Programowanie Obiektowe

Kyrylo Horiunov

kirillgorunov294@gmail.com

144135

28.08.2023

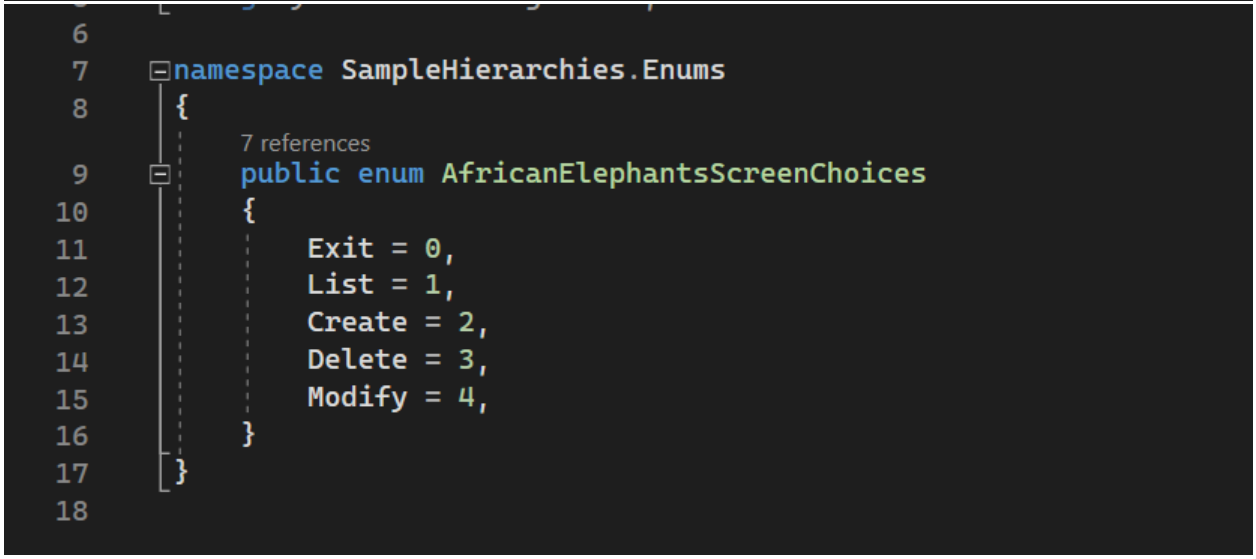
https://github.com/Myrzei4/OOP/tree/final

Rozdzial 1

Dodano:

- Klasy: AfricanElephant, GrizzlyBear, PolarBear, AfricanElephantScreen, GrizzlyBearScreen, PolarBearScreen, SettingsScreen, Settings
- Enumy: AfricanElephantScreenChoices, GrizzlyBearScreenChoices, PolarBearScreenChoices, SettingsScreenChoices
- Interfejsy: IAfricanElephant, IGrizzlyBear, IPolarBear

```
using SampleHierarchies.Interfaces.Data.Mammals;
using System.Xml.Linq;

namespace SampleHierarchies.Data.Mammals;

11 references
public class AfricanElephant  : MammalBase, IAfricanElephant
{


    #region Inteface implimentetion
    3 references
    public float Height { get; set; }

    3 references
    public float Weight { get; set; }

    3 references
    public float TuskLength { get; set; }

    3 references
    public int Lifespan { get; set; }

    3 references
    public string SocialBehavior { get; set; }

#endregion Inteface implimentetion
4 references
public override void Display()
{
    Console.WriteLine("My name is: {0} and I am: {1} years old. My height is {2} meters, my weight is {3} kilograms, my tusk is {4} meters long, I can live {5} years, in social I'm {6}", Name, Age, Height, Weight,TuskLength, Lifespan, SocialBehavior);
}
/// <summary>
///
/// </summary>
/// <param name="name"></param>
/// <param name="age"></param>
/// <param name="height"></param>
/// <param name="weight"></param>
/// <param name="tuskLenght"></param>
/// <param name="lifespan"></param>
/// <param name="socialBehavior"></param>
1 reference
public AfricanElephant(string name, int age ,float height, float weight, float tuskLenght, int lifespan, string socialBehavior) : base(name, age, MammalSpecies.AfricanElephant)
{
    Height = height;
    Weight = weight;
    TuskLength = tuskLenght;
    Lifespan = lifespan;
    SocialBehavior = socialBehavior;
}

}
```



```
6
7   namespace SampleHierarchies.Enums
8   {
        7 references
9       public enum AfricanElephantsScreenChoices
10      {
11          Exit = 0,
12          List = 1,
13          Create = 2,
14          Delete = 3,
15          Modify = 4,
16      }
17  }
18
```

```
try
{
    if (choiceAsString is null)
    {
        throw new ArgumentNullException(nameof(choiceAsString));
    }

    AfricanElephantsScreenChoices choice = (AfricanElephantsScreenChoices)Int32.Parse(choiceAsString);
    switch (choice)
    {
        case AfricanElephantsScreenChoices.List:
            Console.ResetColor();
            Console.Clear();
            ListAfricanElephants();
            break;

        case AfricanElephantsScreenChoices.Create:
            Console.ResetColor();
            Console.Clear();
            AddAfricanElephant();
            break;

        case AfricanElephantsScreenChoices.Delete:
            Console.ResetColor();
            Console.Clear();
            DeleteAfricanElephant();
            break;

        case AfricanElephantsScreenChoices.Modify:
            Console.ResetColor();
            Console.Clear();
            EditAfricanElephantMain();
            break;

        case AfricanElephantsScreenChoices.Exit:
            DisplayLine(8);
            Console.ResetColor();
            Console.Clear();
            return;
    }
```

```
// Validate choice
try
{
    if (choiceAsString is null)
    {
        throw new ArgumentNullException(nameof(choiceAsString));
    }
    SettingsScreenChoices choice = (SettingsScreenChoices)Int32.Parse(choiceAsString);
    switch(choice)
    {

        case SettingsScreenChoices.ChangeColors:
            _settingsService.ChangeColor();
            _settingsService.ApplyColors("settings.json", _settingsService.className);
            Console.Clear();

        break;

        case SettingsScreenChoices.ListOfColors:
            Console.ResetColor();
            Console.WriteLine();
            _settingsService.ListColors();
            _settingsService.ApplyColors("settings.json", _settingsService.className);
            break;

        case SettingsScreenChoices.ListOfScreens:
            Console.ResetColor();
            Console.WriteLine();
            _settingsService.ListScreens();

            break;

        case SettingsScreenChoices.Exit:
            DisplayLine(8);
            Console.ResetColor();
            Console.Clear();

        return;
    }
```

```
10 references
public interface ISettings
{
    #region Interface Members
    /// <summary>
    /// Color settings.
    /// </summary>

    9 references
    string BackgroundColor { get; set; } // Color of background
    9 references
    string ForegroundColor { get; set; } // Color of text

    #endregion // Interface Members
}
```

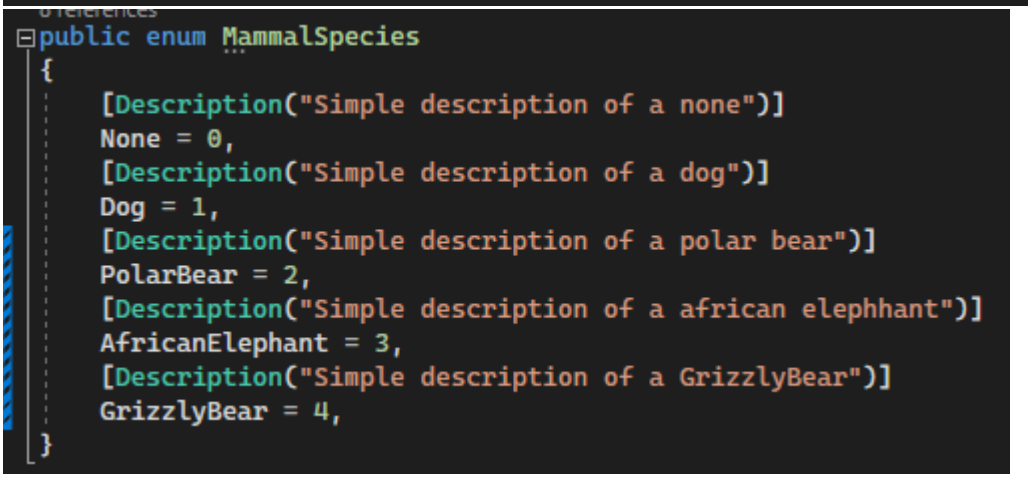
```
public interface IAfricanElephant : IMammal
{
    #region Interface Members
    /// <summary>
    /// Characteristics of Elephant
    /// </summary>
    ///

    float Height { get; set; }


    float Weight { get; set; }


    float TuskLength { get; set; }

    3 references
    int Lifespan { get; set; }

    3 references
    string SocialBehavior { get; set; }

    #endregion // Interface Members
}
```

```
namespace SampleHierarchies.Data
{
    4 references
    public class Settings : ISettings
    {
        #region ISettings Implementation
        9 references
        public string BackgroundColor { get; set; } = "black"; //BG color
        9 references
        public string ForegroundColor { get; set; } = "white"; //FG color
        #endregion //ISetings Implementation
    }
}
```

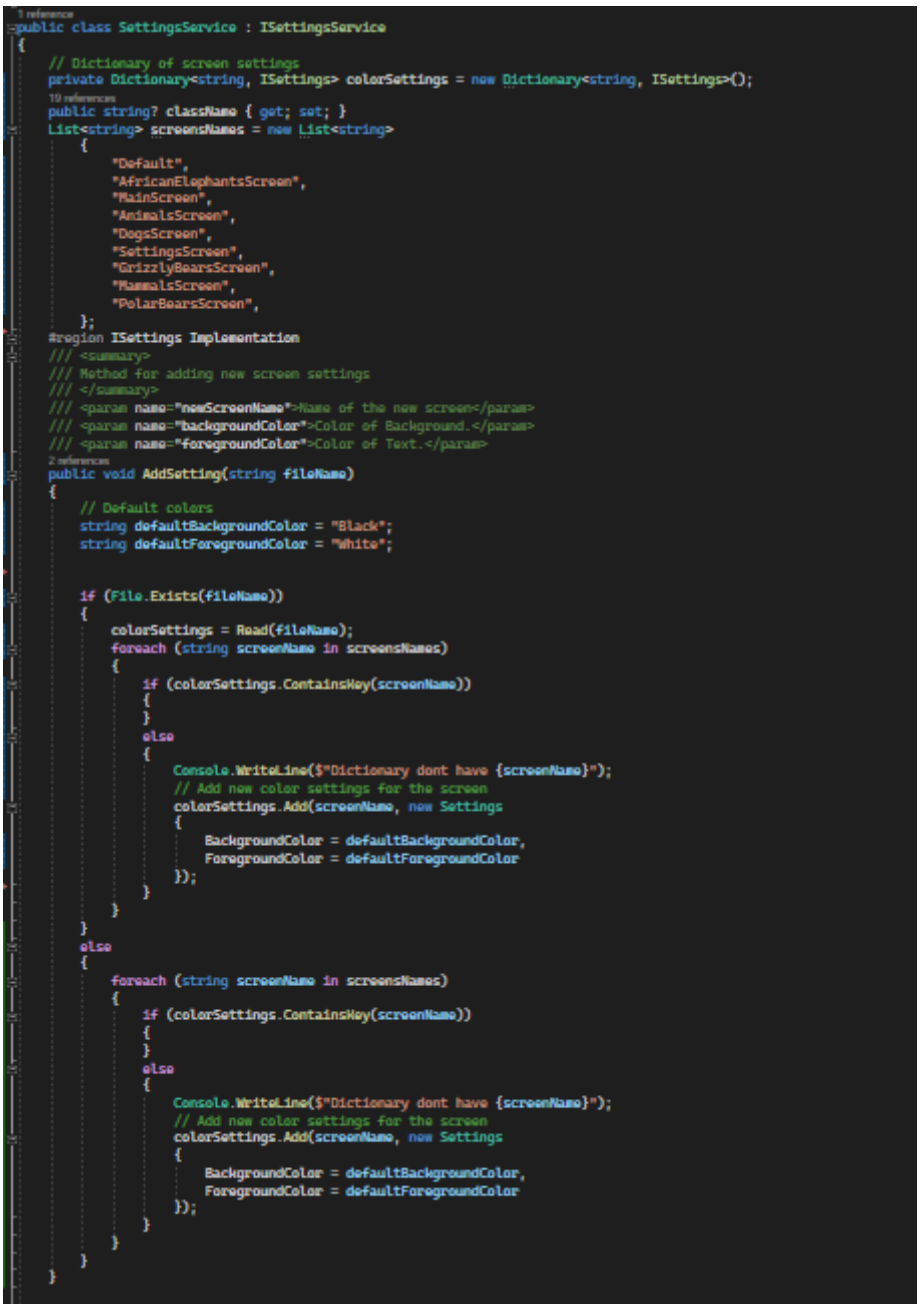
Zmodyfikowano:

- Klasy: Mammals, SettingsService
- Enumy: MammalSpecies, MammalsScreenChoice
- Interfejsy: ISettings, IMammals

```
2 references
public class Mammals : IMammals
{
    #region IMammals Implementation

    /// <inheritdoc/>
    9 references
    public List<IDog> Dogs { get; set; }

    9 references
    public List<IAfricanElephant> AfricanElephants { get; set; }
    9 references
    public List<IPolarBear> PolarBears { get; set; }
    9 references
    public List<IGrizzlyBear> GrizzlyBears { get; set; }
    #endregion // IMammals Implementation

    #region Ctors

    /// <summary>
    /// Default ctor.
    /// </summary>
    1 reference
    public Mammals()
    {
        Dogs = new List<IDog>();
        PolarBears = new List<IPolarBear>();
        AfricanElephants = new List<IAfricanElephant>();
        GrizzlyBears = new List<IGrizzlyBear>();
    }

    #endregion // Ctors
}
```



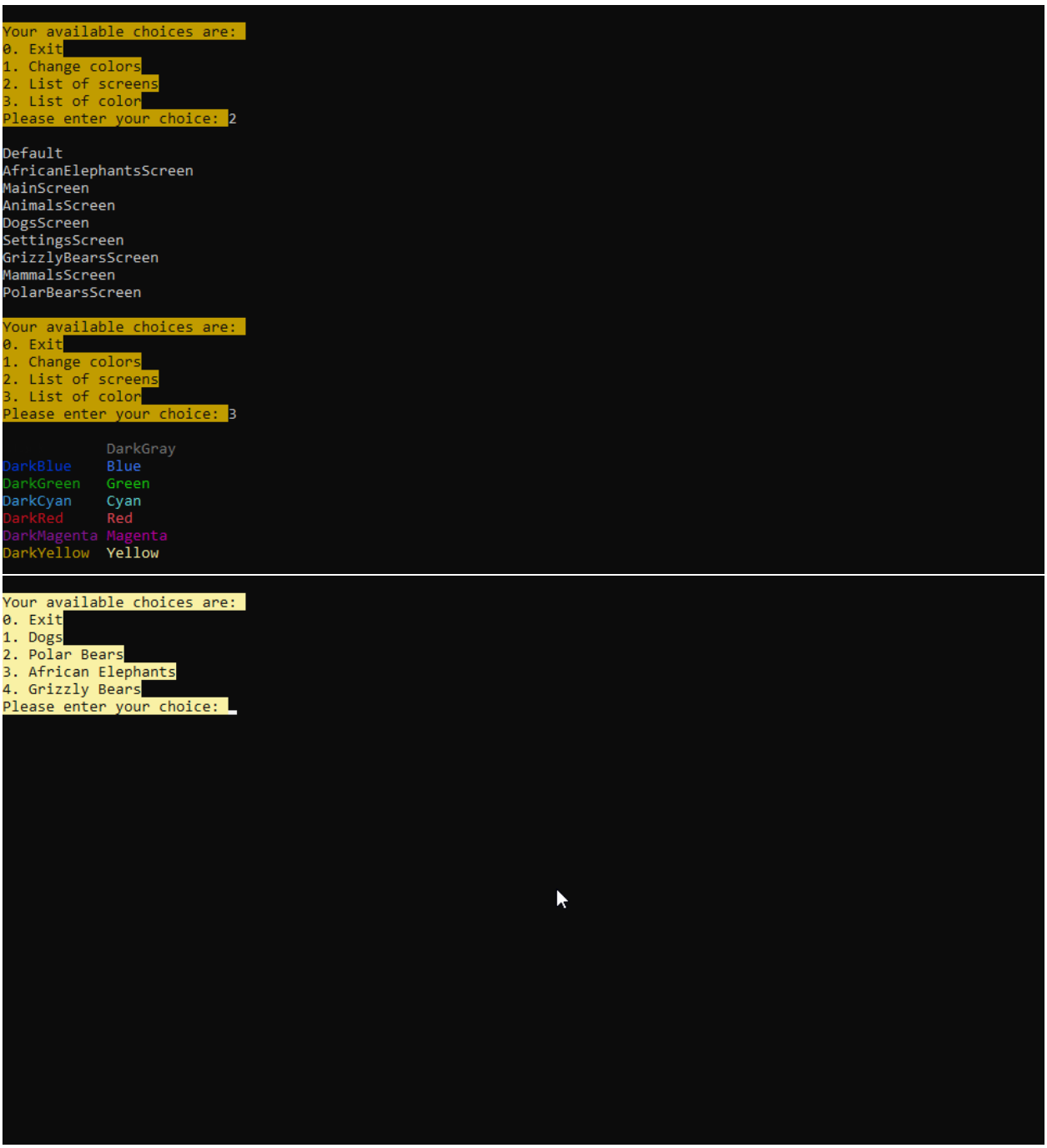
```
public enum MammalSpecies
{
    [Description("Simple description of a none")]
    None = 0,
    [Description("Simple description of a dog")]
    Dog = 1,
    [Description("Simple description of a polar bear")]
    PolarBear = 2,
    [Description("Simple description of a african elephhant")]
    AfricanElephant = 3,
    [Description("Simple description of a GrizzlyBear")]
    GrizzlyBear = 4,
}
```



```
1 reference
public class SettingsService : ISettingsService
{
    // Dictionary of screen settings
    private Dictionary<string, ISettings> colorSettings = new Dictionary<string, ISettings>();
    19 references
    public string? className { get; set; }
    List<string> screensNames = new List<string>
        {
            "Default",
            "AfricanElephantsScreen",
            "MainScreen",
            "AnimalsScreen",
            "DogsScreen",
            "SettingsScreen",
            "GrizzlyBearsScreen",
            "MammalsScreen",
            "PolarBearsScreen",
        };
    #region ISettings Implementation
    /// <summary>
    /// Method for adding new screen settings
    /// </summary>
    /// <param name="newScreenName">Name of the new screen</param>
    /// <param name="backgroundColor">Color of Background.</param>
    /// <param name="foregroundColor">Color of Text.</param>
    2 references
    public void AddSetting(string fileName)
    {
        // Default colors
        string defaultBackgroundColor = "Black";
        string defaultForegroundColor = "White";


        if (File.Exists(fileName))
        {
            colorSettings = Read(fileName);
            foreach (string screenName in screensNames)
            {
                if (colorSettings.ContainsKey(screenName))
                {
                }
                else
                {
                    Console.WriteLine($"Dictionary dont have {screenName}");
                    // Add new color settings for the screen
                    colorSettings.Add(screenName, new Settings
                    {
                        BackgroundColor = defaultBackgroundColor,
                        ForegroundColor = defaultForegroundColor
                    });
                }
            }
        }
        else
        {
            foreach (string screenName in screensNames)
            {
                if (colorSettings.ContainsKey(screenName))
                {
                }
                else
                {
                    Console.WriteLine($"Dictionary dont have {screenName}");
                    // Add new color settings for the screen
                    colorSettings.Add(screenName, new Settings
                    {
                        BackgroundColor = defaultBackgroundColor,
                        ForegroundColor = defaultForegroundColor
                    });
                }
            }
        }
    }
```

```
11 references
public void ApplyColors(string fileName, string className)
{
    Write(fileName);

    if (File.Exists(fileName))
    {
        colorSettings = Read(fileName); // Read the settings from the JSON file
        if (colorSettings.ContainsKey(className))
        {
            ISettings colors = colorSettings[className];

            if (IsColorExist(colors.BackgroundColor) && IsColorExist(colors.ForegroundColor))
            {
                Console.BackgroundColor = (ConsoleColor)Enum.Parse(typeof(ConsoleColor), colors.BackgroundColor);
                Console.ForegroundColor = (ConsoleColor)Enum.Parse(typeof(ConsoleColor), colors.ForegroundColor);

            }
            else
            {

                Console.WriteLine($"Error: color is not available. Applying default settings");

            }
        }
        else
        {
            ISettings colorss = colorSettings["Default"];
            Console.BackgroundColor = (ConsoleColor)Enum.Parse(typeof(ConsoleColor), colorss.BackgroundColor);
            Console.ForegroundColor = (ConsoleColor)Enum.Parse(typeof(ConsoleColor), colorss.ForegroundColor);

        }
    }
    else
    {
        ISettings colorss = colorSettings["Default"];
        Console.BackgroundColor = (ConsoleColor)Enum.Parse(typeof(ConsoleColor), colorss.BackgroundColor);
        Console.ForegroundColor = (ConsoleColor)Enum.Parse(typeof(ConsoleColor), colorss.ForegroundColor);

    }
}
2 references
public bool IsColorExist(string colour)
{
    foreach (ConsoleColor color in Enum.GetValues(typeof(ConsoleColor)))
    {
        if (colour == color.ToString())
        {
            return true;
        }
    }
    return false;
}

2 references
public void ChangeColor()
{
    try
    {
        Console.Write("Enter the name of screen you want to change: ");
        string? screenName = Console.ReadLine();
        string jsonContent = File.ReadAllText("settings.json");
        JObject screens = JObject.Parse(jsonContent);

        var selectedScreen = screens.Properties()
            .FirstOrDefault(prop => prop.Name == screenName);

        if (selectedScreen != null)
        {
            Console.Write("Enter the new background color: ");
            string? newBGcolor = Console.ReadLine();
            selectedScreen.Value["BackgroundColor"] = newBGcolor;

            Console.Write("Enter the new foreground color: ");
            selectedScreen.Value["ForegroundColor"] = Console.ReadLine();

            File.WriteAllText("settings.json", screens.ToString());
            Console.WriteLine("Screen colors updated successfully.");
        }
        else
        {
            Console.WriteLine("Screen not found or colors not updated.");
        }
    }
    catch (Exception)
    {
        // Handle exceptions (file not found, invalid JSON, etc.)
        Console.WriteLine("An error occurred.");
    }
```

```
C:\Users\Мурзич\source\repos\OOP\src\SampleHierarchies.App\bin\Debug\net6.0\SampleHierarchies.App.exe

Your available choices are:
0. Exit
1. Animals
2. Settings
Please enter your choice:
```


Your available choices are:
0. Exit
1. Change colors
2. List of screens
3. List of color
Please enter your choice: 2
Default
AfricanElephantsScreen
MainScreen
AnimalsScreen
DogsScreen
SettingsScreen
GrizzlyBearsScreen
MammalsScreen
PolarBearsScreen
Your available choices are:
0. Exit
1. Change colors
2. List of screens
3. List of color
Please enter your choice: 3
DarkGray
DarkBlue Blue
DarkGreen Green
DarkCyan Cyan
DarkRed Red
DarkMagenta Magenta
DarkYellow Yellow

Your available choices are:
0. Exit
1. Dogs
2. Polar Bears
3. African Elephants
4. Grizzly Bears
Please enter your choice: