# Розробка проектів засобами платформи .NET / Практична робота №1

За першу практичну роботу можна набрати **5 балів**.

## I - Підготовка робочого середовища

Стислий виклад задачі:

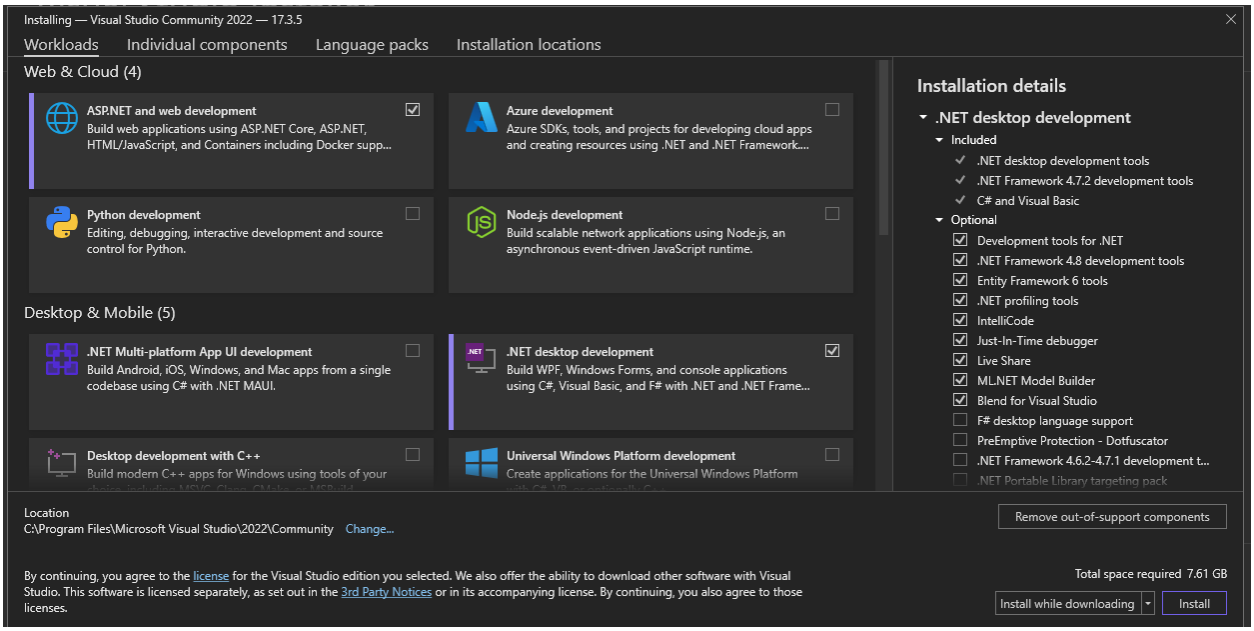
1. Встановити IDE для розробки під .NET.
2. Створити репозиторій на GitHub і надати доступ викладачу.

### Встановити Visual Studio Community 2022

Download from: https://visualstudio.microsoft.com

При інсталяції вибрати пункти:

- ASP.NET and web development
- .NET desktop development




# Встановити SourceTree для роботи з Git

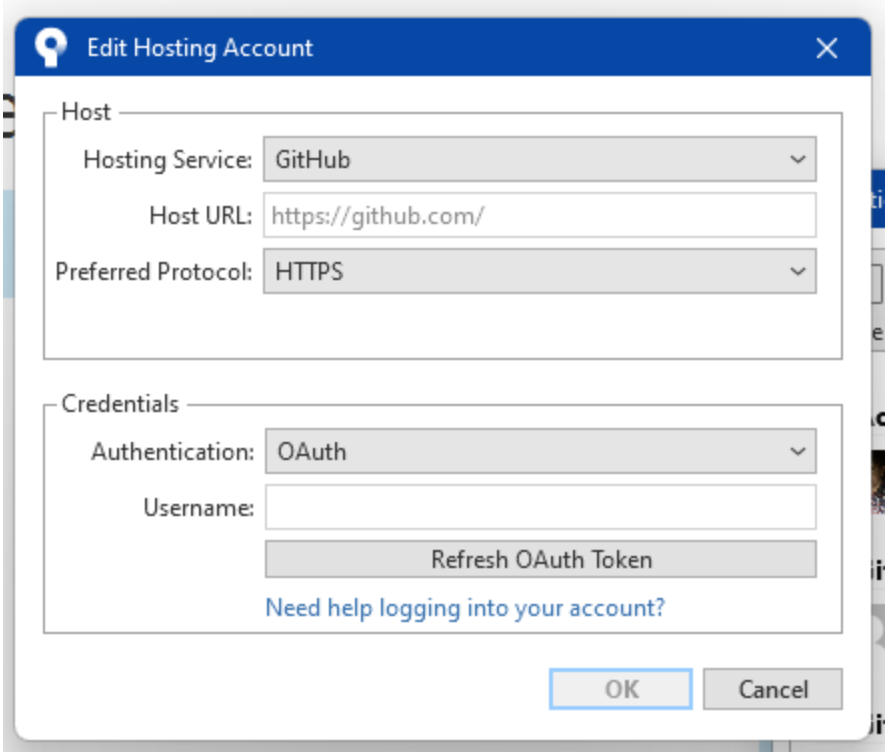
Не обов’язково використовувати саме SourceTree у випадку наявності досвіду роботи з інструментами контролю версій. В методичних вказівках завдання, що стосуються роботи з Git, будуть ілюструватися саме на прикладі SourceTree.

Download from: https://www.sourcetreeapp.com

Пункт реєстрації в Atlassian можна пропустити. При виборі інструментів контролю версій необхідно вибрати Git.

# Авторизувати SourceTree для роботи з GitHub.

Tools > Options > Authentication > Add.

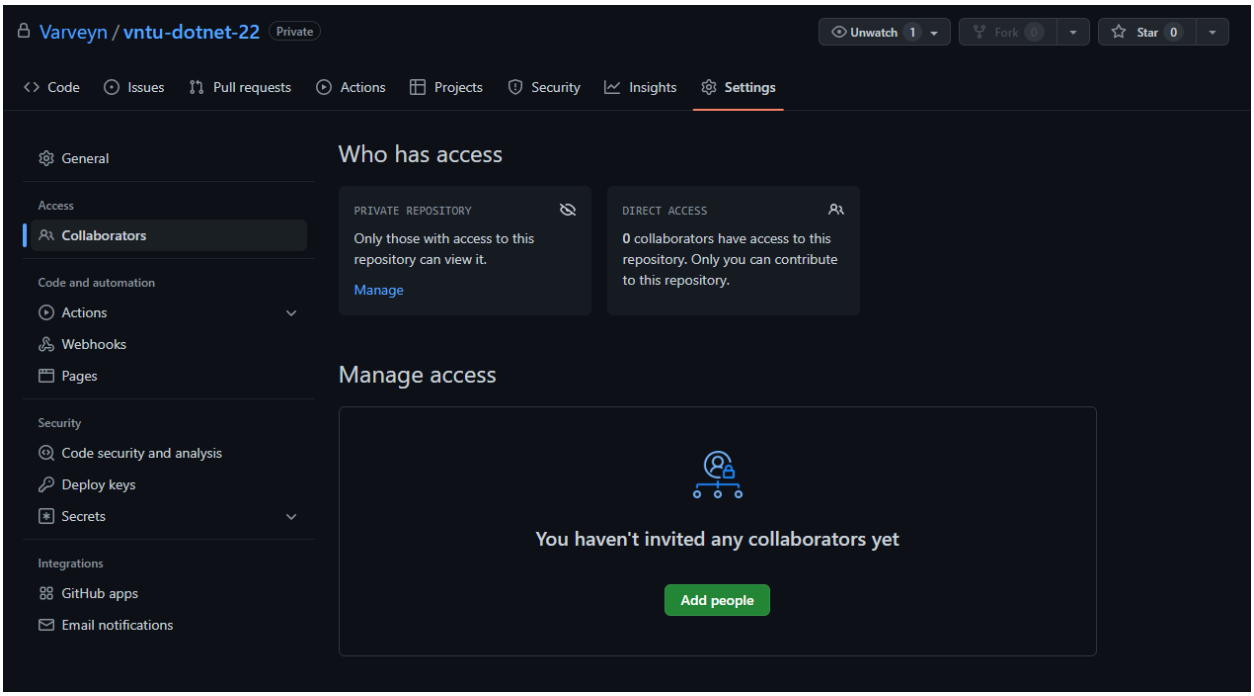



На зображеному діалоговому вікні натиснути “Refresh OAuth Token”, після чого авторизуватись в GitHub.

## Створити репозиторій на GitHub

Цей репозиторій буде використовуватись для здачі практичних робіт.

Надати доступ до репозиторію викладачу: Settings > Collaborators > Add people.




# Клонувати поки-що-порожній репозиторій на комп’ютер

В SourceTree:

Clone > “remote account” > вибрати аккаунт > вибрати репозиторій.

# Створити проект в Visual Studio

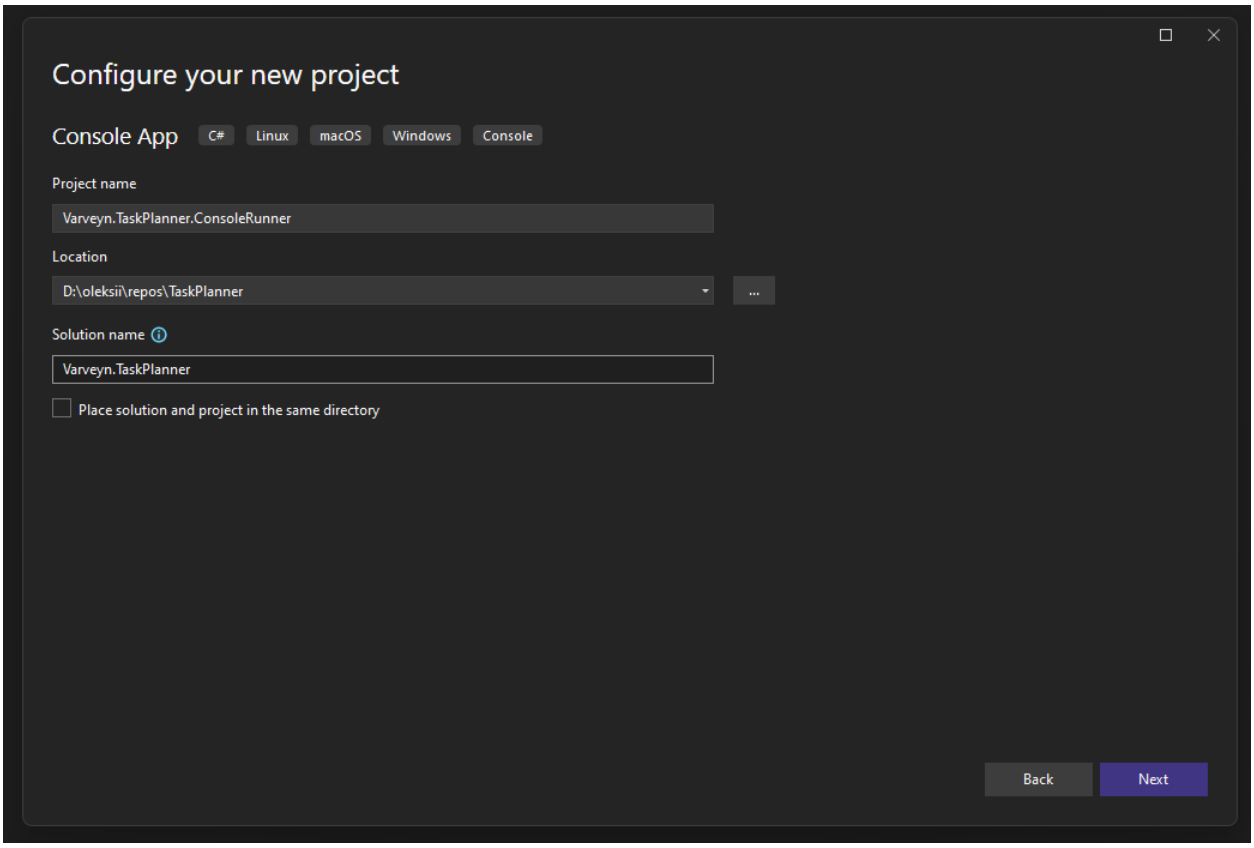
Тип проекту: Console Application.

Target framework: .NET 6.0.

Предметна область практичних завдань - “планувальник задач”. Назва проекту має відповідати предметній області. Можливий формат іменування: **JonDou9000.TaskPlanner**, де

**JonDou9000** - ім’я користувача. Використовується як префікс імені проекту і, відповідно, кореневий простір імен - це має значення для можливих колізій імен з загальнодоступними NuGet-пакетами.

**TaskPlanner** - власне ім’я рішення (solution - найвища структурна одиниця організації коду в екосистемі .NET).




## Зібрати рішення

- Build > Build Solution (F6)

## Запустити проект

- Debug > Start Debugging (F5)
- Debug > Start Without Debugging (Ctrl + F5)

## Підготувати структуру системи керування версіями

1. Push changes to remote repository (origin).
2. Create new branch: **develop**. Checkout. Push to origin.

Практичні завдання виконуються по порядку. Здача практичної роботи виконується наступним чином: студент робить зміни в **develop**, відкриває pull request в **master/main** в GitHub, де теґає викладача. Викладач перевіряє зміни і дає коментарі. Коли практична прийнята, викладач схвалює pull request і заливає його в **master** або **main** branch.

# II - Виконання практичного завдання

1. Створити новий проект - **JonDou9000.TaskPlanner.Domain.Models**.
    a. Тип: **Class Library**.
    b. Target Framework: **.NET 6.0**.
2. У проекті **.Domain.Models** створити папку Enums.
3. В папку Enums додати `enum` `Priority` з наступними значеннями:
    a. None,
    b. Low,
    c. Medium,
    d. High,
    e. Urgent,
4. В папку Enums додати `enum` `Complexity` з наступними значеннями:
    a. None,
    b. Minutes,
    c. Hours,
    d. Days,
    e. Weeks,
5. У проекті **.Domain.Models** додати новий `public` клас: `WorkItem`. Додати в нього наступні auto-implemented properties:
    a. CreationDate : `DateTime`

b. DueDate : `DateTime`

c. Priority : `Priority` (enum)

d. Complexity : `Complexity` (enum)

e. Title : `string`

f. Description : `string`

g. IsCompleted : `bool`

6. В класі `WorkItem` перевизначити метод `ToString()` таким чином, щоб він повертав рядок, що місттить Title, DueDate і Priority. Наприклад: `Do laundry: due 17.10.2022, high priority`. При цьому:

   a. Використати рядкову інтерполяцію (string interpolation).

   b. Дата має бути у форматі `dd.MM.yyyy`.

   c. Слово, що описує пріоритет, має стилістично узгоджуватись з рештою рядка (наприклад, за капіталізацією). Якщо потрібно, використати методи `string.ToLower()` чи `string.ToUpper()`.

7. Створити новий проект - **JonDou9000.TaskPlanner.Domain.Logic**.

   a. Тип: **Class Library**.

   b. Target Framework: **.NET 6.0**.

8. Додати в проекті **.Domain.Logic** reference на проект **.Domain.Models**.

> 💡 Solution Explorer → **JonDou9000.TaskPlanner.Domain.Logic** → Dependencies → Add Project Reference…

9. У проекті **.Domain.Logic** створити клас `SimpleTaskPlanner`**.**

   a. Додати метод `CreatePlan`, який на вхід приймає масив об’єктів типу `WorkItem`, і повертає також масив об’єктів типу `WorkItem`. Метод має впорядкувати масив за наступними критеріями:

      i. Priority - за спаданням (спершу - важливіше).

      ii. DueDate - за зростанням (раніше - спершу).

iii. Title - в алфавітному порядку.

b. Один з варіантів реалізації такого алгоритму:

i. конвертувати масив в `List<WorkItem>` за допомогою метода `ToList()`,

ii. використати метод класу `List<T>` `Sort(Comparison<T> comparer)`,

iii. конвертувати посортований `List<WorkItem>` у масив за допомогою методу `ToArray()`.

> 💡 `Comparison<T>` - це делегат (простими словами - об'єкт-вказівник на метод).

```
private WorkItem[] CreatePlan(WorkItem[] items)
{
    var itemsAsList = items.ToList();

    itemsAsList.Sort(CompareWorkItems);

    return itemsAsList.ToArray();
}

private static int CompareWorkItems(WorkItem firstItem, WorkItem secondItem)
{
    // comparison logic here
}
```

10. У консольному проекті, створеному на початку виконання практичної роботи зробити наступне:

a. додати project reference на проекти **.Domain.Models** та **.Domain.Logic**.

b. У файлі Program.cs стерти вміст і натомість явно визначити `internal static class Program` з методом `public static void Main(string[] args)`. Це - вхідна точка в застосунок.

c. У класі Program реалізувати наступну поведінку:

i. Користувач вводить довільну кількість `WorkItem`. Для вводу використовувати консольний інтерфейс. У нагоді можуть стати статичні методи `DateTime.Parse()`, `Enum.Parse<T>(string value, bool ignoreCase)`.

ii. Створюється новий об’єкт класу `SimpleTaskPlanner`. За його допомогою масив WorkItem впорядковується.

iii. Результат виводиться на екран.

> 💡 Зручним може бути використати конструкцію `foreach`.

# Список посилань


- https://en.wikipedia.org/wiki/NuGet