**Understanding HashSet in C#**

A HashSet<T> is a collection designed to store unique elements. It is particularly useful when you need a set-like behavior where duplicate entries are automatically ignored.

**Useful Methods in HashSet**

1. **Add**  
   Adds an element to the HashSet. Returns true if the item was successfully added; otherwise, false.

HashSet<int> numbers = new HashSet<int>();

numbers.Add(10); // true

numbers.Add(10); // false (duplicate)

1. **Remove**  
   Removes the specified element.

numbers.Remove(10);

1. **Contains**  
   Checks whether the HashSet contains a specific element.

bool exists = numbers.Contains(10);

1. **UnionWith**  
   Combines two sets by adding all unique elements from another collection.

HashSet<int> otherNumbers = new HashSet<int>() { 20, 30 };

numbers.UnionWith(otherNumbers);

1. **IntersectWith**  
   Retains only the elements that are in both the current HashSet and a specified collection.

numbers.IntersectWith(new HashSet<int>() { 10, 20 });

1. **ExceptWith**  
   Removes all elements in the specified collection from the current HashSet.

numbers.ExceptWith(new HashSet<int>() { 10 });

1. **IsSubsetOf / IsSupersetOf**  
   Checks if the current HashSet is a subset or superset of another collection.

bool isSubset = numbers.IsSubsetOf(new HashSet<int>() { 10, 20, 30 });

1. **SetEquals**  
   Checks if two sets contain the same elements.

bool isEqual = numbers.SetEquals(new HashSet<int>() { 10, 20 });

1. **Clear**  
   Removes all elements from the HashSet.

numbers.Clear();

10. **Symetric ExceptWith**

Retaining elements unique to each set using SymmetricExceptWith.

Symmetric difference between set1 and set2

11.**Overlaps**

To Check Wether There are a common element in the Two Collections

