**Звіт**

До проекту

Зі вступу до ООП

Тема «Комп’ютерний моніторинг і прогноз успішності студентів та контроль відвідування занять»

Студента групи ІПЗ-11

Майбороди Ігоря Сергійовича

Проект №46

**Загальний звіт**

1. **Мета:** на основі моделювання бізнес-процесів в системі моніторингу успішності та відвідуваності занять показати шляхи покращення оперативності обліку успішності та відвідуваності занять студентів.
2. **Об’єкти моделювання:** студенти, викладачі, дисципліни, журнал обліку успішності студентів.
3. **Задачі моделювання:** 
   1. Облік присутності студентів на лекціях та лабораторних заняттях
   2. Нарахування балів за виконані роботи та активність на лекціях
   3. Нарахування штрафів за порушення Deadline
   4. Нарахування балів за індивідуальні творчі, ініціативні роботи, участь у конференціях, конкурсах, олімпіадах тощо
   5. Підсумкові відомості перед іспитом
   6. Прогноз результатів іспиту за дисципліною
   7. Підсумкові відомості після іспиту
4. **Аналіз завдань:**

У ході роботи я маю створити консольний застосунок, який би був прототипом електронного журналу оцінок та обліку присутності студентів на заняттях.

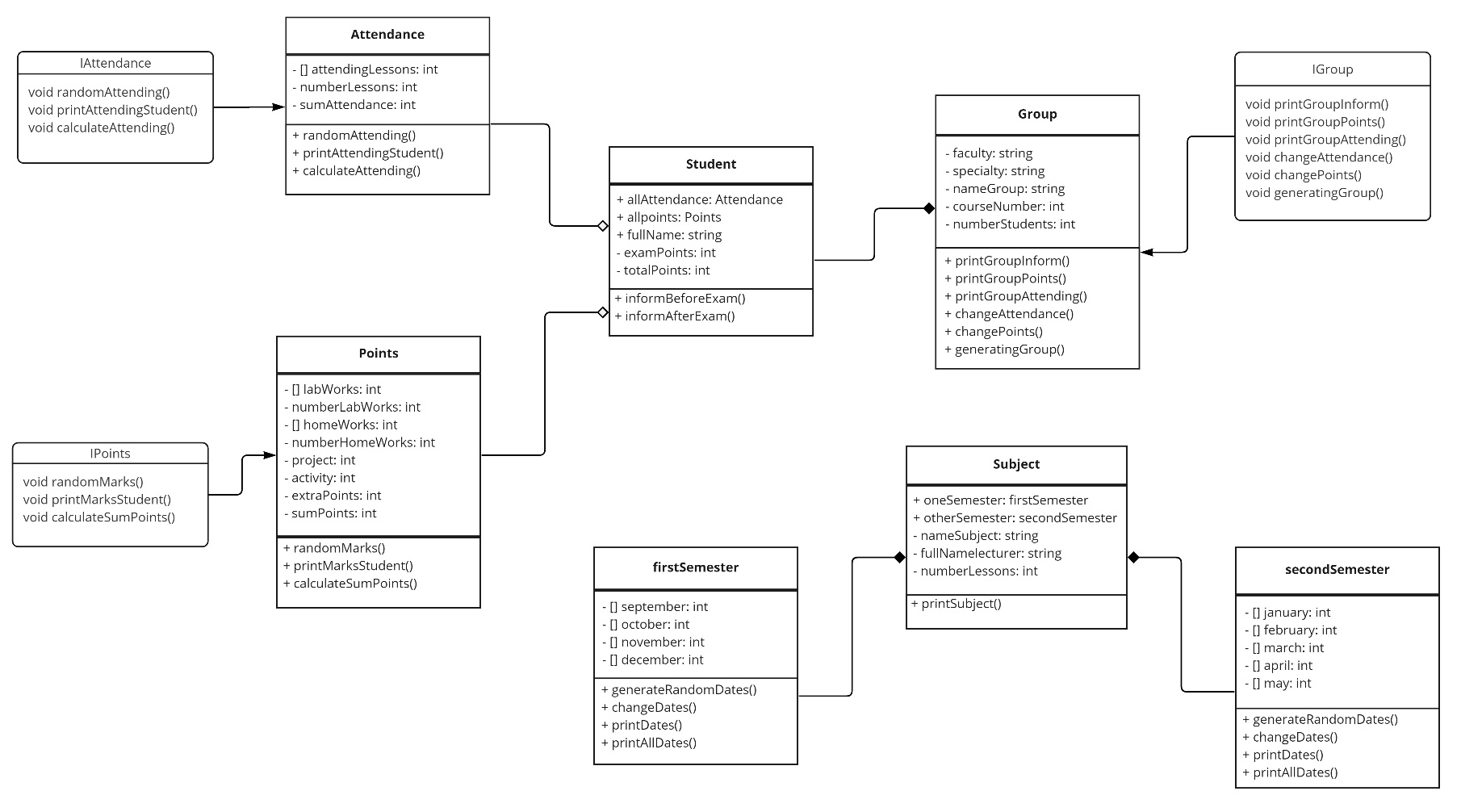
Я спочатку створив каркас своєї програми з 7 класів, обрав зв’язок композицію між класом Student та класом Group, бо група не може існувати без масиву студентів. Між класом Subject та класами firstSemester і secondSemester використана композиція теж, бо кожний предмет складається з 2 семестрів навчання або першого або другого. Агрегація у мене між класами Attendance і Points з класом Student, бо кожний студент має мати інформацію про його оцінки та відвідування занять. Детально можна побачити це на діаграмі класів.

У другій версії моєї програми я маю оголосити всі методи, реалізувати різні конструктори та властивості для кожного класу. Взагалі моя програма складається 2 блоків «Генерація даних» та «Меню». Спочатку кожний користувач має ввести певні дані про дати занять, кількість занять, лабораторних, домашніх робіт тощо. На основі внесених даних генерується журнал оцінок та облік присутності у вигляді 2 таблиць.

Далі у третій та четвертій версії програми я реалізую блок «Меню», саме в ньому користувач може вносити зміни до журналу оцінок та обліку відвідувань занять. Функції меню: «Робота з обліком присутності», «Робота з журналом оцінок», «Отримати відомості про студента до іспиту», «Отримати відомості про студента після іспиту», «Робота з операторами», «Перегляд обліку присутності та журналу оцінок».

Функція прогнозування результатів іспиту знаходиться саме у відомостях студента до іспиту, на консоль виводять поточні дані студента і набрана сума балів за семестр. Ймовірність скласти іспит рахується шляхом знаходження відсотку поточних балів за семестр від 60, а потім віднімається 5%, якщо студент не був присутній на половині занять, і навпаки додається 5%, якщо студент був присутній на більше 50% занять. Приблизний бал за іспит рахується за знайденим відсотком ймовірності від 40 балів.

Одна із функцій меню це Робота з операторами, в ній користувач може порівняти поточну успішність двох студентів, а потім знизити поточний бал першого на 5 і підвищити його у другого студента на 5 балів. Для порівняння використовується бінарний оператор, для віднімання чи додавання – унарний. Функція перегляду обліку присутності та журналу оцінок це просто виведення на екран оновлених даних про студентів на консоль у вигляді 2 таблиць.

1. **Діаграма класів**
2. **Текст програми**

Текст програми можна знайти на моїй сторінці GitHub (там буде і звіт):

[Maiboroda\_Igor](https://github.com/Virtuoso279/Maiboroda_Igor.git)

(посилання: <https://github.com/Virtuoso279/Maiboroda_Igor.git>)

**Звіт до Версії 1**

**«Об'єктно-орієнована декомпозиція предметної області завдання, побудова сценарію та розробка каркасу програми»**

* 1. Ознайомлення з предметною областю та постановкою задачі.
  2. Аналіз рекомендованих джерел інформації
  3. Пошук та аналіз додаткових джерел інформації за потребою.
  4. Розробка сценарію роботи програми
  5. Об'єктно-орієнована декомпозиція предметної області завдання (ідентифікаця об'єктів та їх взаємозв'язків)
  6. Розробка діаграми класів та їх взаємозв'язків. Мають бути враховані такі типи взаємозв'язків:
     1. асоціації;.
     2. агрегації;
     3. композиції.

Для першої версії зв'язок типу «успадкування» не використовувати.

* 1. Написання коду класів або генерація коду класів по розробленій діаграмі класів. Кожний клас має бути описаний в окремому файлі. В класах оголосити поля та методи відповідно до сценарію версії 1 програми. Код методів можна реалізувати в подальших версіях програми
  2. Сторення класу Program та функції Main(). Код функції Main() потрібний як точка входу в програму для її компіляції та запуску.
  3. Збірка програми, компіляція, запуск на виконання.
  4. Результат роботи програми:

ПІБ студента, курс, група

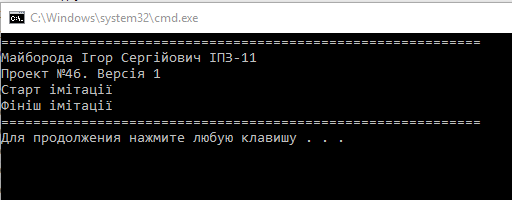
варіант завдання

версія 1

Старт імітації

Фініш імітації

**Результати роботи програми**



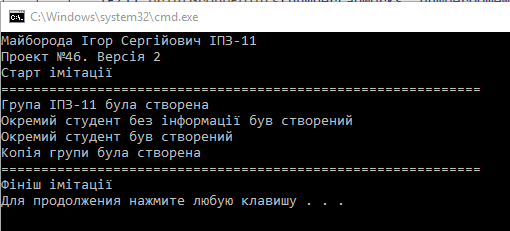
У першій версії просто імітація роботи програми

**Звіт до Версії 2**

**«Конструктори та аксесори класів»**

1. Скопіювати проект програми версії 1. Перейменувати його у версію 2.
2. Додати в кожний файл класу код, що реалізує конструктори класів.
3. Додати до кожного класу код, що реалізує аксесори класів.
4. Відповідно до C# документації <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/constructors> розробити для кожного класу такі типи конструкторів:
   1. Конструктори без параметрів
   2. Конструктори з параметрами (конструктори ініціалізації полів класу)
   3. Статичні конструктори
   4. Закриті конструктори
   5. Конструктори копій
   6. Конструктори, що викликають інші конструктори
5. В конструкторах класів врахувати взаємозв'язки класів:
   1. асоціації
   2. агрегації
   3. композиції
6. Передбачити такі різновиди методів доступу (аксесорів) https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/using-properties
   1. властивості, що автоматично реалізуютьсях
   2. закриті та відкриті властивості
   3. властивості читання та запису
7. У функцію Main() додати код, що створює об'єкти класів. Врахувати взаємозв'язки класів.
8. Вивести на екран протокол роботи конструкторів, аксесорів у вигляді значень атрибутів об’єктів класів

**Результати роботи програми**



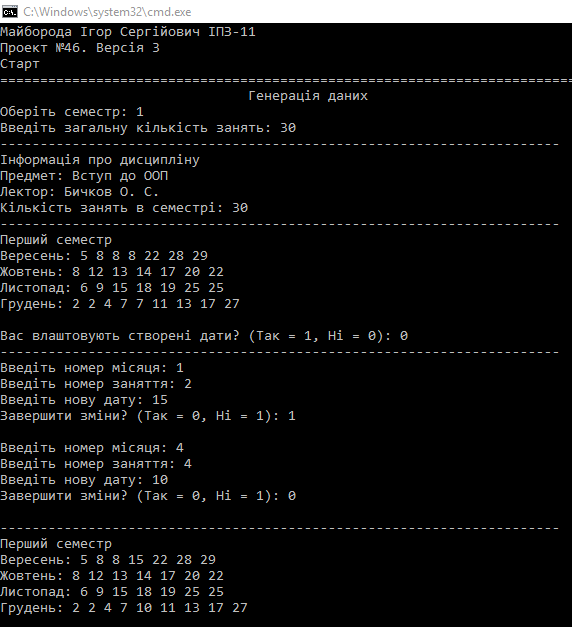
У другій версії теж імітація роботи.

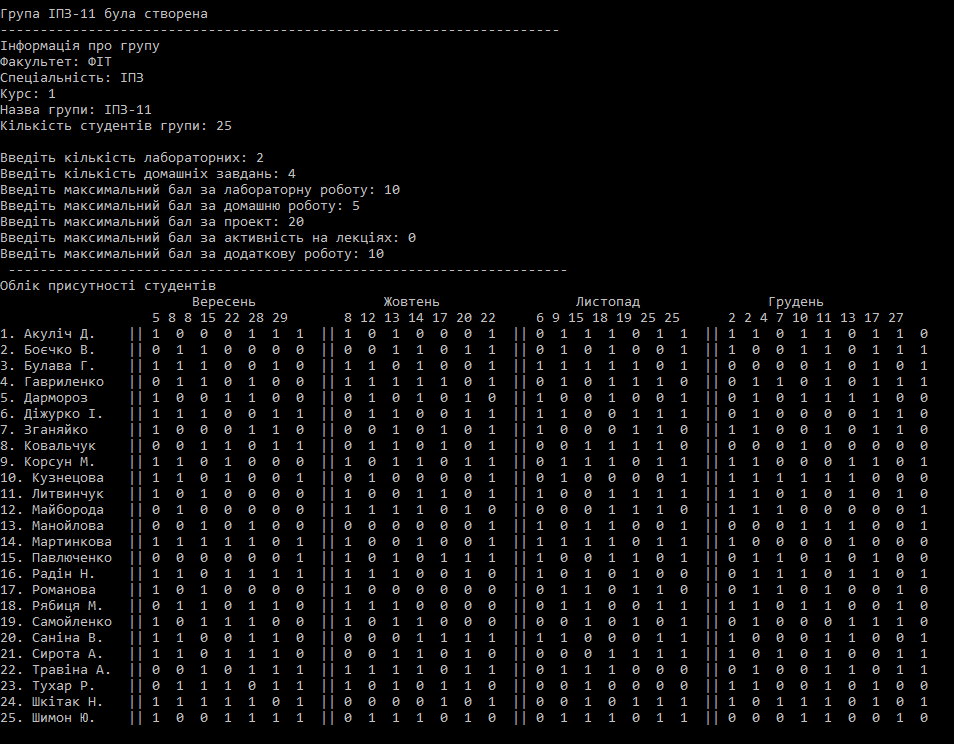
**Звіт до версії 3**

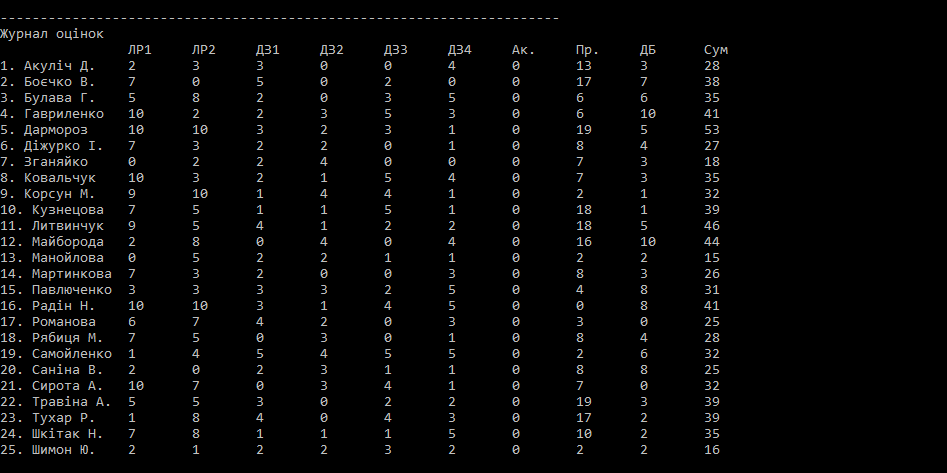
**«Реалізація предикатних функцій та деяких методів відповідно до постановки завдання»**

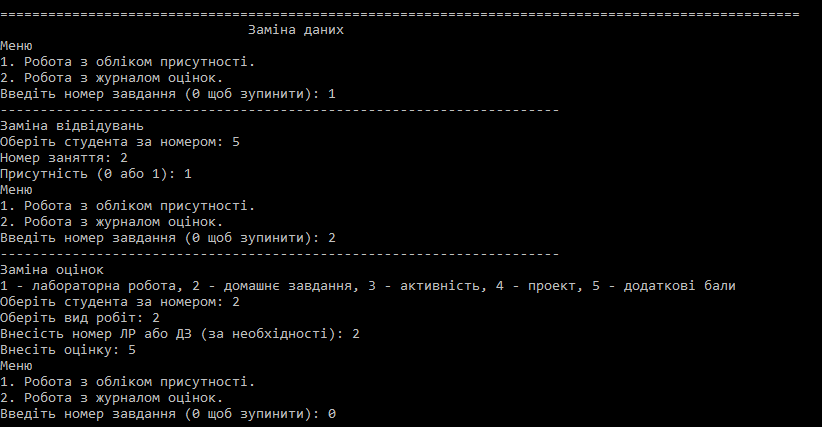
1. Скопіювати проект програми версії 2. Перейменувати його у версію 3.
2. Додати в кожний файл класу код, що реалізує предикатні функції, які повертають значення true/false і служать для визначення стану об'єктів.
3. Узгодити з Product owner (викладачем) пріоритетні задачі, які слід реалізувати в третій версії проєкту.
4. Розробити код методів класів відповідно до узгодженого з Product owner (викладачем) сценарію роботи версії 3 програми.
5. У функцію Main() додати код, що створює об'єкти класів. Врахувати взаємозв'язки класів.
6. Здійснити виклики методів, що узгодження з Product owner (викладачем) відповідно до сценарію роботи прогами.
7. Додатково до протоколу роботи програми версії 2 вивести результати роботи доданих до класів методів.

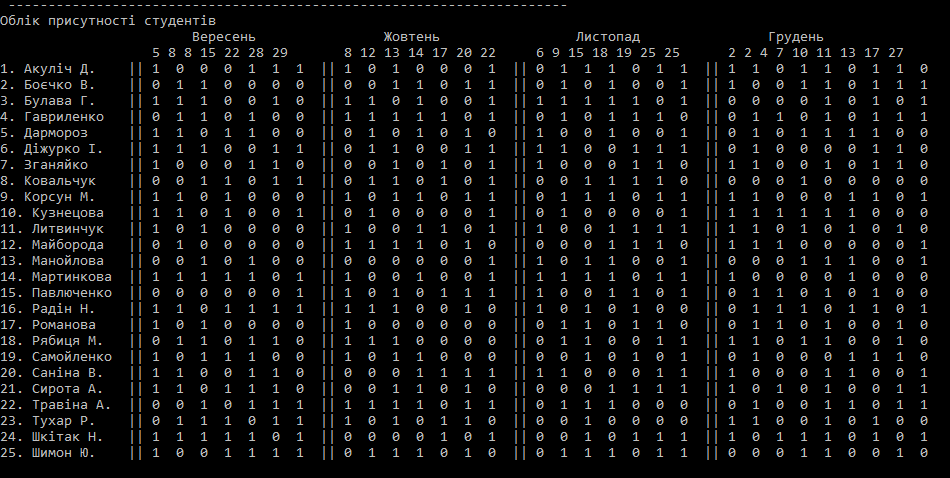
**Результати роботи програми**

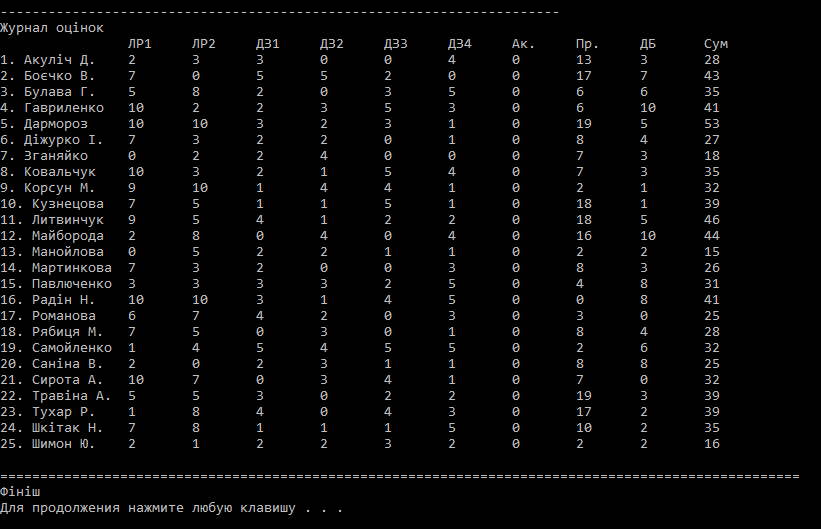












Аналіз достовірності результатів можна провести шляхом перегляду і порівняння даних на скріншотах.

**Звіт до версії 4**

**«Перевантаження операторів та реалізація методів другого пріоритету відповідно до завдання»**

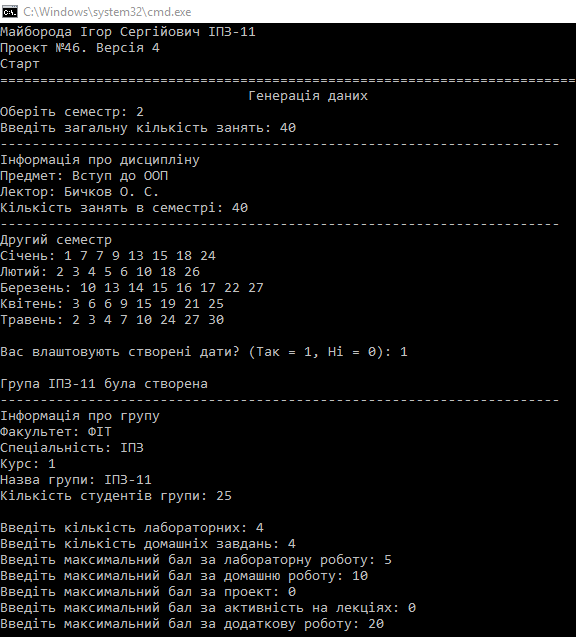
1. Скопіювати проект програми версії 3. Перейменувати його у версію 4.
2. Додати в класи код, що реалізує перевантаження операторів згідно з документацією: https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/operator-overloading

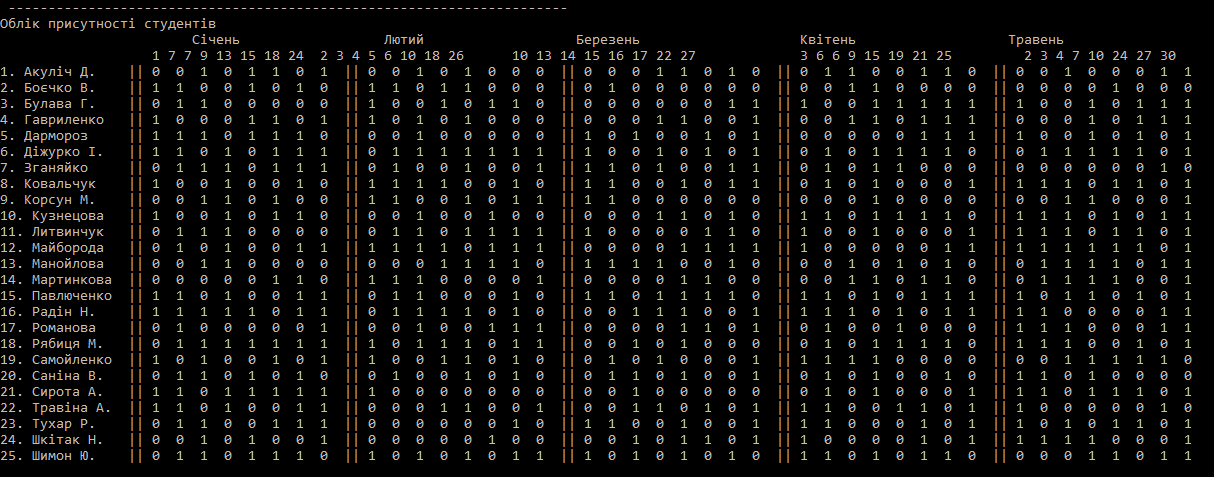
https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/language-specification/expressions#operator-overloading

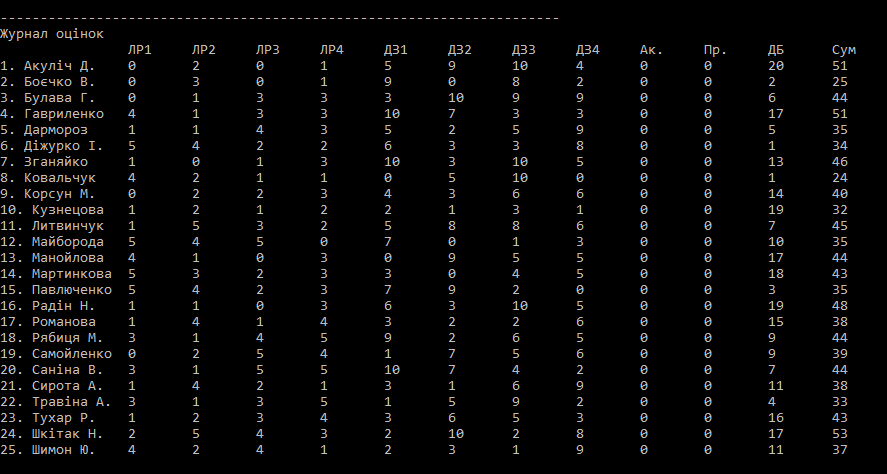
* 1. бінарні (арифметичні) оператори + - \* /
  2. унарні(арифметичні) оператори + - ++,--, !, true, false
  3. оператори порівняння ==. !=, <,>, <=, >=

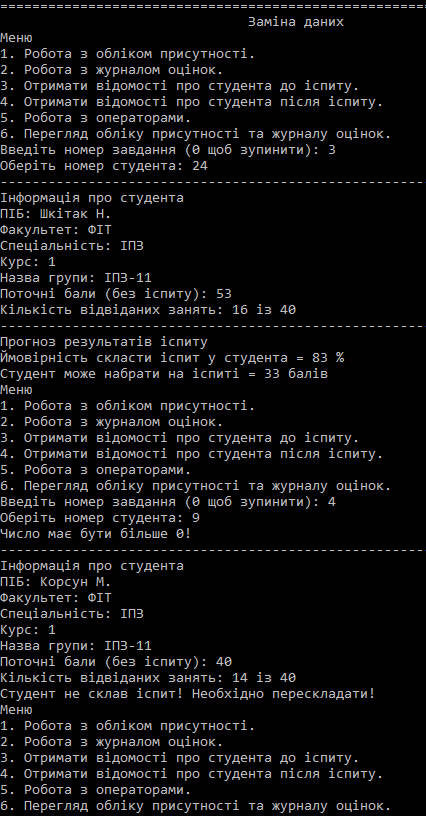
1. Додати в класи код методів, що реалізують сценарій роботи програми і є задачами другого пріоритету, узгодженими з Product owner (викладачем).
2. У функцію Main() додати код, що створює об'єкти класів. Врахувати взаємозв'язки класів.
3. Здійснити виклики методів, що перевантажують унірні та бінарні оператори.
4. Здійснити виклики методів, що узгодження з Product owner (викладачем) та відповідають сценарію роботи програми.
5. Додатково до протоколу роботи програми версії 3 вивести результати роботи доданих до класів методів.

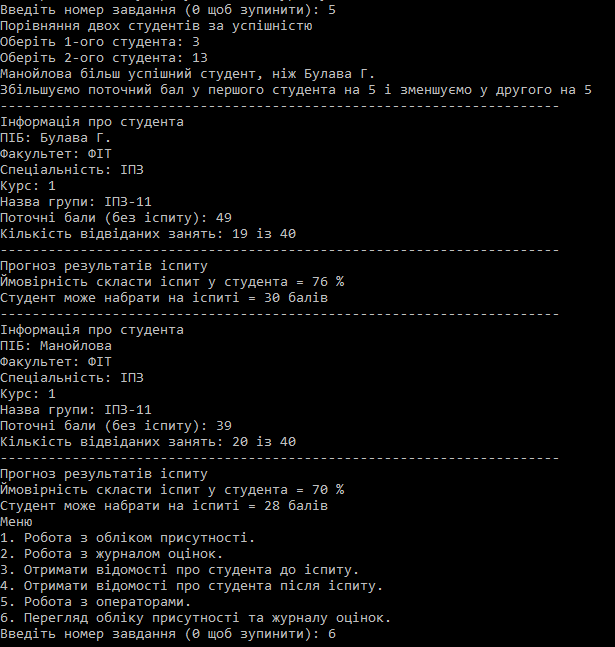
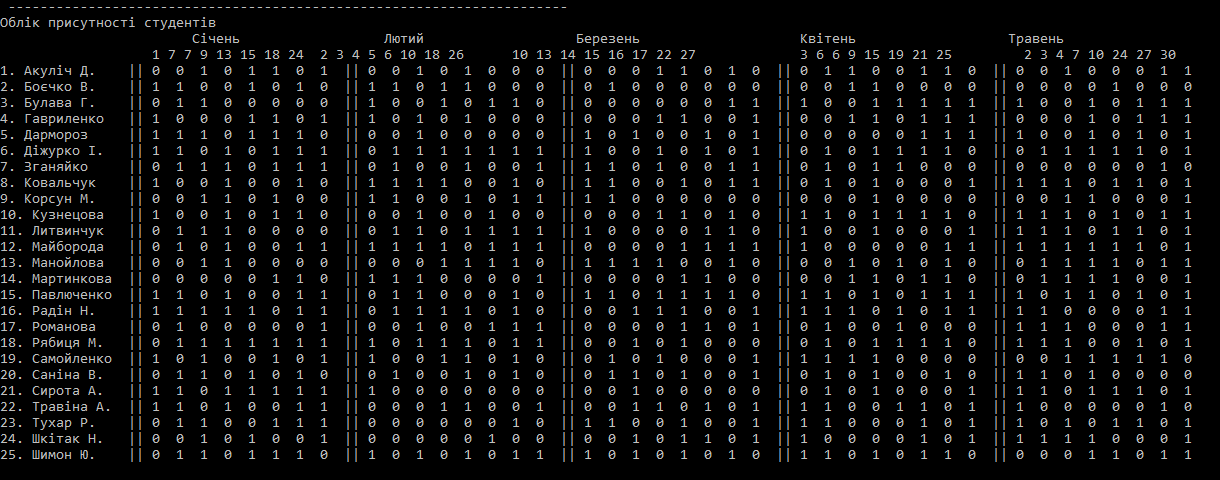
**Результати роботи програми:**

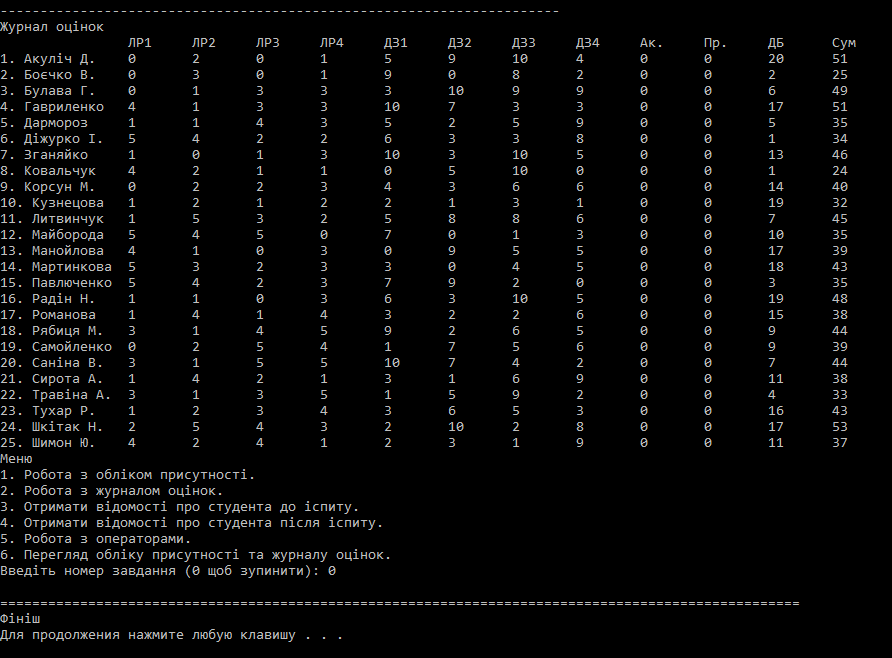












Аналіз достовірності результатів можна провести шляхом перегляду скріншотів вище.

**Звіт до версії 5**

**«Реалізація успадкування класів, інтерфейсів та абстрактних класів»** Скопіювати проект програми версії 4. Перейменувати його у версію 5.

1. Модернізувати діяграму класів, додавши додаткові базові, похідні, абстрактні класи та інтерфейси.
2. Додати в класи код, що реалізує успадкування класів (базові, абстрактні, інтерфейси) відповідно до діаграми класів.
3. Модернізувати конструктори класів, врахувавши конструктори базових класів під час розробки конструкторів похідних класів

<https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/inheritance>

1. Додати в класи код методів, що реалізують сценарій роботи програми і є задачами третього пріоритету, узгодженими з Product owner (викладачем).
2. У функцію Main() додати код, що створює об'єкти базових та похідних класів. Врахувати взаємозв'язки класів.
3. Здійснити виклики методів, що узгодження з Product owner (викладачем) та відповідають сценарію роботи програми.
4. Додатково до протоколу роботи програми версії 4 вивести результати роботи доданих до класів методів.

**Результати роботи програми**

Загалом реалізація і функціоналу у 5-ій версії практично не відрізняється від 4-ої версії, саме тому необхідності дублювати сюди скріншоти я не бачу. Основна відмінність 4-ої та 5-ої версій саме на додаванні 3 інтерфейсів і реалізація їх в 3-ох класах, тобто зміни лише в коді, а побачити їх в консолі чи при компіляції програми не можна.

1. **Загальний висновок**

У ході роботи була програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач моніторингу успішності та обліку відвідування занять студентами. Проект складається з 5 версій.

Щодо оптимізації коду програми, то я можливо краще б реалізував дизайн таблиць під час виведення на консоль обліку присутності та журналу оцінок. Можливо зменшив розмір коду в статичному методі menu, але для цього потрібно було дописати додаткові методи в певні класи.