Lab 2: Working with Self-Referential Data

Goals: The goals of this lab are to review data definitions and practice designing self-referential classes and data, then design methods for a variety of class herarchies using delegation.

Related files:
tester.jar javalib.jar TaskStarter.java TaskManager.java

Submission: Remember to complete the questions in handins before the end of lab!

2.1 Initial scenario

You are designing a reminders program, which allows you to keep track of all the tasks in your busy life.

In this program, a Group is an organizational unit that has a title (such as "Today") and a list of tasks. Each Task has its own descriptive name (such as "Submit Assignment 1") as well as an indication of whether the task has been completed or not.

When you run reminders, you supply it a group; the program should then show you the group's title and the first task (including its description and some way of indicating whether it has been completed or not); if the task list is empty, a friendly congratulations might be in order.

Pressing the right arrow key on the keyboard should allow you to scroll through the tasks in order - it should cycle such that after seeing the last task, you come back to the first. When viewing a task, pressing space bar toggles whether the task is complete or not.

The starter code given contains the code for running a world program in Java. Your job is to complete the methods in Group (shown as stubs) as well as designing any helpers that you need to delegate tasks to other classes/interfaces. (The code for drawing the world is given although it does not currently show the Group title. You may edit the drawing methods to add this if you have time.) **Remember to follow the design recipe for every method you need.** Start by adding templates to all of the classes.

After completing the methods, create a run configuration and run the program!

2.1.1 World programs

Consider the given code for running a world program. World programs should be familiar to you from fundies 1.

At a high level, implementing World Programs require two main ideas:

- **World State:** the data that changes during the lifetime of the program
- **Event Handlers:** the functions that respond to such events as user interaction, the passage of time, and needing to visualize the world state.

Look through the documentation for the Image Library to see what other event handlers is supports. We will see World programs in a future assignment.