Algorithms and Data Structures 2

Sets

Robert Horvick

www.pluralsight.com

Outline

- **Set overview**
- **Set class**
- **Algorithms**
  - Union
  - Intersection
  - Set Difference
  - Symmetric Difference
- **Sample Application**
- **.NET Framework and C++**

What is a set?

- **A collection of any type of comparable object**
- **Example – Sets of integers**
  - Odds = { …, -3, -1, 1, 3, …}
  - Evens = { … -4, -2, 0, 2, 4, … }
  - Negatives = { …, -5, -4, -3, -2, -1 }
  - Positives = {1, 2, 3, 4, 5 … }
  - Neutrals = { 0 }

Introducing The Set Class

```
public class Set<T> :  IEnumerable<T>
    where T: IComparable<T>
{
    private readonly List<T> _items = new List<T>();

    public Set();
    public Set(IEnumerable<T> items);

    public void Add(T item);
    public void AddRange(IEnumerable<T> items);

    public bool Remove(T item);
    public bool Contains(T item);

    public int Count { get; }

    public Set<T> Union(Set<T> other);
    public Set<T> Intersection(Set<T> other);
    public Set<T> Difference(Set<T> other);
    public Set<T> SymmetricDifference(Set<T> other);

    public IEnumerator<T> GetEnumerator();
    System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator();
}
```

Construction

- The items in the collection are stored in a simple container.

```
private readonly List<T> _items = new List<T>();
```

- An empty set can be constructed

```
public Set()
{
}
```

- A set can be constructed from an IEnumerable<T>

```
public Set(IEnumerable<T> items)
{
    AddRange(items);
}
```

Adding Items

- Individual items can be added to the Set

```
public void Add(T item)
{
    if (Contains(item))
    {
        throw new InvalidOperationException("Item already exists in Set");
    }

    _items.Add(item);
}
```

- Multiple items can be added to the Set

```
public void AddRange(IEnumerable<T> items)
{
    foreach (T item in items)
    {
        Add(item);
    }
}
```

Removing Items

- When an item is removed, all items equal to that item are removed.

```
public bool Remove(T item)
{
    return _items.Remove(item);
}
```

Contains and Count

- **The Set Count is the number of items in the collection**
  - Count is also known as “Cardinality”

```
public int Count
{
    get
    {
        return _items.Count;
    }
}
```

- **Contains determines if an item exists in the Set**

```
public bool Contains(T item)
{
    return _items.Contains(item);
}
```

Enumeration

- The Set provides the ability to Enumerate over every item in the Set

```
public IEnumerator<T> GetEnumerator()
{
    return _items.GetEnumerator();
}

IEnumerator System.Collections.IEnumerable.GetEnumerator()
{
    return _items.GetEnumerator();
}
```

Union

- A Set algorithm that compares two Sets and returns a third Set that contains all of the unique items in both Sets.
- The union of {1, 2, 3} and {3, 4, 5} is {1, 2, 3, 4, 5}

```
public Set<T> Union(Set<T> other)
{
    Set<T> result = new Set<T>(_items);
    result.AddRangeSkipDuplicates(other._items);

    return result;
}
```

Intersection

- A Set algorithm that compares two Sets and returns a third Set that contains all of the intersecting, or matching, members of both Sets.
- The intersection of {1, 2, 3} and {2, 3, 4} is {2, 3}

```
public Set<T> Intersection(Set<T> other)
{
    Set<T> result = new Set<T>();

    foreach (T item in _items)
    {
        if (other._items.Contains(item))
        {
            result.Add(item);
        }
    }

    return result;
}
```

Set Difference

- A Set algorithm that takes two inputs Sets (A and B, respectively) and returns all the items of A that are not members of B.
- The Set Difference of { 2, 3, 4 } and { 3, 4, 5 } is { 2 }

```
public Set<T> Difference(Set<T> other)
{
    Set<T> result = new Set<T>(_items);

    foreach (T item in other._items)
    {
        result.Remove(item);
    }

    return result;
}
```

Symmetric Difference

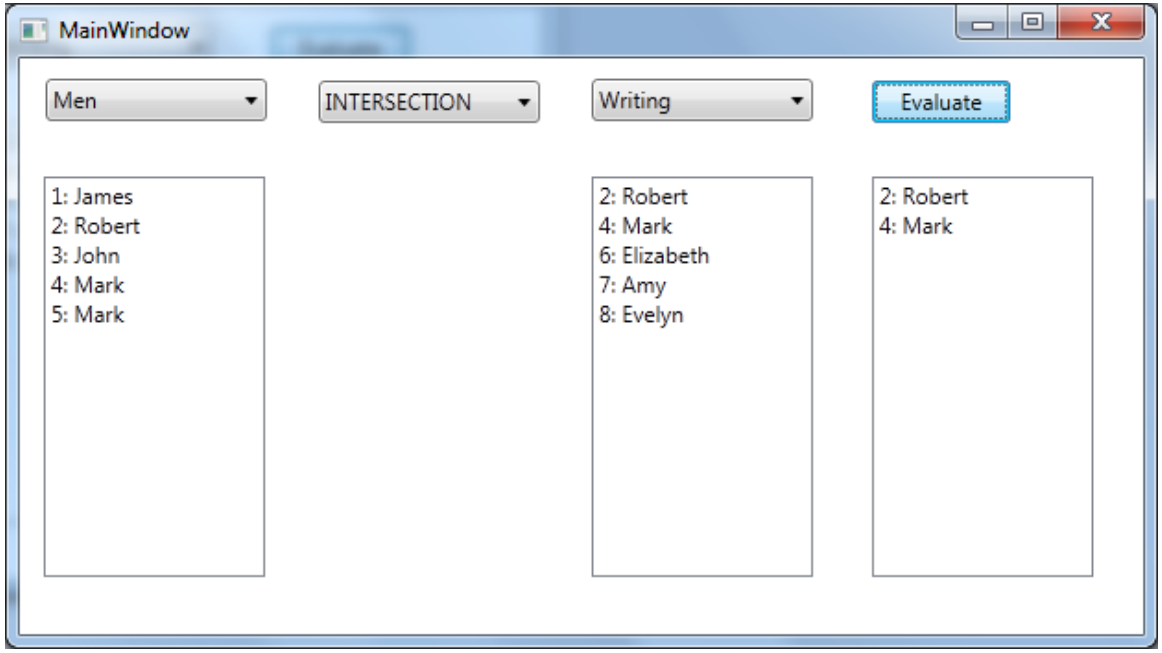
- **A Set algorithm that takes two input Sets and returns a third Set that contains all of the members of both input Sets that are not in the other.**
  - The symmetric difference is the set difference of the intersection and union of the input sets.
- **The symmetric difference of {1, 2, 3} and {2, 3, 4} is {1, 4}**
  - Intersection = {2,3}
  - Union = {1,2,3,4}
  - Set Difference of {1,2,3,4} and {2,3} is {1,4}

```
public Set<T> SymmetricDifference(Set<T> other)
{
    Set<T> intersection = Intersection(other);
    Set<T> union = Union(other);

    return union.Difference(intersection);
}
```

Sample Application

- **Performs the four set algorithms on several sets of sample data**
  - Men
  - Women
  - Class Enrollment



.NET Framework

- **The .NET Framework contains the HashSet<T> class**
  - Method names differ from example Set class

Set Algorithm	HashSet<T> Method
Union	UnionWith
Intersection	Intersect
Set Difference	Except
Symmetric Difference	SymmetricExceptWith

C++

- **C++ implements the set algorithms outside of the collection types.**
  - set_union, set_intersection, set_difference, set_symmetric_difference

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

void main()
{
    std::vector<std::string> peopleInMathClass, peopleInGymClass, peopleInMathAndGym(3);

    peopleInGymClass.push_back("Biff");
    peopleInGymClass.push_back("Marty");

    peopleInMathClass.push_back("Emmett");
    peopleInMathClass.push_back("Marty");

    set_intersection(peopleInGymClass.begin(),  peopleInGymClass.end(),
              peopleInMathClass.begin(), peopleInMathClass.end(),
              peopleInMathAndGym.begin());

    for (std::vector<std::string>::iterator it=peopleInMathAndGym.begin(); it < peopleInMathAndGym.end(); it++ ) {
        std::cout << *it << std::endl;
    }
}
```

Summary

- **Set overview**
- **Set class**
- **Algorithms**
  - Union
  - Intersection
  - Set Difference
  - Symmetric Difference
- **Sample Application**
- **.NET Framework and C++**

References

- **Wikipedia Articles relevant to sets and set theory**
  - http://en.wikipedia.org/wiki/Set_theory
  - http://en.wikipedia.org/wiki/Set_(computer_science)
  - http://en.wikipedia.org/wiki/Set_(mathematics)

- **MSDN Articles for Referenced Types**
  - HashSet: http://msdn.microsoft.com/en-us/library/bb359438.aspx
  - <algorithm> http://msdn.microsoft.com/en-us/library/chzkfc23.aspx