МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 1**

з дисципліни « Об’єктно орієнтоване програмування »

*назва дисципліни*

на тему: « КЛАСИ, ОБ’ЄКТИ »

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: доц. Вдовітченко О.В.

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

# ПОСТАНОВКА ЗАДАЧІ

**Завдання**

На основі отриманого на лекції 1 теоретичного матеріалу розробити консольну програму, яка:

1. Має опис деякого класу (предметну область обрати самостійно, будьте креативними!), клас повинен мати закриті і загальнодоступні поля\* , які описують характеристики класу, та методи, які задають поведінку класу\*\* .

\* хоча б одне поле має бути типу enum, також для характеристик класу треба використовувати не менше 4-5 різних типів даних.

\*\*реалізувати enum i клас в окремих файлах \*.cs.

1. Для кожної характеристики класу сформувати обмеження.

Наприклад, для класу Student обмеження для характеристики string name можуть бути наступні:

* довжина імені не менша за три символи;
* ім’я може містить тільки букви латинського алфавіту.

Обмеження для характеристики Date birthDate можуть бути наступні:

* дата у форматі dd.mm.yyyy;
* дата не менша за 01.01.1960;
* дата не більша за поточну дату.

Обмеження для характеристики double GPA (середній бал) можуть бути наступні:

* GPA – число в діапазоні [0..100].

Перевірка зазначених обмежень має відбуватися в основній програмі. Характеристикам об’єктів можна привласнювати тільки коректні значення.

1. В програмі має бути передбачена можливість зберігати до N об’єктів класу (N>0 i задається користувачем, наприклад, при старті програми\*\*\*).

\*\*\* має бути реалізована перевірка на коректне введення N.

Примітка: використання List не забороняється!

1. В програмі має бути реалізоване наступне меню:

1 – додати об’єкт

2 – вивести на екран об’єкти

3 – знайти об’єкт

4 – видалити об’єкт

5 – демонстрація поведінки

0 – вийти з програми

Під час **додавання** об’єкта користувачем всі його поля мають заповнюватися в ручному або автоматичному (необов’язково) режимах. Передбачити неможливість введення більше ніж N об’єктів.

**Пошук** реалізувати по будь-яким двом характеристикам, в програмі передбачити вибір користувачем характеристики для пошуку (наприклад, шукаємо або по коліру, або по ціні).

**Видалення** об’єкта реалізувати за номером та за однією з характеристик.

При **демонстрації поведінки** необхідно показати виконання всіх реалізованих методів класу.

Програма має закінчувати свою роботу тільки після вибору користувачем пункту меню **«вийти з програми»**.

1. Детально протестувати програму. Мають бути протестовані всі пункти меню, а також пам’ятаємо про перевірку значень, які вводить користувач, на відповідність обмеженням. Проводимо тестування не тільки на коректних значеннях, а також не забуваємо перевірити і некоректні значення, які може ввести користувач.
2. Оформити звіт:

* Титульний аркуш
* Завдання
* Короткий опис класу (опис характеристик та поведінки з точки зору замовника)
* Програмна реалізація класу
* Class diagram
* Код програми
* Результати детального тестування програми (навести скріншоти виконання тестування програми, або скопіювати і вставити у звіт вивід програми на екран)

ХІД РОБОТИ

Опис програми:

Мова програмування: С#, операційна система Windows 11 Prо, Версія 23H2, Збірка ОС 22621.1325, процесор: Apple Silicon M1 Pro 3.20 GHz (ядер: 4), компілятор: Microsoft Visual Studio Community 2022 (64-розрядна версія ARM).

1. Програма виконує роль записника завдань, який заповнює користувач в напівавтоматичному режимі.. Програма містить в собі клас Homework, з відповідними для занять характеристиками (будь-який номер завдання(порядковий, чи сам номер завдання), deadline, предмет, тип завдання, текст задання). Всі з них є загальнодоступними.

public int taskNumber;

public DateTime deadline;

public string subject;

public TaskType taskType;

public string taskText;

1. Програмна реалізація класу:

namespace Lab\_1

{

public class Homework

{

public int taskNumber;

public DateTime deadline;

public string subject;

public TaskType taskType;

public string taskText;

public Homework(int taskNumber, DateTime deadline, string subject, TaskType taskType, string taskText)

{

this.taskNumber = taskNumber;

this.deadline = deadline;

this.subject = subject;

this.taskType = taskType;

this.taskText = taskText;

Storage.Tasks.Add(this);

}

}

}

1. Class Diagram

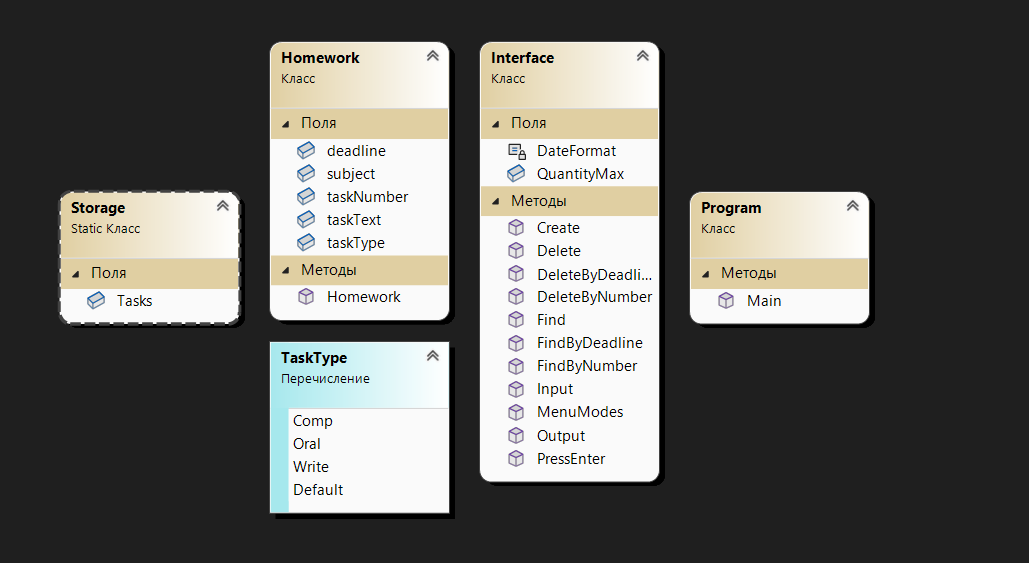


Рисунок 1 – Створена діаграма класу

1. Тестування:

Для початку перевіримо, які значення може прийняти обмежувач кількості завдань (див. рисунок 2)

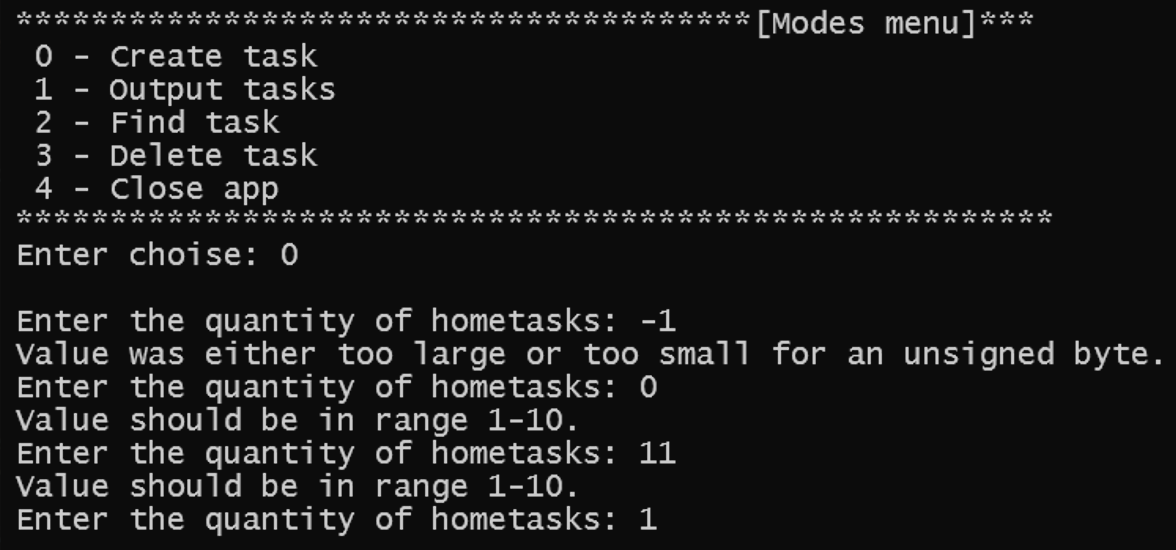


Рисунок 2 – перевірка діапазону введених значень кількості завдань

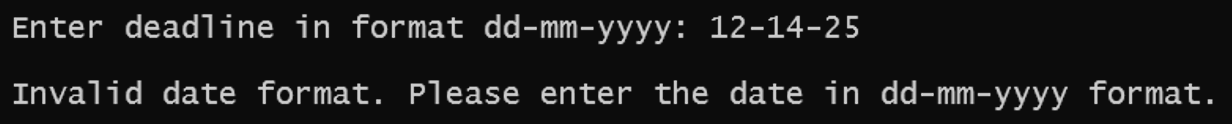


Рисунок 3 – перевірка діапазону введених значень дати

Додамо завдання в список, та перевіримо їх наявність.



Рисунок 4– перевірка введених завдань

Спробуємо додати ще одне завдання, яке не може бути додано через заданий раніше ліміт

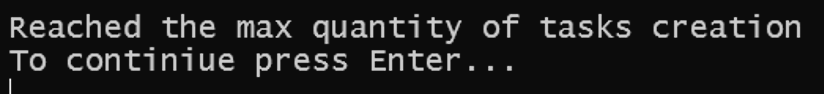


Рисунок 5 – перевірка введення ще одного завдання через ліміт

* 1. Далі спробуємо знайти заняття, які були додані (виконаємо пошук по номеру завдання та дедлайном):

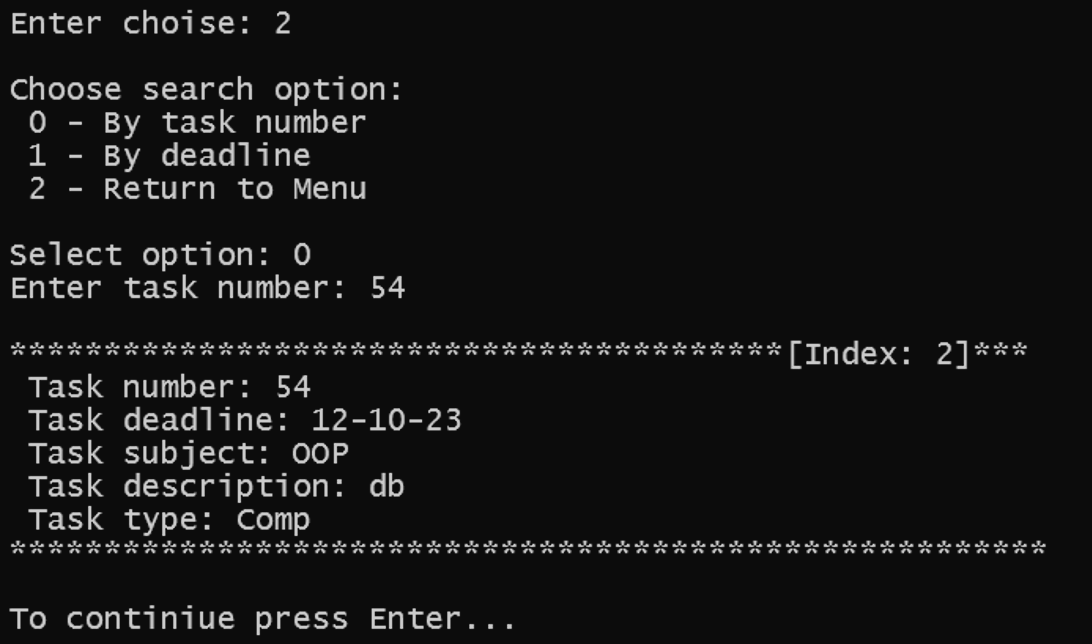


Рисунок 6 – пошук завдання за номером

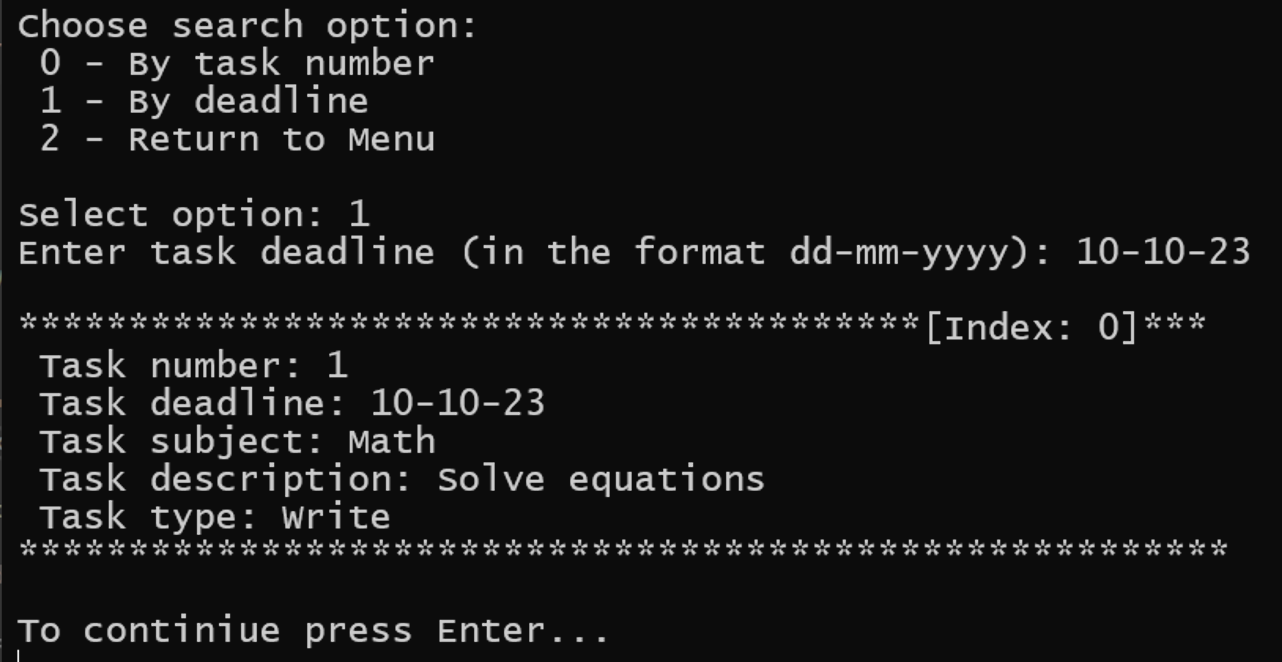


Рисунок 7 – пошук завдання за дедлайном

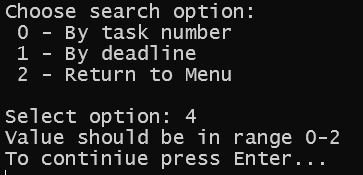


Рисунок 8 – введення некоректного режиму пошуку

Видалимо декілька завдань

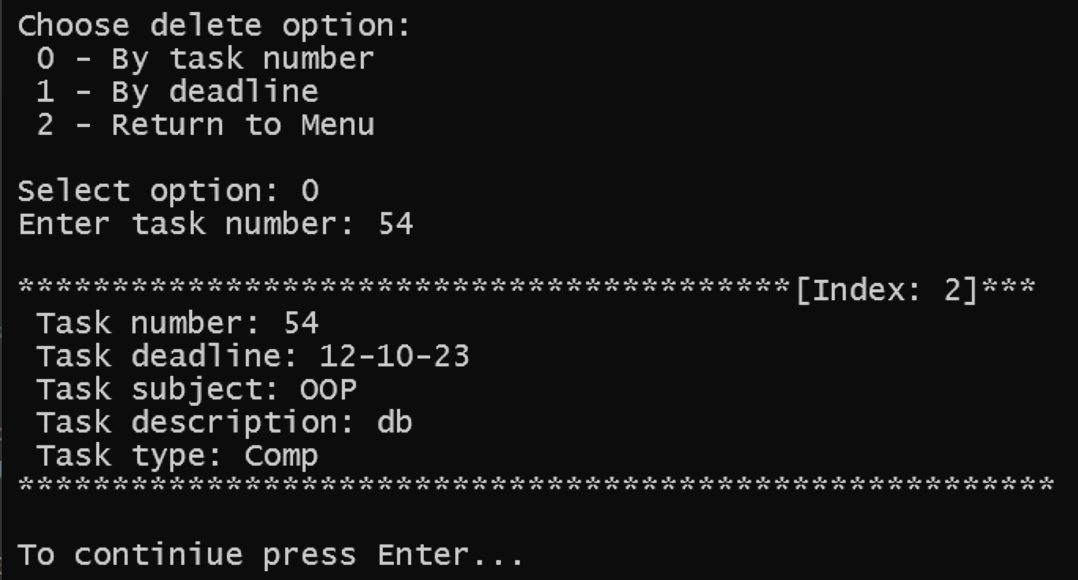
****

Рисунок 9 – видалення завдання за номером

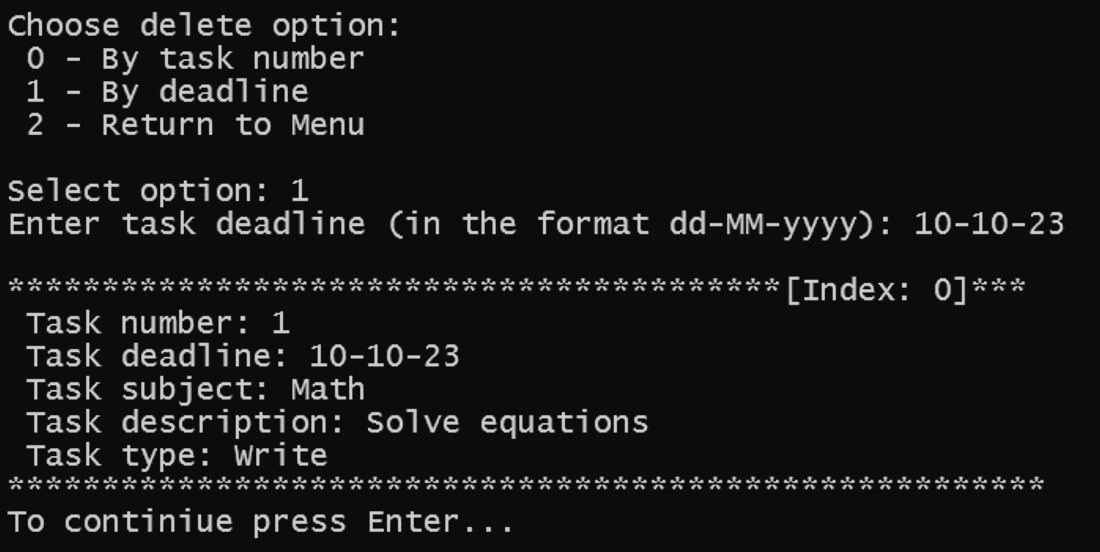
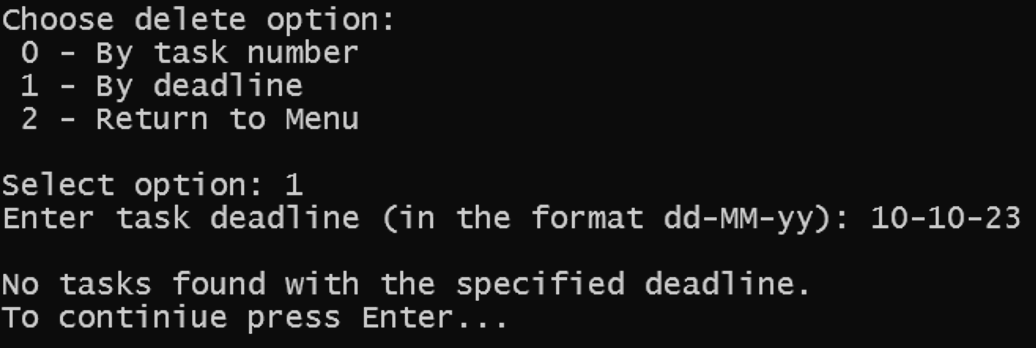
****

Рисунок 10 – видалення завдання за дедлайном

Видалимо всі заняття, які залишились. Також спробуємо видалити заняття, коли їх не існує.

Рисунок 11 – видалення елементів при їх відсутності

# ВИСНОВОК

У ході виконання лабораторної роботи я:

* Навчився створювати класи, встановлювати доступність полів;
* Написав консольний додаток, який використовує створений клас;
* Провів детальне тестування програми, додав скріншоти.

***ДОДАТОК А***

Машинний лістинг програми

**Program.cs:**

using System.Threading.Tasks;

namespace Lab\_1

{

public class Program

{

public static void Main(string[] args)

{

//debug list

var homework1 = new Homework(1, DateTime.Parse("2023-10-10"), "Math", TaskType.Write, "Solve equations");

var homework3 = new Homework(2, DateTime.Parse("2023-10-15"), "English", TaskType.Oral, "Write an essay");

Interface ui = new();

ui.QuantityMax = 5;

while (true)

{

ui.MenuModes();

switch (ui.Input("Enter choise: ", 5, 0))

{

case 0: ui.Create(); ui.PressEnter(); break;

case 1: ui.Output(); ui.PressEnter(); break;

case 2: ui.Find(); ui.PressEnter(); break;

case 3: ui.Delete(); ui.PressEnter(); break;

case 4: Environment.Exit(0); ui.PressEnter(); break;

}

}

}

}

}

***ДОДАТОК Б***

Машинний лістинг класу

**Interrface.cs:**

using Microsoft.VisualBasic;

using System.Globalization;

using System.Transactions;

namespace Lab\_1

{

internal class Interface

{

public int QuantityMax;

const string DateFormat = "dd-MM-yy";

int count = 0;

public void Create()

{

TaskType taskType = TaskType.Default;

#region input

if (count == QuantityMax) Console.WriteLine("Reached the max quantity of tasks creation");

else QuantityMax = Input("Enter the quantity of hometasks: ", 10, 1);

while (count < QuantityMax)

{

int taskNumber = Input("Enter task number: ", UInt16.MaxValue, 0);

Console.Write("Enter deadline in format dd-mm-yy: ");

DateTime mindeadline = new DateTime(2023, 09, 01);

DateTime maxdeadline = new DateTime(2024, 12, 31);

DateTime deadline = maxdeadline;

bool deadlineIsValid = false;

while (!deadlineIsValid)

{

string input = Console.ReadLine();

Console.WriteLine();

DateTime.TryParseExact(input, DateFormat, CultureInfo.InvariantCulture, DateTimeStyles.None, out deadline);

if (deadline > maxdeadline || deadline < mindeadline)

{

Console.Write("Invalid date format. Please enter the date in dd-mm-yy format: ");

deadlineIsValid = false;

}

else deadlineIsValid = true;

}

string subject = "";

bool isValidSubj = false;

while (!isValidSubj)

{

Console.Write("Enter subject: ");

subject = Console.ReadLine();

if (string.IsNullOrWhiteSpace(subject) || string.IsNullOrEmpty(subject))

{

Console.WriteLine("Subject cannot be empty. Please try again.");

continue;

}

if (subject.All(c => char.IsLetter(c) || char.IsSymbol(c)))

isValidSubj = true;

else Console.WriteLine("Invalid characters in the subject. Please use only letters or symbols.");

}

Console.WriteLine();

while (true)

{

Console.WriteLine($"\*\*\*[Task Types]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n" +

$" 0 - Comp \r\n" +

$" 1 - Oral \r\n" +

$" 2 - Write");

Console.WriteLine($"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

while (true)

{

try

{

byte select;

select = Convert.ToByte(Input("Select task type: ", 2, 0));

switch (select)

{

case 0: taskType = TaskType.Comp; break;

case 1: taskType = TaskType.Oral; break;

case 2: taskType = TaskType.Write; break;

default: Console.Write("Input error. Try again"); continue;

}

break;

}

catch (Exception ex) { Console.WriteLine($"{ex.Message}"); }

}

break;

}

Console.Write("Enter task description: ");

string taskText = Console.ReadLine();

Console.WriteLine();

#endregion

Homework task = new Homework(taskNumber, deadline, subject, taskType, taskText);

Console.WriteLine("Task created successfuly"); count++;

}

}

public void Output()

{

foreach (Homework task in Storage.Tasks)

{

Console.WriteLine("" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[Index: {Storage.Tasks.IndexOf(task)}]\*\*\*\n" +

$" Task number: {task.taskNumber}\n" +

$" Task deadline: {task.deadline.ToString(DateFormat)}\n" +

$" Task subject: {task.subject}\n" +

$" Task description: {task.taskText}\n" +

$" Task type: {task.taskType}\n" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

public void Find()

{

while (true)

{

Console.WriteLine($"Choose search option: \n" +

$" 0 - By task number\n" +

$" 1 - By deadline\n" +

$" 2 - Return to Menu\n");

Console.Write("Select option: ");

byte select = byte.Parse(Console.ReadLine());

if (select >= 3) Console.WriteLine("Value should be in range 0-2");

switch (select)

{

case 0: FindByNumber(); break;

case 1: FindByDeadline(); break;

case 2: return;

}

break;

}

}

public void FindByNumber()

{

Console.Write("Enter task number: ");

int inputNumber = Convert.ToInt32(Console.ReadLine());

Console.WriteLine();

foreach (Homework task in Storage.Tasks)

{

if (inputNumber == task.taskNumber)

{

Console.WriteLine("" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[Index: {Storage.Tasks.IndexOf(task)}]\*\*\*\n" +

$" Task number: {task.taskNumber.ToString()}\n" +

$" Task deadline: {task.deadline.ToString(DateFormat)}\n" +

$" Task subject: {task.subject}\n" +

$" Task description: {task.taskText}\n" +

$" Task type: {task.taskType}\n" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

}

public void FindByDeadline()

{

bool inputDateValid = false;

while (!inputDateValid)

{

Console.Write("Enter task deadline (in the format dd-mm-yy): ");

string inputDateString = Console.ReadLine();

DateTime inputDate;

if (DateTime.TryParseExact(inputDateString, DateFormat, CultureInfo.InvariantCulture, DateTimeStyles.None, out inputDate))

{

inputDateValid = true;

}

else if (!DateTime.TryParseExact(inputDateString, DateFormat, CultureInfo.InvariantCulture, DateTimeStyles.None, out inputDate))

{

Console.WriteLine("Invalid date format. Please enter the date in dd-mm-yy format.");

return;

}

Console.WriteLine();

foreach (Homework task in Storage.Tasks)

{

if (inputDate == task.deadline)

{

Console.WriteLine("" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[Index: {Storage.Tasks.IndexOf(task)}]\*\*\*\n" +

$" Task number: {task.taskNumber}\n" +

$" Task deadline: {task.deadline.ToString(DateFormat)}\n" +

$" Task subject: {task.subject}\n" +

$" Task description: {task.taskText}\n" +

$" Task type: {task.taskType}\n" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

}

}

public void Delete()

{

while (true)

{

Console.WriteLine($"Choose delete option: \n" +

$" 0 - By task number\n" +

$" 1 - By deadline\n" +

$" 2 - Return to Menu\n");

Console.Write("Select option: ");

byte select = byte.Parse(Console.ReadLine());

if (select >= 3) throw new Exception("Value should be in range 0-2");

switch (select)

{

case 0: DeleteByNumber(); break;

case 1: DeleteByDeadline(); break;

case 2: return;

}

break;

}

}

public void DeleteByNumber()

{

bool found = false;

int inputNumber = Input("Enter task number: ", UInt16.MaxValue, 1);

Console.WriteLine();

List<Homework> tasksToDelete = new List<Homework>();

for (int i = 0; i < Storage.Tasks.Count; i++)

{

Homework task = Storage.Tasks[i];

if (inputNumber == task.taskNumber)

{

tasksToDelete.Add(task);

}

}

if (tasksToDelete.Count == 0)

{

Console.WriteLine("No tasks found with the specified number.");

return;

}

Console.WriteLine("Tasks with the specified number:");

for (int i = 0; i < tasksToDelete.Count; i++)

{

Console.WriteLine($"[{i + 1}] Task number: {tasksToDelete[i].taskNumber}");

Console.WriteLine($" Task deadline: {tasksToDelete[i].deadline.ToString(DateFormat)}");

Console.WriteLine($" Task subject: {tasksToDelete[i].subject}");

Console.WriteLine($" Task description: {tasksToDelete[i].taskText}");

Console.WriteLine($" Task type: {tasksToDelete[i].taskType}");

Console.WriteLine();

}

Console.Write("Enter the number of the task to delete: ");

if (int.TryParse(Console.ReadLine(), out int selectedTaskIndex) && selectedTaskIndex >= 1 && selectedTaskIndex <= tasksToDelete.Count)

{

// Remove the selected task

Storage.Tasks.Remove(tasksToDelete[selectedTaskIndex - 1]);

Console.WriteLine("Task deleted successfully.");

}

else

{

Console.WriteLine("Invalid input. Task was not deleted.");

}

}

public void DeleteByDeadline()

{

Console.Write("Enter task deadline (in the format dd-MM-yy): ");

string inputDateString = Console.ReadLine();

DateTime inputDate;

if (!DateTime.TryParseExact(inputDateString, DateFormat, CultureInfo.InvariantCulture, DateTimeStyles.None, out inputDate))

{

Console.WriteLine("Invalid date format. Please enter the date in dd-MM-yy format.");

return;

}

Console.WriteLine();

List<Homework> tasksToDelete = new List<Homework>();

for (int i = 0; i < Storage.Tasks.Count; i++)

{

Homework task = Storage.Tasks[i];

if (inputDate == task.deadline)

{

tasksToDelete.Add(task);

}

}

if (tasksToDelete.Count == 0)

{

Console.WriteLine("No tasks found with the specified deadline.");

return;

}

Console.WriteLine("Tasks with the specified deadline:");

for (int i = 0; i < tasksToDelete.Count; i++)

{

Console.WriteLine($"[{i + 1}] Task number: {tasksToDelete[i].taskNumber}");

Console.WriteLine($" Task deadline: {tasksToDelete[i].deadline.ToString(DateFormat)}");

Console.WriteLine($" Task subject: {tasksToDelete[i].subject}");

Console.WriteLine($" Task description: {tasksToDelete[i].taskText}");

Console.WriteLine($" Task type: {tasksToDelete[i].taskType}");

Console.WriteLine();

}

Console.Write("Enter the number of the task to delete: ");

if (int.TryParse(Console.ReadLine(), out int selectedTaskIndex) && selectedTaskIndex >= 1 && selectedTaskIndex <= tasksToDelete.Count)

{

// Remove the selected task

Storage.Tasks.Remove(tasksToDelete[selectedTaskIndex - 1]);

Console.WriteLine("Task deleted successfully.");

}

else

{

Console.WriteLine("Invalid input. Task was not deleted.");

}

}

#region UI

public void MenuModes()

{

Console.Clear();

Console.WriteLine("" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[Modes menu]\*\*\*\n" +

$" 0 - Create task\n" +

$" 1 - Output tasks\n" +

$" 2 - Find task\n" +

$" 3 - Delete task \n" +

$" 4 - Close app \n" +

$"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

public ushort Input(string text, ushort up\_range, ushort down\_range)

{

while (true)

{

ushort input;

try

{

Console.Write(text);

input = Byte.Parse(Console.ReadLine());

if (input > up\_range || input < down\_range)

{

throw new Exception($"Value should be in range {down\_range}-{up\_range}.");

Console.WriteLine();

}

}

catch (Exception ex)

{

Console.WriteLine($"{ex.Message}");

continue;

}

Console.WriteLine();

return input;

}

}

public void PressEnter()

{

while (true)

{

Console.WriteLine("To continiue press Enter...");

var key = Console.ReadKey(intercept: true);

if (key.Key == ConsoleKey.Enter)

{

Console.Clear();

break; // Выход из цикла, если нажата клавиша Enter

}

else Console.WriteLine("Pressed another key");

}

}

#endregion

}

}

***ДОДАТОК В***

Машинний лістинг класу

**Homework.cs:**

namespace Lab\_1

{

public class Homework

{

public int taskNumber;

public DateTime deadline;

public string subject;

public TaskType taskType;

public string taskText;

public Homework(int taskNumber, DateTime deadline, string subject, TaskType taskType, string taskText)

{

this.taskNumber = taskNumber;

this.deadline = deadline;

this.subject = subject;

this.taskType = taskType;

this.taskText = taskText;

Storage.Tasks.Add(this);

}

}

}

***ДОДАТОК Г***

Машинний лістинг класу

**EnumFile.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_1

{

public enum TaskType

{

Comp,

Oral,

Write,

Default

};

}

***ДОДАТОК Ґ***

Машинний лістинг класу

**Storage.cs:**

namespace Lab\_1

{

internal static class Storage

{

public static List<Homework> Tasks = new List<Homework>();

}

}