*November 24, 2020*

# Task 7

Create a generic class **XList**, objects of which are lists, i.e., it directly or indirectly implements the **List** interface. The class provides some additional functionality allowing for creating lists and operating on them.
The class should define constructors

- `public XList(Collection<T> coll)`
  which ensures that the list being created contains the same elements as the collection coll;
- `public XList(T... args)`
  which ensures that the list being created contains all elements passed as comma-separated arguments or an array.

Create also static factory methods **of** of a similar functionality

- `public static <T> XList<T> of(Collection<T> coll)`
- `public static <T> XList<T> of(T... args)`

and, for the special case of lists of **String**s

- `public static XList<String> charsOf(String str)`
  which returns a list of **String**s containing individual characters from str;
- `public static XList<String> tokensOf(String str, String sep)`
  which returns a list of lexemes ("words") from str, where sep is a regular expression defining the separator between lexemes;
- `public static XList<String> tokensOf(String str)`
  which returns a list of lexemes ("words") from str separated by any non-empty sequences of white characters.

Also, define the following non-static methods:

- `public XList<T> union(Collection<? extends T> coll)`
  which returns a new **XList** containing all elements of *this* list and all elements of coll;
- `public XList<T> union(T... arr)`
  as the previous one, but with a different type of arguments;
- `public XList<T> diff(Collection<? extends T> col)`
  which returns a new **XList** containing elements of *this* list, but only those which do *not* occur in coll;
- `public XList<T> diff(T... arr)`
  as the previous one, but with a different type of arguments;
- `public XList<T> unique()`
  which returns a new **XList** with elements of *this* list, but without repetitions (the order of elements should be preserved);

- `public XList<XList<T>> combine()`
  which can be invoked on a list elements of which are themselves lists and returns also a list of lists. Each of the resulting lists contains exactly one element from each input list in all possible combinations. Example: let *this* list contains, as its elements, three lists, so it has a form
  ```
        [ ["a", "b"], ["x"], ["1", "2"] ].
  ```
  Then the resulting list of lists would be
  ```
        [ ["a", "x",  "1"], ["a", "x",  "2"],
          ["b", "x",  "1"], ["b", "x",  "2"] ]
  ```
- `public <R> XList<R> collect(Function<? super T, ? extends R> fun)`
  which returns a new list, elements of which are the results of applying the **fun** function on elements of *this* list;
- `public String join(String sep)`
  returning a string which is the concatenation of the strings representing the elements of *this* list, separated by the separator sep;
- `public String join()`
  as before, but without separators;
- `public void forEachWithIndex(BiConsumer<? super T, Integer> cons)`
  which iterates over *this* list and applies the consumer cons (its **accept** function) to all pairs (element, i), where element is an element of the list and i is its index.

The following program

download *SXList.java*

```
    // imports

    public class XList {
        public static void main(String[] args) {
              // some data to use later
            Integer[] ints = {88, 99};
            Set<Integer> set =
                        new HashSet<>(Arrays.asList(4, 7));

            System.out.println("*** Creating XLists");
            XList<Integer> list1 = new XList<>(1, 3, 9);
            XList<Integer> list2 = XList.of(5, 6);
            XList<Integer> list3 = new XList<>(ints);
            XList<Integer> list4 = XList.of(ints);
            XList<Integer> list5 = new XList<>(set);
            XList<Integer> list6 = XList.of(set);
            System.out.println(list1);
            System.out.println(list2);
            System.out.println(list3);
            System.out.println(list4);
            System.out.println(list5);
            System.out.println(list6);
```

```java
System.out.println("*** Special funcs for Strings");
XList<String> slist1 = XList.charsOf("ab cd efg");
XList<String> slist2 = XList.tokensOf("ab cd efg");
XList<String> slist3 = XList.tokensOf("A-B-C", "-");
System.out.println(slist1);
System.out.println(slist2);
System.out.println(slist3);

System.out.println("*** Union");
List<Integer> m1 = list1.union(list2);
System.out.println(m1);
m1.add(11);
System.out.println(m1);
XList<Integer> m2 = (XList<Integer>) m1;
XList<Integer> m3 =
        m2.union(ints).union(XList.of(4, 4));
System.out.println(m2);
System.out.println(m3);
m3 = m3.union(set);
System.out.println(m3);

System.out.println("*** Diff");
System.out.println(m3.diff(set));
System.out.println(XList.of(set).diff(m3));

System.out.println("*** Unique");
XList<Integer> uniq = m3.unique();
System.out.println(uniq);

System.out.println("*** Combinations");
List<String> sa = Arrays.asList("a", "b");
List<String> sb = Arrays.asList("X");
XList<String> sc = XList.charsOf("12");
XList toCombine = XList.of(sa, sb, sc);
XList<XList<String>> cres = toCombine.combine();
System.out.println(cres);

System.out.println("*** Collect and join");
XList<String> j1 = cres.collect(li -> li.join());
System.out.println(j1.join(" "));
XList<String> j2 =cres.collect(li -> li.join("-"));
System.out.println(j2.join(" "));

System.out.println("*** ForEachWithIndex");
XList<Integer> lmod =
                    XList.of(1,2,8, 10, 11, 30, 3, 4);
            lmod.forEachWithIndex( (e, i) -> lmod.set(i, e*2));
            System.out.println(lmod);
            lmod.forEachWithIndex( (e, i) -> {
                if (i % 2 == 0) lmod.remove(e);
            });
            System.out.println(lmod);
            lmod.forEachWithIndex( (e, i) -> {
                if (i % 2 == 0) lmod.remove(i);
            });
            System.out.println(lmod);
        }
    }
```

should print something like

```
*** Creating XLists
[1, 3, 9]
[5, 6]
[88, 99]
[88, 99]
[4, 7]
[4, 7]
*** Special funcs for Strings
[a, b,  , c, d,  , e, f, g]
[ab, cd, efg]
[A, B, C]
*** Union
[1, 3, 9, 5, 6]
[1, 3, 9, 5, 6, 11]
[1, 3, 9, 5, 6, 11]
[1, 3, 9, 5, 6, 11, 88, 99, 4, 4]
[1, 3, 9, 5, 6, 11, 88, 99, 4, 4, 4, 7]
*** Diff
[1, 3, 9, 5, 6, 11, 88, 99]
[]
*** Unique
[1, 3, 9, 5, 6, 11, 88, 99, 4, 7]
*** Combinations
[[a, X, 1], [a, X, 2], [b, X, 1], [b, X, 2]]
*** Collect and join
aX1 aX2 bX1 bX2
a-X-1 a-X-2 b-X-1 b-X-2
*** ForEachWithIndex
[2, 4, 16, 20, 22, 60, 6, 8]
[4, 16, 22, 60, 8]
[16, 22, 60, 8]
```

*Deadline: Dec 11 (inclusive)*