Part I

In the first part of this homework you are asked to specify and implement a generic version of two sorting algorithms:

- selection sort
- bubble sort

In the generic version both the list of entries is generic, and the order is a parameter to the sort. The implementation to write for the bubble sort is the most efficient one discussed in class, where it checks after a pass if a swap took place to continue making more passes to the list.

Part II

In this part of the homework you will test one of the sorting algorithms; you choose the sort to test. Cases to consider for the testing:

- an empty list
- a list with one element
- a list with two elements; two cases: a. list is in order, b. list is out of order.
- a list with many elements: (several cases here)
  - a. list already in order (that is already sorted)
  - b. list in the opposite order (that is, sorted in the reversed order.)
  - c. list almost ordered (that is, is almost ordered but a few elements are out of order).
  - d. list entries are in random order. Test several lists of this kind for several different sizes.

Many entries means that size is >= 20 entries and you could choose it to be in the thousands or millions.

For all these cases, initially use as object Integer and give it int numbers which will be boxed to Integer instances; and use as Order ‘<=’. Therefore in all these tests your lists is sorted in increasing order.

After testing all these cases, take two cases of type d. and sort the list where the order is “>={”, that is the resulting sorted list will be in decreasing order.

To create a list of a whole bunch of Integer numbers use the class java.util.Random (read its API); and to get a random integer use the method nextInt().

Example:

a. The statement below creates a Random instance:

```java
Random gen = new Random();
```

b. the statement below fills list (of type List<Integer>) with random Integer instances, where MAX_SIZE is a constant value defined in the test source:

```java
int i = 0;
while (i < MAX_SIZE){
    list.add(gen.nextInt());
    i = i + 1;
}
```
For the testing, after invoking sort on a list you must check that the list is “in order” with respect to the order given to sort. Thus, you must write a utility *generic* method to check whether a list is in order with respect to a given *Order* instance. After you invoke the sort, use an *assertTrue* statement on an invocation of that method to verify that the list is indeed ordered.

**Part III**

In this part you will test sort using an object with several properties; for this case use a mock object with 3 properties.

```java
class MockSortObject {
    private int id;
    private int hwk;
    private int test;
}
```

For hwk and test use *Random’s nextInt (int n)* to generate numbers in the range: 0.. n-1, with n = 101; for id use *nextInt* but make sure to make it positive by using *Math.abs*.

Make a list of many entries of type *MockSortObject*; sort the list for the following orders:

a. Increasing by id;

b. decreasing by hwk.

c. increasing by test.

and for each case proceed to test that the list is ordered.

**What to submit**

a. source code for the two sort methods.

b. source code for the tests.

c. Report of success or failure on this homework.