Лабораторна робота № 4

Проектування та реалізація класів засобами C#. Модульне тестування

Вимоги до роботи:

- У розроблених класах необхідно реалізувати такі концепції ООП, як "успадкування" та "поліморфізм". Потрібно описати *базовий клас* (клас-предок) та наслідувати від нього.
- Протестувати *у повному обсязі усі реалізовані можливості* з використанням тестового проекту та *юніт-тестів*.
- Частину функціональності класів треба *реалізувати з використанням властивостей*.
- Необхідно реалізувати підтримку *серіалізації* об'єктів класів.
- Потрібно описати *власний інтерфейс*, описати у ньому принаймні 5 методів та властивостей та реалізувати їх у класі.
- У завданнях на узагальнені (параметризовані) класи завдання пункту d) передбачають реалізацію параметризованих версій відповідних інтерфейсів.
- Один з методів потрібно реалізувати як *метод-розширення*.
- Кожний клас *обов'язково має містити* принаймні 3 конструктори, хоча б один статичний метод та перевизначення методу `ToString()`.
- При написанні класів *не можна використовувати класи та методи розширень із просторів імен* `System.Linq`, `System.Collection.Generic` та `System.Collection` (списки, асоціативні масиви, черги, стеки, множини і т. п.). У варіантах, у назві класів у яких є «*», обмеження на використання колекцій відсутнє. Використання масивів допускається у всіх варіантах.
- Реалізація інтерфейсу `IEquatable` має містити метод порівняння `Equals`, який не є тотожнім до операції ==.
- З використанням тегу <summary> додати коментарі до методів.
- *Протестувати у повному обсязі усі реалізовані можливості* з використанням тестового проекту та *юніт-тестів*.

Варіанти завдань

Варіант № 1

Написати клас "пряма". Реалізація має містити:

a) статичний метод перевірки паралельності прямих, метод знаходження точки перетину прямих, метод обчислення відстані від точки до прямої;
b) властивості `IsVertical` та `IsHorizontal`;
c) операцію порівняння прямих;
d) підтримку інтерфейсів `ICloneable`, `IEquatable`.

Варіант № 2

Написати узагальнений клас для реалізації *однонапрямленого списку* (без

зворотних зв'язків). Реалізація має містити:

a) функції-члени для вставки елемента в початок списку, видалення елемента з початку списку, пошуку елемента у списку, видалення 1-го входження елемента, вставка елемента у список за заданим елементом, добавлення у поточний список усіх елементів іншого списку;
b) властивості `Count`, `Head` та індексатор;
c) операції ==, + та * (має повертати список спільних елементів двох списків);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 3

Написати реалізацію класу *"булевий вектор"*. Реалізація має містити:

a) статичний метод Distance для знаходження відстані Хеммінга між двома булевими векторами та методи RotateLeft, RotateRight (циклічні зсуви);
b) індексатор, властивості Dim та Norm (норма Хеммінга);
c) операції !, &, |, ^, <<, та >>;
d) підтримку інтерфейсів IEquatable, ICloneable, IEnumerable.

Варіант № 4

Написати клас для реалізації *векторів* 3-*вимірного простору* з дійсними коефіцієнтами. Реалізація має містити:

a) метод `Length` обчислення довжини вектора та статичну функцію знаходження відстані між векторами;
b) властивості для звертання до координат вектора та властивості `IsInteger` та `IsPositive`.
c) операції над векторами (+, –, | — паралельність, скалярний та векторний добутки);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`, `IComparer` та `IComparable`.

Варіант № 5

Написати параметризований клас для реалізації двонапрямленого списку (із зворотними зв'язками). Реалізація має містити:

a) функції-члени для вставки елемента після і перед заданим елементом, видалення заданого елемента списку, пошуку елемента у списку;
b) властивості `Count`, `Head`, `Tail` та індексатор;
c) операції "==", "*" (має повертати список спільних елементів двох списків) та "–" (має повертати список елементів першого списку, які не належать другому списку);
d) підтримку інтерфейсів `ICloneable`, `ICollection`, `IEquatable`.

Варіант № 6

Написати клас для реалізації бінарних відношень на множині {1,2,…, *N*}. Реалізація має містити:

a) статичний метод `Inverse` (має повертати обернене відношення), методи `Add`

(для додавання пари елементів, які перебувають у відношенні), `Remove` (видалення пари елементів, які перебувають у відношенні), `Clear` та `Contains` (перевірка входження пари у бінарне відношення);
b) властивості `N`, `Count` (кількість пар, які перебувають у відношенні), `IsIdentity` (є відношенням ідентичності) та індексатор;
c) операції + (об'єднання), – (різниця) та `*` (добуток відношень).
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 7

Написати клас для подання числової прямокутної матриці. Реалізація має містити:
a) метод для обчислення норми матриці та метод для множення матриці на вектор;
b) індексатор та властивості `RowCount`, `ColumnCount`;
c) операції над матрицями (множення на число, додавання, віднімання, множення матриць, порівняння);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 8

Написати реалізацію класу "коло" та його нащадка — класу "циліндр". Реалізація останнього класу має містити визначення циліндра у 3-вимірному координатному просторі, вісь якого паралельна осі OZ. Передбачити:
a) функції-члени SurfaceArea, Volume, Contains (перевірка належності точки циліндру);
b) властивості Radius, Height та індексатор (має надавати доступ до центрів основ циліндра);
c) операції "==", "+" (зсув циліндра на заданий вектор), "*" (має повертати циліндр з масштабованим радіусом), < (має повертати True, якщо 1-й циліндр міститься усередині 2-го);
d) підтримку інтерфейсів ICloneable, IEquatable, IComparable (вважати, що більшим є циліндр більшого об'єму).

Варіант № 9

Написати реалізацію класу "*прямокутний паралелепіпед*". Реалізація має містити:
a) статичний метод `Surface` (обчислення площі поверхні) та метод `Translate` (паралельне перенесення поточного паралелепіпеда на заданий вектор);
b) властивості `Vertices` (вершини), `Volume` (об'єм) та `Center` (центр);
c) операції "==", ">" та "`*`" (множення координат усіх вершин на число);
d) підтримку інтерфейсів `ICloneable`, `IEquatable` та `IEnumerable`.

Варіант № 10

Написати реалізацію класу "*булева функція*". Реалізація має містити:
a) методи `GetNumber` (має повертати номер вектора значень функції), `GetDual` (знаходження двоїстої функції) та `Reduce` (видалення фіктивних змінних);
b) індексатор, властивості `IsIdenticallyTrue` (перевірка загальнозначущос-

ті), `IsIdenticallyFalse` (перевірка суперечливості), `IsSelfDual` (самодвоїстість), `IsMonotonic` (монотонність), `IsLinear` (лінійність);

c) визначення операцій `!`, `&`, `|`, `^` та ==;
d) підтримку інтерфейсів `IEquatable`, `ICloneable`, `IEnumerable`.

Варіант № 11

Написати узагальнений клас із двома параметрами для реалізації асоціативного масиву. Реалізація має містити:

a) методи пошуку, добавлення та видалення елементів;
b) властивості `Count`, `Keys`, `Values` та індексатор;
c) операції "==" та "<=" (має повертати `true`, якщо усі пари ключ-значення першого асоціативного масиву належать другому масиву);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 12

Написати реалізацію класу "*еліпс*". Вважати, що осі еліпсі паралельні осям координат. Реалізація має містити:

a) методи `Area` — обчислення площі еліпса та `Contains` — перевірка належності точки фігурі, яка обмежується еліпсом;
b) властивості `Center`, `A`, `B`, `Eccentricity`;
c) операції + (зсув еліпса на заданий вектор), == та * (масштабування еліпса);
d) підтримку інтерфейсів `IEquatable`, `ICloneable`.

Варіант № 13

Написати клас для реалізації *векторів n-вимірного простору* з дійсними коефіцієнтами. Реалізація має містити:

a) статичний метод знаходження евклідової відстані між векторами, метод `Abs` та метод `ToArray()`;
b) індексатор та властивості `Length` (кількість координат) та `Norm` (норма);
c) операції "==", "+", "−", "*" (скалярний добуток векторів).
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IComparable`.

Варіант № 14

Написати дві різні реалізації узагальненого класу, які реалізують поняття "*множина елементів*". Реалізація має містити:

a) методи для добавлення, вилучення та пошуку елементів;
b) властивості `Count` та `IsSingleton` (містить один елемент);
c) операції + (об'єднання), * (перетин) та − (різниця);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 15

Написати узагальнений клас для реалізації *впорядкованого списку*. Реалізація має містити:

a) функції-члени для вставки елемента у список, видалення елемента із списку, `IndexOf` (знаходження позиції першого входження елемента);
b) властивості `Count`, `Min`, `Max` та індексатор;
c) операції ==, ^ (симетрична різниця двох списків), <=;
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 16

Написати реалізацію класу "*трикутник*". Реалізація класу має містити:
a) статичний метод `Area` — знаходження площі та метод `Interior` — перевірка того, чи лежить точка усередині трикутника;
b) властивості `Vertices` (вершини трикутника), `IsIsosceles` та `IsRight-Angled` (перевірка, чи є трикутник рівнобедреним чи прямокутним);
c) операції "+" — паралельне перенесення трикутника на заданий вектор, "==" та "/" (зменшення координат);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable` та `IEquatable`.

Варіант № 17

Написати клас для реалізації *точок у 2-вимірному просторі*. Реалізація має містити:
a) метод `Deconstruct` для повернення координат точки, функцію, яка повертає номер чверті, у якій лежить точка, `GetSymmetric` (має повертати точку, симетричну відносно заданої координатної осі чи точки), та статичні функції `Distance` (знаходження відстані між точками) та `Parse`;
b) властивості для звертання до координат точки та властивості `OnAxes` (перевірка того, чи лежить точка на осі координат);
c) покоординатні операції над точками (+, –, *);
d) підтримку інтерфейсів `ICloneable`, `IComparable`, `IEquatable`.

Варіант № 18

Написати реалізацію класу "*ламана лінія*". Реалізація має містити:
a) методи `Length` (обчислення довжини), `PassThrough` (перевірка проходження через точку);
b) властивості `Vertices` — вершини ламаної та `SelfCrossing` — перевірка наявності самоперетинів;
c) операцію + (паралельний перенос ламаної на заданий вектор), * (множення усіх координат на задане число) та "==";
d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IEnumerable`.

Варіант № 19

Написати реалізацію класу "*Номер у готелі*"*. Реалізація має містити:
a) методи `MakeEmpty` (виселити усіх жильців з номеру), `Contains` (перевірка того, що особа проживає у номері);
b) властивості `Number`, `Count` — поточна кількість людей у номері, `Capacity`

— максимальна місткість номера, `Dwellers` (список імен мешканців кімнати);
c) операції "+" та "−" (поселення та виселення жильця у номер);
d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IEnumerable`.

Варіант № 20

Написати реалізацію класу "*відрізок*". Реалізація має містити:
a) методи `Length` (повертає довжину відрізка), `Translate` (зсув на заданий вектор) та статичний метод `AreParallel` — перевірка паралельності відрізків;
b) властивості `EndPoints` (кінці) та `Intersects` (перевірка того, чи перетинає відрізок координатні осі);
c) операції == та <= (вважати, що коротший відрізок є "більшим");
d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IComparable`.

Варіант № 21

Написати клас "*наддовге ціле число*"* для реалізації цілого числа з довільною (як завгодно великою) кількістю цифр (при виконанні завдання не можна використовувати бібліотечний клас `BigInteger`). Реалізація має містити:
a) методи `ToDouble` та `Parse`;
b) властивості `DigitCount` та `IsEven`.
c) операції додавання, віднімання, множення, операцію неявного перетворення із типу `int` у тип "наддовге ціле число";
d) підтримку інтерфейсів `ICloneable`, `IEquatable` і `IComparable`.

Варіант № 22

Написати реалізацію класу "*Автобус*"*. Реалізація має містити:
a) методи посадки та висадки одного або кількох пасажирів, методи `Accelerate` та `Brake` (збільшення та зменшення швидкості). Реалізація має враховувати `MaxSpeed` та `Capacity`;
b) властивості `Speed`, `Capacity` (максимальна кількість пасажирів), `MaxSpeed`, `HasEmptySeats` (є вільні місця), `Passengers` (список імен пасажирів);
c) операції "+" та "−" (посадка та висадка пасажира із заданим прізвищем);
d) підтримку інтерфейсів `ICloneable`, `IEquatable`.

Варіант № 23

Написати реалізацію класу "*коло*". Реалізація має містити:
a) статичний метод `Intersect` (перевірка того, що два кола перетинаються), методи `Length` — обчислення довжини кола та `Contains` — перевірка належності точки кругу, який обмежується колом;
b) властивості `Center`, `Radius`, `InSingleQuarter` (перевірка того, що коло цілком лежить у якійсь одній чверті);
c) операції + (зсув кола на заданий вектор), == та * (розтяг/стиснення кола);

d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IComparable` (вважати, що коло є "більшим" за усі кола, які у ньому містяться).

Варіант № 24

Реалізувати узагальнений клас, що реалізує масив довільної кількості вимірів. Параметром класу є тип елементів масиву. Реалізація має містити:

a) методи `CopyTo` та `IndicesOf` (повертає масив індексів);
b) індексатор та властивість `DimCount`, значенням якої є кількість вимірів масиву;
c) операції: ==, !=, >, <;
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 25

Написати клас "квадратний тричлен". Реалізація має містити:

a) метод обчислення значення тричлена у точці та метод `GetRoots();`
b) властивості `Vertex` (вершина) та `DeadSquare` (перевірка того, що тричлен є точним квадратом);
c) операції додавання, віднімання і порівняння тричленів;
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 26

Написати узагальнений клас, який реалізує мультимножину елементів заданого типу (елементи можуть повторюватися кілька разів). Реалізація має містити:

a) методи для добавлення і вилучення елементів та метод, який повертає кількість входжень заданого елемента;
b) властивості `Count` та `Set` (набір різних елементів, які входять у мультимножину);
c) операції + (об'єднання), * (перетин) та − (різниця);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 27

Написати узагальнений клас Histogram, який реалізує гістограму (полігон частот) елементів заданого типу. Реалізація має містити:

a) методи для добавлення і вилучення потрібного числа екземплярів заданого елемента у гістограмі та метод, який повертає елементи максимальної частоти;
b) властивості Count і Set (набір різних елементів, які входять у мультимножину) та індексатор;
c) операції + (об'єднання), * (перетин) та − (різниця);
d) підтримку інтерфейсів ICloneable, IEnumerable, IEquatable.

Варіант № 28

Написати параметризований клас для подання *многочленів із коефіцієнтами заданого типу*. Реалізація має містити:

a) метод обчислення значення многочлена у точці та метод `ToArray()`;
b) властивості `Degree`, `MaxCoefficient` та індексатор;
c) операції додавання, віднімання і множення многочленів та операції порівняння і неявного перетворення із типу `T` у многочлен (`T` — тип коефіцієнтів многочлена);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 29

Написати узагальнений клас, який реалізує впорядковану множину елементів заданого типу. Реалізація має містити:
a) методи для добавлення, вилучення та перевірки належності елементів;
b) властивість `Count`, `IsEmpty`, `Min` та `Max`;
c) операції + (об'єднання), * (перетин) та ^ (симетрична різниця);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 30

Написати реалізацію класу "промінь". Реалізація має містити:
a) метод, який перевіряє, чи належить задана точка променю та статичний метод перевірки паралельності променів;
b) властивості `Vertex` (вершина променя), `CrossOX` та `CrossOY` (перевірка того, чи перетинає промінь відповідні осі координат);
c) операцію порівняння променів, операцію "+" (зсув вершини променя) та операцію "*" — знаходження точки перетину променів (якщо промені не перетинаються, має повертати `null`, якщо вершина одного з променів належить іншому, то результатом операції вважати цю вершину);
d) підтримку інтерфейсів `ICloneable`, `IEquatable` та `IEnumerable` (ітератор має послідовно повертати усі точки променя із цілими абсцисами, починаючи із вершини).

Варіант № 31

Написати реалізацію класу "предикат". Реалізація має містити:
a) метод `TruthSet` (має повертати множину істинності предиката) та `GetBasicSet(int id)` (повертає відповідну базисну множину предиката);
b) властивості `IdenticallyTrue` (загальнозначущий), `IdenticallyFalse` (тотожно фальшивий) та `Homogeneous` (однорідний);
c) операції `!`, `&&`, `||`, та `^`;
d) підтримку інтерфейсів `IEquatable`, `ICloneable`, `IEnumerable`.

Варіант № 32

Написати реалізацію класу "чотирикутник". Реалізація класу має містити:
a) метод `Perimeter` — знаходження периметра та метод `GetRect`, який має повертати мінімальний прямокутник, який містить у собі чотирикутник і має сторони паралельні осям координат;

b) властивості `Vertices` (вершини чотирикутника), `IsParallelogram` та `IsRect` (перевірка, чи є чотирикутник паралелограмом чи прямокутником);
c) Операції "+" — паралельний перенос чотирикутника на заданий вектор, "==" та "!=";
d) підтримку інтерфейсів `ICloneable` та `IEquatable`.

Варіант № 33

Написати реалізацію класу "книжкова полиця*" (попередньо написати клас книга). Реалізація має містити:
a) методи для додавання, пошуку та видалення книг із полиці;
b) властивості Count, Authors (автори книг) та Titles (назви книг);
c) операції ==, + (результат — полиця, які містить усі книги полиць аргументів), <= (усі книги з першої полиці містяться на другій полиці);
d) підтримку інтерфейсів IEquatable, ICloneable, IEnumerable.

Варіант № 34

Написати узагальнений клас із двома параметрами для реалізації асоціативного масиву, пари елементів якого упорядковані за ключем. Реалізація має містити:
a) методи пошуку, добавлення та видалення елементів;
b) властивості `Count`, `Keys`, `Values` та індексатор;
c) операції "==" та "`&&`" (має повертати перетин двох асоціативних масивів, який містить лише ті пари "ключ-значення", що входять у обидва аргументи);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IComparable`.

Варіант № 35

Написати реалізацію класу "рівнобічна трапеція". Реалізація має містити:
a) методи `Length` — обчислення периметра та `Area` — обчислення площі;
b) властивості для довжин основ та бічної сторони (задання сторін не має порушувати нерівності: будь-яка сторона менша за суму інших сторін);
c) операції == та * (розтяг/стиснення трапеції);
d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IComparable` (вважати, що трапеція є "більшою" за усі трапеції з меншою площею).

Варіант № 36

Написати реалізацію класу "клієнт банку*" (попередньо написати клас "банківська операція"). Реалізація має містити:
a) методи для виконання операцій зарахування та зняття (з перевіркою допустимості виконання операції) коштів з рахунку;
b) властивості `Name` (ім'я клієнта), `Account`, `Balance` (поточний баланс), `Operations` (список операцій);
c) операції > (порівняння балансів), + (злиття двох рахунків того самого клієнта);
d) підтримку інтерфейсів `IEquatable`, `ICloneable`, `IEnumerable` (ітерація по операціям клієнта).

Варіант № 37

Написати реалізацію класу для *подання многокутників*. Реалізація класу має містити:

a) методи `Insert` (добавлення вершини у задану позицію), `Remove` (видалення вершини), `Area` та статичний метод `Overlap` (два многокутники-параметри мають спільні точки);
b) властивості `IsEquilateral`, `IsConvex`, `Perimeter` — периметр та `Points` — вершини многокутника;
c) операції `*` (множення координат усіх вершин на задане число), + (зсув многокутника на заданий вектор) та "==".
d) підтримку інтерфейсів `ICloneable`, `IEnumerable` та `IEquatable`.

Варіант № 38

Написати дві різні реалізації узагальненого класу "черга". Реалізація має містити:

a) методи для добавлення (у кінець черги), видалення (з початку черги) та пошуку елемента черги;
b) властивості `Count`, `Head`, `Tail` (останні дві властивості відповідають за елементи, який розташовані на початку та у кінці черги відповідно);
c) операції == та + (конкатенація черг);
d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IEnumerable`.

Варіант № 39

Написати реалізацію класу "магазин*" (попередньо написати клас товар). Реалізація має містити:

a) методи для додавання, пошуку та видалення товарів у магазин і метод, який повертає загальну вартість усіх товарів;
b) властивості `Count`, `Prices` (ціни) та `Names` (назви товарів);
c) операції ==, `%` (збільшення ціни на заданий відсоток);
d) підтримку інтерфейсів `IEquatable`, `ICloneable`, `IEnumerable`.

Варіант № 40

Написати реалізацію узагальненого класу "черга з пріоритетом". Реалізація має містити:

a) методи для добавлення, видалення (елемента з найбільшим пріоритетом з початку черги) та пошуку елемента черги;
b) властивості `Count`, `Max`, `Min` (останні дві властивості відповідають за елементи, який розташовані на початку та у кінці черги відповідно);
c) операції == та + (конкатенація черг);
d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IEnumerable`.

Варіант № 41

Реалізувати узагальнений клас, що реалізує *масив довільної кількості вимірів*. Параметром класу є тип елементів масиву. Реалізація має містити:

a) методи `CopyTo` та `IndicesOf` (повертає масив індексів);
b) індексатор та властивість `Shape`, значенням якої є кортеж розмірностей масиву по кожному виміру;
c) операції: ==, `!=`, >, <;
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 42

Написати клас для реалізації діапазону цілих чисел. Реалізація має містити:
a) Функції-члени `Contains`, `Increase` (збільшення кінця діапазону на задане значення), статичну функцію `Parse` (передбачити можливість включати / виключати кінці діапазону за допомогою символів дужок "[" та "(" відповідно);
b) властивості `From`, `To`, `Length` та `Step`;
c) операції ==, + (має повертати зсунутий діапазон, до початку та кінця якого додані відповідні значення), * (має повертати масштабований діапазон, початок та кінець якого помножені на числовий множник);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 43

Написати реалізацію класу "офіс*" (попередньо написати клас "працівник" із полями код, прізвище, дата народження, посада, безпосередній керівник). Реалізація має містити:
a) методи для додавання, пошуку та видалення працівника і метод `GetSubordinates`, який має повертати список підпорядкованих працівників;
b) властивості `Count`, `Employees`;
c) операції ==, + та − (включення або виключення працівника із штату);
підтримку інтерфейсів `IEquatable`, `ICloneable`, `IEnumerable`.

Варіант № 44

Написати параметризований клас з двома параметрами для реалізації *упорядкованих бінарних дерев*, у вершинах яких зберігаються елементи заданого типу (один параметр відповідає типу ключів вершин, другий — типу елементів). Реалізація має містити:
a) методи вставки та видалення вершин з дерева (за ключем) та метод пошуку у дереві (за ключем);
b) властивості `Count`, `Value` (значення, яке зберігається у корені дерева) та `Height` (висота дерева);
c) операції ==, <= (приймає значення `true`, якщо 1-ий аргумент є піддеревом 2-го аргументу);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 45

Написати реалізацію класу "Маршрутне таксі*". Реалізація має містити:

a) методи посадки та висадки кількох пасажирів, методи ChangeSpeed (збільшення чи зменшення швидкості). Реалізація має враховувати MaxSpeed та Capacity;
b) властивості Speed (швидкість), Capacity (максимальна кількість пасажирів), MaxSpeed, IsFull (усі місця зайняті), Seats (словник, ключами якого є номери місць, а значеннями — імена пасажирів);
c) операції "+" та "−" (посадка та висадка пасажира із заданим іменем);
d) підтримку інтерфейсів ICloneable, IEquatable, IEnumerable.

Варіант № 46

Написати реалізацію класу для подання многокутників. Реалізація класу має містити:
a) методи `Rotate`, `Insert` та `Remove` (поворот, добавлення та видалення вершин у задану позицію);
b) властивості `IsConvex` — перевірка випуклості, `Perimeter` — периметр та `Points` — вершини многокутника;
c) операції `*` (множення координат усіх вершин на задане число), + (зсув многокутника на заданий вектор) та "==".
d) підтримку інтерфейсів `ICloneable`, `IEnumerable` та `IEquatable`.

Варіант № 47

Написати параметризований клас з двома параметрами для реалізації *упорядкованих бінарних дерев*, у вершинах яких зберігаються елементи заданого типу (один параметр відповідає типу ключів вершин, другий — типу елементів). Реалізація має містити:
a) методи добавлення та видалення вершин з дерева (за ключем) та метод `Successor` (має повертати наступний до його аргументу ключ, який зберігається у дереві);
b) властивості `Count`, `Keys`, `Max` (значення з найбільшим ключем, яке зберігається у дереві) та `Root` (корінь дерева, піддеревом якого є поточне дерево);
c) операції ==, + (об'єднання двох дерев);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 48

Написати дві різні реалізації узагальненого класу "дек". Реалізація має містити:
a) методи для добавлення (у кінець та початок), видалення (з початку та кінця) та пошуку елемента у деку;
b) властивості Count, Head, Tail (останні дві властивості відповідають за елементи, який розташовані на початку та у кінці деку відповідно);
c) операції ==, | (конкатенація вмісту) та & (знаходження спільних елементів) ;
d) підтримку інтерфейсів IEquatable, ICloneable та IEnumerable.

Варіант № 49

Написати клас для реалізації бінарних відношень* на множинах {1,2,…, *M*} та {1,2,…, *N*}. Реалізація має містити:

a) метод `Add` (для додавання кількох пар елементів, які перебувають у відношенні), метод `Remove` (видалення кількох пар елементів, які перебувають у відношенні), `Projection` (1-ша або друга проекція відношення), та метод `Related` (перевірка того, що елементи перебувають у бінарному відношенні);
b) властивості `Count` (кількість пар, які перебувають у відношенні), `M`, `N` та індексатор;
c) операції `!` (обернене відношення), `|` (об'єднання), `&` (перетин відношень) та `*` (добуток відношень).
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 50

Написати реалізацію класу "коло" та його нащадка — класу "конус". Реалізація останнього класу має містити визначення геометричного тіла у 3-вимірному координатному просторі, вісь якого паралельна осі OZ. Передбачити:

a) функції-члени `SurfaceArea`, `Volume`, `Contains` (перевірка належності точки конусу);
b) властивості `Radius`, `Height`, `SlantHeight` та `Vertex`;
c) операції "==", "+" (зсув на заданий вектор), "`*`" (має повертати конус з масштабованим радіусом), < (має повертати `true`, якщо 1-й конус має меншу площу бічної поверхні ніж 2-й);
d) підтримку інтерфейсів `ICloneable`, `IEquatable`, `IComparable` (вважати, що більшим є конус більшого об'єму).

Варіант № 51

Написати реалізацію класу "прямокутний паралелепіпед". Реалізація має містити:

a) методи `Surface` (обчислення площі поверхні) та `Translate` (паралельне перенесення на заданий вектор);
b) властивості `Vertices` (вершини) та `Volume` (об'єм);
c) підтримку інтерфейсів `ICloneable` та `IEnumerable`;
d) операції "==", ">" та "`*`" (множення на число).

Варіант № 52

Написати узагальнений клас, який реалізує множину елементів заданого типу. Реалізація має містити:

a) методи для додавання, вилучення та пошуку елементів;
b) властивість `Count` та статичну властивість `U` (універсальна множина);
c) операції + (об'єднання), * (перетин), ! (доповнення) та − (різниця);
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 53

Написати клас "Перестановка*" для реалізації перестановок на множині {1,2,…, *N*}. Реалізація має містити:

a) методи `InvariantElements` (повертає масив елементів, які залишаються на місті), `GetCycles` (повертає розклад перестановки у добуток циклів);
b) властивості `N` та `IsEven` (перевірка парності перестановки) та індексатор;
c) операції "−" (знаходження оберненої перестановки), "*" (добуток перестановок).
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 54

Написати реалізацію класу "еліпс". Вважати, що осі еліпсі паралельні осям координат. Реалізація має містити:

a) методи Area — обчислення площі еліпса та Contains — перевірка належності точки фігурі, яка обмежується еліпсом;
b) властивості Center, A, B, Eccentricity;
c) операції + (зсув еліпса на заданий вектор), == та * (розтяг/стиснення еліпса);
d) підтримку інтерфейсів IEquatable, ICloneable.

Варіант № 55

Написати клас для реалізації векторів *n*-вимірного простору з дійсними коефіцієнтами. Реалізація має містити:

a) статичний метод знаходження манхетенської відстані між векторами, метод `Dot` (скалярний добуток векторів) та метод `Ceiling` (заокруглення усіх координат уверх);
b) індексатор і властивості `Dim` (розмірність) та `Max` (максимальна координата вектора);
c) операції "==", "+", "<", "*" (покоординатний добуток векторів).
d) підтримку інтерфейсів `ICloneable`, `IEnumerable`, `IEquatable`.

Варіант № 56

Написати узагальнений клас для реалізації впорядкованого списку. Реалізація має містити:

a) функції-члени для вставки елемента у список, видалення елемента із списку, IndexOf (знаходження позиції першого входження елемента);
b) властивості Count, Min, Max та індексатор;
c) операції ==, ^ (симетрична різниця двох списків), <=;
d) підтримку інтерфейсів ICloneable, IEnumerable, IEquatable.

Варіант № 57

Написати реалізацію класу "ламана лінія". Реалізація має містити:

a) методи `Length` (обчислення довжини), `PassThrough` (перевірка проходження через точку);

b) властивості `Vertices` — вершини ламаної та `SelfCrossing` — перевірка наявності само перетинів;
c) операцію + (паралельне перенесення ламаної на заданий вектор), * (множення усіх координат на задане число) та "==";
d) підтримку інтерфейсів IEquatable, ICloneable та IEnumerable.

Варіант № 58

Написати клас для реалізації *точок у 2-вимірному просторі*. Реалізація має містити:
a) методи `Norm` (відстань від початку координат), та `Abs` (покоординатний модуль), функцію, яка повертає номер чверті, у якій лежить точка та статичну функцію для зчитування точки з рядка;
b) властивості для звертання до координат точки та властивості `OnBisectors` (перевірка того, чи лежить точка на бісектрисах координатних чвертей);
c) покоординатні операції над точками (+, −, *, /, ==, <);
d) підтримку інтерфейсів `ICloneable`, `IEquatable`.

Варіант № 59

Написати реалізацію класу "*Номер у готелі*"*. Реалізація має містити:
a) методи `MakeEmpty` (виселити усіх жильців з номеру), `Contains` (перевірка того, що особа проживає у номері);
b) властивості `Number`, `Count` — поточна кількість людей у номері, `Capacity` — максимальна місткість номера, `Dwellers` (список імен мешканців кімнати);
c) операції "+" та "−" (поселення та виселення жильця у номер);
d) підтримку інтерфейсів `IEquatable`, `ICloneable` та `IEnumerable`.

Варіант № 60

Написати реалізацію класу "*трикутник*". Реалізація класу має містити:
a) статичний метод `Parse` — зчитування із рядка та метод `OnSide` — перевірка того, чи лежить точка на межі трикутника;
b) властивості `Vertices` (вершини трикутника), `IsEquilateral` та `IsRight-Angled` (перевірка, чи є трикутник правильним чи прямокутним);
c) Операції "−" — паралельне перенесення трикутника на заданий вектор, "==" та "*" (масштабування);
d) підтримку інтерфейсів `ICloneable`, `IEquatable` та `IComparable` (вважати, що більшим є трикутник більшої площі).