

8 Assignment

Portfolio Problems

Finish all the lab for this week and include it in your portfolio.

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* $\langle T \rangle$, *MList* $\langle T \rangle$, and *ConsList* $\langle T \rangle$ that you have seen already in the lab. It includes the *filter* method that consumes an *ISelect* $\langle T \rangle$ object and produces a list of all elements from *this* list that satisfy the given predicate.
- *ISelect* $\langle T \rangle$ 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 than 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 $\langle T \rangle$ by using all four classes that implement the *ISelect* $\langle T \rangle$ 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:

- *PrintBalloon* that prints the balloon data as a *String* to the system output
- *PaintBalloon* that paints the balloon data in the *Canvas*
- *PrintCity* that prints the city data as a *String* to the system output
- *PaintCity* that paints the city as a small circle in the *Canvas*.

Hint: Compute the location of the city on a map of USA that would fit into your *Canvas* from its latitude and longitude. We will draw more cities later, so make sure you can use this code later.

4. Now design the method *showAll* in the classes that represent a list of $\langle T \rangle$. 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. ■