Wyższa Szkoła Bankowa

## Programowanie Obiektowe Sprawozdanie do ćwiczenia nr 2

Imię i nazwisko: Mykhailo Zaliznyi

Adres email: <u>zaliznyimh@gmail.com</u>

Numer albumu: 144388

Data: 27.08.2023

Link do repo na Github:

https://github.com/zaliznyimh/ProgramowanieObiektoweZadanie2

## Rozdział 1 – Zadania do zrealizowania

1) Tworzymy klasę **ScreenLineEntry** z następującymi polami:

```
Epublic class ScreenLineEntry

{

/// <summary>
/// Properties for lines which will be outputted to the console
/// Backgroundcolor
/// ForegroundColor
/// Text
/// </summary>

CCLINOK: 1
public ConsoleColor BackgroundColor { get; set; }

CCLINOK: 1
public ConsoleColor ForegroundColor { get; set; }

CCLINOK: 1
public string? Text { get; set; }

}
```

2) Tworzymy klasę **ScreenDefinition** z listem typu **ScreenLineEntry**:

3) Dodajemy klasę **ScreenLineHelper** w którą będzie Dictionary.

```
Ссылок: 4

= public class ScreenLineHelper

{

/// <summary>
/// Dictionary which contain value and lines
/// </summary>

Ссылок: 4
public Dictionary<ScreenLineEnum, ScreenDefinition>? ScreenLine { get; set;}
```

4) W klasę **ScreenDefinitionService** dodałem 2 metody **ScreenLineHelper Load()**, **ShowLines()** 

```
/// <summary>
/// Method which reads property from JSON file

/// </summary>
/// <param name="jsonFileName"></param>

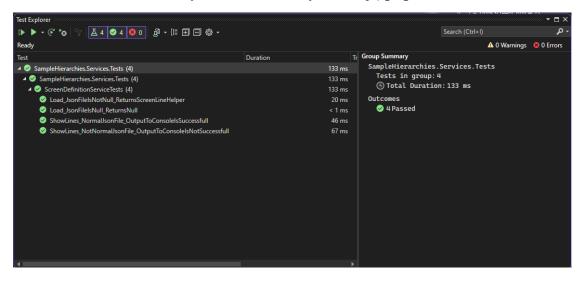
Ccbs/Ock 3
public static ScreenLineHelper? Load(string jsonFileName)
{
    try
    {
        string jsonString = File.ReadAllText(jsonFileName);
        ScreenLineHelper? screenLineHelper = JsonConvert.DeserializeObject<ScreenLineHelper>(jsonString);
    return screenLineHelper;
    }
    catch (Exception ex)
    {
        Console.WriteLine("There is an error during reading json file." + ex.Message);
        return null;
    }
}
```

```
public string? ShowLines(string jsonFileName, ScreenLineEnum property, int lineID)

__screenLineHelper = Load(jsonFileName);

if (_screenLineHelper != null && _screenLineHelper.ScreenLine != null) {
        Console.BackgroundColor = _screenLineHelper.ScreenLine[property].LineEntries[lineID].BackgroundColor;
        Console.ForegroundColor = _screenLineHelper.ScreenLine[property].LineEntries[lineID].ForegroundColor;
        Console.WriteLine(_screenLineHelper.ScreenLine[property].LineEntries[lineID].Text);
        Console.ResetColor();
        return "Ok";
    }
    else
    {
        Console.WriteLine("Your JSON file is null here.");
        return null;
    }
}
```

5) Dalej dodałem nowy project o nazwie **SampleHierarchies.Services.Tests**Testy z klasy **ScreenDefinitionServiceTests**(Wszystkie Unit Testy są w projekcie)
Również można zobaczyć, że Unit Testy działają poprawnie.



```
[TestMethod]

    ∅ | 0 references

public void Load_JsonFileIsNotNull_ReturnsScreenLineHelper()
    // Arrange
    var screenDefinitionService = new ScreenDefinitionService();
    var result = ScreenDefinitionService.Load(notNullJsonFileNameLoad);
    // Assert
    Assert.IsNotNull(result);
[TestMethod]
| 0 references
public void Load_JsonFileIsNull_ReturnsNull()
    // Arrange
    var screenDefinitionService = new ScreenDefinitionService();
    var result = ScreenDefinitionService.Load(isNullJsonFileNameLoad);
    // Assert
    Assert.IsNull(result);
```

6) Tworzymy pliki JSON które przechowują wszystkie linii dla każdego ekranu Na przykład, JSON file dla **MainScreen** wygląda tak. Dla każdej linii również jest ustawiona kolorystyka linii.

## Tak wygląda metoda Show() w klasie MainScreen

```
public override void Show()
{
    while (true)
    {
        Console.Clear();
        _screenDefinitionService.ShowLines(jsonFileName, ScreenLineEnum.Menu, 0); // Screen history: MainScreen
        _screenDefinitionService.ShowLines(jsonFileName, ScreenLineEnum.Show, 0); // Your available choices are:
        _screenDefinitionService.ShowLines(jsonFileName, ScreenLineEnum.Show, 1); // 0. Exit
        _screenDefinitionService.ShowLines(jsonFileName, ScreenLineEnum.Show, 2); // 1. Animals
        _screenDefinitionService.ShowLines(jsonFileName, ScreenLineEnum.Show, 3); // 2. Settings
        _screenDefinitionService.ShowLines(jsonFileName, ScreenLineEnum.Show, 4); // Please enter your choice:
```

Tak wygląda konsola dla ekranu MainScreen-a

```
Screen history: MainScreen

Your available choices are:

0. Exit

1. Animals

2. Settings

Please enter your choice:
```

Konsola AnimalsScreen-a

```
Screen history: MainScreen -> AnimalsScreen
Your available choices are:
0. Exit
1. Animals
2. Save to file
3. Read from file
Please enter your choice:
```

Konsola MammalsScreen-a

```
Screen history: MainScreen -> AnimalsScreen -> MammalsScreen

Your available choices are:

3. Exit

1. Dogs

2. Wolves(Wolfs)

3. Dolphins

4. Bengal Tiger

Please enter your choice:
```

8) Dla każdego ekranu istnieje własny JSON file z liniami

MainScreen - EnMainScreenLines.json,

AnimalsScreen - EnAnimalsScreenLines.json,

MammalsScreen - EnMammalsScreenLines.json,

DogsScreen - EnDogsScreenLines.json,

DolphinsScreen - EnDolphinsScreenLines.json,

BengalTigerScreen - EnTigersScreenLines.json,

WolfsScreen - EnWolfsScreenLines.json;

8) Historia ekranów również jest napisana w plikach JSON