8 Abstracting Over the Data Type

Portfolio Problems

1. Finish Lab 8 and include all the work in your portfolio.

2. Return to the Problem 6.2 from Assignment 6. Change the definition of the `ABST` and the `Ordering` so they can deal with a binary tree of `Song`s, or `Images`, or any other data type.

Pair Programming Assignment

8.1 Problem

Abstracting Over the Data Type

Download the file *Expressions.java*. It includes the implementation and some sample tests of the classes that represent an arithmetic expression where the values can only be integers, and the only operation allowed is addition.

A. Study the class diagram for this class hierarchy. Extend the example so that the expressions can also include multiplication.

   Hint: Add the class `Times`.

B. Design the method `toString` that produces a `String` representation of this expression with parentheses surrounding every binary expression. Define examples that represent the following expressions and include tests that verify that they have been correctly rendered as `String`s':

```
(2 + (3 + 4))
((3 + 5) * ((2 * 3) + 5))
```

C. We now want to represent relational expressions (that compare two integer values and produce a boolean value). We limit our choices to the *greater than* and *equal to* comparisons. We also want to represent boolean expressions, *and* as well as *or*.

   Change the definitions so that they are parametrized over the type of data you will use.

The `IExp` interface is parametrized only over the type of value it represents when evaluated.

The `BinOp` class needs to be parametrized over the type of operands it receives, as well as the type of value it produces.

D. Add the necessary class definitions so you can represent *relational* and *arithmetic* expressions.

   Make sure you have examples for each of them, as well as tests for the `eval` method.

E. Now design two new classes `IntVar` and `BoolVar` that will represent a variable of the appropriate type in the expression and `implements` `IExp`. It needs to keep track of its name, e.g. `x`, or `width`, etc.

   It should include a method `substInt` for the class `IntVar` and the method `substBool` for the class `BoolVar` that consumes a `String` and an argument of the appropriate type and produces an instance of a `Value` that represents the given value, provided the given `String` matches the variable name. In all other cases it just returns `this`.

   Of course, it has to include the method `eval`. However, this method should `throw` an exception, indicating that an expression with a variable in it cannot be evaluated.

F. Design the method `noVars`, a predicate that verifies that the expression does not contain any variables.

G. Design the methods `substInt` and `substBool` for the entire `IExp` class hierarchy, that produces a new `IExp` in which every occurrence of `Var` that matches the given name is replaced with an instance of the class `Value` with the given value. Throw an exception if there is an attempt to substitute a `boolean` value for the identifier that represents an `int` value as well as if there is an attempt to substitute a `int` value for the identifier that represents an `boolean` value.

8.2 Problem

Game Design

The game design project you start here will last for three weeks. During the first week you will work on the design of the game on paper, producing a well written proposal that describes what you plan to accomplish. During the second week you will design enough of the game so that all key objects can be displayed, their interactions are defined, and some of the behavior has been defined as well. You will finish the design in the third week. The final version will be due on 19 November.

We will have *code walks* when you present your code to the class during the class on November 23rd, and, if we run our of time then, we will finish on November 30th.

You will then switch partners for the remainder of the term and work on one of the two games - making the changes suggested during the code walks, adding functionality and 'special effects', or just catching up with the test coverage. The final projects will be presented during the last two classes on December 7th and 9th.

Game Options

A. **The Map Finder**

   The canvas displays a network of nodes (cities with roads between them, airports with flights that connect them, intersections with roads that go from one intersection to another, a train network, or an Internet network).

   User selects the origin and the destination and the program finds the routing that leads to the destination and displays it.

   It may produce a list of places to travel through, or may animate the routing by highlighting each segment in turn.

   The beginner version of this game allows only maps with no cycles - so that there is a unique way of getting from one place to another.

   You may create the routing instructions that actually tell the driver what to do: drive 30 miles North, turn left, drive 20 miles East, etc.

B. **Traffic Simulation**

   The canvas displays at least one intersection with traffic moving in four different directions (East, West, North, and South). Cars enter the road at random times, move along, and watch for the traffic light that controls the intersection.

   A more complex version allows turns, have the cars move at different speeds, or has multiple lanes of traffic.

   The game collects data about the traffic flow to help the traffic engineers adjust the traffic light timing.

Another version of the game allows the controller to set the traffic light timing before the simulation begins.

C. **Frogger**

Any variation of the classic frogger game. The player tries to move from one side of the canvas to another, avoiding some of the moving objects (fish that will eat it) while using other moving objects (lily pads) to help it get along.

Score is kept of successful crossings as well as lives lost.

This week's tasks

Write a proposal for the game you plan to design. It should be written as you would write a paper for your English class: properly designed paragraphs, spell checked, organized so that the reader really understands what you are proposing. It should contain the following sections.

A. *Abstract*

A high level description of the game. Think of this as an abstract, or an advertisement you may want to put on a web page if you are selling the game. No details, just an outline.

B. *Player's Guide*

A description of how the game is played. This should include a sketch of how the canvas looks at the start, what controls does the player have, what are their behaviors, and what are the goals of the game.

Write this for the most basic version of your game.

C. *Design Document*

This section is written for the programmer who has to continue working on your program after you have left the company, because you found a better job.

Start by brainstorming about the roles of various objects in the game and think what classes you may need to represent them. If you need a class to represent a collection of objects, you may just call it *Collection of LilyPods* without specifying whether it will be a list or a binary search tree, or something else. (We will soon see other ways or defining composite data.)

Once you have this done, write down a for each object or each collection what information it will represent, what data will be used to represent that information, and what are the actions you expect this object to perform.

Finally, add a class diagram that shows the relationship between the classes described earlier.

Write this for the most basic version of your game.

D. *Extensions*

Describe at least two ways in which you can enhance the game and make it more interesting and challenging for the player.

Special Instructions:

Submit the electronic copy with your other work and leave a paper copy of your game proposal in Professor Proulx's mailbox in 202 WVH by noon on Friday.