2 Designing Classes

Practice Problems

Practice problems help you get started, if some of the lab and lecture material is not clear. You are not required to do these problems, but make sure you understand how you would solve them. Solving them on paper is a great preparation for the exams.

Work out as complete programs the following exercises from the textbook:

Problems:

1. Problem 2.4 on page 17
2. Problem 3.1 on page 25
3. Problem 4.4 on page 33
4. Problem 5.3 on page 44
5. Problem 5.9 on page 51
6. Problem 10.2 on page 97
7. Problem 10.5 on page 105
8. Problem 14.1 on page 140
9. Problem 14.7 on page 144

Pair Programming Assignment

2.1 Problem

A. Convert the data representation for the US cities from the previous assignment to data definitions in *FunJava* language. (Class `City`)

B. Make sure you have a separate class `Loc` for the location given in the latitude and longitude coordinates.

C. Define the class hierarchy `IRoute` that represent a list of cities. Follow the DESIGN RECIPE FOR DATA DEFINITIONS. Remember to make examples of data.

D. Define the class hierarchy `ILoState` that represents that represent a list of states identified by a `String` (typically two letters — the same format as is used in the `City` class. Follow the DESIGN RECIPE FOR DATA DEFINITIONS. Remember to make examples of data.

E. Define the class `Capitol` that represents a capitol of a state. It has a field that represents the city, and a field that represents the list of neighboring states.

F. Define the class hierarchy `IStateMap` that represents a list of `Capitols` of all states on the map.

G. Make examples of this data for the six New England states.

2.2 Problem

Design the classes to represent shapes `IShape` we may want to draw on the *Canvas*. We have circles, disks, rectangles, lines, and a combination of shapes - one on the top, another on the bottom.

Here is what you need to know about each of the shapes:

- A circle has the position of the center, the radius, and a color.
- A disk has the position of the center, the radius, and a color.
- A rectangle has the position of the NW corner, the height and the width, as well as a color.
- A line has the position of the start and end points and a color.
- A *combo* shape consists of the top and the bottom shape.
- The library `colors.jar` defines the `interface IColor` and six classes that implement this interface: `Red`, `Blue`, `Green`, `Yellow`, `Black`, and `White`. The *constructor* for each of these classes consumes no arguments.

A. Draw a class diagram of class hierarchy that represent shapes.

B. Define a class `CartPt` to represent a Cartesian point with integer coordinates.

C. Define the classes that represent shapes. Use the `CartPt` class to represent the positions of shapes. Use the `IColor` type to represent the colors of the shapes.

D. Make examples of data: Draw a picture on paper that you want your shape to represent. Then write the data definitions that will represent this picture.

2.3 Problem

Creative Project

We continue with the design of an interactive game in the style you have done in the first course. A game consist of several different objects. The object move either on each tick of the clock, or in response to the keys (typically the arrow keys). There may be other changes in the game object over the time or in response to the key events (x key launches a shot, an animal gets hungrier as the time goes on, ...). The objects interact in some predefined manner. Finally, something (the state of an object, the interaction between objects) triggers the end of the game.

A. Write down the description of the simple version of the game that has been approved by the instructor.

B. For each object (or a collection of objects *e.g. a list of objects*) that will be used in the game do the following:

   (a) Describe briefly its behavior during the game: does it change with the clock tick?, does in respond to key events?, does it interact with another object in the game?

   (b) Design a class that represents this object. Often, the only information the class needs to keep is the position of the object.

   Include the description of the behavior as a *block comment* before the class definition.

   Make examples of this data at several stages in the game.

   (c) Identify the essential information you will need to keep track of as the *World* scene changes. This will include objects in the classes defined above, as well as possible other data (game score, a flag indicating whether the game ended, etc.).

   (d) Design the class `GameWorld` that includes all objects involved in the game and any data identified in the previous step.

   (e) Make examples of the initial `GameWorld` and a couple of intermediate *worlds* you expect to see in the game, as well as the state of the world at the end of the game.

(f) Design a picture that will represent each of the objects in your game as well as the entire scene.

A picture is composed of disks, circles (outlines), rectangles, lines and text in six possible colors: red, blue, green yellow, white, or black. The picture should be an instance of the class hierarchy `IShape` defined in the previous problem.