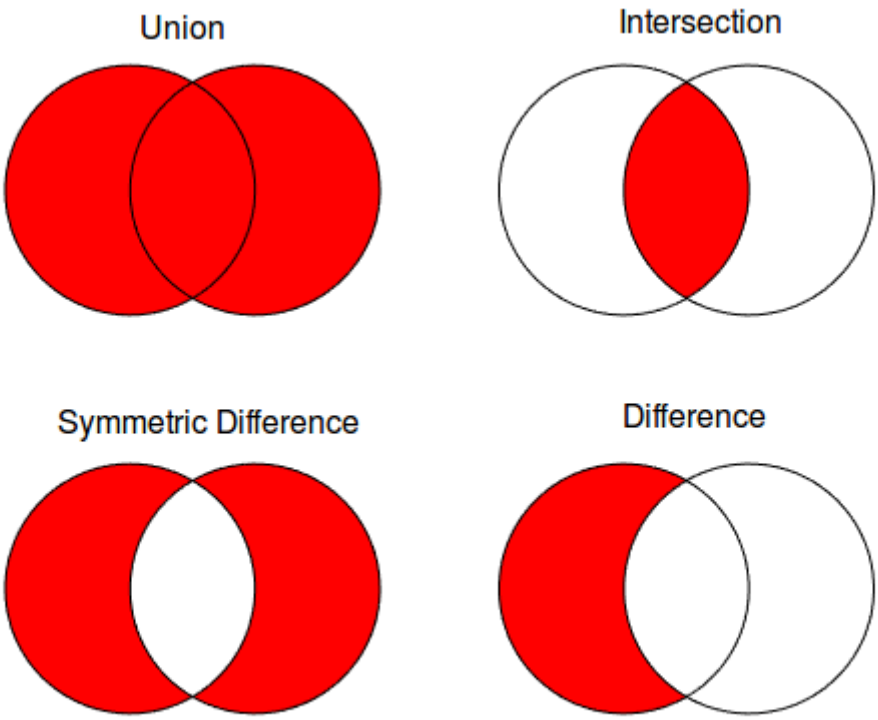
Intset

The purpose of this assignment is to use test-first-design to implement a class for representing sets of integers. You will implement the set class in Java after writing your tests. Java already offers some set classes in its libraries, but you are not allowed to use these, otherwise, the assignment becomes very trivial. Instead, try to implement your set class using a data structure such as a list (ArrayList in Java). You have already been provided with a test file and an incomplete stub class. Before you start, read through the Testing chapter of the reader. Throughout this course we expect you to use JUnit 5.

Sets

A quick recap on sets: a set is a collection of elements (in this case, integers) in which each element appears no more than once. A set supports the following operations:



- **Union:** C is a set which is the union of sets A and B if C contains only elements of A or B or both and no other elements.
  Notation: C = A ∪ B
- **Intersection:** C is a set which is the intersection of sets A and B if C contains only elements that are present in both A and B.
  Notation: C = A ∩ B
- **Difference:** C is a set which is the difference of sets A and B if C contains only those elements that are present in A, but not in B.
  Notation: C = A \ B
  Note that C = A \ B is different from C = B \ A

- **Symmetric Difference:** C is a set which is the symmetric difference of sets A and B if C contains only those elements that are present in either A or B, but not in both.
  Notation: C = A $\oplus$ B or C = A $\triangle$ B

Assignment

To start, open the given `pom.xml` in an IDE of your choice (we recommend IntelliJ). Most modern IDEs allow opening a pom file for the project without you having to create a new project.

The work for this assignment will be done in two files: `IntSet.java` and `IntSetTest.java`. The `IntSet.java` is a stub class. It contains most of the methods your class should support. Not all operations in the list below are provided as a stub though. Do not forget to add and implement these yourself!

The `IntSetTest.java` is an empty test class. You will have to add the test methods here.

Use the method as described in the lectures and reader for using JUnit to test each of the given operations carefully and completely. Take care to also test the constructor itself. Always write the test(s) first, and *then* implement the corresponding code in such a way that it passes all tests.

The set you will have to implement also has a capacity, i.e. a maximum number of elements it can contain.

Operations

Below you will find a list of all the operations your class has to support. We expect every single one of these operations to be tested! When writing the tests, think about all the possible cases that your code could encounter. At the same time, take into consideration all the corner cases (null, empty sets, illegal examples, etc.). **Do not just test what a method should do, also test what a method shouldn't do!**

The operations are as follows:

- Creating a new empty set (constructor). Make sure to check that the creation is successful and that indeed the new set is empty.
- Checking if a set is empty (`isEmpty`).
- Checking if a set contains a value (`has`).
- Adding elements to the set (`add`). Make sure to test adding the same element twice.
- Removing elements from the set (`remove`). Make sure to test removing non-existing items.
- Union (`union`).
- Intersection (`intersect`).
- Difference (`difference`).
- Symmetric difference (`symmetricDifference`)
- Retrieving an array representation of the set (`getArray`). Also test that this method does not influence the content of the set itself.
- Retrieving the number of elements in the set (`getCount`).
- Retrieving the capacity of the set (`getCapacity`).
- Retrieving a string representation of the set (`toString`). Again, test that this method does not influence the content of the set itself. Make sure to test all possible types of sets with this method.

Please note that you have to think of explicit test cases yourself and that not all of these are listed here. We will, among other things, be looking at how many and which cases you managed to test, so try to be thorough. At

the same time, pay attention that you are not testing the same thing multiple times. Note that in the given code, not all operations have a stub yet and for some of them you will have to implement the entire method.

General remarks

Documentation is important for all of the assignments. Therefore, please make sure you properly document your code with meaningful comments and explanations, while at the same not being too overzealous. In most cases, this means that Javadoc is sufficient. While test classes are technically not part of the actual code, they should still be somewhat readable and maintainable.