Instructions:

1. Create a new Java project and import the skeleton code
2. Complete the undone implementation according to the requirements.
3. Submit your code through the link in Canvas: https://canvas.ust.hk/courses/42336/assignments/207441 You are allowed to submit multiple times within the examination duration. The final score is given based on your last submission.
4. You must not modify the completed methods given in the skeleton code (otherwise we may fail to grade your work).

Questions:

You are given a class `SortedList<T>`. During the construction of a sorted list:

```
public SortedList(ArrayList<T> list) {
    this.list = list;
    filter(list);
    bubbleSort(list);
}
```

It takes an `ArrayList<T>`, first applies the `filter` method to only keep some of the elements in `list`.

Then it applies `bubbleSort` to sort the list according to the order defined by the generic parameter `T`.

Task 1:

Implement the `filter` method:

```
void filter(ArrayList<T> list)
```

Filtering criteria:

> This method should only keep the elements at index 1, 4, 7, 10, ..., 3*k+1, ... from `list` (k starting from zero) and discard elements at other positions.

Hint:

Create a new list to contain those needed elements. Replace the old list with the new list.

Task 2:

Implement the `bubbleSort` method:

```
void bubbleSort(ArrayList<T> list)
```

Requirement:

This method will sort `list` according to the order of `T`. You **must not** use the sort function from the Java library. Instead, you are required to implement the **bubble sort** algorithm.

Hint:

Here is how you can do bubble sort for an integer array:

```
void bubbleSort(int arr[])
{
    int n = arr.length;
    for (int i = 0; i < n-1; i++)
        for (int j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
}
```

Task 3:

You are given a class `Student`. Your task is to modify it so that we can put it into a `SortedList` as follows:

```
ArrayList<Student> unordered = new ArrayList<Student> (...);
SortedList<Student> sorted = new SortedList<Student>(unordered);
```

Background on lexicographical order:

The lexicographical order of Student objects concerning fields (age, score, name) is defined as follows:

> When two students have different age, their relative order is defined by the order of their ages.
>
> When two students have the same age but different scores, their relative order is defined by the order of their scores.
>
> When two students have the same age and score, their relative order is defined by the order of their names.

In this task, you are required to define the order on Student as the lexicographical order concerning fields (age, score, name).

Hint:

Implement the `Comparable` interface for `Student`. Invoke the `compareTo` methods for the data members one by one to find the first data member that differs.

Test:

Run `Test.Main`, the expected output is

```
Test student passed!
```

To be able to run the test, the code in `Test.java` reads data from files. Thus, you need to maintain the following directory structure of your project:

```
src/
students.txt
results_xx.txt
```