Lab Exercise 8
CS 2334

October 15, 2015

Introduction

In this lab, you will experiment with using HashMaps and Enumerated data types in Java. You will implement a class of enums that has a custom class as the value. In the Driver class, you will create a HashMap that maps from a String to the enumerated data type. In addition, your implementation will iterate over the HashMap.

Learning Objectives

By the end of this laboratory exercise, you should be able to:

1. Create an enumerated data type
2. Create and add items to a HashMap
3. Iterate over a HashMap in order to print out information

Proper Academic Conduct

This lab is to be done individually. Do not look at or discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.

Preparation

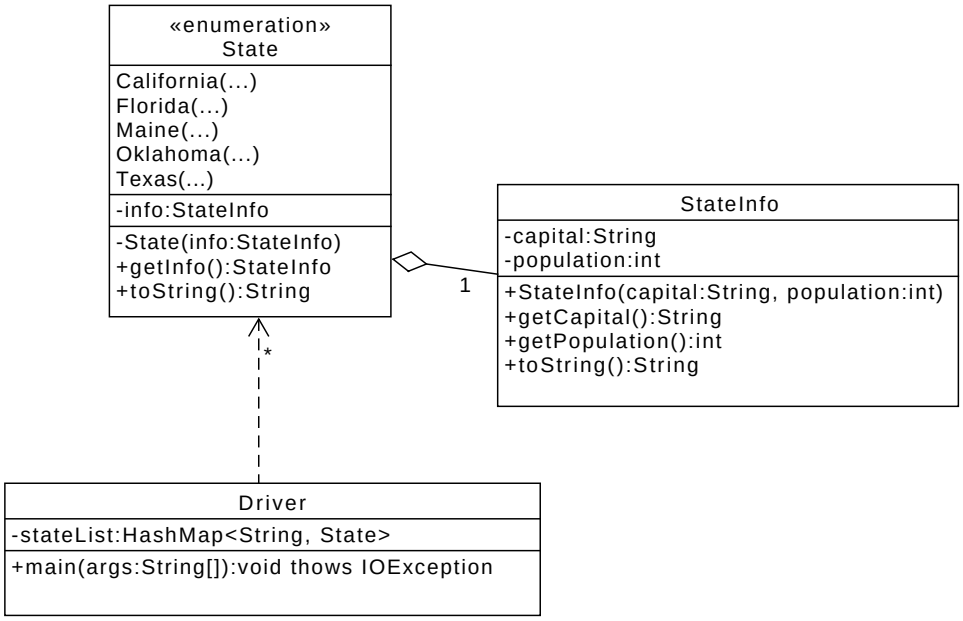
1. Create a new project in your eclipse workspace.

   (a) In Eclipse, select *File/New/Java Project*

   (b) Name the project *lab8* and use the default package

Representing Different States

Below is the UML representation of the lab. Your task will be to implement this set of classes and an associated set of JUnit test procedures.



We will only be using the following states for this lab: *California, Florida, Maine, Oklahoma, and Texas*. The keys used with the HashMap are the two letter abbreviations for the states: *CA, FL, ME, OK, and TX respectively*. Below is the information we will be using for testing:

- California: capital - Sacramento, population - 38,802,500
- Florida: capital - Tallahassee, population - 19,893,297

- Maine: capital - Augusta, population - 1,330,089
- Oklahoma: capital - Oklahoma City, population - 3,878,051
- Texas: capital - Austin, population - 26,956,958

Be sure you spell the capitals correctly and use the correct populations.

Lab 8: Specific Instructions

1. Create a new Java class for both the enum class State and the StateInfo class described in the UML diagram
    - Be sure that the class name is exactly as shown
    - You must use the default package, meaning that the package field must be left blank
2. Implement the attributes and methods for each class
    - Use the same spelling for instance variables and method names as shown in the UML
    - Do not add functionality to the classes beyond what has been specified
    - Don't forget to document as you go!
3. Create *Lab8Test* class and use JUnit tests to thoroughly test all of your code
    - You need to convince yourself that everything is working properly
    - Make sure that you cover all the classes and methods while creating your test. Keep in mind that we have our own tests that we will use for grading.

Driver Class

The Driver class will create and populate a HashMap with Strings as keys and State objects as values. It will also present the user with an option to select a single state or all of the states in the HashMap about which to print information. If the user chooses to select a specific state, the list of states in the HashMap is presented to the user.

- Main menu:

  Please choose an option:
  1: Choose a state
  2: All states

- State menu

  Please choose from the following states: [TX, FL, ME, OK, CA]

  Note: Sets are unordered, so the list of states could be in any order. The formatting occurs when a Set is printed.

Use a *BufferedReader* to take in the input. Your code will need to be able to handle any input that the user could choose, e.g. numbers other than 1 and 2, letters, states not listed, etc. If incorrect input is given, then your program must reprompt until a correct input is given.

Once all of the necessary information is obtained from the user, your program must print the report on the state (or states) and exit.

Example Interactions

```
Please choose an option:
1: Choose a state
2: All states
2
ME - Maine, capital: Augusta, population: 1330089
OK - Oklahoma, capital: Oklahoma City, population: 3878051
TX - Texas, capital: Austin, population: 26956958
CA - California, capital: Sacramento, population: 38802500
FL - Florida, capital: Tallahassee, population: 19893297
```

```
Please choose an option:
1: Choose a state
2: All states
adfa
Please enter an integer.
1
Please choose from the following state: [ME, OK, TX, CA, FL]
aaf
Please choose one of the states listed.
ca
CA - California, capital: Sacramento, population: 38802500
```

Final Steps

1. Generate Javadoc using Eclipse.
   - Select *Project/Generate Javadoc...*
   - Make sure that your project is selected, as are the classes: Driver, State, StateInfo, and Lab8Test (check for these individually!)
   - Select *Private* visibility
   - Use the default destination folder
   - Click *Finish*
2. Open the *lab8/doc/index.html* file using your favorite web browser or Eclipse (double clicking in the package explorer will open the web page). Check to make sure that that all of your classes are listed and that all of your documented methods have the necessary documentation.
3. If you complete the above instructions during lab, you may have your implementation checked by one of the TAs.

Submission Instructions

- All required components (source code and compiled documentation) are due at 11:59pm on Friday, October 16th.
- Prepare your submission file:
  1. Select the project in the *Package Explorer* window.
  2. Right-click. Select *Export*

3. Select *General/Archive File*
4. Expand your project and verify that both the *src* and *doc* folders are selected, as well as all of their contents
5. Enter the archive file name: *lab6.zip* (note that you may want to browse to a different destination folder)
6. Select *Save in zip format*
7. Click *Finish*

- Submit your zip file to the lab8 folder on D2L.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 5 points

(5) The program is properly formatted (including indentation, curly brace and semicolon locations).

(3) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 10 points

(10) The program is using proper data types and method calls.

(7) There is one error in data type or method call selection.

(4) There are two errors in data type or method call selection.

(0) There are three or more errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented.

(7) A component of one required method is missing.

(4) A method is not implemented or components are missing from two methods.

(0) Two or more required methods are not implemented or components from three or more methods are missing.

Unit Tests: 10 points

(10) A full set of unit tests has been implemented.

(8) One key component is not tested properly by the unit tests.

(6) Two key components are not tested properly by the unit tests.

(4) Three key components are not tested properly by the unit tests.

(2) Four key components are not tested properly by the unit tests.

(0) Five or more required methods are not implemented or components from three or more methods are missing.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.

(10) The program fails one test.

(5) The program fails two tests.

(0) The program fails three or more tests.

Execution: 15 points

(15) The program executes with no errors.

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements.

(0) The method documentation is missing more than two of the required documentation elements.

Inline Documentation: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used and has the correct contents.

(5) The correct zip file name is used, but one required component is missing.

(0) An incorrect zip file name is used or more than one required component is missing.