1.1 Assignment 1: Intset

The purpose of this assignment is to use test-first-design to implement a class for representing sets of integers. You will implement a set class in Java after writing your test class and stub class. Java already offers some set classes in its libraries, but you cannot 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).

Remember: a set is a collection of elements (in our case integer values) in which each element appears at most once. A set has the following operations:

- **Union**: C is the union of sets A and B if C contains all elements of A and B together and no other elements.
  Notation: $C = A \cup B$.
- **Intersection**: C is the intersection of A and B if C contains only those elements that are present in A and in B
  Notation: $C = A \cap B$.
- **Difference**: C is the difference of A and B if C contains only those elements that are present in A but not in B.
  Notation: $C = A \setminus B$.
  Note that $C = A \setminus B$ is different from $C = B \setminus A$.
- **Symmetric difference**: C is the symmetric difference of A and B if C contains only those elements that are present in A or in B but not in both. Notation: $C = A \oplus B$.

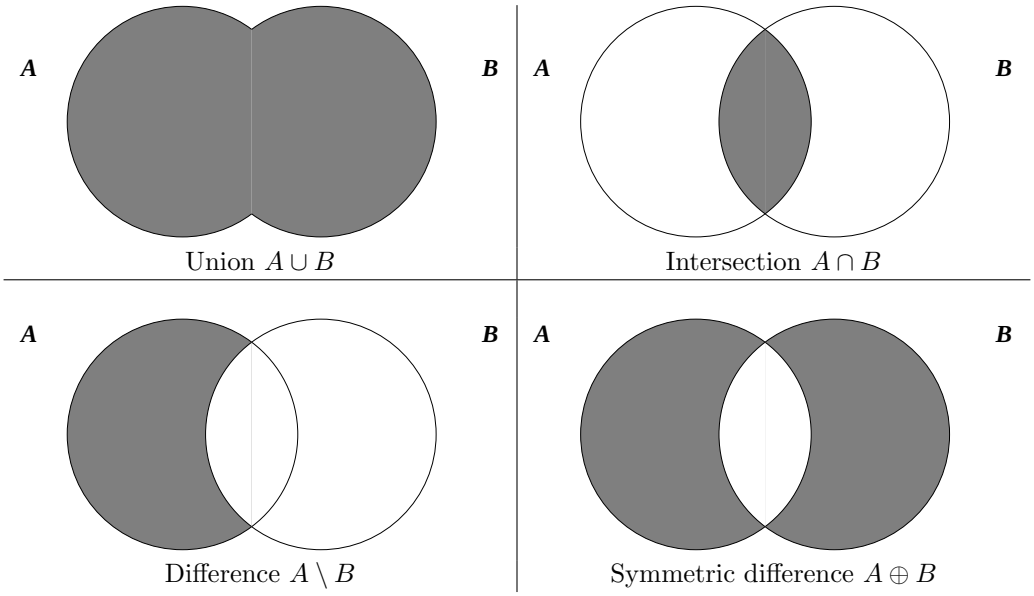

Figure 1.1: Operations on sets

- Make a new eclipse project with the given file IntSet.java as first source file.
- Use the method as described in the lectures for using Junit to test each of the given operations carefully and completely. Take care of also testing the constructor itself. Always first write the test(s) and then implement the corresponding code in such a way that the test will succeed.

  A broad overview of operations that should be tested is given below. Please note that you have to think of explicit test cases yourself and that not all of these are listed here. Furthermore, for some operations listed above you need to implement the entire method, as opposed to filling in the blanks (e.g. symmetric difference).

  – Creating a new empty set

- Checking if a set is empty
- Adding and removing elements from the set (including removing non-existing elements, adding the same element twice, etc)
- Checking if an element is part of the set (with method has)
- Returning the number of elements in a set using getCount
- Creating new sets by using one or more of the above operations.
- Returning an array representation using getArray (and also test that this method will not influence the content of the set itself)
- Returning a String-representation of the set using toString (and, again, test that this method will not influence the content of the set itself)
- ...