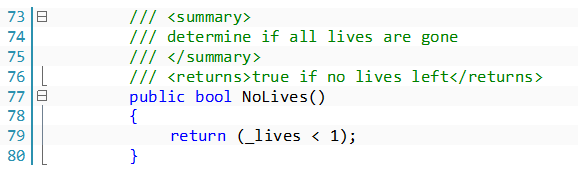


* 1. Initialize these property values in the **InitializeMission** method of the **Controller** class.

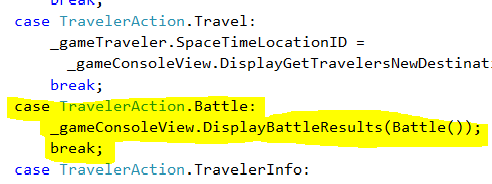
Note: The **BattleIndex** is a value ranging between 1 and 100 and represents their skill level. The higher the number, the more likely they are to win a battle over an opponent with a lower **BattleIndex** value. Given that battles are calculated in a weighted random fashion, it is possible for someone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



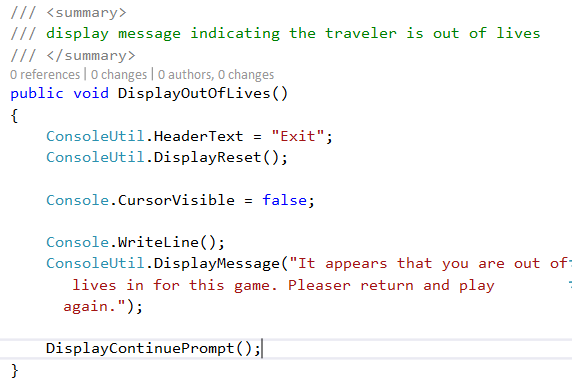
* 1. Add a **NoLives** method to the **Traveler** class.



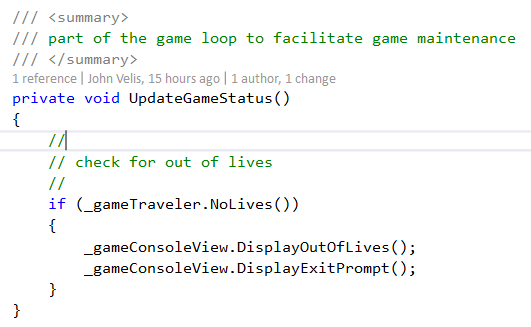
1. Prepare the game for battle.
   1. Create a file ***BattleEnums.cs*** and add the **BattleAction** and **BattleResult** enums. (refer to printout)
   2. Add the **Battle** and **ProcessBattle** methods to the **Controller** class. (refer to printout)
   3. Add the **DisplayGetBattleActionChoice** method to the **ConsoleView** class. (refer to printout)
   4. Update the menu.
      1. Add a **DisplayGameStatus** method to the **ConsoleView** class.. (refer to printout)
      2. Update the **DisplayGetTravelersActionChoice** method in the **ConsoleView** class to include “Battle” as a choice. (refer to printout)
   5. Update the game loop.
      1. Add **Battle** to the **TravelerAction** enum.
      2. Modify the **ManageGameLoop** method in the **Controller** class to handle the **Battle** **TravelerAction** and call the **DisplayBattleResults** method from the **ConsoleView** class.



* + 1. Add a **DisplayOutOfLives** method in the **ConsoleView** class**.**

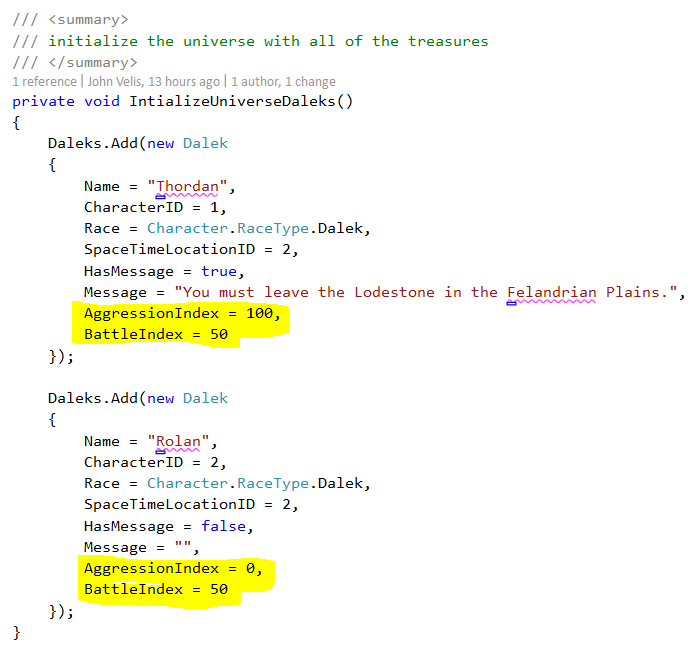


* + 1. Modify the **UpdateGameStatus** method to monitor the traveler’s current number of lives and, if lives are 0, call the **DisplayOutOfLives** and **DisplayExitPrompt** methods in the **ConsoleView** class.



1. Prepare the Daleks for battle.
2. Create a file **IBattle.cs** and develop the **IBattle** interface. (refer to printout)
3. Implement the **IBattle** interface with the **Dalek** class.
4. 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.



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