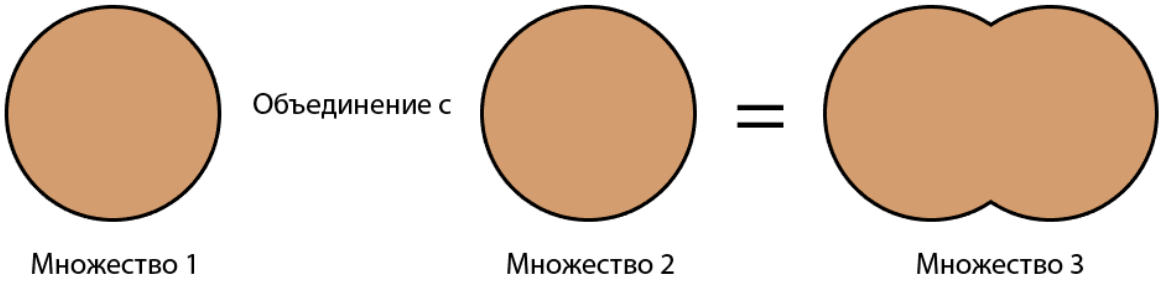
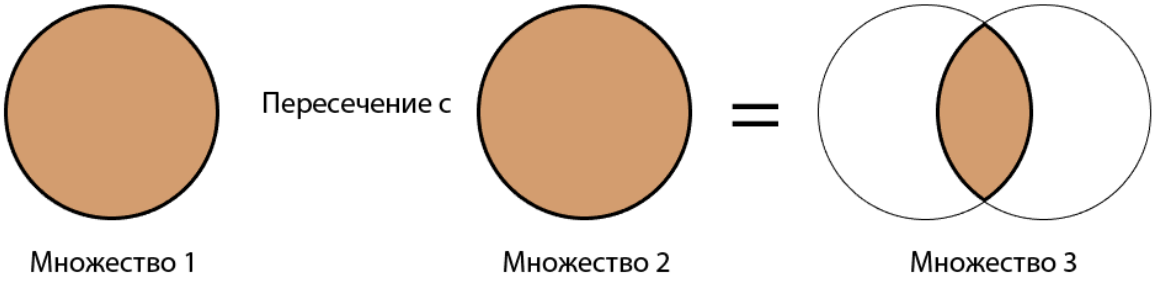
Создайте класс коллекции Set – множество чисел типа int. Множество – структура данных, наподобие списка, с несколькими особенностями: все элементы множества уникальны. Т.е. в множестве не может присутствовать 2 одинаковых элемента. Если попытаться добавить элемент, который уже присутствует во множестве, ничего не должно происходить.

Также для множества определяются специфичные операции:

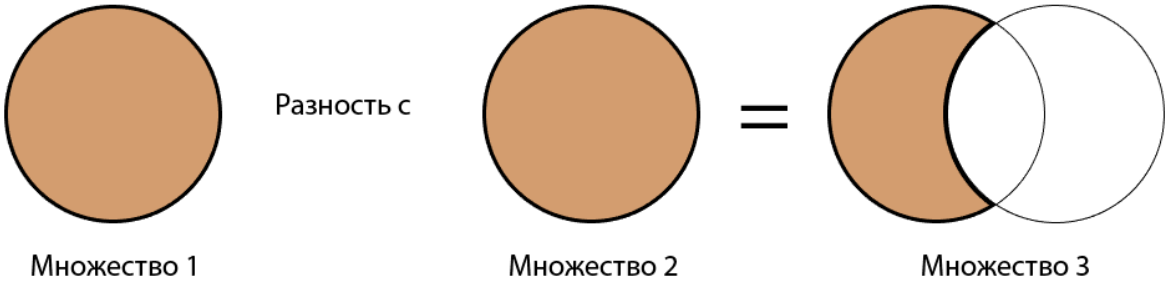
- Объединение 2 множеств – получение нового множества на основании 2 других, состоящее изо всех элементов обоих множеств, без повторов.



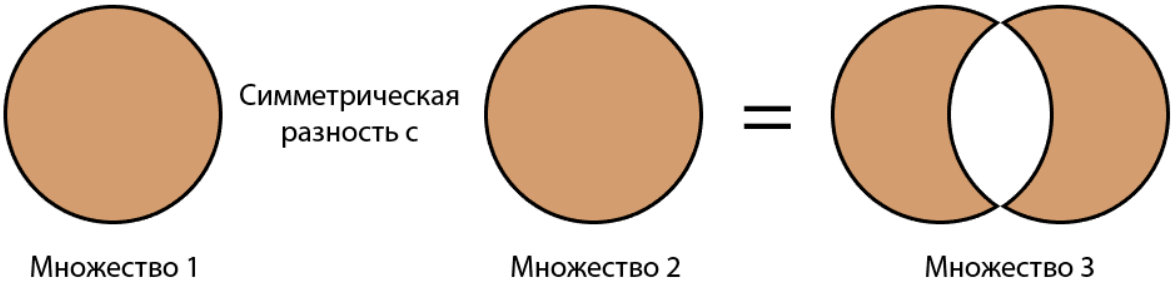
- Пересечение 2 множеств – получение нового множества на основании 2 других, состоящее только из тех элементов, которые есть и в первом, и во втором множестве.



- Разность 2 множеств – получение нового множества, состоящего только из тех элементов первого множества, которых нет во втором.



- Симметрическая разность 2 множеств – получение нового множества, состоящего из элементов первого и второго множеств, без элементов, которые присутствуют и там, и там.



1. Внутри класса элементы могут храниться в виде списка.
2. Нужно сделать конструктор, который будет инициализировать множество на основе массива (параметр – массив чисел).
3. Написать свойство для количества элементов.
4. Написать методы Add и Remove для добавления и удаления элементов из множества, соответственно. Добавление должно быть реализовано таким образом, чтобы повторки не добавлялись.
5. Реализуйте метод для вывода всех элементов множества на консоль.
6. Операции над множествами (объединение, пересечение, разность, симметрическая разность) должны быть реализованы через методы (можно обычные, можно статические). Пример их работы:

```
Set s1 = new Set(new int[] { 3, 4, 7, 2, -3 });
Set s2 = new Set(new int[] { 6, 3, 1, -3, 8, 13, 2 });

// Объединение s1 и s2
Set s3 = s1.Union(s2);
// s3 = { 3, 4, 7, 2, -3, 6, 1, 8, 13 }

// Пересечение s1 и s2
Set s4 = s1.Intersect(s2);
// s4 = { 3, 2, -3 }

// Разность s1 и s2
Set s5 = s1.Difference(s2);
// s5 = { 4, 7 }

// Симметрическая разность s1 и s2
Set s6 = s1.SymmetricDifference(s2);
// s6 = { 4, 7, 6, 1, 8, 13 }
```

7. Реализуйте метод проверки, является ли одно множество подмножеством другим, т.е. есть ли все элементы первого множества во втором:

```
Set s7 = new Set(new int[] { 1, 2, 6 });
Set s8 = new Set(new int[] { 3, 5, 8, 4 });
Set s9 = new Set(new int[] { 3, 4, 5, 6, 7, 8 });

// Проверка, является ли s7 подмножеством множества s9
s7.IsSubsetOf(s9);
// false, т.к. в множестве s9 нет элементов 1 и 2, которые есть в s7

// Проверка, является ли s8 подмножеством множества s9
s8.IsSubsetOf(s9);
 // true, т.к. s8 состоит из элементов s9
```

Часть 2.

1. Переделайте класс, заменив методы действия над множествами на перегрузки операторов (которые работают также):
   - Объединение множеств – с помощью оператора бинарного + :

```
// Объединение s1 и s2
Set s3 = s1 + s2;
```

   - Пересечение множеств – с помощью оператора * :

```
// Пересечение s1 и s2
Set s4 = s1 * s2;
```

- Разность множеств – с помощью оператора бинарного - :

```
// Разность s1 и s2
Set s5 = s1 - s2;
```

- Симметрическую разность множеств – с помощью оператора % :

```
// Симметрическая разность s1 и s2
Set s6 = s1 % s2;
```

- Проверку на подмножество – с помощью операторов > и < :

```
// Проверка, является ли s7 подмножеством множества s9
s9 > s7;

// Проверка, является ли s8 подмножеством множества s9
s8 < s9;
```

2. Сделайте класс Set обобщённым с типом-параметром T. T.e. чтобы он мог хранить не только целые числа, но любые типы данных. Если элементы сравнивались у вас с помощью операторов == и !=, замените их на вызовы метода Equals.
3. Продемонстрируйте работу вашего класса для 3 различных типов данных, один из которых должен быть вашим классом из предыдущих заданий.