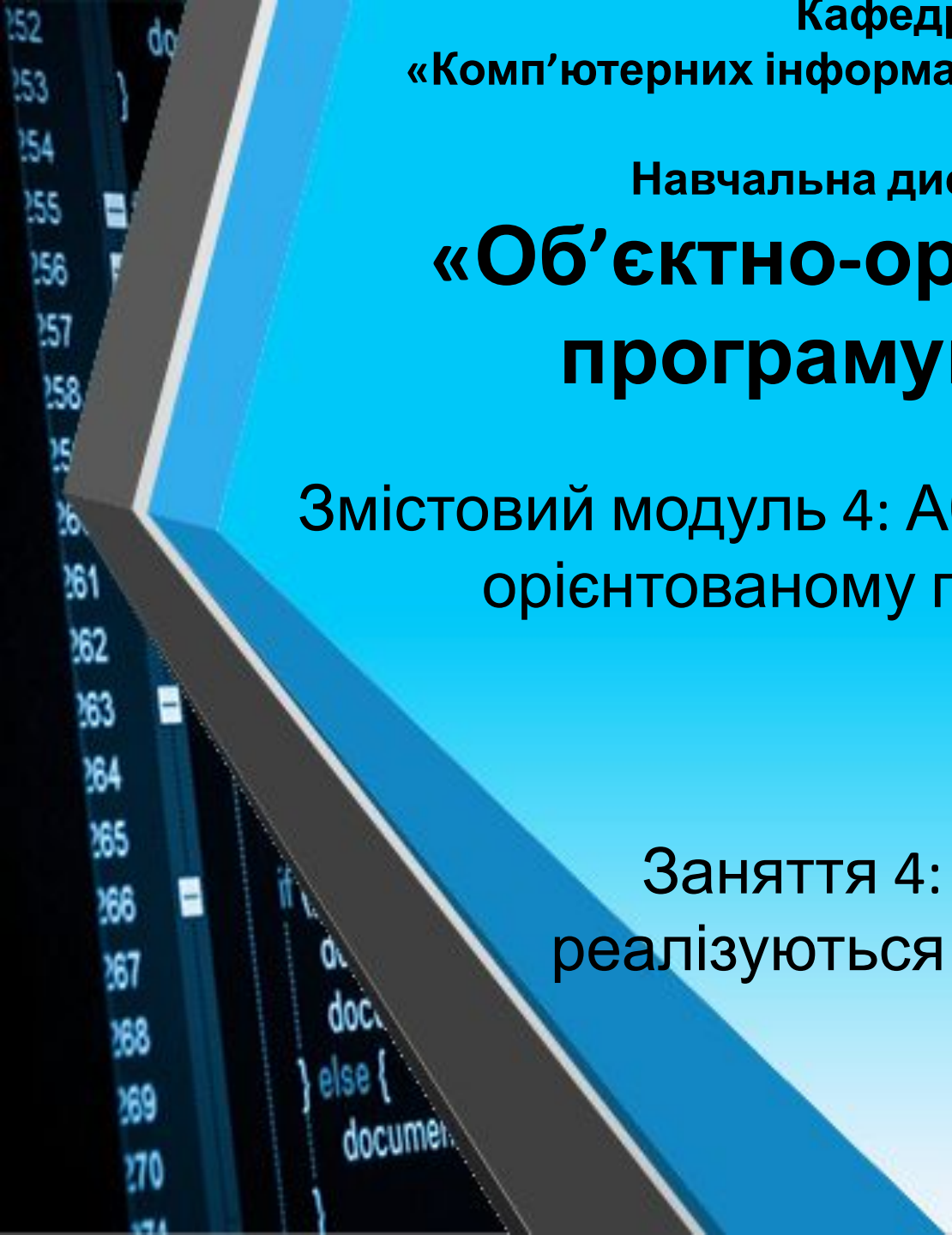


Кафедра
«Комп’ютерних інформаційних технологій»

Навчальна дисципліна:

«Об’єктно-орієнтоване програмування»

Змістовий модуль 4: Абстракції в об’єктно-орієнтованому програмуванні

Заняття 4: Інтерфейси, що реалізуються в класах-колекціях

ЛІТЕРАТУРА

1. *Шилдт Герберт - C# 4.0 полное руководство – 2011*
2. *Эндрю Троелсен "Язык программирования C# 5.0 и платформа .NET 4.5" – 2013*
3. *Стиллмен, Дж. Грин Изучаем C#. 3-е изд – 2017*
4. *Джозеф Албахари C# 6.0. Справочник. Полное описание языка – 2017*

Навчальні питання

1. *Узагальнені та неузагальнені колекції*
2. *Види класів колекцій*

Класи колекції

Щоб допомогти в подоланні обмежень простого масиву, бібліотеки базових класів .NET поставляються з декількома просторами імен, що містять класи колекцій

Класи колекції поділяються на:

- неузагальнені колекції (що в основному знаходяться в просторі імен System.Collections)
- узагальнені колекції (що в основному знаходяться в просторі імен System.Collections.Generic)

Класи колекції

Неузагальнений клас	Призначення	Інтерфейси, що реалізовуються
ArrayList	Представляє колекцію динамічно змінюваного розміру, що містить об'єкти в певному порядку	IList, ICollection, IEnumerable і ICloneable
BitArray	Управляє компактним масивом бітових значень, які представляються як булеві, де true означає встановлений (1) біт, а false - невстановлений (0) біт	ICollection, IEnumerable ICloneable
Hashtable	Представляє колекцію пар "ключ/значення", організованих на основі хеш-кода ключа	IDictionary, ICollection, IEnumerable і Icloneable
Queue	Представляє стандартну чергу об'єктів, FIFO	ICollection, IEnumerable і ICloneable
SortedList	Представляє колекцію пар "ключ/значення", відсортованих по ключу і доступних по ключу і по індексу	IDictionary, ICollection, IEnumerable і ICloneable
Stack	Представляє стек LIFO	ICollection, IEnumerable і ICloneable

Класи колекції

Узагальнений клас	Призначення	Підтримувані основні інтерфейси
Dictionary<TKey, TValue>	Представляє узагальнену колекцію ключів і значень	ICollection<T>, IDictionary<TKey, TValue>, IEnumerable<T>
LinkedList<T>	Представляє двозв'язний список	ICollection<T>, IEnumerable<T>
List<T>	Послідовний список елементів з динамічно змінюваним розміром	ICollection<T>,IEnumerable<T>, IList<T>
Queue<T>	Узагальнена реалізація черги - списку, що працює по алгоритму "перший увійшов, - перший вийшов" (FIFO)	ICollection, IEnumerable<T>
SortedDictionary <TKey, TValue>	Узагальнена реалізація словника - відсортованої безлічі пар "ключ/значення"	ICollection<T>, IDictionary<TKey, TValue>, IEnumerable<T>
SortedSet<T>	Представляє колекцію об'єктів, підтримуваних в сортованому порядку без дублювання	ICollection<T>, IEnumerable<T>, ISet<T>

Черга

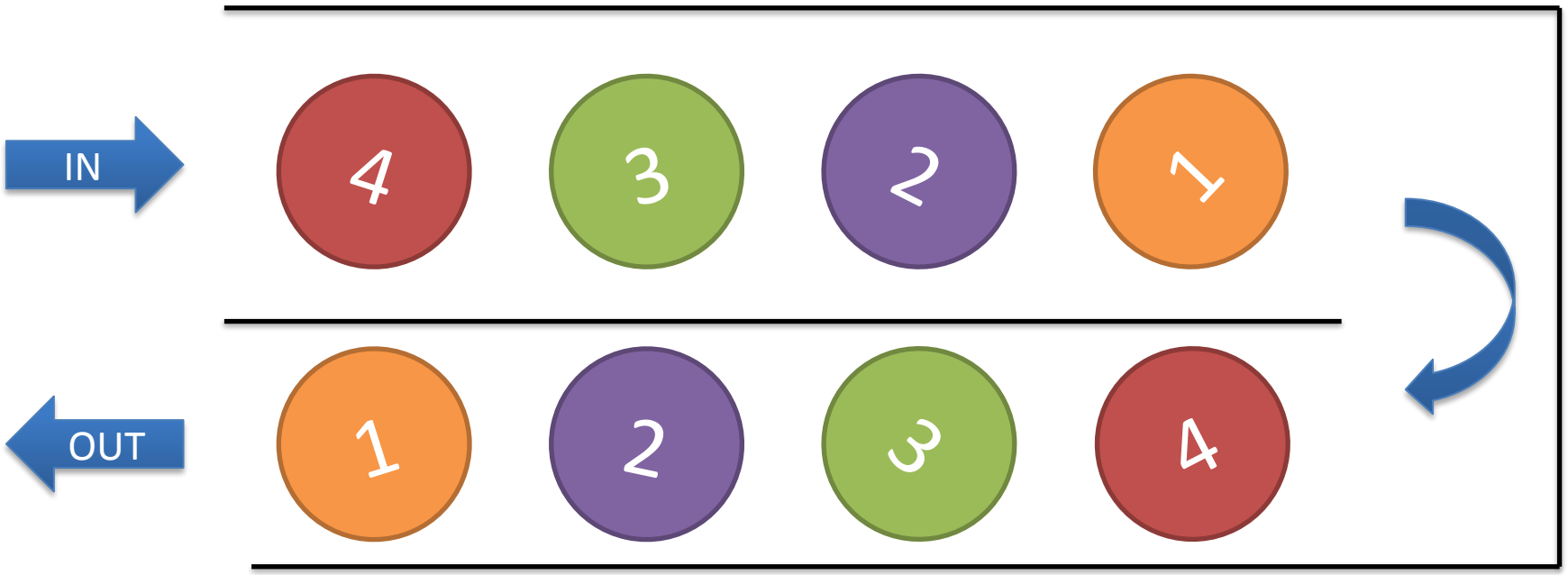
Черга (Queue) – колекція, в якій елементи обробляються за схемою "перший увійшов, перший вийшов" (FIFO)

У .NET є неузагальнений клас Queue і узагальнений клас Queue<T> у просторі імен

Метод	Призначення
Enqueue ()	Додає елемент в кінець черги
Dequeue ()	Читає і видаляє елемент з голови черги. Якщо в черзі більше немає елементів на момент виклику методу Dequeue(), генерується виключення
Peek ()	Метод читає елемент з голови черги, але не видаляє його
Count	Повертає кількість елементів в черги
TrimExcess ()	Змінює місткість черги
Contains ()	Перевіряє наявність елементу в черзі і повертає true, якщо шуканий елемент в ній міститься
CopyTo ()	Дозволяє копіювати елементи черги в існуючий масив
ToArray ()	Повертає новий масив, що містить елементи черги

ЧЕРГА

QUEUE – FIFO



Черга

```
Queue<Person> peopleQ = new Queue<Person>();
peopleQ.Enqueue(new Person { FirstName = "Homer", Age = 47 });
peopleQ.Enqueue(new Person { FirstName = "Marge", Age = 45 });
peopleQ.Enqueue(new Person { FirstName = "Lisa", Age = 9 });
Console.WriteLine($"{peopleQ.Peek()} is first in line!");
// // Видалити усіх з черги.
GetCoffee(peopleQ.Dequeue());
GetCoffee(peopleQ.Dequeue());
GetCoffee(peopleQ.Dequeue());
try{
    GetCoffee(peopleQ.Dequeue());
}
catch (InvalidOperationException e){
    Console.WriteLine($"Error! {e.Message}");
    }
```

```
d:\Documents\Visual Studio 2017\Projects\C
Homer is first in line!
Coffee to Homer
Coffee to Marge
Coffee to Lisa
Error! Очередь пуста.
```

```
static void GetCoffee(Person person)
{
        Console.WriteLine($"Coffee to {person}");
}
```

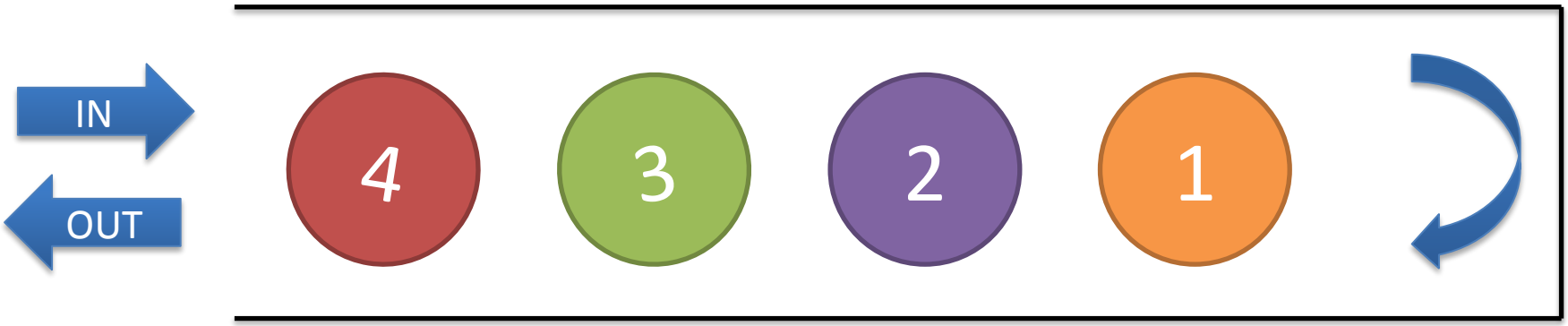
Стек

Стек (Stack) - це контейнер, в якому елемент, доданий до стека останнім, читається першим (LIFO)

Методи	Опис
Push()	Додає елемент у вершину стека
Pop()	Видаляє і повертає елемент з вершини стека. Якщо стек порожній, генерується виключення типу InvalidOperationException
Peek()	Повертає елемент з вершини стека, не видаляючи його при цьому
Count	Повертає кількість елементів в стеку
Contains()	Перевіряє наявність елементу в стеку і повертає true у разі знаходження його там.
CopyTo()	Копіює елементи стека в існуючий масив
ToArray()	Повертає новий масив, що містить усі елементи стека

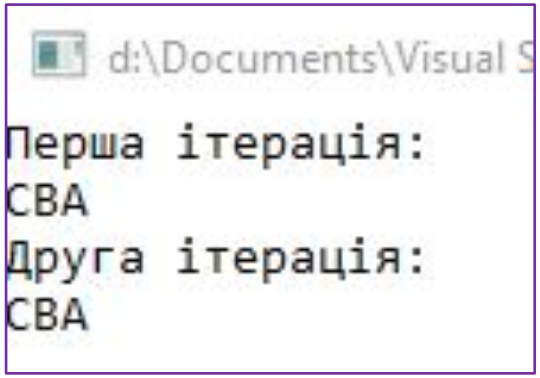
Стек

STACK – LIFO



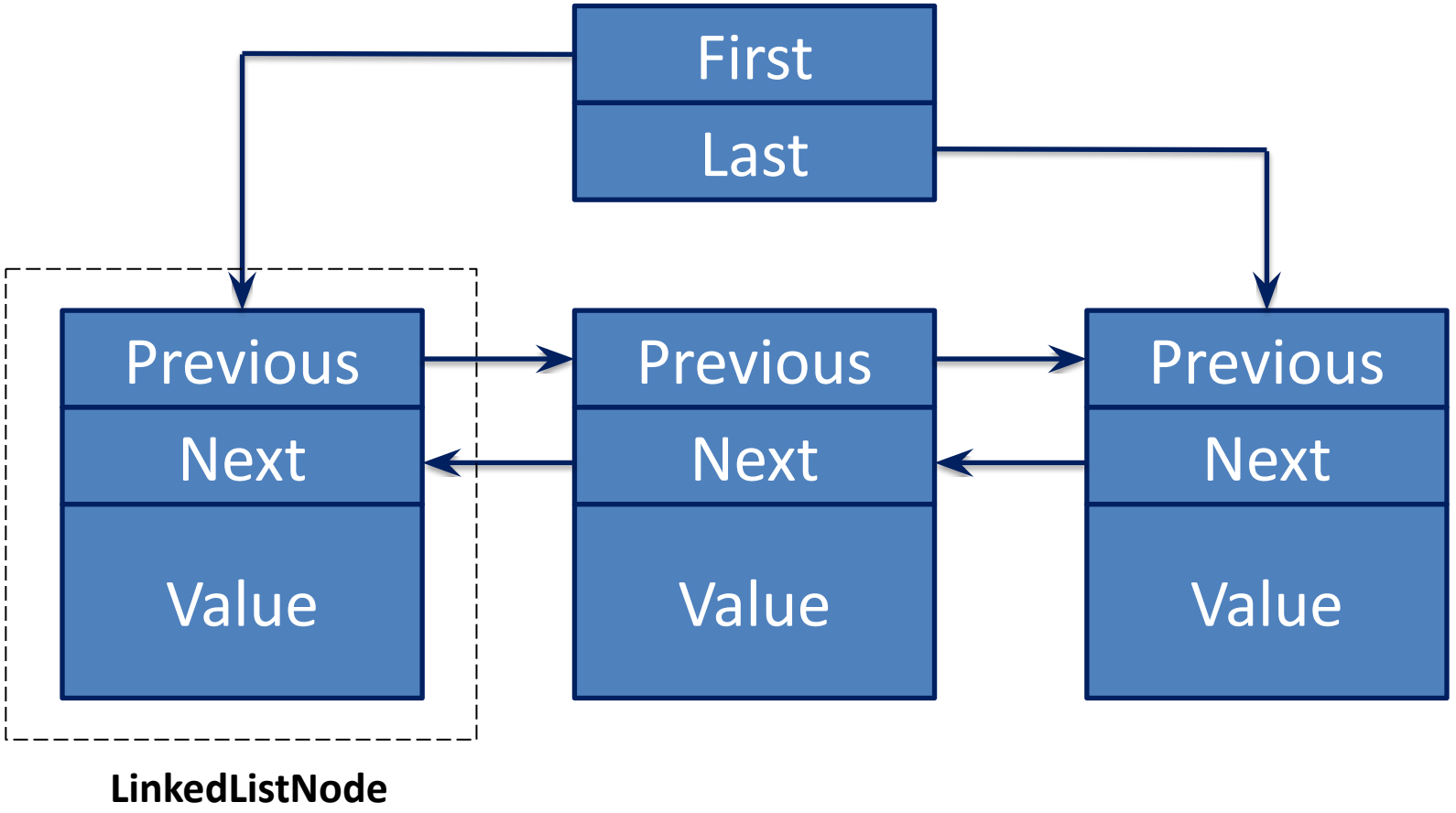
Стек

```
Stack<string> alphabet = new Stack<string>();
alphabet.Push("A");
alphabet.Push("B");
alphabet.Push("C");
Console.WriteLine("Перша ітерація:");
foreach (string item in alphabet){
  Console.Write(item);
}
Console.WriteLine("\nДруга ітерація: ");
while (alphabet.Count > 0){
  Console.Write(alphabet.Pop());
}
```



Двозв'язний список

Клас `LinkedList<T>` – двозв'язний список, в якому кожен елемент посилається на наступний і попередній



Член	Призначення
AddAfter (node, newNode)	вставляє вузол newNode в список після вузла node
AddAfter (node, value)	вставляє в список новий вузол із значенням value після вузла node
AddBefore (node, newNode)	вставляє в список вузол newNode перед вузлом node
AddBefore (node, value)	вставляє в список новий вузол із значенням value перед вузлом node
AddFirst (node)	вставляє новий вузол в початок списку
AddFirst (T value)	вставляє новий вузол із значенням value в початок списку
AddLast (node)	вставляє новий вузол в кінець списку
AddLast (value)	вставляє новий вузол із значенням value в кінець списку
RemoveFirst ()	видаляє перший вузол зі списку. Після цього новим першим вузлом стає вузол, наступний за видаленим
RemoveLast ()	видаляє останній вузол зі списку

Двозв'язний список

```
LinkedList<int> list = new LinkedList<int>();
list.AddFirst(1);
list.AddLast(5);
list.AddBefore(list.Last, 4);
list.AddAfter(list.First, 2);
list.Remove(list.Last);
foreach(var needle in list)
    Console.Write($"{needle} ");//1 2 4

list.AddFirst(new Person());

Console.ReadKey();
```

Person.Person()

Аргумент 1: не удается преобразовать из "ConsoleApp11.Person" в "int".

Елементи узагальнених типів працюють типом заданим при ініціалізації

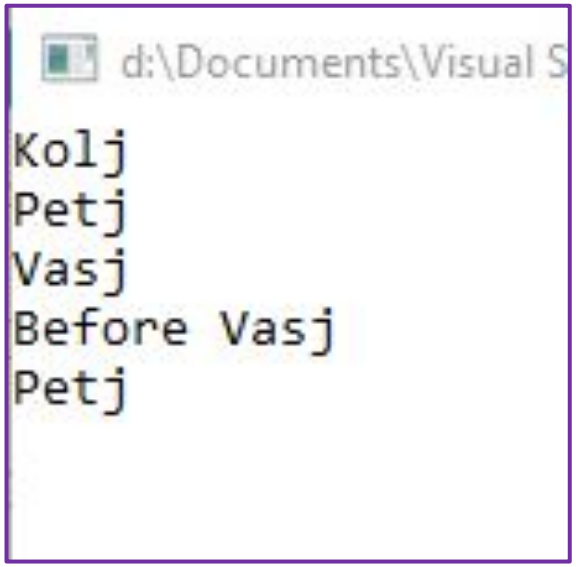
Двозв'язний список

```
var list = new LinkedList<Person>();
LinkedListNode<Person> vasj = list.AddLast(new Person
                    {FirstName= "Vasj", Age = 22});

var petj = list.AddFirst(new Person {FirstName ="Petj", Age = 23});

var kolj = list.AddBefore(petj,new Person{FirstName ="Kolj“
                                  ,Age= 19});
foreach (var pers in list){
    Console.WriteLine(pers);
}

Console.WriteLine($"Before {vasj.Value}");
Console.WriteLine(vasj.Previous.Value);
```



Набори

Набори HashSet<T> та SortedSet<T> містять тільки унікальні об’єкти

Унікальність визначається за хеш-кодом об’єктів

```
var set = new HashSet<Person>();
set.Add(new Person {FirstName = "Vasj", Age=22});
set.Add(new Person { FirstName = "Kolj", Age = 21 });
set.Add(new Person { FirstName = "Vasj", Age = 22 });
set.Add(new Person { FirstName = "Petj", Age = 20 });
set.Add(new Person { FirstName = "Kolj", Age = 21 });
foreach (var pers in set){
   Console.WriteLine(pers);
}
```

Набори

```
static void Main(string[] args)
{

    SortedSet<char> ss = new SortedSet<char>();
    SortedSet<char> ss1 = new SortedSet<char>();

    ss.Add('C');
    ss.Add('B');
    ss.Add('A');
    ss.Add('Z');
    ShowColl(ss, "Перша колекція: ");

    ss1.Add('X');
    ss1.Add('Y');
    ss1.Add('Z');
    ShowColl(ss1, "Друга колекція: ");

    ss.SymmetricExceptWith(ss1);
    ShowColl(ss, "Виключення (однакові елементи) двох множин: ");

    ss.UnionWith(ss1);
    ShowColl(ss, "Об'єднання двох множин: ");

    ss.ExceptWith(ss1);
    ShowColl(ss, "Віднімання множин");

    Console.ReadLine();
```

```
static void ShowColl(SortedSet<char> ss, string s)
{
    Console.WriteLine(s);
    foreach (char ch in ss)
        Console.Write(ch + " ");
    Console.WriteLine("\n");
}
```

```
Перша колекц?я:
A B C Z

Друга колекц?я:
X Y Z

Виключення (однаков? елементи) двох множин:
A B C X Y

Об'єднання двох множин:
A B C X Y Z

В?дн?мання множин
A B C
```

GetHashCode

Equals

Набори

```
class X
{
    int f;
    int d;
    public X() { }

    public X(int f, int d)
    {
        this.f = f;
        this.d = d;


    }

    public override bool Equals(object obj)
    {
        var x = obj as X;
        return x != null &&
               f == x.f &&
               d == x.d;
    }
}
```

```
public override int GetHashCode()
{
    var hash = 19;
    // уникаємо хеш-колізій, використовуємо (невелике) просте число
    hash = hash * 37 + f.GetHashCode();
    hash = hash * 37 + d.GetHashCode();
    return hash;
}
```

```
static void Main(string[] args)
{
    X x1 = new X(2, 4);
    X x2 = new X(2, 4);

    Console.WriteLine(x1.Equals(x2));
    Console.WriteLine(x1.GetHashCode());
    Console.WriteLine(x2.GetHashCode());
    Console.ReadKey();

}
```

```
True
46104728
12289376
```

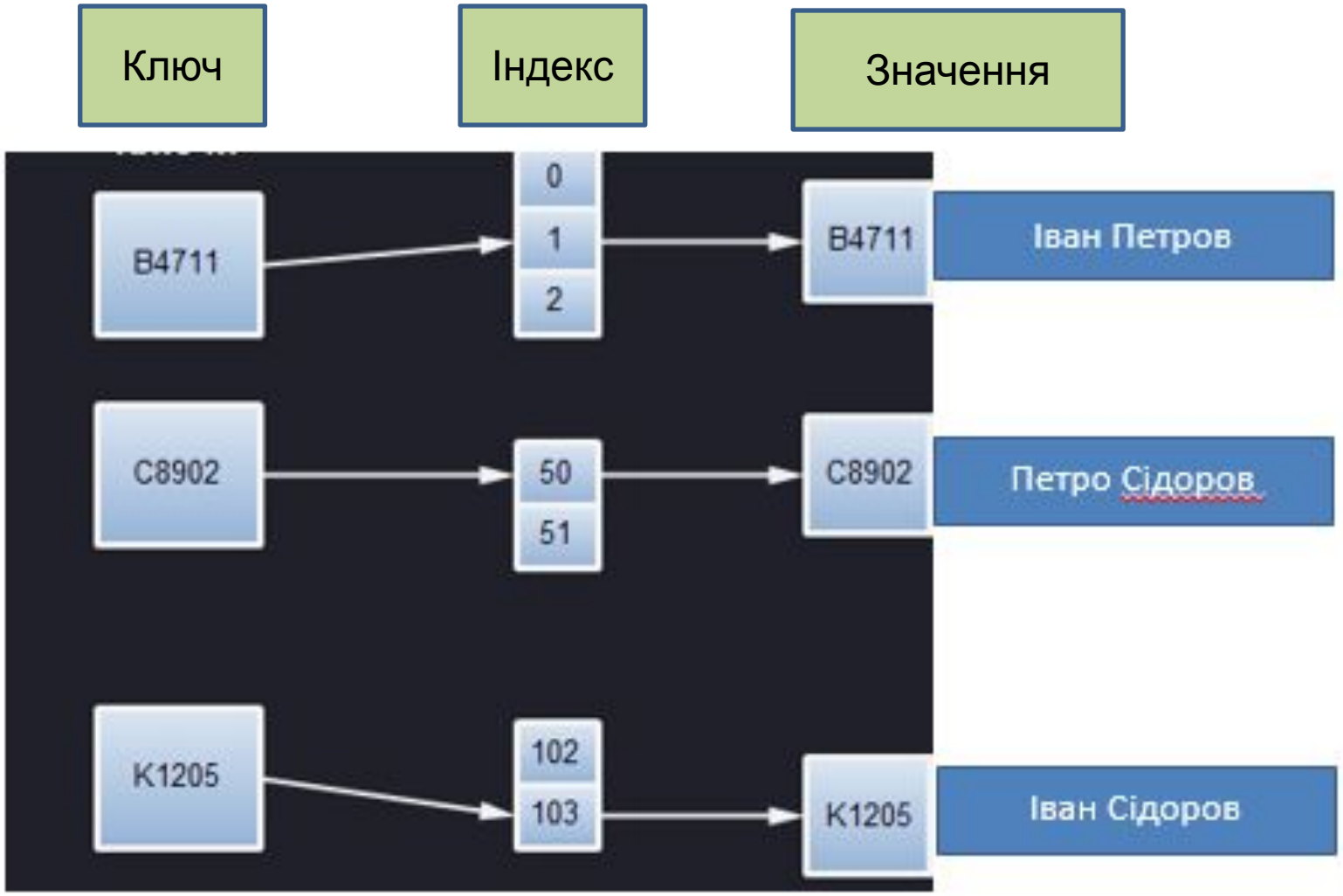
```
True
26089
26089
```

Словники

Словники – колекції в яких кожен елемент є парою ключ-значення

Словник	Внутрішня структура	Індекс	Швидкість
Несортовані			
`Dictionary<K,V>`	Хеш-таблиця	Ні	висока
`Hashtable`	Хеш-таблиця	Ні	висока
`ListDictionary`		Ні	низька
`OrderedDictionary`	Хеш-таблиця + масив	Так	висока
Сортовані			
`SortedDictionary<K,V>`		Ні	середня
`SortedList<K,V>`	Пара масивів	Так	середня
`SortedList`	Пара масивів	Так	середня

Словники


Ключ
Індекс
Значення
B4711
C8902
K1205
0
1
2
50
51
102
103
B4711
C8902
K1205
Іван Петров
Петро Сідоров
Іван Сідоров

Словники

```
class UserInfo
{

    public static Dictionary<int, string> MyDic(int i)
    {
        Dictionary<int, string> dic = new Dictionary<int, string>();
        Console.WriteLine("Введіть ім'я працівника: \n");
        string s;
        for (int j = 0; j < i; j++)
        {
            Console.Write("Name{0} --> ", j);
            s = Console.ReadLine();
            dic.Add(j, s);
            Console.Clear();
        }
        return dic;
    }
```

```
static void Main(string[] args)
{
    Console.Write("Скільки працівників додати? ");
    try
    {
        int i = int.Parse(Console.ReadLine());
        Dictionary<int, string> dic = UserInfo.MyDic(i);

        // Отримати колекцію ключів
        ICollection<int> keys = dic.Keys;

        Console.WriteLine("База даних містить: ");
        foreach (int j in keys)
            Console.WriteLine("ID -> {0}  Name -> {1}", j, dic[j]);
    }
    catch (FormatException)
    {
        Console.WriteLine("Помилка");
    }
    Console.ReadKey();
```

```
База даних м?стить:
ID -> 0  Name -> Petro
ID -> 1  Name -> Ivan
ID -> 2  Name -> Masha
```

Завдання на самостійну підготовку

1. *Законспектувати основні визначення*
2. *Виконати всі приклади, що були продемонстровані*