CS 342 Project 3: Data Structure Visualization

Teams and Overview

This is the second major project and team project. It is much more open-ended than previous assignments, but some basic ground rules will be given on this page.

As noted when you formed teams for the Monopoly project, your team for this project must be entirely different from the last. In more precise words, for all pairs *a*, *b* of people working on your Project 3 team, *a* and *b* must not have worked together on the Monopoly project. Teams must have 3 or 4 students.

This project kicks off early during Week 10 and will wrap up near the end of Week 12.

Functional Requirements

You must visualize some data structure, and illustrate how it changes. (Your target end user might be another CS student who is learning how to use the structure. As a side motive behind this project, you'll learn whatever you're visualizing more deeply than you did before. You might choose something you think you need to understand better for the learning experience.)

Choose from among these data structures (some are technically abstract data types):

- stack
- queue
- linked list
- doubly linked list
- heap
- priority queue
- chained hash table
- binary search tree
- red-black tree
- treap
- 2-3-4 tree
- B tree

You must illustrate your structure and its basic operations. Generally, this includes creation, insertion, deletion, and search (or lookup) - but it depends on your precise structure. Red-black trees are sufficiently complicated that I'd say leaving out deletion still yields a reasonable project.

Alternatively, you could build a graph and focus on a graph algorithm, like BFS, DFS, topological sort, Prim's and/or Kruskal's algorithm, Dijkstra's algorithm.

Your program should be able to handle varying sizes, but you can put a reasonable limit on the size. It varies based on the structure and we can discuss, but, for example, I'd say supporting trees with about six levels is reasonable.

The user should be able to control the basic operations and your program should illustrate the change happening. For operations that require multiple steps, the user should be able to see the change happening one step at a time.

If you're interested in some topic that's not on the list above, ask. (It should probably be something that's in the classic Cormen/Leiserson/Rivest/Stein text, and I would, in general, recommend that as a reference on the operations for many of the above.)

Technical Requirements

You must create a Java applet or application meeting at least the following:

- You must use classes and objects.
- You must implement the data structure you're illustrating in a class or classes. You must build it from primitive types (and arrays, if applicable). (You may **not** use anything already built.) Presumably, you've implemented some of these structures in languages where you manage memory more directly, but you'll get the experience of implementing a data structure in Java.
- You must use one or more of the layout managers taught in class. I'm imagining and expecting you'll leverage the lesson on using BorderLayout with panels.
- Your GUI and graphics must scale with different reasonable window sizes (say from 600 by 400 pixels up).
- You must implement a ScaledPoint class, described below, and use it to represent coordinates in your back-end.

ScaledPoint Class

You must make all of your graphics resize-able and one basic requirement is that you define a ScaledPoint class to help with this. Objects of this class will store locations on the screen as horizontal coordinates that are percentages of the width and vertical coordinates that are percentages of the height of the drawing space. For example, a point (100, 200) in a 400 x 1000 window would be represented as (0.25, 0.2). This class will hold all of the functionality for storing points and scaling them to the proper size in pixels.

Required public functions are:

1. default and initializer constructors
2. methods for setting either coordinate based upon its pixel coordinate value at the overall screen size in that dimension (e.g., 100 and 400 in the *x* direction for the example above)
3. methods for getting the pixel value of either coordinate for a given drawing canvas width or height (e.g., a function that would take in 800 and return 200 for the *x*-coordinate in the example above).

Add other functions as necessary or to support the simplicity/maintainability of the required functions.

As you do your drawing work, store coordinates using this class, instead of absolute coordinates.

Deliverables and Schedule

Here are some basic ideas, which may be firmed up later:

- During Week 10, you'll form teams and get a start on the project. You should pick topics and teams on Tuesday, but we'll wait until Thursday to make them official in case some students have trouble finding teams.
- Between classes during Week 10, you should do some high level planning and start sketching what your GUI will look like using pencil and paper. Bring your sketches in progress to class on Thursday, 10/29. We'll leave around a third of the class time for working on this. At the end of class, you'll either turn in your sketches, or we'll do a check that you've completed them.
- By the middle of Week 11, you'll be required to make an electronic submission of your ScaledPoint class and back-end classes used for implementing your data structure. This is not a correctness check, but, like before, a checkpoint to force you to spread the project work out reasonably so you finish successfully on time.
- By Thursday's class in Week 11, you should have at least a partially-functioning GUI running.
- At some point late in Week 11 or early in Week 12, you'll be required to do basic usability testing of your project and be test users for other student's projects.
- Your project should be mostly done by class time on Tuesday of Week 12. Around that point in time, you'll do code reviews of another team's code.
- The project is tentatively due on Thursday, 11/12. We'll see how demos go with Monopoly before firming up details. For a project like this where it's creative, it might be nice to have presentations of projects to your classmates (not just us).