

Sets and Dictionaries Advanced

Sets and Multi-Dictionaries, Nested Dictionaries



SoftUni Team

Technical Trainers



SoftUni

Software University

<https://about.softuni.bg/>

1. Dictionary<K, V> Overview

2. Multi-Dictionaries

- A Key Holds Multiple Values

3. Nested Dictionaries

- A Dictionary Holding Another Dictionary

4. Set<T>

- **HashSet<T>** and **SortedSet<T>**
- **List<T>** vs **Set<T>**

sli.do

#csharp-advanced




Dictionary<K, V> Overview

Collection of Keys Mapped to Values

Associative Arrays (Maps, Dictionaries)

- Associative arrays are arrays indexed by keys
 - Not by the numbers 0, 1, 2, ... (like arrays)
- Hold a set of pairs {**key** → **value**}



Key	Value
John Smith	+1-555-8976
Lisa White	+1-555-1234
Sam Doe	+1-555-5030

- **Dictionary<K, V>**: collection of {key, value} pairs
- Keys are **unique**, each mapping to a value
- **Dictionary<K, V>** keeps the keys in their **order of addition**

```
var fruits = new Dictionary<string, double>();  
fruits["banana"] = 2.20;  
fruits["apple"] = 1.40;  
fruits["kiwi"] = 3.20;  
Console.WriteLine(string.Join(", ", fruits.Keys));
```

- **SortedDictionary<K, V>**: collection of {key, value} pairs
 - Keeps its **keys** always **sorted**
 - Implemented internally by a balanced search tree

```
var fruits = new SortedDictionary<string, double>();  
fruits["kiwi"] = 4.50;  
fruits["orange"] = 2.50;  
fruits["banana"] = 2.20;  
Console.WriteLine(string.Join(", ", fruits.Keys));
```

- Add(key, value) method

```
var airplanes = new Dictionary<string, int>();  
airplanes.Add("Boeing 737", 130);  
airplanes.Add("Airbus A320", 150);
```

- Remove(key) method

```
var airplanes = new Dictionary<string, int>();  
airplanes.Add("Boeing 737", 130);  
airplanes.Remove("Boeing 737");
```


- ContainsKey(key) – fast!

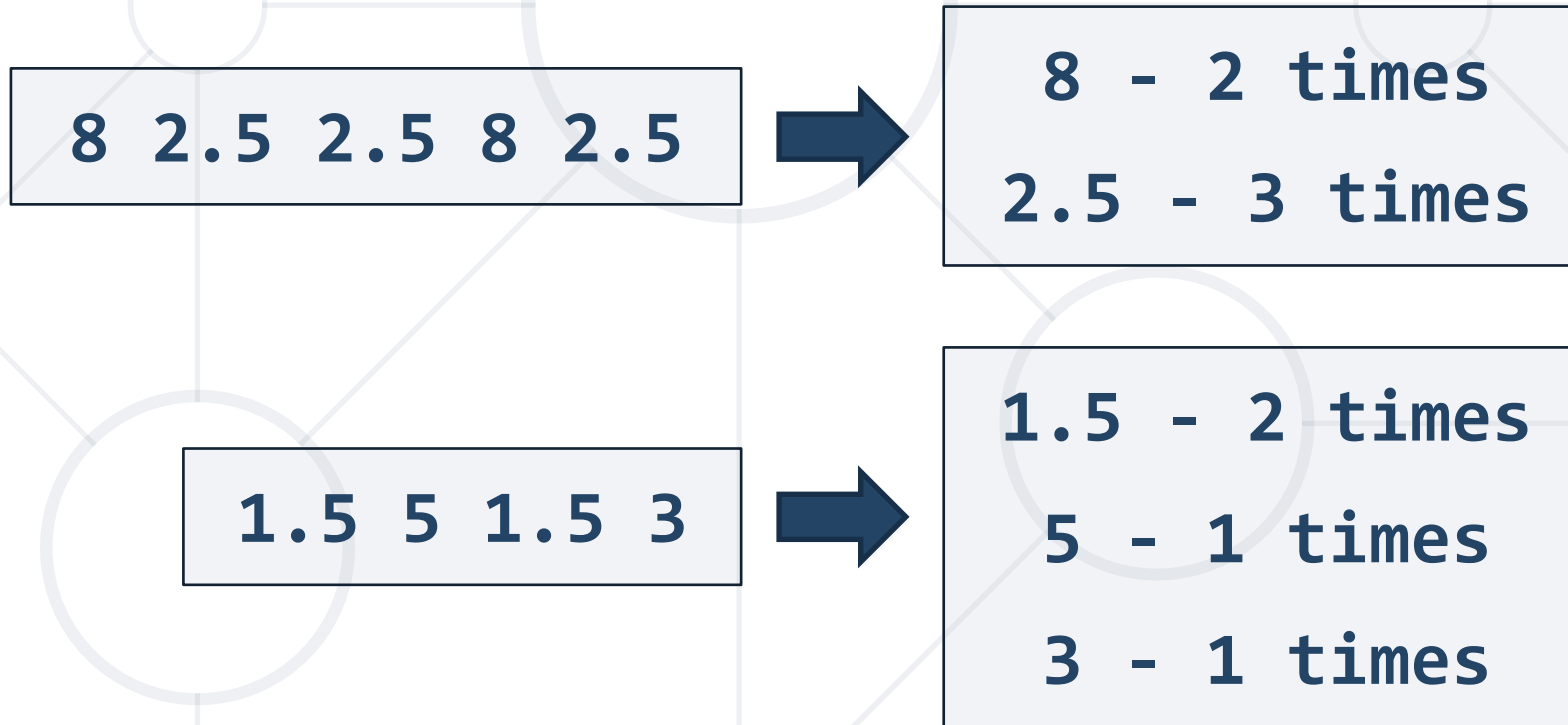
```
var dictionary = new Dictionary<string, int>();  
dictionary.Add("Airbus A320", 150);  
if (dictionary.ContainsKey("Airbus A320"))  
    Console.WriteLine($"Airbus A320 key exists");
```

- ContainsValue(value) – slow!

```
var dictionary = new Dictionary<string, int>();  
dictionary.Add("Airbus A320", 150);  
Console.WriteLine(airplanes.ContainsValue(150)); // True  
Console.WriteLine(airplanes.ContainsValue(100)); // False
```

Problem: Count Same Values in Array

- Read a list of **real numbers** and print them along with their **number of occurrences**



Check your solution here: <https://judge.softuni.org/Contests/Practice/Index/1465#0>

Solution: Count Same Values in Array

```
double[] nums = Console.ReadLine().Split(' ')
    .Select(double.Parse).ToArray();
var counts = new Dictionary<double, int>();
foreach (var num in nums)
    if (counts.ContainsKey(num))
        counts[num]++;
    else
        counts[num] = 1;
foreach (var num in counts)
    Console.WriteLine($"{num.Key} - {num.Value} times");
```

counts[num] hold show many times num occurs in nums

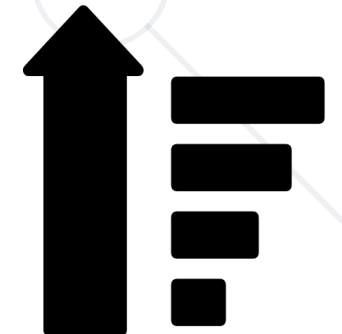
- Using OrderBy() to sort collections:

```
List<int> nums = { 1, 5, 2, 4, 3 };  
nums = nums  
    .OrderBy(num => num)  
    .ToList();
```



- Using OrderByDescending() to sort collections:

```
List<int> nums = { 1, 5, 2, 4, 3 };  
nums = nums.OrderByDescending(num => num).ToList();  
Console.WriteLine(String.Join(", ", nums));
```



Sorting Collections by Multiple Criteria

- Using ThenBy() to sort collections by multiple criteria:

```
var products = new Dictionary<int, string>();  
Dictionary<int, string> sortedDict = products  
    .OrderBy(pair => pair.Value)  
    .ThenBy(pair => pair.Key)  
    .ToDictionary(pair => pair.Key,  
                  pair => pair.Value);
```



Problem: Largest 3 Numbers

- Read a list of integers
- Print the **largest 3** of them (or less for shorter lists)
- Print them in **descending order**

10 30 15 20 50 5



50 30 20

1 2 3



3 2 1

20 30



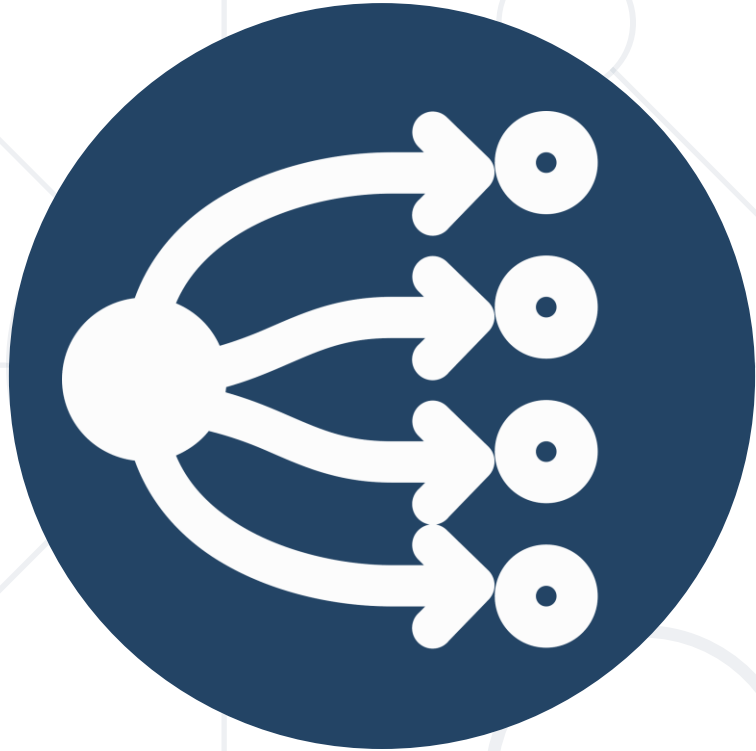
30 20

Check your solution here: <https://judge.softuni.org/Contests/Practice/Index/1465#2>

Solution: Largest 3 Numbers

```
int[] numbers = Console.ReadLine()
    .Split()
    .Select(int.Parse)
    .OrderByDescending(n => n)
    .ToArray();
int count = numbers.Length >= 3 ? 3 : numbers.Length;
for (int i = 0; i < count; i++)
    Console.Write($"{numbers[i]} ");
```

Check your solution here: <https://judge.softuni.org/Contests/Practice/Index/1465#2>




Multi-Dictionaries

Dictionaries Holding a List of Values

Multi-Dictionaries

- A dictionary could hold a **set of values** by given key
 - Example: student may have multiple grades:
 - Peter → [5, 5, 6]
 - Kevin → [6, 6, 3, 4, 6]



```
var grades = new Dictionary<string, List<int>>();  
grades["Peter"] = new List<int>();  
grades["Peter"].Add(5);  
grades["Kevin"] = new List<int>() { 6, 6, 3, 4, 6 };  
Console.WriteLine(string.Join(" ", grades["Kevin"]));
```

Adding Elements to Multi-Dictionary

```
static void AddStudentGrade(  
    Dictionary<string, List<double>> grades,  
    string studentName, double grade)  
{  
    if (! grades.ContainsKey(studentName))  
        grades.Add(studentName, new List<double>());  
    grades[studentName].Add(grade);  
}
```

Ensure that the list of grades **exist** for the target student

```
AddStudentGrade(grades, "Peter", 6);  
AddStudentGrade(grades, "Maria", 5);
```

Problem: Average Student Grades

- Write a program to read student **names + grades**
- Print the **students + average grade** for each student

6

Barney 5.20

Melissa 5.50

Melissa 2.50

Ted 2.00

Melissa 3.46

Ted 3.00



Barney -> 5.20 (avg: 5.20)

Melissa -> 5.50 2.50 3.46 (avg: 3.82)

Ted -> 2.00 3.00 (avg: 2.50)

Check your solution here: <https://judge.softuni.org/Contests/Practice/Index/1465#1>

Solution: Average Student Grades (1)

```
var grades = new Dictionary<string, List<decimal>>();  
var n = int.Parse(Console.ReadLine());  
for (int i = 0; i < n; i++) {  
    var tokens = Console.ReadLine().Split();  
    var name = tokens[0];  
    var grade = decimal.Parse(tokens[1]);  
    if (!grades.ContainsKey(name))  
        grades[name] = new List<decimal>();  
    grades[name].Add(grade);  
}  
// continues on next slide...
```

**Make sure the
list is initialized**

**Add the grade
into the list**

Solution: Average Student Grades (2)

```
foreach (var (name, studentsGrades) in grades)
{
    var average = studentGrades.Average();
    Console.Write($"{name} -> ");
    foreach (var grade in studentGrades)
        Console.Write($"{grade:f2} ");
    Console.WriteLine($"(avg: {average:f2})");
}
```

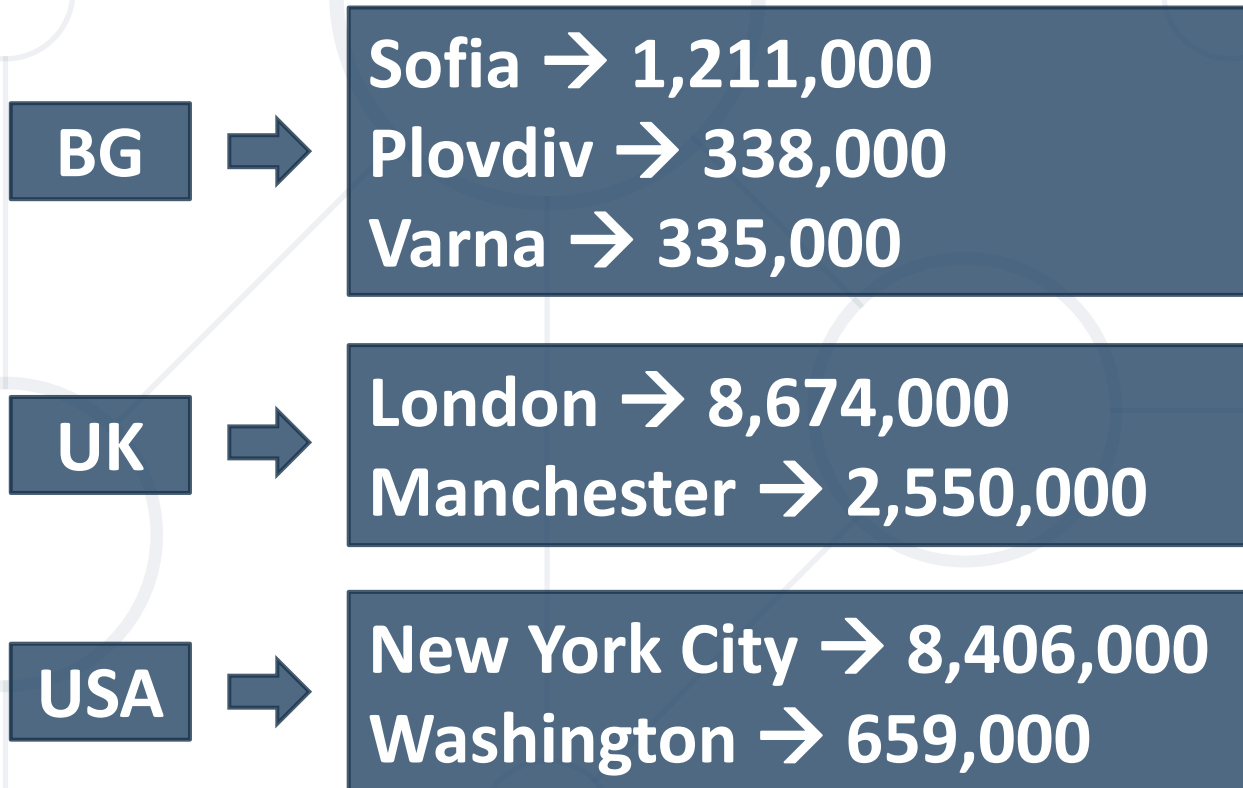
Europe	{Bulgaria → Sofia} {France → Paris} {Germany → Berlin}
Asia	{China → Beijing} {India → New Delhi}
Africa	{Nigeria → Abuja} {Kenya → Nairobi}

Nested Dictionaries

Dictionaries Holding Other Dictionaries

Nested Dictionaries

- A dictionary may hold another **dictionary** as value
- Example: population by country and city



Nested Dictionaries: Initialization

```
var population = new Dictionary<string, Dictionary<string, int>> {  
    {"BG",  
        new Dictionary<string, int> {  
            { "Sofia", 1_211_000 },  
            { "Plovdiv", 338_000 },  
            { "Varna", 335_000 },  
        }  
    },  
    {"UK",  
        new Dictionary<string, int> {  
            { "London", 8_674_000 },  
            { "Manchester", 2_550_000 },  
        }  
    }, ...  
};
```


Nested Dictionaries: Printing

```
foreach (var (countryName, towns) in population)
{
    Console.WriteLine("Country: " + countryName);
    foreach (var (townName, townPop) in towns)
        Console.WriteLine(
            $"\\tTown {townName} --> population {townPop}");
}
```

Microsoft Visual Studio Debug Console

```
Country: BG
    Town Sofia --> population 1211000
    Town Plovdiv --> population 338000
    Town Varna --> population 335000
Country: UK
    Town London --> population 8674000
    Town Manchester --> population 2550000
```

Nested Dictionaries: Adding New Entry

```
AddPopulation("China", "Shanghai", 24_300_000);
AddPopulation("China", "Beijing", 18_800_000);
AddPopulation("China", "Shenzhen", 12_700_000);
AddPopulation("BG", "Stara Zagora", 250_000);

void AddPopulation(string country, string town, int townPop)
{
    if (! population.ContainsKey(country))
        population[country] = new Dictionary<string, int>();
    population[country][town] = townPop;
}
```

Problem: Product Shop

- Write a program to keep information about **food shops**
 - The input holds triples: **{shop, product, price}**
 - If you receive an existing {shop + product}, **replace the price**
- Your output must be **ordered by shop name**

```
lidl, juice, 2.30  
kaufland, banana, 1.10  
lidl, grape, 2.20  
Revision
```

End command



```
kaufland->  
Product: banana, Price: 1.1  
lidl->  
Product: juice, Price: 2.3  
Product: grape, Price: 2.2
```

Solution: Product Shop (1)

```
var shops = new Dictionary<string, Dictionary<string, double>>();  
string line;  
while ((line = Console.ReadLine()) != "Revision")  
{  
    string[] productsInfo = line.Split(", ");  
    string shop = productsInfo[0];  
    string product = productsInfo[1];  
    double price = double.Parse(productsInfo[2]);  
    // continues on next slide...
```

Solution: Product Shop (2)

```
if (!shops.ContainsKey(shop))
{
    shops.Add(shop, new Dictionary<string, double>());
}
shops[shop].Add(product, price);
}
var orderedShops = shops.OrderBy(s => s.Key)
    .ToDictionary(x => x.Key, x => x.Value);
// TODO: Print the ordered dictionary
```

Make sure the inner dictionary is initialized

Problem: Cities by Continent and Country

- Write a program to read **continents**, **countries** and their **cities**, put them in a nested dictionary and print them

6

Europe Bulgaria Sofia

Asia China Beijing

Asia Japan Tokyo

Europe Poland Warsaw

Europe Germany Berlin

Europe Poland Poznan



Europe:

Bulgaria -> Sofia

Poland -> Warsaw, Poznan

Germany -> Berlin

Asia:

China -> Beijing

Japan -> Tokyo

Check your solution here: <https://judge.softuni.org/Contests/Practice/Index/1465#3>

Solution: Cities by Continent and Country (1)

```
var continentsData =  
    new Dictionary<string, Dictionary<string, List<string>>>>();  
var n = int.Parse(Console.ReadLine());  
for (int i = 0; i < n; i++) {  
    var tokens = Console.ReadLine().Split();  
    var continent = tokens[0];  
    var country = tokens[1];  
    var city = tokens[2];  
    // continues on next slide...
```

Solution: Cities by Continent and Country (2)

```
if (!continentsData.ContainsKey(continent)) {  
    continentsData[continent] = new Dictionary<string, List<string>>();  
}  
if (!continentsData[continent].ContainsKey(country)) {  
    continentsData[continent][country] = new List<string>();  
}  
continentsData[continent][country].Add(city);  
}
```

Initialize continent

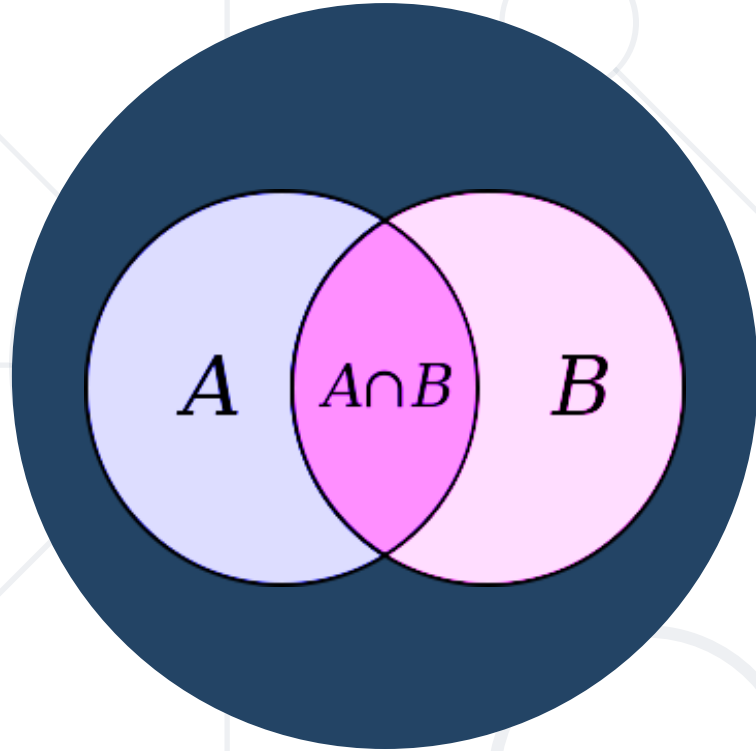
Initialize cities

Append the city
to the country

// continues on next slide...

Solution: Cities by Continent and Country (3)

```
foreach (var (continentName, countries) in continentsData)
{
    Console.WriteLine($"{continentName}:");
    foreach (var (countryName, cities) in countries)
    {
        // TODO: Print each country with its cities
    }
}
```



Set<T>

HashSet<T> and SortedSet<T>

Sets in C#

- A set keeps **unique elements**
 - Allows **add / remove / search** elements
 - Very **fast performance**
 - Example: Towns = {London, Tokyo, Paris, Rome}
- **HashSet<T>**
 - Keeps a set of elements in a **hash-table**
 - Elements are in **no particular order**
 - Similar to **List<T>**, but more efficient implementation



HashSet<T> – Example

```
HashSet<string> set = new HashSet<string>();  
set.Add("Peter");  
set.Add("Peter"); // Existing element → not added again  
set.Add("George");  
Console.WriteLine(string.Join(", ", set)); // Peter, George  
Console.WriteLine(set.Contains("Maria")); // False  
Console.WriteLine(set.Contains("Peter")); // True  
set.Remove("Peter");  
Console.WriteLine(set.Count); // 1
```

- List<T>

- Fast "add", **slow** "search" and "remove" (pass through each element)
- **Duplicates** are allowed
- The insertion **order** is guaranteed

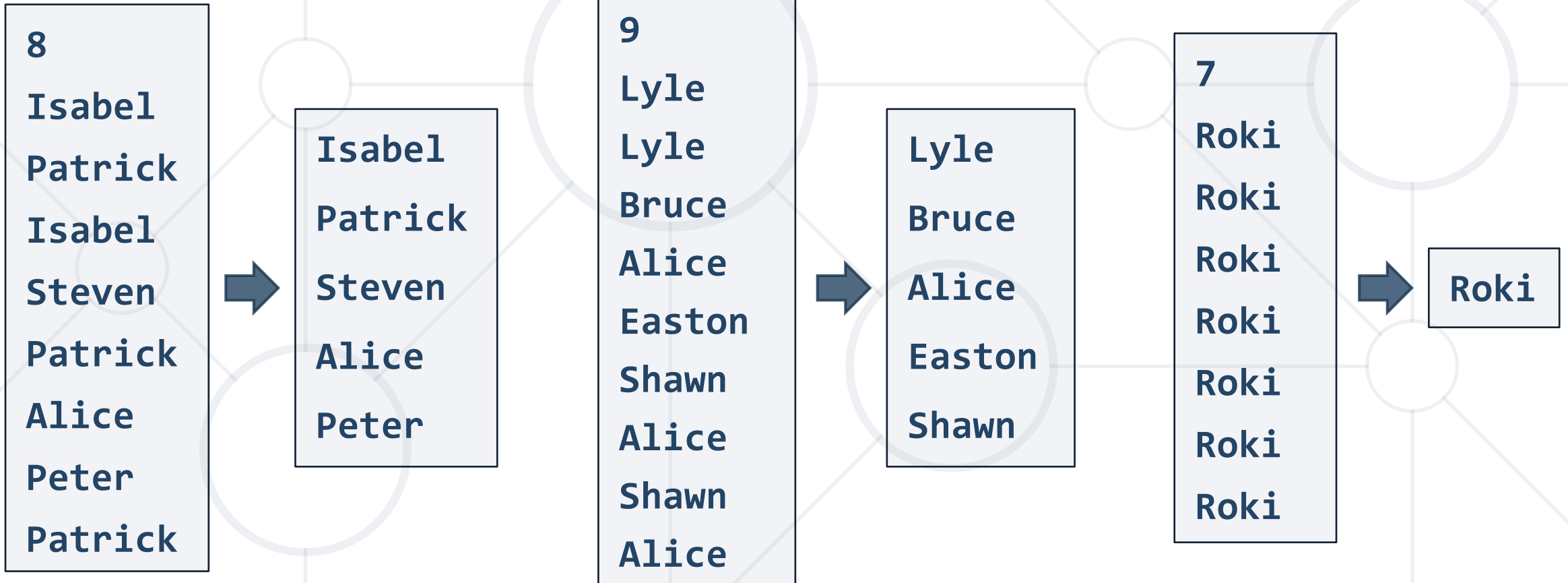
- HashSet<T>

- Fast "add", "search" and "remove" thanks to **hash-table**
- **No duplicates** are allowed
- Does not guarantee the insertion **order**



Problem: Record Unique Names

- Read a sequence of names and print only the **unique ones**



Check your solution here: <https://judge.softuni.org/Contests/Practice/Index/1465#4>

Solution: Record Unique Names

```
var names = new HashSet<string>();  
var n = int.Parse(Console.ReadLine());  
for (int i = 0; i < n; i++)  
{  
    var name = Console.ReadLine();  
    names.Add(name);  
}  
foreach (var name in names)  
    Console.WriteLine(name);
```

HashSet stores
unique values

Adds non-existing names only

SortedSet<T>

- SortedSet<T>
- The elements are **ordered incrementally**

```
var set = new SortedSet<string>();  
set.Add("Peter");  
set.Add("Peter");  
set.Add("George");  
set.Add("Maria");  
set.Add("Alice");  
Console.WriteLine(string.Join(", ", set));
```

Microsoft Visual Studio Debug Console

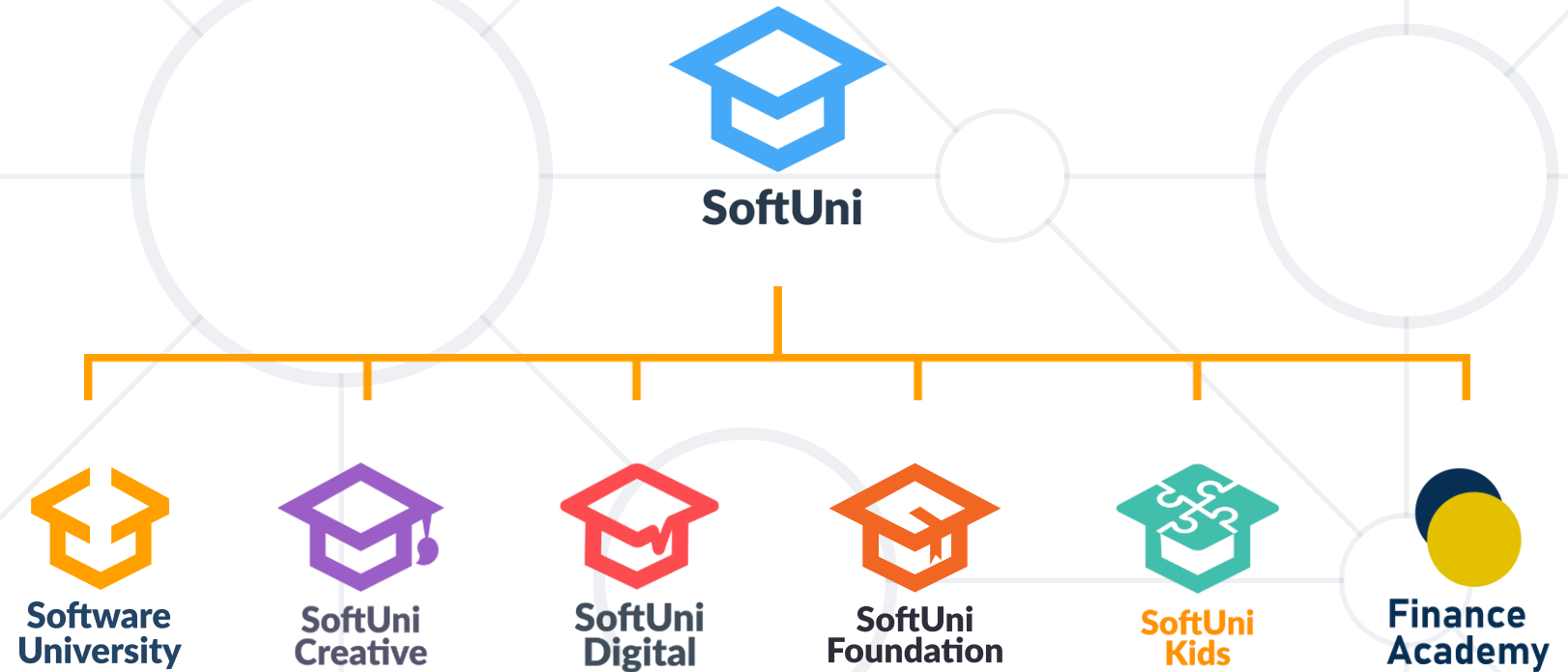
Alice, George, Maria, Peter



- Multi-dictionaries allow **keeping a collection** as a **dictionary value**
- Nested dictionaries **allow keeping a dictionary** as **dictionary value**
- Sets allow keeping **unique values** in **unspecified order**
 - No duplicates
 - Fast add, search & remove



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**



**Coca-Cola HBC
Bulgaria**

 **Flutter**TM
International

INDEAVR
Serving the high achievers



AMBITIONED

 **DRAFT
KINGS**



BOSCH

 **Postbank**
Решения за твоето утре

 **PHAR
VISION**



SmartIT

DXC
TECHNOLOGY

createX

- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

