CS-420-1: Object-Oriented Design
Fall 2019
Northeastern Illinois University
Homework #4: Due Tuesday, 09/24/19 at 5:00 p.m.
Collections

- Download the files provided for you from the course website and unzip them into a folder.
- Similar to Homework #1, import the `homework4.project`.
- Other than to uncomment the tests, do not modify the code in `homework4.project` → `src` → `test` → `java`.
- Make sure that your test runner is set to Gradle.

Problem 1:

1. In `homework4.project` → `src` → `main` → `java`, modify the `LabeledPoint` class to do the following:
   - It should implement the `Comparable` interface.
   - The `compareTo` method should compare first by x-values. If the x-values are the same, then compare by y-values, if the y-values are the same, then compare by the label values.
2. Uncomment the first 4 tests in the `LabeledPointTest` file and run them. If they all pass, then the method has been created correctly.
3. Create a static method in the `LabeledPoint` class named `treeSetPoints` that does not take any parameters and returns a `TreeSet` of the following `LabeledPoints` (added in this order).

```
(2, 0), "magic"
(-2, 0), "blah"
(-2, 0), "bar";
(-2, -2), "map"
```

Problem 2:

1. In `homework4.project` → `src` → `main` → `java`, create a java class named `Task`.
2. The `Task` class should be very simple with two properly encapsulated instance variables named `priority` (`int`) and `description` (`String`). Create a constructor and getters for the instance variables.
3. Create a class named `TaskComparator` that implements the `Comparator` interface to compare two `Task` objects.
4. The `compare` method should return the difference between the first parameter's priority and the second parameter's priority.

5. Create a `TaskDemo` class with the `main` method. You should prompt the user to find out whether they are ready to enter tasks. If they are, continue to prompt them to enter a task (description and priority) until they enter `n`. Do not worry about error handling (incorrect usage of y/n, bad input, etc).

6. After all the tasks are all entered, print out the tasks in order of highest priority (1) to lowest priority.

7. **IMPORTANT:** To run your code using the console (System.in), you will need to change the default build runner. Under preferences/system, change "Build and run using" to IntelliJ.

8. Things to consider: Use the correct `Collection` structure to store the tasks! Create helper methods so that you don't have a huge amount of code inside the `main` method.

9. Format your output to match the output below.

```
Ready to enter tasks? (y/n): y
Enter task description: Do laundry
Enter task priority: 7
Another task? (y/n): y
Enter task description: Do dishes
Enter task priority: 3
Another task? (y/n): y
Enter task description: Take out garbage
Enter task priority: 1
Another task? (y/n): y
Enter task description: Walk dogs
Enter task priority: 2
Another task? (y/n): n

Your tasks are:
1, Take out garbage
2, Walk dogs
3, Do dishes
7, Do laundry
```

Problem 2:

1. In `homework4.project` → `src` → `main` → `java`, design an object-oriented approach (constructors, instance variables, etc - multiple classes if necessary) to solve the following problem:

2. Read all the words from the `readFict.txt` file provided in `homework4.project` → `src` → `main` → `resources`. Note that all the words in this file are separated by spaces and are lowercase with no punctuation.

3. Add the words to a map whose keys are the phone keypad spellings of the word (for example, the letters g, h, and i correspond to the number 4, and whose values are sets of words with the same code. For example, `26337` is mapped to the set { `"Andes", "coder", "codes", . . .`}.

4. Then keep prompting the user for numbers and print out all words in the dictionary that can be "spelled" with that number.

5. The path to the file should **not** be a hard-coded path (as this means that I would most likely have to change it to run on my computer). It should be a relative path (or similar to HW2).

6. Suggestions: For reading from the file, you may want to consider using the `FileReader` class wrapped in a `BufferedReader` (now that you understand binary I/O, text I/O options are very similar). Notice the similarity between BufferedInputStream and BufferedReader and a FileInputStream and FileReader.

7. Reminder: Similarly to Problem 2, to run your code using the console (System.in), you will need to change the default build runner. Under preferences/system, change "Build and run using" to IntelliJ.

Submitting your homework to D2L

- In IntelliJ, under `File`, choose `"Export to Zip File"`.
  Use the default name of `homework4.project.zip`.
- Submit this file to D2L.