| OOP and Data Structures | Discussion questions |
|---|---|
| CS2110 Summer 2019 | Java essentials |

Before you get started, you should browse the notes on the Java String API in the Java HyperText.

The purpose of this problem set is for you to get practice. As long as this is a good attempt, with most things right, you will get 100%. Solve these problems during discussion!!! Do it with 1 or 2 neighbors, working together. Go ahead! Discuss things with those around you. If you have a question, ask the TA or the people around you. It's best if you upload your solution at the end of discussion, but you have until 5PM Wednesday.

Here are some things to keep in mind while working on these problems:

- Remember to commit your work (Team → Commit) after each meaningful chunk of work. While committing, review the list of changed files (you can double click on them to see the changes), and give a meaningful description of what has changed.
- Try to use the methods of class `String` as much as possible: avoid loops, recursion, etc. You can find the `String` documentation by googling Java 11 String; clicking "Method" next to "Summary" in the top left will jump to the list of available methods.
- Here are some helpful `String` methods to look at:
  - `s.length()`, `s.charAt(int)`, `s.substring(int)`, `s.substring(int,int)`, `s.trim()`,
  - `s.startsWith(char)`, `s.endsWith(char)`, `s.compareTo(String)`, `s.indexOf(char)`, `s.lastIndexOf(char)`

1. Create a project for this recitation, and a class called `Discuss1`. Be sure to set up source control (Team → Share) and create an initial commit (Team → Commit).
2. Create a method `public static boolean containsVowel(String s)` in class `Discuss1`. This method should return `true` if string `s` contains a vowel (a, e, i, o, u) and `false` otherwise. Write a reasonable javadoc comment for the method.
3. Create a `public static void main(String[] args)` method in class `Discuss1`. This allows the class to be run. Make it call `containsVowel` on the String `"Hello world!"` to convince yourself that it works properly. Be sure to add a brief javadoc comment to document your `main` method.
4. Add a method `public static String dateToString(int d, int m, int y)`, where `m` is a month (in 1...12), `d` is a day within the month, and `y` is a year. This method returns the long form of the date. For example, `dateToString(28,8,2018)` should return `"28 August 2000"`. Assume the arguments of a call form a valid date (this type of assumption is called a *precondition* of the method).

   Don't forget to add a Javadoc comment. Use `@param` tags in the Javadoc to document the preconditions (Eclipse should insert these for you). For example, you might write:

   ```
   1 /**
   2  * ...
   3  * @param m an integer in the range 1..12, representing the month
   4  * ...
   5  */
   ```

5. Modify your `main` method to convince yourself that your method `dateToString` works.
6. Add a method `public static int countEs(String s)` that returns the number of 'e's in `s`. Document and test `countEs`.
7. View your commit history by right clicking on your project and selecting Team → Show in History. See if your list of commits makes sense.

8. You can view the Javadoc that you created for your class by selecting Project → Generate Javadoc in the top menu. Keep the default options, then open `doc/index.html`. Verify that your documentation would be useful to someone without access to the code.

9. If you worked with others, form a group on CMS (one student should invite the others; the others need to accept). Submit the file `Discuss1.java`