Exercises module 16 – Generic lists

16.1 – List of integers

1. Create a list of integers like this:

```
List<int> nums = new ArrayList<>();
```

   Does it work?

2. By design Java collections can’t store primitive types in a collection, read more about it here: http://stackoverflow.com/questions/2504959

3. Fix the code above to make it work and add 10 random numbers to the array.

4. Now create a new class named **IntegerHelper** that contains a method named **maxValue** that takes a List of integers as input and returns the integer with the highest value in the provided list.

5. Then create a second method in the helper class called **evenNumbers** that will take a list of integers as input and return a list that contains all the even numbers found in the provided list.

   Verify that it works.

16.2 – Set operations

1. A neat way to create a list and initialize it with values at the same time is to write:

```
List<Integer> listA = Arrays.asList(1,2,3,4,5,6);
List<Integer> listB = Arrays.asList(5,6,7,8,9,10,11,12,13);
```

2. We need to write a **helper class** that will contain various common set operations on lists of integers.

   Including

   - Union
     Returns a new list that contains the elements from both lists, excluding duplicates.
   - Intersect
     Returns the intersection of the two lists that only contains those element that exists in both lists.
   - Except
     Returns a list of the items in the first list that does not exist in the second lists.

   Sample usage and result:

```
ListHelper helper = new ListHelper();

List<Integer> union = helper.union(listA, listB);         //1,2,3,4,5,6,7,8,9
List<Integer> intersect = helper.intersect(listA, listB); //4,5,6
List<Integer> except = helper.except(listA, listB);       //1,2,3
```

3. Now your job is to implement the three methods above using only the **List/ArrayList** collection types.

   The operations we have implemented above is called Set operations working on sets of data.

16.3 – Lists of lists

1. Creating lists of lists can be a useful data-structure for certain scenarios.

   The following code creates a List of List data structure and your job is to implement the **printAllNames** method below so that it prints out all the names in the data structure.

   The code:

```
ArrayList<ArrayList<String>> names = new ArrayList<ArrayList<String>>();

ArrayList<String> namesStartingWithA = new ArrayList<String>();
namesStartingWithA.add("Anders");
namesStartingWithA.add("Andreas");
namesStartingWithA.add("Anna");

ArrayList<String> namesStartingWithB = new ArrayList<String>();
namesStartingWithB.add("Berit");
namesStartingWithB.add("Bertil");
namesStartingWithB.add("Bo");

ArrayList<String> namesStartingWithC = new ArrayList<String>();
namesStartingWithC.add("Cecilia");
namesStartingWithC.add("Carter");
namesStartingWithC.add("Carl");

names.add(namesStartingWithA);
names.add(namesStartingWithB);
names.add(namesStartingWithC);

printAllNames(names);
```

   The output should be:

```
Anders
Andreas
Anna
Berit
Bertil
Bo
Cecilia
Carter
Carl
```

Questions and concepts to study further on your own:

- Java boxing and unboxing
  http://docs.oracle.com/javase/1.5.0/docs/guide/language/autoboxing.html
- Why do we use autoboxing and unboxing in Java?
  https://stackoverflow.com/questions/27647407
- List vs ArrayList
- java primitive types vs reference types
- 18 Java ArrayList Programming Examples
  http://javaconceptoftheday.com/java-arraylist-programming-examples/
- Set operations
  https://en.wikipedia.org/wiki/Set_(mathematics)