8 Assignment

Portfolios: Abstracting over the Datatype: Generics

1. As usual, finish the Lab 8.
2. Recall the class *Song* from Lab 6. Make three examples of lists of songs as instances of AList¡Song¿, each containing at least four songs. Now define any classes you may need so that you can use the methods defined for AList¡T¿ to produce each of the following:
   - Does the list contain only songs that cost less than 10.00?
   - Doeas the list contains a song by "Beatles"?
   - Produce a list of those songs from this list that cost less than some given number.
   - Produce from this list a list of all songs by a given artist.

Abstracting with Interfaces and Function Objects Abstracting with Generics: Type Parameters

8.1 Problem

You will start this assignment with the given code. We will deal with two classes of data, *Balloon* and *City*. The goal will be to design common methods that work for both classes without changes.

Here is a brief inventory of the classes that are provided:

- *City* — the information includes the name, state, zip code, latitude, and longitude
- *Balloon* — the information includes the x and y coordinates of the center, the radius, and the color of the balloon
- *AList*<*T*>, *MTList*<*T*>, and *ConsList*<*T*> that you have seen already in the lab. It includes the *filter* method that consumes an *ISelect*<*T*> object and produces a list of all elements from *this* list that satisfy the given predicate.
- *ISelect*<*T*> interface that represents a *select* predicate

- *Examples* — the class with some examples of data and some tests already implemented

1. Design the following classes that implement the *ISelect* interface:

   - *RedBalloon* — that selects only the red balloons
   - *SmallBalloon* — that selects all balloons with the radius smaller that the value given to the constructor.
   - *Below40th* — that selects only the cities that are below 40th parallel of latitude
   - *InState* — that selects only the cities in the given state.

   Make sure you test all these classes.

2. Design and run tests for the method *filter* in the classes that represent a list of <*T*> by using all four classes that implement the *ISelect*<*T*> interface.

3. Add the following interface to your project:

   ```
   interface ShowMe<T>{
     public void display(T t);
   }
   ```

   Now design the following classes that implement the *ShowMe* interface:

   - *PaintBalloon* that paints the balloon data in the *Canvas*
   - *PaintCity* that paints the city as a small circle in the *Canvas*.

     Hint: To be able to draw the city on a Canvas, assume that the latitude and longitude lines are parallel, and that the entire map is drawn between the 65th and 125th longitude line and between 25th and 55th parallels (latitude lines).

4. Now design the method *showAll* in the classes that represent a list of <*T*>. Test is by using all four classes that implement the *ShowMe* interface.

5. Java Collections Framework — the libraries we will soon use — provides the following interface:

   ```
   public interface Comparator<T>{
     /* produce int < 0 if op1 is before op 2
      * produce 0 if op1 is the same as op2
      * produce int > 0 if op1 is after op2
      * in your desired ordering
      */
     public int compare(T op1, T op2);
   }
   ```

   Design the following classes that implement the *Comparator<T>* interface with methods that perform the following comparisons:

   - *B1HigherThanB2* that determines whether balloon-1 is closer to the top than balloon-2
   - *B1SmallerThanB2* that determines whether balloon-1 has smaller radius than balloon-2
   - *C1BeforeC2* that determines whether the name of the city-1 is lexicographically before the name of the city-2
   - *C1StateBeforeC2* that determines whether the state of the city-1 is lexicographically before the name of the city-2

6. Design the *sort* method for the classes that represent a list of *<T>* using the given instance of the *Comparator*. Test your program (and all helper methods) using all four of the classes defined above. ∎

8.2 Problem - Extra Credit

You can now generalize the graph traversal solution from Assignment 7 to draw a graph of state capitals with lines connecting neighboring states (and all *Four Corners* states). Try to generalize your solution so you would be able to draw the map and show the path from one city to another.

You can hand in a text document that describes what you have tried to do and how far did you get, if you do not complete the whole program.

We suggest that you first work only with the New England states, to make the data entry less painful. Soon, we will read data like this as an input from a file.

∎