Comparable Interface - Code Reading and Exploration

Due date: by the end of the recitation session

Introduction

This activity sends you to a real life application to explore uses of the interface that you should have seen previously, namely the `Comparable<E>` interface in Java.

Many open source projects host their source code in publicly accessible repositories. GitHub is an example of such a repository. In this activity you will browse through the source code of a particular project hosted on GitHub. You do not need a GitHub account to complete this activity.
The project that we will be looking at is OpenMRS, http://openmrs.org/, and its GitHub repository is at https://github.com/openmrs/openmrs-core

Worksheet: `https://goo.gl/GVlIhK`

Part 1 Familiarize yourself with OpenMRS

Visit the websites (not the GitHub repository) of the OpenMRS project and try to learn what
the goal of the project is.
Go to the GitHub repository for OpenMRS (use the above link to the OpenMRS-core) and explore the information available on the homepage for the project.
If you are unfamiliar with GitHub organization, you may need to spend a bit of time reading through that page. If any member of your group has used/worked with GitHub before, they should explain to all other members how the page is organized and what various things mean.

Answer the questions in Part 1 on the worksheet (they are here just as a reference for all group members):

1. What is the purpose/goal of this project?
2. How many programmers contributed code to the project?
3. When was the most recent modification to the codebase?
4. What is the username of the person who submitted the last change?

Part 2 Explore the code

Start with OpenMRS and search for the keyword "compareTo" (the case does not matter, but you are looking for examples that match this spelling). Locate four different examples of a call to the `compareTo` method. For each of them identify the type of the object that the method is called on. Explore (i.e., find the files and look at them) the four classes that you identified.

Answer the questions in Part 2 on the worksheet (they are here just as a reference for all group members):

1. List the four different calls to the `compareTo` method. For each call specify the file name and line number where it occurs and specify the type of object on which it is called (you may need to open the actual source code file to figure out the type of the object).
2. For two of the classes that you named in the above table, find the following information.
    a. What is the header for that class?
    b. What is the type of the parameter to the `compareTo` method? How many parameters are there?
    c. What is the return type of the `compareTo` method?

Part 3 Familiar `compareTo` in the `String` class

Recall that you used the `compareTo` method with the `String` objects.

Here are the code fragments that show the header for the `String` class and the source code for the `compareTo` method:

```
public final class String
   implements java.io.Serializable, Comparable<String>, CharSequence

-------------

public int compareTo(String anotherString) {

  int len1 = value.length;
  int len2 = anotherString.value.length;
  int lim = Math.min(len1, len2);
  char v1[] = value;
  char v2[] = anotherString.value;
  int k = 0;
  while (k < lim) {
      char c1 = v1[k];
      char c2 = v2[k];
      if (c1 != c2) {
        return c1 - c2;
      }
      k++;
  }
  return len1 - len2;
}
```

Read through this function and figure out what it is doing (try to figure out **exactly** what each line does and why).

Answer the questions in Part 3 on the worksheet (they are here just as a reference for all group members):

1. What is the type of the parameter and the return type for the `compareTo` method of the `String` class.
2. Given the following lines of code determine the return value of the call to `compareTo` method, i.e., what value is stored in the variable `result`? (Do not just run that code! Trace the `compareTo` method given in the instructions.
   ```
   String str1 = "hello";
   String str2 = "help";
   int result = str1.compareTo(str2);
   ```
3. How many iterations of the `while` loop in the `compareTo` method are executed when the above call is made?

Part 4 `Comparable<E>` interface

Looking at all the examples above you learned a few things about the `Comparable<E>` interface.

Answer the questions in Part 4 on the worksheet (they are here just as a reference for all group members):

1. What do the headers of three classes (`String` class and two classes you picked from OpenMRS) have in common?
2. What can you say about the parameter type of the `compareTo` method?

3. What can you say about the return type of the `compareTo` method?

Part 5 Reading the API for `Comparable<E>` interface

The Java API documentation for `Comparable<E>` interface can be found at https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html
Read through that documentation page. It may contain some material that is hard to follow at this point, but you should be getting used to reading the documentation even if you do not understand every detail.

Consider this alternative definition of the `compareTo` method for the `String` class (the modified lines are marked):

```
public int compareTo(String anotherString) {

  int len1 = value.length;
  int len2 = anotherString.value.length;
  int lim = Math.min(len1, len2);
  char v1[] = value;
  char v2[] = anotherString.value;
  int k = 0;
  while (k < lim) {
      char c1 = v1[k];
      char c2 = v2[k];
      if (c1 != c2) {
        return c2 - c1;           // <--- modified
      }
      k++;
  }
  return len2 - len1;             // <--- modified
}
```

Answer the questions in Part 5 on the worksheet (they are here just as a reference for all group members):

1. The first sentence of the description states that "This interface imposes a total ordering on the objects of each class that implements it." What do you think is meant by "total ordering"?
2. The first sentence of the second paragraph states that "Lists (and arrays) of objects that implement this interface can be sorted automatically by `Collections.sort` (and `Arrays.sort`)." What do you think is meant by "automatically"?
3. Locate the *Method Detail* for the `compareTo` method. Describe in your own words what is the meaning of the value returned by this method?
4. If a call is made to `Arrays.sort` giving it as an argument an array of `String` objects, what do you think will happen to an array of strings?
5. Assume that the `compareTo` method in the `String` class is modified as shown in the instructions (the modified version). What do you think the result of calling `Arrays.sort` on an array of strings will be now?