ICT 4361 — Java Programming Exercise 5

Purpose:

This exercise will familiarize you with using polymorphic containers and text processing in Java.

Our approach will be based on a simple Mad-Libs exercise, based on an exercise for the MIT introductory course to Java (MIT OpenCourseWare Java Preparation Course). We will take this exercise, and then add a file-parsing front-end on it for the next homework exercise.

There are many ways to extend this exercise.

You can check out 

Madlibs.org to see how a web front end is added to a similar capability.

What To Hand In:

Please hand in a listing for each program requested, formatted in an easy-to-read style.

Ensure your name, and the name of the file is available in a comment at the top of the file.

Also, ensure that you have a sample of the output from the program.

If your program fails to compile, hand in your error listing as your output.

For each question asked, provide one or two sentences summarizing your answer. Please be both complete and succinct.

Problems:

1. Start with the homework starter files. These will give you a good framework for creating your Madlibs. Describe why they will not compile as they are.
2. Examine the `MadLibEntry` abstract class, and create the two derived classes `TextEntry` and `Slot`. Be sure to implement all the abstract methods.
3. Add fields and methods to the `MadLib` class. These should include a container for `MadLibEntries`, appropriate constructors, and the methods outlined in the class starter.
4. Compile and run the `MadHello` program. When you have implemented the above, it should compile and run
5. Compile and run the `MadHamlet` program.
6. *Optional:* Run the JUnit tests and make them all pass

Notes:

- You can put the sample classes into your NetBeans environment by putting all the files except `HW5TestSuite.java` and `MadLibTest.java` into the source directory, and these

remaining 2 files into the test directory. You can also simply use `ant` with the provided `build.xml` file.

- While you could create the `TextEntry` and `Slot` at the bottom of the `MadLibEntry.java`, please instead create them as separate, public classes.
- Note that while doing input and output in this framework, you should always use the `ui` instance passed in as a parameter, rather that System.out or System.in. This will ensure that all your input and output happens properly.
- Note that the `MadLib` class must store an ordered list of `String`s. This is best handled by using one of the `Collections` classes in the `java.util` package. Various methods in your `MadLib` class will iterate through the collection. For example, a `List` of `MadLibEntry` would be a reasonable collection, instantiated by, say, a `LinkedList` of `MadLibEntry`.
- If your ordered list stores `MadLibEntry` instances, it will be able to store both `TextEntry` objects ***and*** `Slot` objects.
- Your `addString` method can create a new `TextEntry` and add it to your list
- Your `addSlot` method can create a new `Slot` and add it to your list
- Your `doLib` method should first go through each item on your list, and invoke its `doLib` method Afterward, it should build a result string by appending the result of the `madLibString` method for each item on your list Finally, it should write the resulting string to the output.
- `MadHello` is the simplest test program using this set of classes, so try running it first.
- For the optional part, note that the format of the prompts is very important. You must prompt with the string `Please enter a` *`type`*`:` . That is, the words "Please enter a", a space, the kind of word, a colon, and another space. You must also be sure not to add extra spaces when printing the templates or MadLib, and finally when printing out the template, you must print out `Slot`s as `<type>`. That is, a less-than, the name of the slot, and then a greater-than.
- For fun, you may note that a `JOptionPaneUI` implementation of the `UserInterface` is provided. Simply replacing the `StdStreamUI` with an instance of `JOptionPaneUI` will make another form of GUI.

Evaluation Criteria:

Criteria	Weight
Answer for problem 1	10
MadLibEntry class	15
TextEntry derived class and test	10
SlotEntry derived class and test	15
MadLib class completion	20
MadHello program and output	15
MadHamlet program and output	15