Виконала: Коноз Наталія

Варіант: 7

**DOT NET**   
**Лабораторна робота № 4**

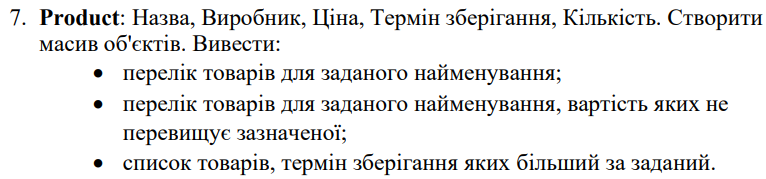
Тема: Використання принципів та відносин об’єктно-орієнтованого програмування у мові C#

Мета: набути навички в написанні програм із застосуванням принципів та відносин об’єктно-орієнтованого програмування (ООП).

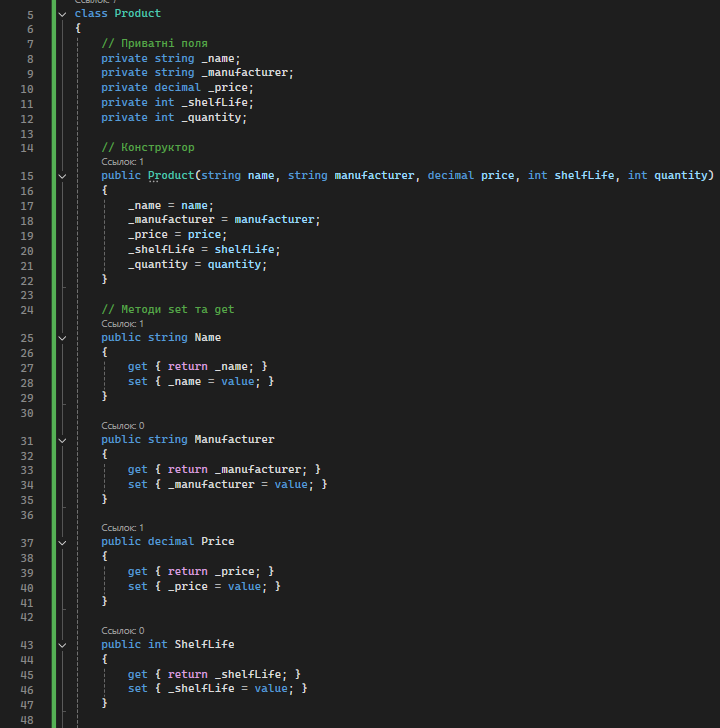
Індивідуальне завдання:

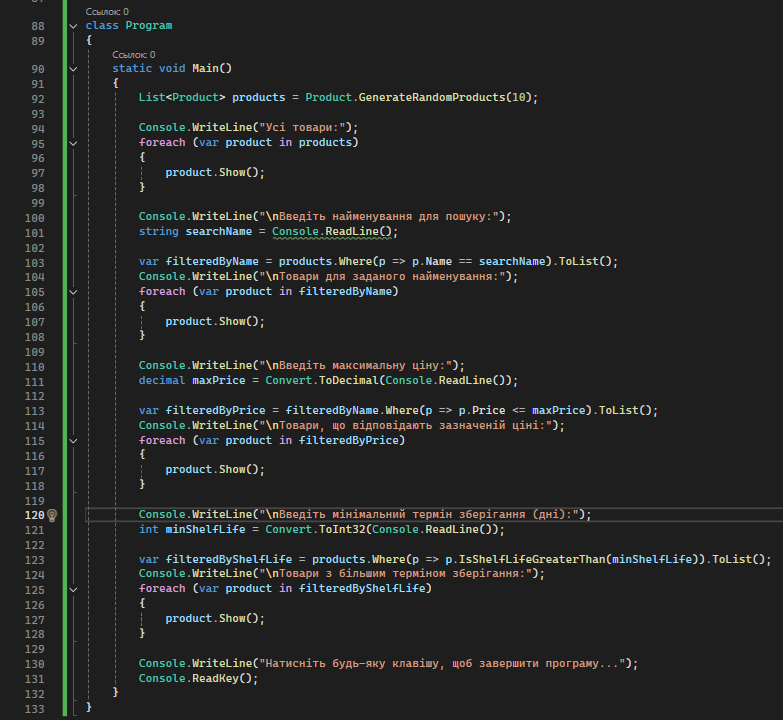
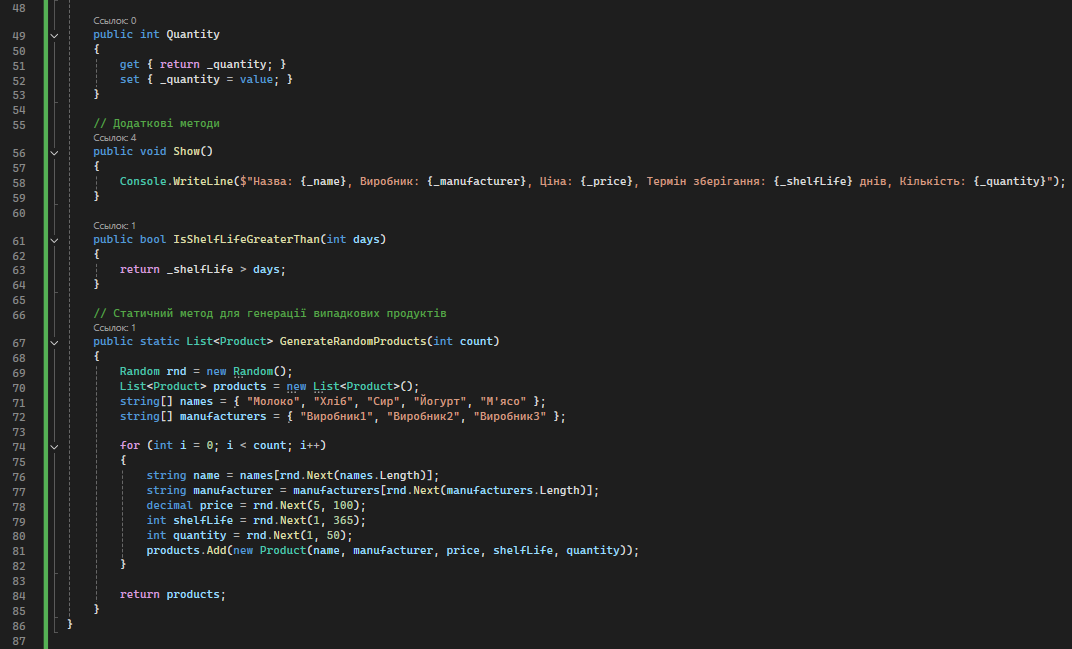
**1.**

Розробити клас згідно до свого варіанта. Включити до класу методи set (…), get (…), show (…) та ін., використовуючи принцип інкапсуляції. Окрім гетерів, сетерів конструкторів та інших стандартних методів розробити не менше двох додаткових методів, які реалізують власний функціонал об’єкту відповідної предметної галузі. Написати програму, яка створює список, масив чи колекцію, об’єктів на основі відповідного класу. Під час створення використати випадковий підхід для генерації числової та текстової інформації об’єктів списку. Передбачити можливість додання, редагування та видалення об’єктів зі списку за певними критеріями. В програмі реалізувати функціонал за своїм варіантом

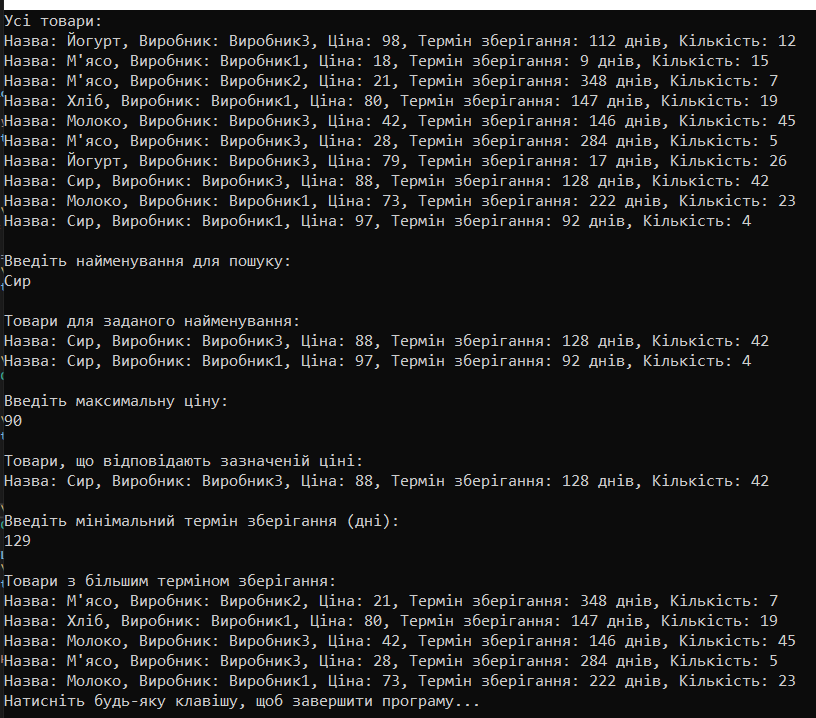


Код:





Що виводиться на екран:



**2.**

2.1 Для класу з минулого завдання, ретельно занурившись у предметну область власного варіанта, створити абстрактний клас чи інтерфейс, який має стати предком, зв’язати ці класи відносинами спадкування. Наприклад для класу Book (Книга) можна створити клас Reader (Засіб читання).

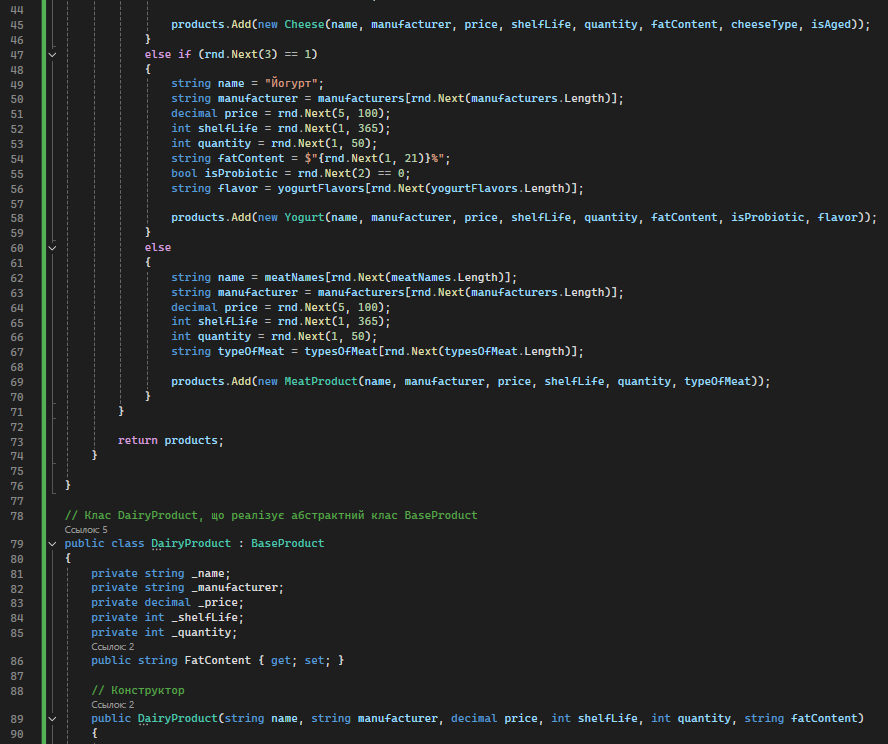
2.2 Окрім вашого класу додати до абстрактного класу чи інтерфейсу, створеному у попередньому пункті, ще два класи-спадкоємця першого рівня. Наприклад до класу Book (Книга) можна додати класи Newspaper (Газета) Magazine (Журнал) з відповідними полями та методами.

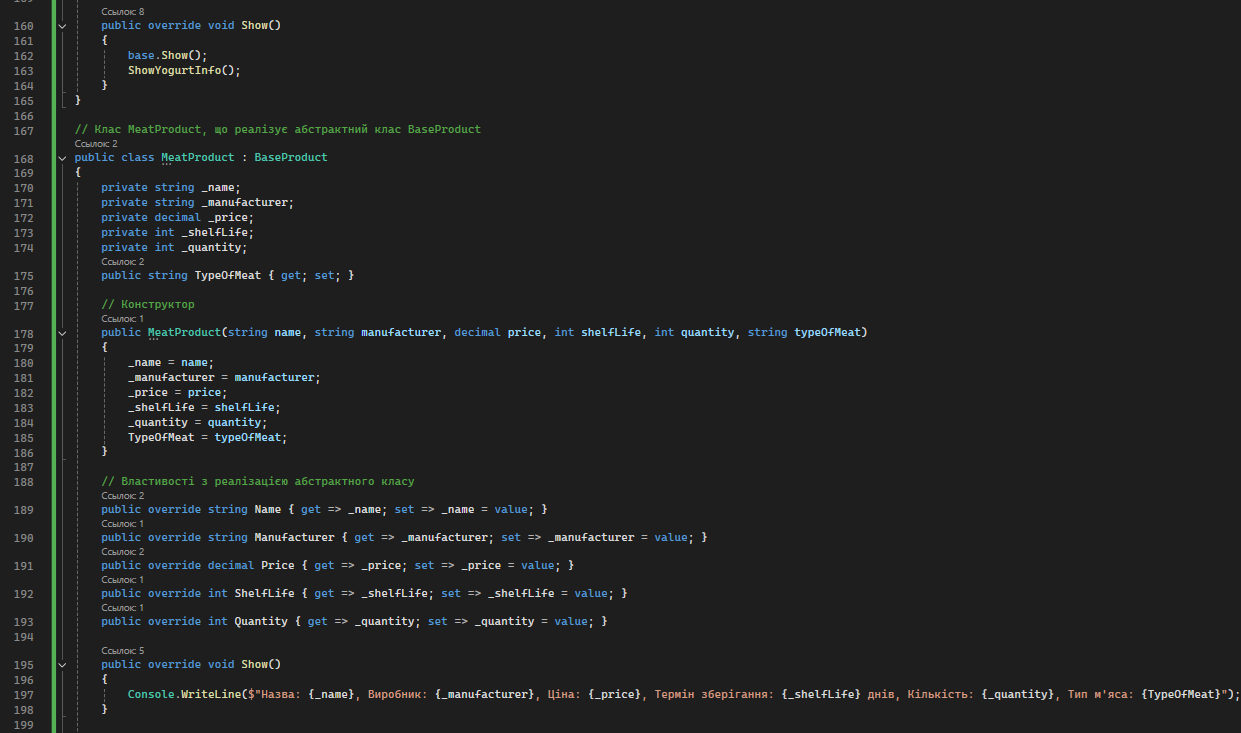
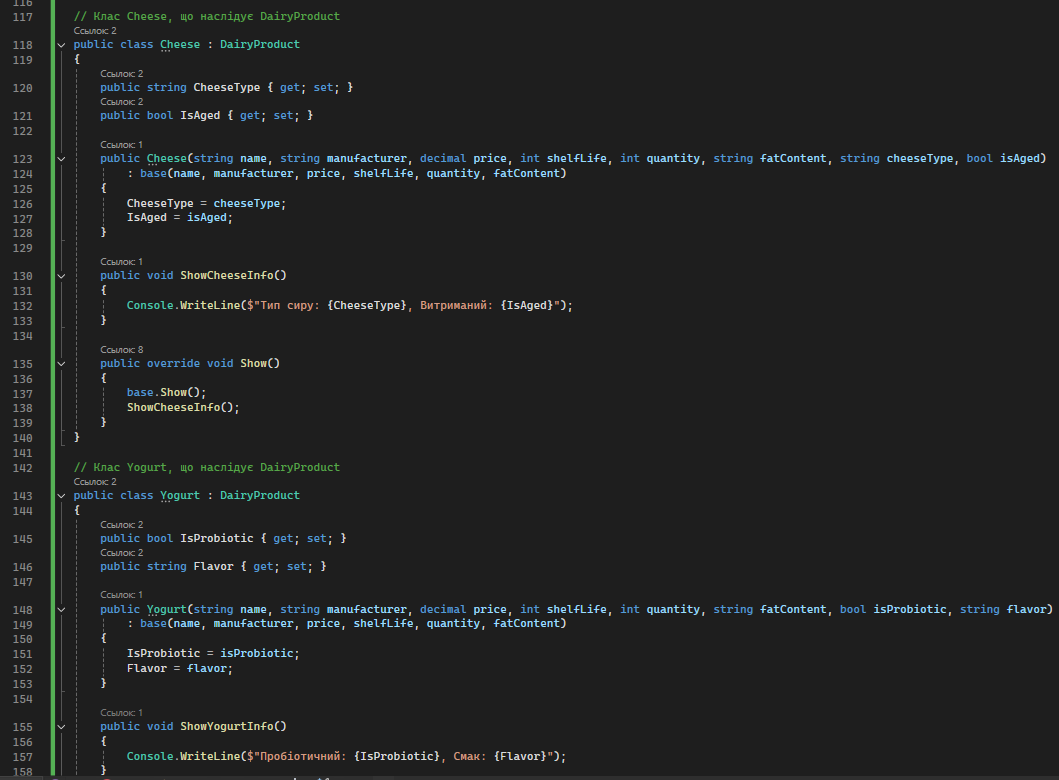
2.3 Для одного з доданих у попередньому завданні класів створити не менше двох спадкоємців другого рівня спадкування. Наприклад до класу Magazine (Журнал) можна додати класи PMagazine (Паперовий журнал), EMagazine (Електронний журнал), IMagazine (Інтернет журнал). У кожного нащадка має бути не менш ніж на два поля та метода більше ніж у предка.

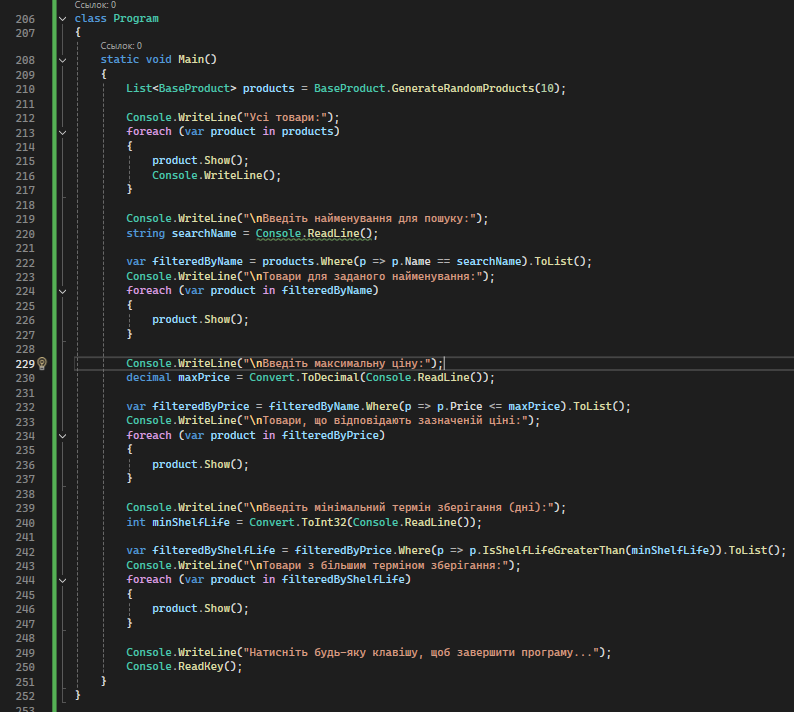
2.4 Створити масив чи колекцію об’єктів усіх рівнів спадкування. Виконати відповідні функції над поліморфним списком із завдання власного варіанта.

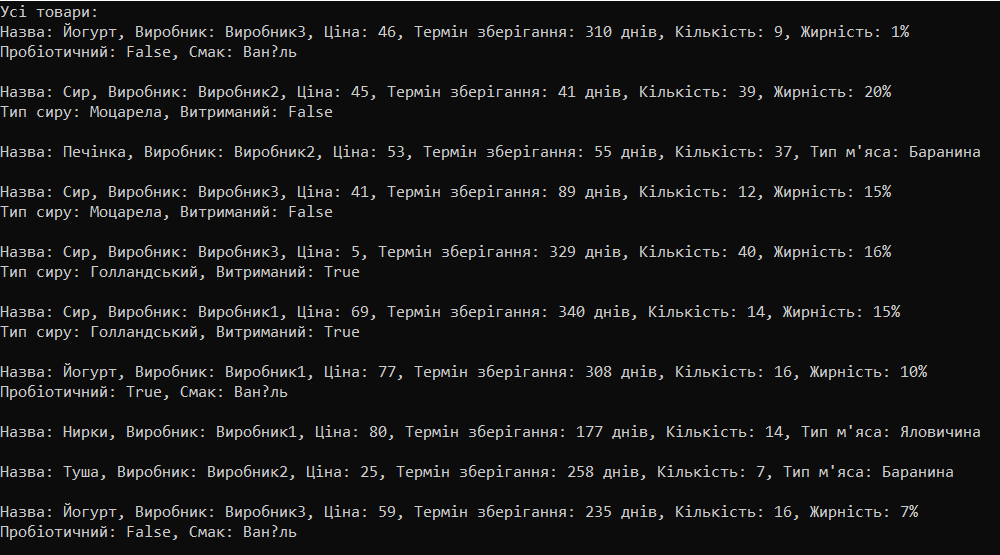
Код:

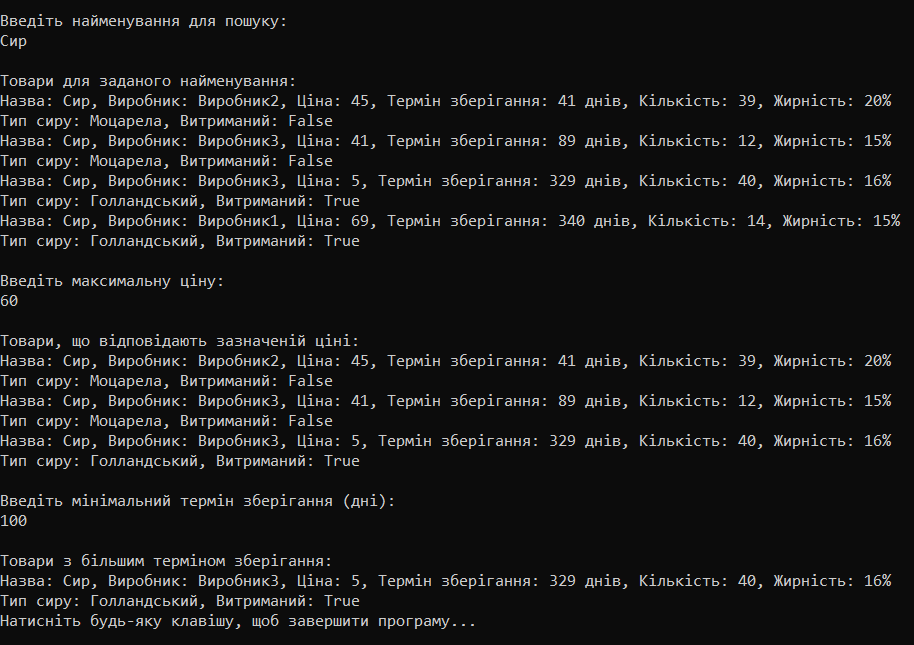




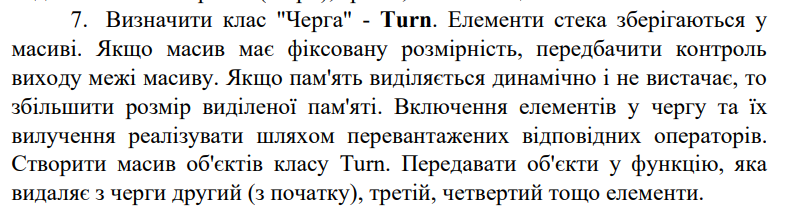


 Що виводиться на екран:

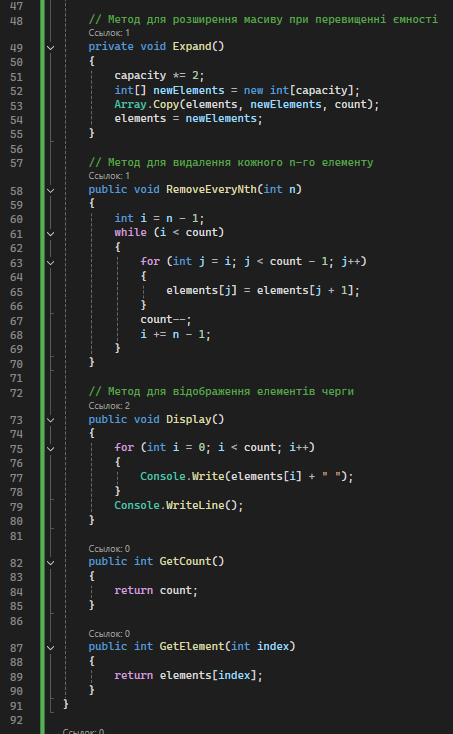
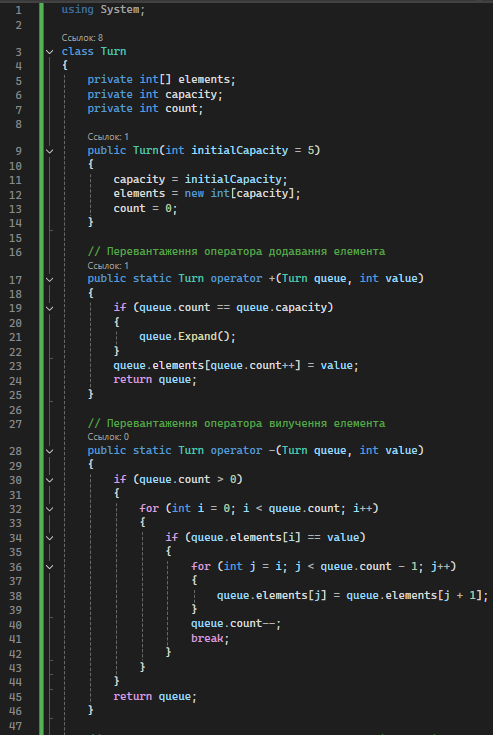


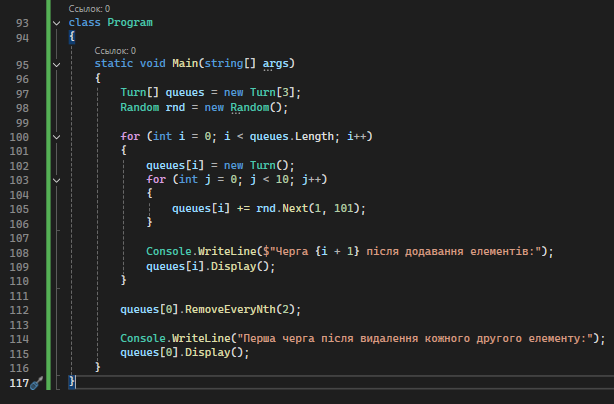


**3.** Розробити клас та перевантажити оператори згідно до свого варіанта. В головній функції програми протестувати роботу створених класів на прикладі використання окремих об’єктів та масивів чи колекцій цих об’єктів. Під час створення об’єктів застосувати випадковий підхід чи зчитування інформації з файлів.

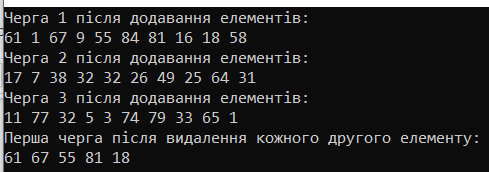


Код:





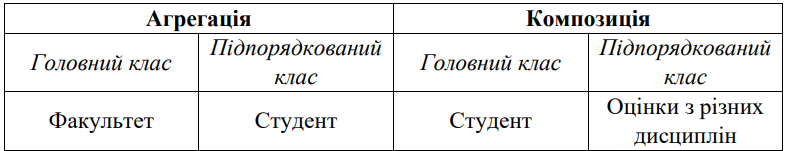
Що виводиться на екран:



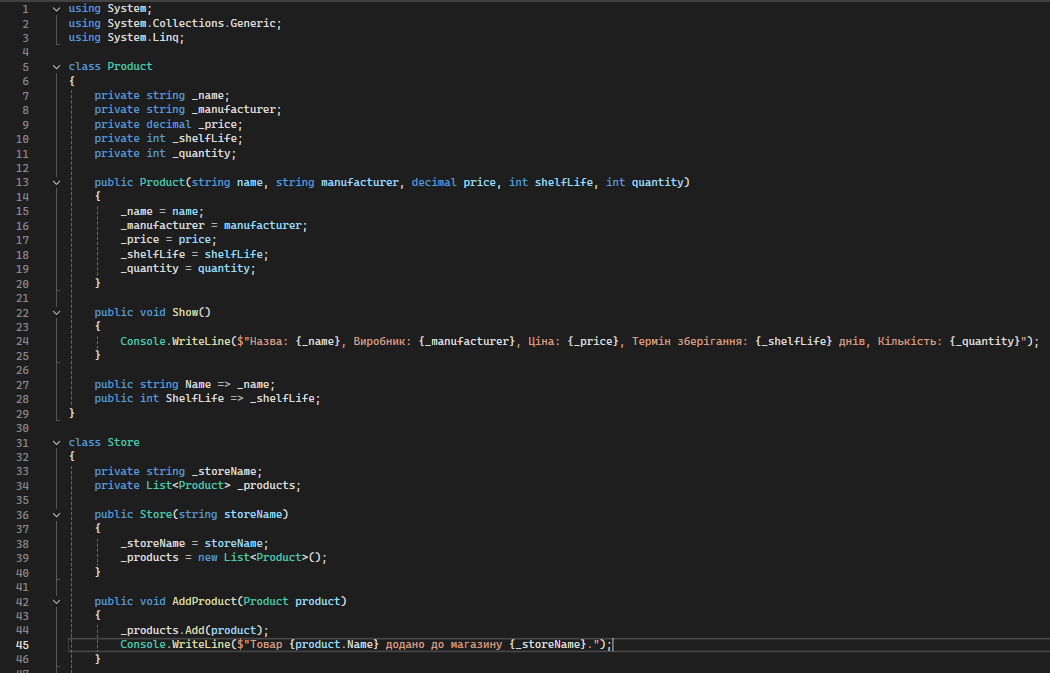
**4.**

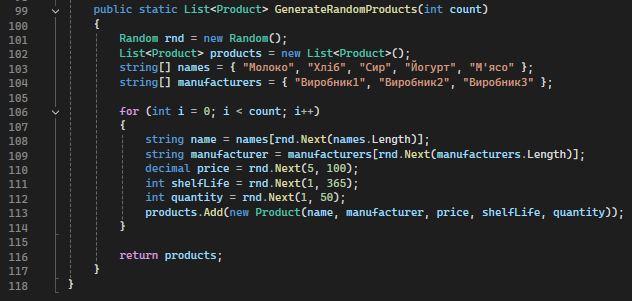
4.1 Для класу першого згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини агрегації. Продемонструвати роботу відносин, створивши декілька об’єктів відповідних класів та позбавляючи чи додаючи властивості до цих об’єктів.

4.2 Для класу першого завдання згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини композиції. Продемонструвати роботу відносин, створивши декілька об’єктів відповідних класів та позбавляючи чи додаючи властивості до цих об’єктів.

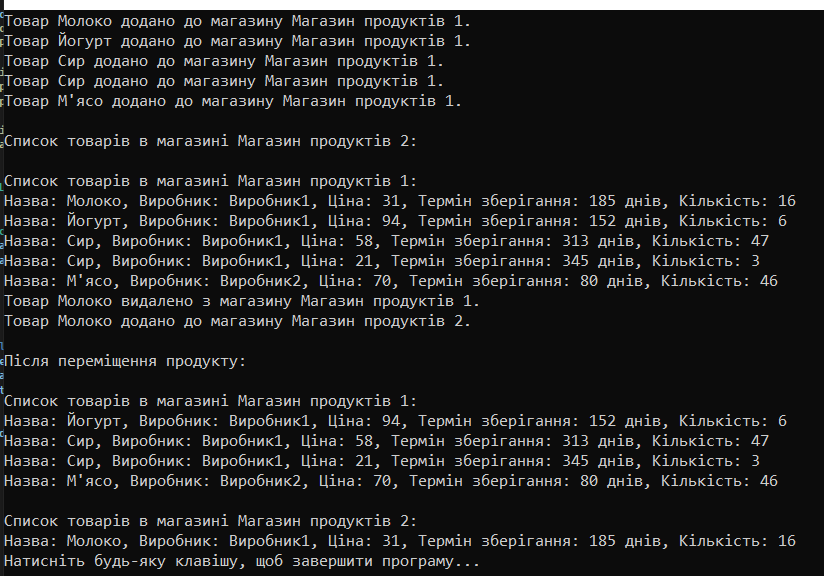
 **Агрегація:**

Код:



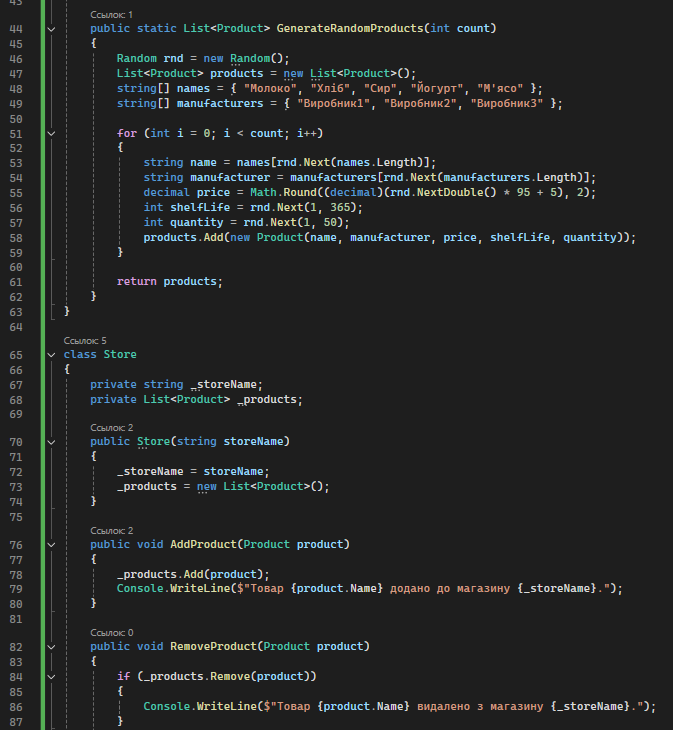
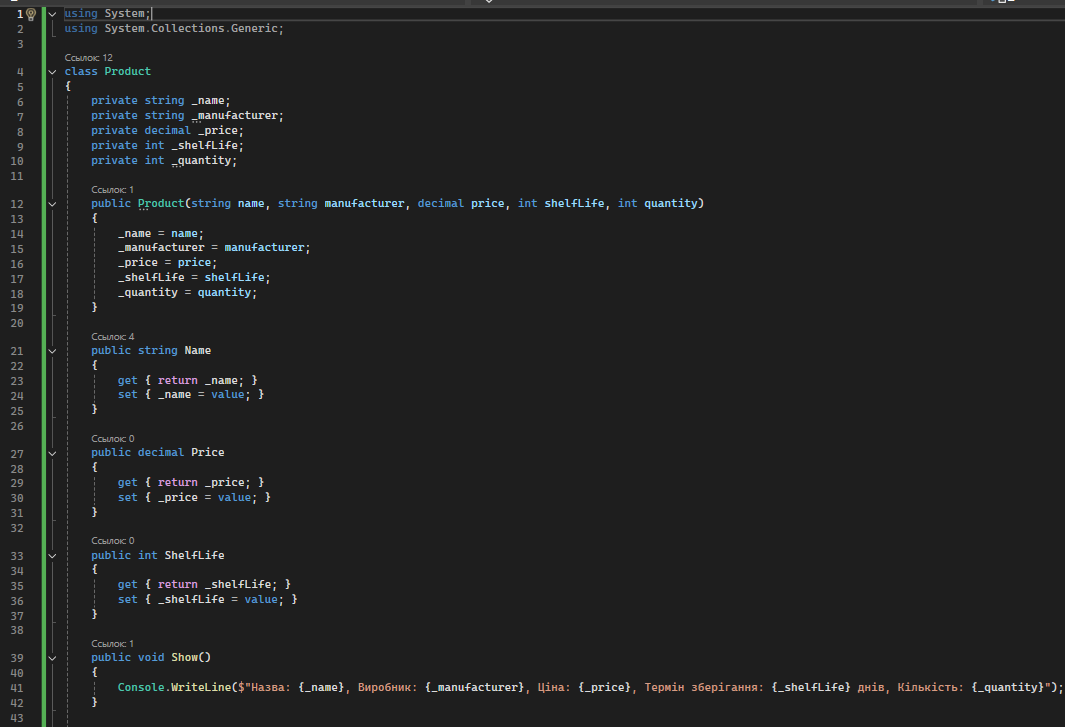


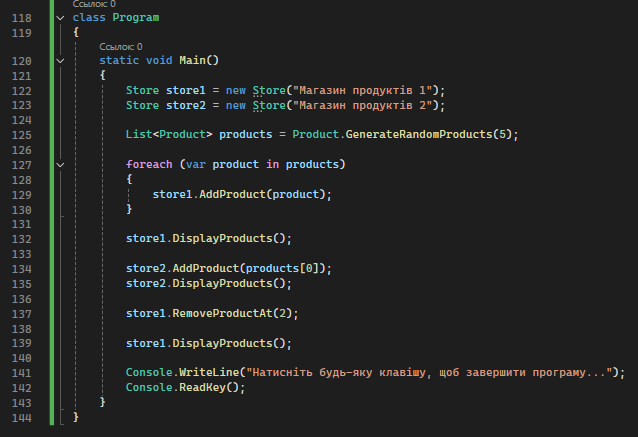
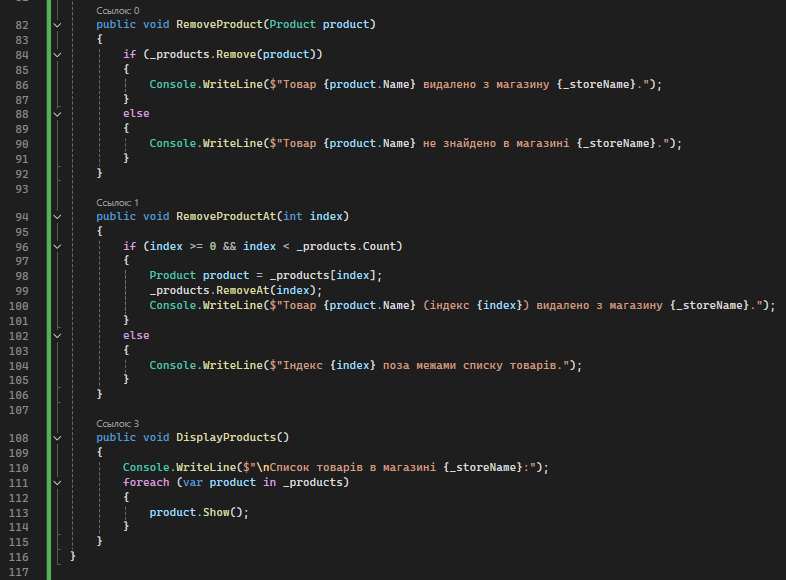
Що виводиться на екран:



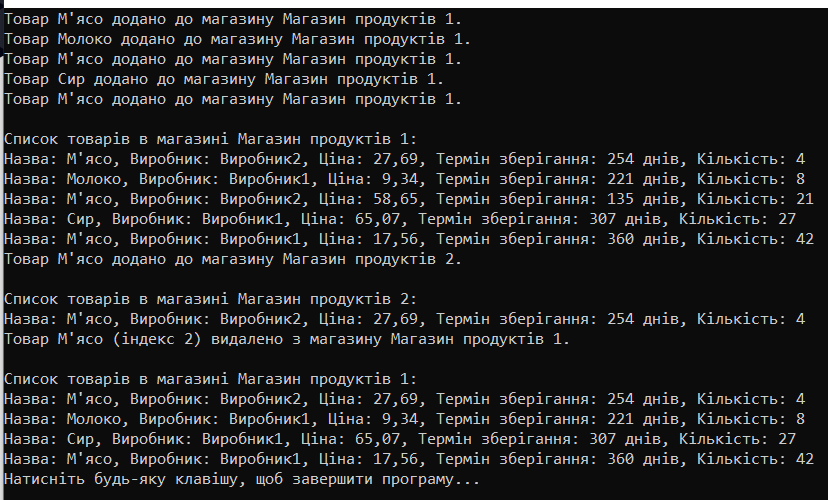
**Композиція:**

Код:





Що виводиться на екран:



Висновки:

У виконаній роботі на тему "Використання принципів та відносин об’єктно-орієнтованого програмування у мові C#" було реалізовано клас Product з властивостями товарів та методами для їх управління, продемонстровано генерацію випадкових об'єктів і можливості їх додавання, редагування та видалення. Було створено абстрактний клас та класи-нащадки для реалізації спадкування та поліморфізму. Реалізовано перевантаження операторів для класу "Черга" (Turn) з контролем за розміром масиву та динамічним виділенням пам’яті. Також продемонстровано відносини агрегації та композиції, що підкреслило важливість принципів об’єктно-орієнтованого програмування для створення ефективних програм у C#.