Знайомство з ООП. Поглиблене вивчення мови C#

Тема 4. Колекції

Колекції

- У C# колекція – це сукупність об'єктів, розміщених в одному контейнері, що надає можливість їх послідовної обробки. В області обробки множинних даних колекції є потужною альтернативою масивам.
- Середовище .NET Framework містить інтерфейси та класи, в яких визначено і реалізовано різні типи колекцій. Колекції спрощують рішення багатьох завдань програмування завдяки тому, що пропонують готові рішення для створення цілого ряду типових, але часто трудомістких для розробки структур даних
- Головна перевага колекцій полягає в тому, що вони стандартизують обробку груп об'єктів у програмі. Колекції розроблені на основі інтерфейсів .NET. Є також можливість реалізувати власну колекцію

Типи колекцій

- **Неузагальнені колекції** реалізують такі структури даних, як динамічні масиви, стеки, черги та словники. Вони оперують елементами класу object. Тому неузагальнені колекції можуть зберігати дані будь-якого типу, причому одна колекція може містити дані різних типів. Класи та інтерфейси узагальнених колекцій описані в просторі імен System.Collections.
- **Узагальнені колекції** забезпечують узагальнену реалізацію декількох неузагальнених колекцій. Такі колекції є типізованими, тому в узагальненій колекції можуть зберігатися тільки елементи даних, сумісні за типом з колекцією. Завдяки цьому виключається випадкова розбіжність типів. Узагальнені колекції оголошено в просторі імен System.Collections.Generic.

Класи неузагальнених колекцій

- **ArrayList**. Колекція виду "динамічний масив", яка дозволяє працювати з її елементами як з елементами масиву. Розмір масиву можна змінювати під час виконання.
- **Queue**. Колекція виду "черга" – список, який діє за принципом "першим зайшов – першим вийшов".
- **Stack**. Колекція виду "стек" – список, який діє за принципом "першим зайшов – останнім вийшов".
- **Hashtable**. Колекція пар "ключ - значення", які впорядковані за геш-кодом ключа".
- **SortedList.** Колекція виду "сортований список", яка містить пари "ключ-значення".

Класи неузагальнених колекцій визначено в просторі імен System.Collections.

Класи узагальнених колекцій

- **List<T>**. Колекція виду "динамічний масив", яка дозволяє працювати з її елементами як з елементами масиву. Розмір масиву можна змінювати під час виконання.
- **Queue<T>**. Колекція виду "черга" – список, який діє за принципом "першим зайшов – першим вийшов".
- **Stack<T>**. Колекція виду "стек" – список, який діє за принципом "першим зайшов – останнім вийшов".
- **Dictionary<Tkey, TValue>**. Колекція виду "словник", яка зберігає пари "ключ-значення".
- **SortedDictionary<TKey, TValue>**. Колекція виду "сортований словник". Містить пари "ключ-значення".
- **SortedList<TKey, TValue>.** Колекція виду "сортований список", яка містить пари "ключ-значення".
- **SortedSet<T>**. Створює сортовану множину. HashSet<T>. Зберігає ряд унікальних значень за допомогою хеш-таблиці.

Класи узагальнених колекцій визначено в просторі імен System.Collections.Generic.

Клас ArrayList

- У C# стандартні масиви мають фіксовану довжину, яка не може змінитися під час виконання програми. Клас ArrayList призначений для підтримки динамічних масивів, які за необхідності можуть збільшуватися або скорочуватися.
- Об'єктом класу ArrayList є масив змінної довжини, елементами якого є об'єктні посилання. Будь-який об'єкт класу ArrayList створюється з деяким початковим розміром. З перевищенням цього розміру колекція автоматично подвоюється. У разі видалення об'єктів масив можна скоротити.

Найвживаніші методи класу ArrayList

- Метод Add додає елемент наприкінці списку.
- Метод Insert вставляє елемент у середину списку, посуваючи всі елементи з індексами, не меншими за index.
- Методи AddRange та InsertRange використовують для додавання або вставляння діапазону елементів. Цей діапазон задається колекцією (наприклад, масивом).
- Метод SetRange не вставляє діапазон елементів, а замінює ними інші, починаючи із заданого індексу.
- Метод Sort впорядковує елементи динамічного масиву за заданим об'єктом comparer правилом порівняння. Якщо параметр comparer відсутній або має значення null, то числа та дати впорядковуються за зростанням, а символи - за абеткою в порядку зростання.
- Метод Reverse використовують для зміни порядку елементів на протилежний.

Колекція ArrayList може містити об’єкти різних типів

```
ArrayList a = new ArrayList();
a.Add("Комп'ютер");
a.Add("Програмування");
a.Add("C# - це круто!");
a.Add(1);
a.Add('r');
a.Add(3.45);
a.Add(true);
a.Add(DateTime.Now);
a.Add(new Object());
a.Add(new { Name = "Петро", LastName = "Петренко" });

foreach (object obj in a)
    Console.WriteLine(obj);

// Можна працювати як зі звичайним масивом
for (int i = 0; i < a.Count; i++)
    Console.WriteLine(a[i]);
```

Клас List<T>

- Клас List<T> реалізує динамічний масив, який може змінювати розмір під час виконання програми. У C# стандартні масиви мають фіксовану довжину, яка не може змінюватися після створення масиву. Але часто конкретна довжина масиву є невідомою. Для таких ситуацій і призначений клас List<T>.
- Він визначає масив змінної довжини, який складається з посилань на об'єкти і може динамічно збільшувати і зменшувати свій розмір. Колекцію типу List<T> створюють з початковим розміром. Якщо цей розмір перевищується, то вона автоматично розширюється. При видаленні об'єктів колекція автоматично скорочується.
- Колекції класу List<T> широко застосовують в практиці програмування на C#.

Найвживаніші методи класу List<T>

- void Add (T item). Додає новий елемент в в кінець викликаючої колекції
- void AddRange (ICollection collection). Додає елементи колекції collection в кінець викликаючої колекції
- int BinarySearch (T item). Шукає у викликаючій колекції значення, задане параметром item. Повертає індекс знайденого елемента. Якщо значення не знайдено, повертає від'ємне значення. Список повинен бути відсортований
- void RemoveAt (int index). Видаляє з викликаючої колекції елемент за вказаним індексом index
- void Sort (). Сортує викликаючу колекцію за зростанням

Колекція List<T> може містити об’єкти лише вказаного певного типу T

```
List<string> names = new List<string>();
names.Add("Bill Gates");      // Засновник Microsoft
names.Add("Steven Jobs");     // Засновник Apple
names.Add("Sergey Brin");     // Засновник Google
names.Add("Mark Zuckerberg"); // Засновник Facebook
//names.Add('A');  - Помилка !!!

foreach (string str in names)
      Console.WriteLine(str);

// Можна звертатися через індекс
Console.WriteLine("Засновник Microsoft - {0}", names[0]);
```

Структура даних «Черга»

Класи Queue та Queue<T>

- Черга – це список, додавання елементів у який виконується в один кінець (хвіст), а вибірка проводиться з іншого кінця (голови). Інші операції з чергою не визначені. У ході вибірки елемент виключається з черги. Говорять, що чергу реалізує принцип обслуговування FIFO (fist in – fist out, “першим прийшов, – першим вийшов”).
- У програмуванні черги застосовують для зберігання таких елементів, як активні процеси, списки призупинених транзакцій в базі даних або пакети даних, отримані через мережу.
- Колекція Queue реалізована як для нетипізованих колекцій, так і для типізованих. Клас Queue визначено в просторі імен System.Collections, а клас Queue<T> - в просторі імен System.Collections.Generic.

Найвживаніші методи класу Queue<T>

- public T Dequeue(). Вибирає об'єкт з початку черги
- public void Enqueue(T item). Додає об'єкт item в кінець черги
- public T Peek(). Читає елемент з початку черги, не видаляючи його
- public virtual T[] ToArray(). Повертає масив, який містить копії об'єктів черги

Структура даних «Стек»

Класи Stack та Stack<T>

- Стек – це список, що діє за принципом "першим прийшов – останнім вийшов" (last-in, first-out – LIFO).
- Стек – одна з найважливіших структур даних в обчислювальній техніці. Він часто застосовується в системному програмному забезпеченні, компіляторах, а також в програмах відстеження в зворотному порядку на основі штучного інтелекту.
- Колекція Stack реалізована як для нетипізованих колекцій, так і для типізованих. Клас Stack визначено в просторі імен System.Collections, а клас Stack<T> - в просторі імен System.Collections.Generic.

Найвживаніші методи класу Stack<T>

- public T Peek(). Читає об'єкт з вершини стеку, але не видаляє його
- public T Pop(). Витягує об'єкт зі стеку
- public void Push(T item). Розміщує у стеку об'єкт item

Клас Hashtable

Клас Hashtable призначений для створення колекції пар об’єктів типу («ключ», «значення»), в якій для зберігання об'єктів використовується геш-кодування-таблиця. У геш-таблиці для зберігання інформації застосовують механізм гешування (hashing). Сутність гешування полягає в тому, що для визначення унікального значення, яке називають геш-кодом, використовується інформаційний вміст відповідного йому ключа. Геш-кодування-код потім використовується як індекс, за яким у таблиці відшукуються дані, відповідні цьому ключу. Перетворення ключа в геш-кодування-код виконується автоматично. Перевага гешування в тому, що воно дозволяє скорочувати час виконання таких операцій, як пошук, прочитування і запис даних, навіть для великих обсягів інформації.

Клас Hashtable

У класі Hashtable, окрім властивостей, визначених у реалізованих ним інтерфейсах, визначено дві власні public-властивості:

- public virtual ICollection Keys { get; } //дозволяє отримати колекцію ключів
- public virtual ICollection Values { get; } //дозволяє отримати колекцію значень

Для додавання елементу в геш-таблицю необхідно викликати метод Add(), який приймає два окремі аргументи: ключ і значення. Важливо зазначити, що геш-таблиця не гарантує збереження порядку елементів, оскільки гешування зазвичай не застосовується до відсортованих таблиць.

Ключі повинні бути унікальними, а значення можуть повторюватися

```
Hashtable currencies = new Hashtable();
currencies.Add("US", "Dollar");
currencies.Add("Japan", "Yen");
currencies.Add("France", "Euro");
currencies.Add("Australia", "Dollar");

Console.WriteLine("US Currency: {0}", currencies["US"]);

foreach (DictionaryEntry x in currencies)
    Console.WriteLine("{0}    {1}", x.Key, x.Value);
```

Клас Dictionary<TKey, TValue>

- Клас Dictionary<TKey, TValue> дозволяє зберігати пари "ключ-значення" в колекції як у словнику.Кожен такий об'єкт належить до структури KeyValuePair <TKey, TValue>.
- Завдяки властивостям Key і Value, які є у цій структурі, можна отримати ключ і значення елемента в словнику.
- Словники є динамічними, розширюючись при необхідності.

Найвживаніші елементи класу Dictionary<TKey, TValue>

- void Add(TKey key, TValue value). Додає в колекцію пару "ключ-значення", задану параметрами key і value. Якщо ключ вже є у словнику, його значення не змінюється і генерується виняткова ситуація ArgumentException
- bool ContainsKey(TKey key). Повертає логічне значення true, якщо колекція містить ключ key, в іншому випадку – логічне значення false
- bool ContainsValue(TValue value). Повертає логічне значення true, якщо колекція містить значення value, в іншому випадку – логічне значення false
- bool Remove(TKey key). Видаляє з колекції елемент, ключ якого дорівнює key. Повертає true при успіху, і false – в протилежному випадку

Домашнє завдання

Напишіть клас Automobile (Автомобіль) з полями «Назва» та «Максимальна швидкість» автомобіля. Додайте до класу необхідні методи, конструктори та властивості.

Зробити колекцію об’єктів типу Automobile (типізований список List). Додати до колекції кілька різних об’єктів типу Automobile. Вивести на екран інформацію про усі автомобілі в колекції. Вивести на екран назву автомобіля, який має найвищу максимальну швидкість.