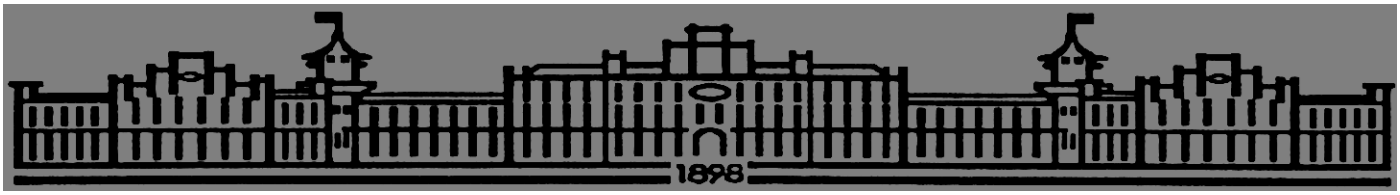

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № __1___

з дисципліни “Основи програмування 2”
тема “Об’єкти та класи”

Виконав
студент І курсу
групи КП-03
Сидоренко Олександр Олександрович
(*прізвище, ім’я, по батькові*)

Перевірив
“____” “_____” 20___ р.
викладач
Гадиняк Руслан Анатолійович
(*прізвище, ім’я, по батькові*)

варіант № 16

Київ 2020

Мета роботи

Створення і використання класів для об'єктів з даними без логіки та об'єктів динамічних колекцій елементів.

Генерування та порядкова обробка великої кількості даних у форматі CSV.

Постановка завдання

Частина 1. Генератор CSV

Створити консольну програму, що на основі двох заданих аргументів командного рядка генерує у текстовий файл дані у форматі CSV.
Перший аргумент задає шлях до файлу, другий - кількість рядків у CSV таблиці з даними, які будуть згенеровані. Наприклад, `**dotnet run ./out.csv 13**`. Якщо не задано одного з аргументів, або у аргументах є помилка - друкувати про це повідомлення в консолі і завершувати роботу програми. До аргументу з назвою файлу не додавати в програмі жодних додаткових рядків, не робити обмеження лише на `.csv` розширення, воно може бути будь-яким і бути відсутнім. Кількість рядків даних - невід'ємне число.
Дані, які генеруються у файл описують сутності із лабораторної роботи №5 минулого семестру. CSV файл має містити перший рядок із назвами стовпців (не входить в кількість рядків, які потрібно генерувати). Ідентифікатори генерувати унікальними в рамках файлу. Для генерації рядкових даних можна задати масив з рядками і випадковим чином вибирати звідти рядки, їх можна склеювати.
Достатньо зберігати дані без CSV екранування, якщо вони будуть генеруватись з відповідними обмеженнями на дані.

Частина 2.

Створити консольну програму, що зчитує CSV дані заданих за варіантом сутностей з 2-х файлів, виконує їх перетворення і записує результат як CSV у 3-ій файл.
Попередньо згенерувати вхідні файли за допомогою програми-генератора з першої частини завдання. Перший з файлів має містити близько 10-20 сутностей, другий - 100000-200000 сутностей.
Шляхи до 3-х файлів достатньо задати в коді (але 1 раз в головній функції) без аргументів

командного рядка.

Перенести у код роботи структуру даних із лабораторної роботи №5 і зробити її класом. Додати у клас два конструктора: без параметрів і з параметрами для всіх полів. Одне із полів має бути цілочисельним і з назвою **id** - ідентифікатор об'єкта. Додати в клас реалізацію стандартного метода ToString (див. додаток).

Створити клас (ListEntity, Entity замінити на назву типу сутності) для списку сутностей за варіантом на основі масиву об'єктів (див. додаток).

Реалізувати функцію (замінити entity і entities на назву сутності в однині і множині, за варіантом):

```
static ListEntity ReadAllEntities(string filePath);
```

У цій функції зчитувати текст з CSV файлу **порядково** (див додатки), у об'єкт власної реалізації списка із об'єктами типу сутності за варіантом. **Файл можна зчитати лише один раз**.
Перевіряти рядки CSV, якщо у них помилка (наприклад, кількість даних у рядку не рівна кількості стовпців) викидати помилку, що завершить роботу програми.

Використати ReadAllEntities для обох вхідних файлів і вивести кількість зчитаних рядків даних і перші 10 рядків даних (якщо є) у консоль для кожного файлу окремо.

Дії (створити на кожну по функції):

- Створити новий об'єкт ListEntity і переписати в нього всі елементи з перших двох списків так, щоби в новому списку не повторювались ідентифікатори сутностей (всі ідентифікатори були унікальні). Тобто, не переписувати у новий список сутність, якщо у списку вже є інша сутність з таким ідентифікатором.
- Обрати будь-яке числове поле вашого типу сутності (але не ідентифікатор) і знайти по списку середнє арифметичне значення по цьому полю.
- Видалити зі списку всі об'єкти, значення відповідного поля яких менше за знайдене середнє арифметичне.

Створити функцію (замінити entity і entities на назву сутності в однині і множині, за варіантом):

```
static void WriteAllEntities(string filePath, ListEntity entities);
```

Записати результат (модифікований третій список) у вихідний файл в форматі CSV порядково (див додатки).

Обмеження
При реалізації завдань заборонено:

- використання стандартних типів колекцій і інших алгоритмів, що дані у завданні, їх поведінку необхідно реалізувати самостійно.
- використання статичних полів.

Функції мають мати одне призначення, без зайвих параметрів, повертати результат там, де це можливо.

Назви функцій, типів, полів і змінних мають слідувати одному стилю і відповідати їх призначенню.
Назва функції (метода) має містити хоча б одне дієслово, що описує дію, яку вона виконує.

Приклади результатів

Частина 1:

dotnet run ./out 10

```
Перша частина успішно виконана! Сутності згенеровано та збережено у файл
`./out`.

Натисніть [Enter], щоб виконати другу частину:
```

Помилки:

dotnet run ./out -10

```
Unhandled exception. System.Exception: ERROR! Only positive numbers!
   at lab1__semester_2_.Program.Main(String[] args) in
/home/alex/projects/progbase2/lab1/Program.cs:line 105
```

dotnet run out 10

```
Unhandled exception. System.Exception: ERROR! Incorrect file path! First
argument must be started with `./`
   at lab1__semester_2_.Program.Main(String[] args) in
/home/alex/projects/progbase2/lab1/Program.cs:line 92
```

dotnet run ./out abc

```
Unhandled exception. System.Exception: ERROR! Only integers in the 2nd
argument!
   at lab1__semester_2_.Program.Main(String[] args) in
/home/alex/projects/progbase2/lab1/Program.cs:line 101
```

dotnet run ./out 10 20

```
Unhandled exception. System.Exception: ERROR! Should be 2 arguments but have 3
   at lab1__semester_2_.Program.Main(String[] args) in
/home/alex/projects/progbase2/lab1/Program.cs:line 88
```

dotnet run ./out

```
Unhandled exception. System.Exception: ERROR! Should be 2 arguments but have 1
   at lab1__semester_2_.Program.Main(String[] args) in
/home/alex/projects/progbase2/lab1/Program.cs:line 88
```

Частина 2:

[Enter]

```
Кількість елементів списку #1: 10
Перші 10 елементів списку:
ID: 6; Name: Мова програмування C#; Group: KP-92; Semester: 4
ID: 1; Name: Мова програмування C#; Group: KP-02; Semester: 4
ID: 7; Name: Основи програмування; Group: KP-03; Semester: 5
ID: 10; Name: Веб-дизайн; Group: KP-03; Semester: 1
ID: 5; Name: Основи програмування; Group: KP-92; Semester: 4
ID: 8; Name: Основи програмування; Group: KP-91; Semester: 2
ID: 9; Name: Веб-дизайн; Group: KP-01; Semester: 2
ID: 4; Name: Веб-програмування; Group: KP-91; Semester: 1
```

```
ID: 3; Name: Мова програмування C#; Group: КР-91; Semester: 4
ID: 2; Name: Мова програмування C#; Group: КР-92; Semester: 2

Кількість елементів списку #2: 100000
Перші 10 елементів списку:
ID: 2006; Name: Основи програмування; Group: КР-01; Semester: 4
ID: 26117; Name: Веб-програмування; Group: КР-01; Semester: 1
ID: 30061; Name: Мова програмування C#; Group: КР-92; Semester: 1
ID: 141108; Name: Основи програмування; Group: КР-91; Semester: 4
ID: 91725; Name: Веб-дизайн; Group: КР-01; Semester: 4
ID: 15807; Name: Веб-дизайн; Group: КР-02; Semester: 1
ID: 26768; Name: Веб-дизайн; Group: КР-92; Semester: 2
ID: 109274; Name: Веб-дизайн; Group: КР-91; Semester: 5
ID: 117757; Name: Мова програмування C#; Group: КР-02; Semester: 4
ID: 146626; Name: Веб-програмування; Group: КР-02; Semester: 1

Друга частина виконана успішно! Дані з обох вхідних файлів оброблено та
записано у файл `./file3.csv`
```

Помилки:

```
Натисніть [Enter], щоб виконати другу частину: d
Помилка! Натисніть [Enter]!
```

Висновки

Під час виконання лабораторної роботи я навчився створювати і використовувати класи для об'єктів з даними без логіки та об'єктів динамічних колекцій елементів.

Також генерував та порядково обробляв велику кількость даних у форматі CSV.