

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

**Факультет управления и информатики в технологических системах**

**Кафедра**[**корпоративных информационных систем и программирования**](https://vsuet.ru/obuchenie/faculties/uits/k_kisip)

**Направление подготовки 09.03.02**

***(шифр)***

**Информационные системы и технологии**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование направления подготовки)*

**Отчет**

**по практическому заданию № 9**

Выполнил студент гр. У-213

Боровкова Анастасия Дмитриевна

(ф.и.о)

**\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***(****подпись****)***

Проверил(а):

Демичев С.Е.

**\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***(****подпись****)***

Воронеж – 2023

using System;

using System.IO.IsolatedStorage;

using System.Diagnostics;

namespace myAppName

{

public class AppSettings

{

// Our isolated storage settings

IsolatedStorageSettings settings;

// The isolated storage key names of our settings

const string InstalledonSettingKeyName = "InstalledonSetting";

const string SeenTutorialSettingKeyName = "SeenTutorialSetting";

const string FontSizeSettingKeyName = "FontSizeSetting";

const string FontTypeSettingKeyName = "FontTypeSetting";

// The default value of our settings

const long InstalledonSettingDefault = 0;

const bool SeenTutorialSettingDefault = false;

const int FontSizeSettingDefault = 30;

const string FontTypeSettingDefault = "";

/// <summary>

/// Constructor that gets the application settings.

/// </summary>

public AppSettings()

{

try

{

// Get the settings for this application.

settings = IsolatedStorageSettings.ApplicationSettings;

}

catch (Exception e)

{

Debug.WriteLine("Exception while using IsolatedStorageSettings: " + e.ToString());

}

}

/// <summary>

/// Update a setting value for our application. If the setting does not

/// exist, then add the setting.

/// </summary>

/// <param name="Key"></param>

/// <param name="value"></param>

/// <returns></returns>

public bool AddOrUpdateValue(string Key, Object value)

{

bool valueChanged = false;

// If the key exists

if (settings.Contains(Key))

{

// If the value has changed

if (settings[Key] != value)

{

// Store the new value

settings[Key] = value;

valueChanged = true;

}

}

// Otherwise create the key.

else

{

settings.Add(Key, value);

valueChanged = true;

}

return valueChanged;

}

/// <summary>

/// Get the current value of the setting, or if it is not found, set the

/// setting to the default setting.

/// </summary>

/// <typeparam name="valueType"></typeparam>

/// <param name="Key"></param>

/// <param name="defaultValue"></param>

/// <returns></returns>

public valueType GetValueOrDefault<valueType>(string Key, valueType defaultValue)

{

valueType value;

// If the key exists, retrieve the value.

if (settings.Contains(Key))

{

value = (valueType)settings[Key];

}

// Otherwise, use the default value.

else

{

value = defaultValue;

}

return value;

}

/// <summary>

/// Save the settings.

/// </summary>

public void Save()

{

settings.Save();

}

public long InstalledonSetting

{

get

{

return GetValueOrDefault<long>(InstalledonSettingKeyName, InstalledonSettingDefault);

}

set

{

if (AddOrUpdateValue(InstalledonSettingKeyName, value))

{

Save();

}

}

}

public int FontSizeSetting

{

get

{

return GetValueOrDefault<int>(FontSizeSettingKeyName, FontSizeSettingDefault);

}

set

{

if (AddOrUpdateValue(FontSizeSettingKeyName, value))

{

Save();

}

}

}

public string FontTypeSetting

{

get

{

return GetValueOrDefault<string>(FontTypeSettingKeyName, FontTypeSettingDefault);

}

set

{

if (AddOrUpdateValue(FontTypeSettingKeyName, value))

{

Save();

}

}

}

public bool SeenTutorialSetting

{

get

{

return GetValueOrDefault<bool>(SeenTutorialSettingKeyName, SeenTutorialSettingDefault);

}

set

{

if (AddOrUpdateValue(SeenTutorialSettingKeyName, value))

{

Save();

}

}

}

}