Advanced Programming – Assignment 1
Easy operations on collections of identifiers

Important notice on forehand
When necessary, use the following methods for the assignments below.

N.B. For the following methods to work, the delimiter of the Scanner-object has to be set to the empty String by calling the `useDelimiter()` method.

```
// Method to read 1 character.
char nextChar (Scanner in) {
    return in.next().charAt(0);
}
```

The three methods below have in common that they do not read a character from input, but return a boolean indicating whether a specific character could be read with `nextChar()`.

```
// Method to check if the next character to be read when
// calling nextChar() is equal to the provided character.

boolean nextCharIs(Scanner in, char c) {
    return in.hasNext(Pattern.quote(c+""));
}
```

To be able to use the class Pattern, include the following import in your program:

```
import java.util.regex.Pattern;

// Method to check if the next character to be read when
// calling nextChar() is a digit.

boolean nextCharIsDigit (Scanner in) {
    return in.hasNext("[0-9]");
}

// Method to check if the next character to be read when
// calling nextChar() is a letter.

boolean nextCharIsLetter (Scanner in) {
    return in.hasNext("[a-zA-Z]");
}
```

Description assignment

An interactive program is a program that has interaction between the user and the program. Concretely: the program asks questions and the user gives answers, after which the program processes these answers and prints the results.

For this assignment a program must be written that reads two collections of identifiers from standard input, applies four operations on these collections, and prints the results of these operations in standard output.

The **identifiers**, which form the elements of the collections used in this assignment, must have the following properties:

- Only alphanumeric characters are allowed as elements of an identifier.
- An identifier begins with a letter.
- Identifiers have a length of at least 1 character.

The collections, used in this assignment, have to meet to following properties:

- Only identifiers, following the restrictions used in this assignments, are allowed as elements of a collection.
- Collections can contain a minimum of 0 and a maximum of 20 identifiers.

The four **operations** on collections are defined as follows:

1. *difference* : all elements contained in the 1st but not the 2nd collection.
2. *intersection* : all elements contained in both collections.
3. *union* : all elements of both collections. (N.B. Collections do not contain duplicate elements per definition.)
4. *symmetric difference* : all elements of both collections that are not contained in the intersection.

Input to the program is through standard input (the keyboard) and output uses standard output (the screen). The program asks two times for a collection of identifiers and subsequently computes the difference, intersection, union, and symmetric difference. The program is supposed to repeat the process described above until CTRL-D (or in the case of Windows: CTRL-Z) is entered as input to one of the questions of the program. E.g. :

```
Give the first collection : {aap noot mies mies2}
Give the second collection : {aal w xyz noot}
difference   = aap mies mies2
intersection = noot
union        = aap noot mies2 aal w xyz mies
sym. diff.   = aap mies aal w xyz mies2

Give the first collection : {een twee drie}
Give the second collection : {}
difference   = een twee drie
intersection =
union        = een twee drie
sym. diff.   = een twee drie

Give the first collection :
Give the first collection : {piet jan henk
```

```
Missing '}'
Give the first collection : {piet jan henk}
Give the second collection : ^D
```

Note that the text following the colon was typed by the user.

Collections on the input can contain a maximum of only 10 identifiers, to ensure that it is always possible to determine the union and symmetric difference. The identifiers in a collection on the input are delimited by spaces and are preceded and closed by respectively '{' and '}'. Because collections are unordered by definition, there are no restrictions on the order of identifiers in the input and output of a collection. On improper input, a clear error message has to be given in one line. Absence of input is not seen as an error, but should lead to a repeat of the question.

For the first **meeting** with the assistant you should have :

- studied the assignment and recognized which objects need to be used in the implementation
- written a specification for each object
- created a design of the program

A specification for an object has the following requirements:

- specify the object in an interface
- mention for each method in the interface the PRE- and POST-condition
- specify the object by describing its properties using
  - Elements (which elements make up the object?)
  - Structure (relations between elements)
  - Domain (which values are allowed?)

Use Javadoc (a how-to has been posted on Blackboard 'Use of Javadoc') for specifying the Elements, Structure and Domain of each ADT and specifying the PRE-and POST-condition of each method. For methods that throw an Exception, "@throws Exception" should be used to describe in which case an Exception is thrown. The case when no Exception is thrown, should be described in the POST-condition.

Once you completed the program and thoroughly tested it, all the program's Java files can be uploaded and combined into one .jar-file or a zip-file to be handed in via Practool.