COGS 125 / CSE 175
Introduction to Artificial Intelligence
Specification for Laboratory Exercise #0

David C. Noelle

Due: 11:59 P.M. on Friday, September 20, 2019

Overview

This initial laboratory exercise has two primary learning goals. First, the exercise will provide you with an opportunity to refresh your knowledge of Java™ program generation in the Eclipse integrated development environment (IDE) that will be used throughout this course. In this way, this exercise will provide practice in some of the fundamental skills that will be needed to successfully complete future assignments. Second, this exercise will provide you with experience with some specific Java™ classes for maintaining sets and lists that will be useful in upcoming programming assignments.

The programming exercise involves implementing a new class that records a collection of named locations on a map and calculates the minimum area rectagle that surrounds all of the given locations.

Submission for Evaluation

To complete this assignment, you must generate one Java™ class files: `MapBox.java`. This is the only file that you should submit for evaluation.

To submit your completed assignment for evaluation, log onto the class site on CatCourses and navigate to the "Assignments" section. Then, locate "Exercise #0" and select the option to submit your solution for this assignment. Provide your program source file as an attachment. Do not upload any other files as part of this submission. Comments to the teaching team should appear as header comments in your Java™ source code files.

Submissions must arrive by 11:59 P.M. on Friday, September 20th. Please allow time for potential system slowness immediately prior to this deadline. You may submit exercise solutions multiple times, and only the most recently submitted version will be evaluated. As discussed in the course syllabus, late exercises will not be evaluated.

Activities

You are to provide a Java™ class that implements a collection of named locations, here called a "map box". In addition to storing locations, your class should be able to filter out duplicate locations and calculate a bounding box around all locations that are stored. Your provided Java™ source code must be compatible with a collection of provided Java™ classes that implement individual locations and a top-level driver for testing your "map box" implementation. Your exercise solution will be evaluated by combining your submitted class file with copies of the provided utility class files and testing the resulting complete program against a variety of test cases. In other words, *your solution must work with the provided utilities, without any modifications to these provided files*.

More specifically, you are to provide a class called `MapBox` in a source code file named "`MapBox.java`". Your class must have the following features ...

- a default constructor that allocates any needed storage
- a public member variable called "`locations`" that holds a list of objects of the type `Location`
- four public accessor member functions that return (as type `double`) the coordinates of the bounding box:
  - `Westmost` — returns the minimum x-axis coordinate
  - `Eastmost` — returns the maximum x-axis coordinate
  - `Southmost` — returns the minimum y-axis coordinate
  - `Northmost` — returns the maximum y-axis coordinate
- a public member function called `recordLocation` that takes a single `Location` object as an argument and adds that object to the `locations` list, but only if the given `Location` object does not have the same name as any previously recorded location

In general, your classes should allow the `main` method in the provided `Ezero` class to output correct solutions, including the coordinates of the bounding box and a complete list of locations, without duplicates. Note that this means that the member functions of the class that you implement should write *no output* to the standard output stream, as this will clutter the output produced by the `Ezero` class `main` method. If you include any statements that write output in your code (perhaps as tools for debugging) these should be removed prior to submitting your source code file for evaluation. You will not receive credit for the exercise if your solution produces extraneous output.

The Java™ utility classes that you are required to use are provided in a ZIP archive file called "`EX0.zip`" which is available in the "Assignments" section of the class CatCourses site, under "Exercise #0". These utilities include:

- `Location` — This object encodes a location on the map. It includes a location name, a longitude (x-axis coordinate), and a latitude (y-axis coordinate).

- `Ezero` — This object provides a top-level driver that tests your solution. Your code must allow `Ezero` to produce correct output.

The contents of these Java™ utility files will be discussed during a course laboratory session, and inline comments in these files should assist in your understanding of the provided code. Questions are welcome, however, and should be directed to the teaching team.

Your submission will be evaluated for both accuracy and efficiency. Not only must your solution allow `Ezero` to produce correct output, but it must do so with time complexity that is strictly linear in the number of locations. This means that any check for duplicate locations must be done in *constant time*. During the laboratory session in which this exercise is introduced, members of the teaching team will discuss methods to efficiently check for duplicates.

As for all work done in this class, submitted solutions should reflect the understanding and effort of the individual student making the submission. Not a single line of computer code should be shared between course participants. If there is ever any doubt concerning the propriety of a given interaction, it is the student's responsibility to approach the instructor and clarify the situation *prior* to the submission of work results. Also, helpful conversations with fellow students, or any other person (including members of the teaching team), should be explicitly mentioned in submitted assignments (e.g., in comments in the submitted source code files). These comments should also explicitly mention any written resources, including online resources, that were used to complete the exercise. Citations should clearly identify the source of any help received (e.g., "Dr. David Noelle" instead of "a member of the teaching team", "the Oracle Java Technology Reference API Specifications at `www.oracle.com/technetwork/java/index-jsp-142903.html`" instead of "a Java web page"). Failure to appropriately cite sources is called *plagiarism*, and it will not be tolerated!

The members of the teaching team stand ready to help you with the learning process embodied by this exercise. Please do not hesitate to request their assistance.