Lesson 36 - Collections V

`Dictionary<U, V>`

A **dictionary** is a collection of unique key/value pairs where a key is used to access the corresponding value. Dictionaries are used in database indexing, cache implementations, and so on.

The C# generic collection `Dictionary<K, V>` class requires all key/value pairs be of the same type K, V. Duplicate keys are **not permitted** to ensure that every key/value pair is unique.

`Dictionary<K, V>` properties include:

- `Count` - Gets the number of key/value pairs contained in the dictionary.
- `Item[K key]` - Gets the value associated with the specified key in the dictionary. Item is the indexer and is not required when accessing an element. You only need to use the brackets [] and key value.
- `Keys` - Gets an indexed collection containing only the keys contained in the dictionary.

`Dictionary<K, V>` methods include:

- `Add(K key, V value)` - Adds the key, value pair to the dictionary.
- `Remove(K key)` - Removes the key/value pair related to the specified key from the dictionary.

```
Dictionary<string, int> dict = new();

dict.Add("Ek", 1);
dict.Add("One", 1);
dict.Add("Do", 2);
dict.Add("Two", 2);

DictionaryPrinter(dict);

dict.Remove("One");

DictionaryPrinter(dict, "Removed");
```

```
static void DictionaryPrinter<Tvalue>(Dictionary<string, Tvalue> dict, string
name = "Dictionary")
{
    Console.WriteLine($"\n{name}");
    foreach (var item in dict)
        Console.WriteLine($"Key:{item.Key}\t Value:{item.Value}");
}
```

In the above example, the dictionary `dict` uses strings as it's keys and integers as the values.

Here are the additional `Dictionary<K, V>` properties and methods:

- `Values` - Gets an indexed collection containing only the values in the dictionary.
- `Clear()` - Removes all the key/value pairs from the dictionary.
- `ContainsKey(K key)` - Returns true if the specified key is present in the dictionary.
- `ContainsValue(V value)` - Returns true if the specified value is present in the dictionary.

`HashSet<T>`

A **hash set** is a set of unique values where duplicates are not allowed.

C# includes the `HashSet<T>` class in the generic collections namespace. All `HashSet<T>` elements are required to be of the same type `T`. Hash sets are different from other collections because they are simply a set of values. They do not have index positions and elements cannot be ordered.

> The `HashSet<T>` class provides high-performance set operations. HashSets allow fast lookup, addition, and removal of items, and can be used to implement either dynamic sets of items or lookup tables that allow finding an item by its key (e.g., finding the phone number of a person by the last name).

`HashSet<T>` **properties** include:

- `Count` Returns the number of values in the hash set.

And **methods** include:

- `Add(T t)` Adds a value (t) to the hash set.
- `IsSubsetOf(ICollection c)` Returns true if the hash set is a subset of the specified collection (c).

```
HashSet<int> hs = new();

hs.Add(1);
```

```
hs.Add(2);
hs.Add(3);
hs.Add(4);
hs.Add(5);

HashSetPrinter(hs);
hs.Remove(2);
HashSetPrinter(hs, "HashSet 1");

HashSet<int> hs2 = new();
hs2.Add(1);
hs2.Add(5);
HashSetPrinter(hs2, "HashSet 2");

Console.WriteLine("\nHS2 is subset of HS: " + hs2.IsSubsetOf(hs));

static void HashSetPrinter<T>(HashSet<T> hashSet, string name = "HashSet")
{
    Console.WriteLine($"\n{name} \nCount:{hashSet.Count}");
    foreach (var item in hashSet)
        Console.WriteLine(item);
}
```

Here are additional `HashSet<T>` methods:

- `Remove(T t)` Removes the value (t) from the hash set.
- `Clear()` Removes all the elements form the hash set.
- `Contains(T t)` Returns true when a value (t) is present in the hash set.
- `ToString()` Creates a string from the hash set.
- `IsSupersetOf(ICollection c)` Returns true if the hash set is a superset of the specified collection.
- `UnionWith(ICollection c)` Applies set union operation on the hash set and the specified collection (c).
- `IntersectWith(ICollection c)` Applies set intersection operation on the hash set and the specified collection (c).
- `ExceptWith(ICollection c)` Applies set difference operation on the hash set and the specified collection (c).