Sets and Dictionaries Advanced

Sets and Multi-Dictionaries, Nested Dictionaries

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg/

Table of Contents

Dictionary<K, V>

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 Smith	+1-555-1234
Sam Doe	+1-555-5030

Dictionary

- **Dictionary<K, V>** - collection of key and value pairs
- Keys are **unique**
- 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;
```

Sorted Dictionary

- **SortedDictionary<K, V>**
- Keeps its keys always sorted
- Uses a balanced search tree

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

Built-In Methods

- **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");
```

Built-In Methods (2)

- **ContainsKey(key)** – very **fast** operation

```
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** operation

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

Problem: Count Same Values in Array

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

```
8 2.5 2.5 8 2.5
```

```
8 - 2 times
2.5 - 3 times
```

```
1.5 5 1.5 3
```

```
1.5 - 2 times
5 - 1 times
3 - 1 times
```

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] will hold how many times num occurs in nums

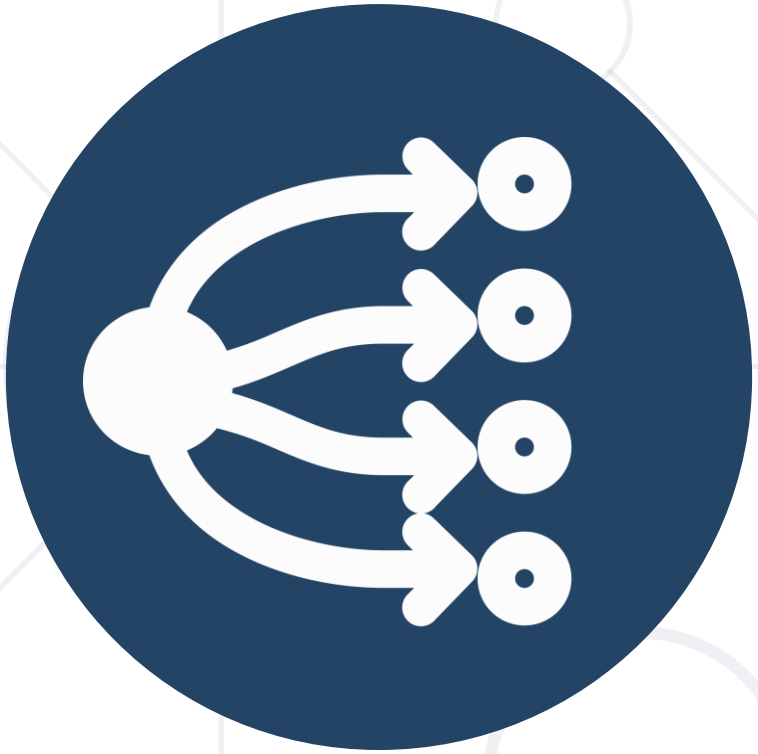
Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3178#3

Iterating Through a Dictionary

- Using **foreach loop**
- Iterates through objects of type **KeyValuePair<K, V>**
- Cannot modify the dictionary (**read-only**)

```
var fruits = new Dictionary<string, double>();
fruits.Add("banana", 2.20);
fruits.Add("kiwi", 4.50);
fruits.Add("orange", 3.20);
foreach (var fruit in fruits)
  Console.WriteLine($"{fruit.Key} -> {fruit.Value}");
```

fruit.Key -> fruit name
fruit.Value -> fruit price



Multi-Dictionaries

Multi-Dictionaries

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

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

Problem: Average Student Grades

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

```
6
Ivancho 5.20
Mariika 5.50
Mariika 2.50
Stamat 2.00
Mariika 3.46
Stamat 3.00
```

```
Ivancho -> 5.20 (avg: 5.20)
Mariika -> 5.50 2.50 3.46 (avg: 3.82)
Stamat -> 2.00 3.00 (avg: 2.50)
```

Solution: Average Student Grades (1)

```
var grades = new Dictionary<string, List<double>>();

var n = int.Parse(Console.ReadLine());

for (int i = 0; i < n; i++) {
  var tokens = Console.ReadLine().Split();
  var name = tokens[0];
  var grade = double.Parse(tokens[1]);
  if (!grades.ContainsKey(name))
    grades[name] = new List<double>();
  grades[name].Add(grade);
}

// continues on next slide ...
```

Make sure the list is initialized

Add grade into the list

Solution: Average Student Grades (2)

```
foreach (var pair in grades)
{
  var name = pair.Key;

  var studentGrades = pair.Value;

  var average = studentGrades.Average();

  Console.Write($"{name} -> ");

  foreach (var grade in studentGrades)
    Console.Write($"{grade:f2} ");

  Console.WriteLine($"(avg: {average:f2})");
}
```

KeyValuePair<string, List<double>

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3178#4

Nested Dictionaries

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

BG →
Sofia → 1,211,000
Plovdiv → 338,657

UK →
London → 8,674,000
Manchester → 2,550,000

USA →
New York City, NY → 8,406,000
Washington, DC → 658,893

Problem: Product Shop

- Write a program that stores information about **food shops**
- If you receive a shop you already have received **add the product**
- Your output must be **ordered by shop name**

{shop}, {product}, {price}

```
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

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3178#5

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
```

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);
}
// continues on next slide...
```

Initialize continent

Initialize cities

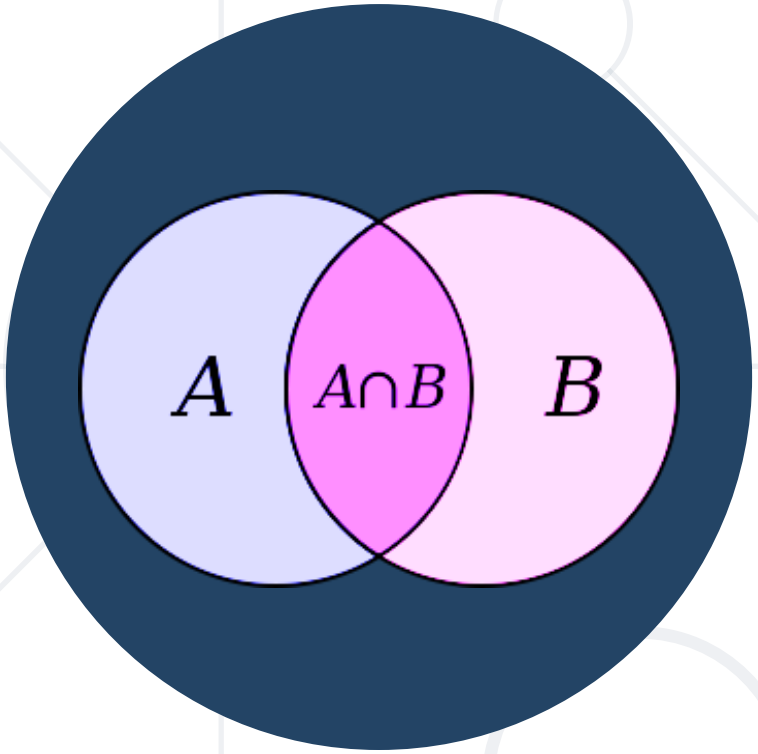
Append a city to the country

Solution: Cities by Continent and Country (3)

```
foreach (var continentCountries in continentsData) {
  var continentName = continentCountries.Key;
  Console.WriteLine($"{continentName}:");
  foreach (var countryCities in continentCountries.Value) {
    var countryName = countryCities.Key;
    var cities = countryCities.Value;
    // TODO: Print each country with its cities
  }
}
```

Cities in the country

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3178#6



HashSet<T> and SortedSet<T>

Sets in C#

- A set keeps **unique elements**
  - Allows **add** / **remove** / **search** elements
  - Very **fast performance**
- **`HashSet<T>`**
  - Keeps a set of elements in a **hash-table**
  - Elements are in **no particular order**
  - Similar to **`List<T>`**, but a different implementation

List<T> vs HashSet<T>

- **List<T>**
  - Fast "add", slow "search" and "remove" (pass through each element)
  - Duplicates are **allowed**
  - Insertion order is **guaranteed**
- **HashSet<T>**
  - Fast "add", "search" and "remove" thanks to **hash-table**
  - Does not allow duplicates
  - Does not guarantee the insertion order

HashSet<T> – Example

```
HashSet<string> set = new HashSet<string>();
set.Add("Pesho");
set.Add("Pesho"); // Not added again
set.Add("Gosho");
Console.WriteLine(string.Join(", ", set)); // Pesho, Gosho
Console.WriteLine(set.Contains("Georgi")); // false
Console.WriteLine(set.Contains("Pesho")); // true
set.Remove("Pesho");
Console.WriteLine(set.Count); // 1
```

Problem: Record Unique Names

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

```
8
Ivan
Pesho
Ivan
Stamat
Pesho
Alice
Peter
Pesho
```

```
Ivan
Pesho
Stamat
Alice
Peter
```

```
8
Lyle
Lyle
Bruce
Alice
Easton
Shawn
Alice
Shawn
```

```
Lyle
Bruce
Alice
Easton
Shawn
```

```
8
Roki
Roki
Roki
Roki
Roki
Roki
Roki
Roki
```

```
Roki
```

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

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3178#8

SortedSet<T>

- The **SortedSet<T>** class holds elements **ordered incrementally**

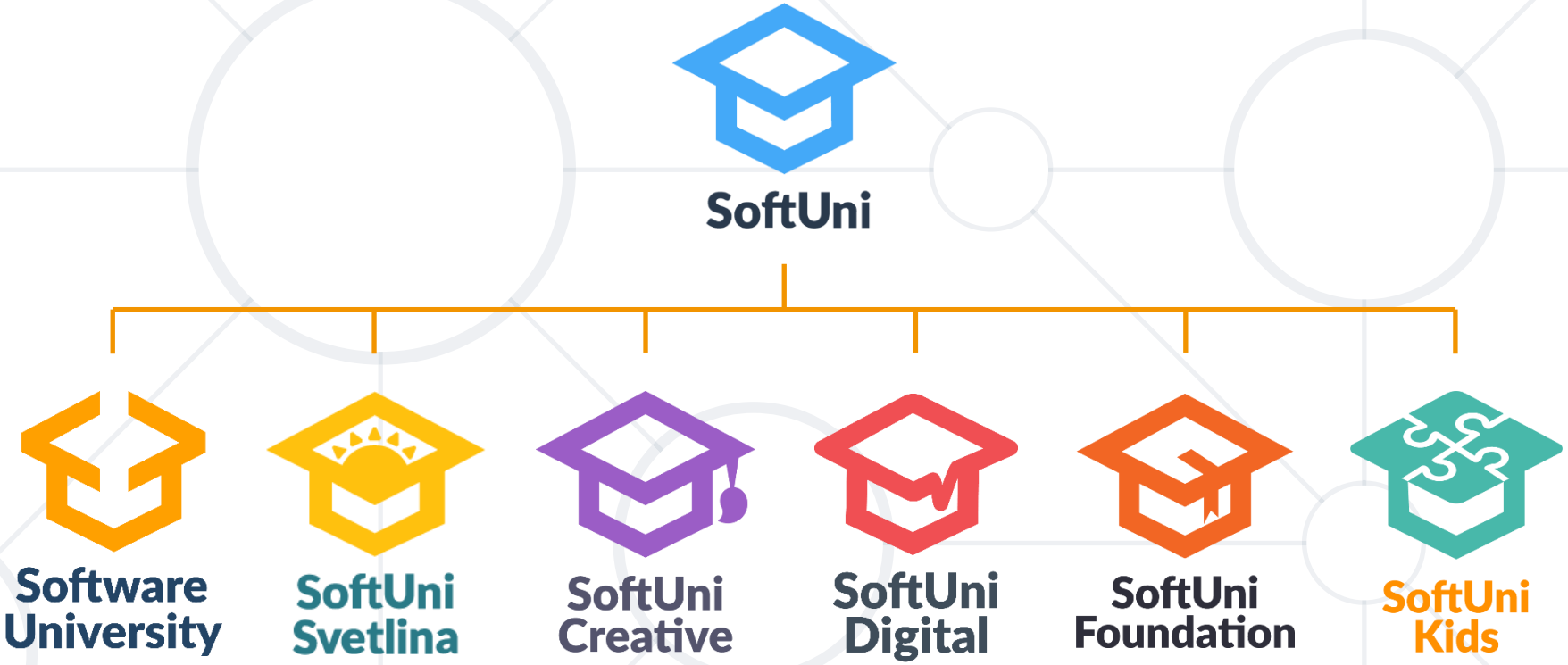
```
var set = new SortedSet<string>();
set.Add("Pesho");
set.Add("Pesho");
set.Add("Gosho");
set.Add("Maria");
set.Add("Alice");
Console.WriteLine(string.Join(", ", set));
```

Alice, Gosho, Maria, Pesho

Summary

- 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?



License

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