Programming Structures and Abstractions (CS 2334)
Project 4

October 29, 2009

Introduction

In this project, you will be creating a graphical user interface for viewing, manipulating and executing Finch Actions. This project brings together your work in project 3 and labs 4 & 5.

Your program will maintain a single "master" Finch Action list that will be manipulated by the user. At any given time, the user may view the entire master list or just those actions that match a specified "filter" name. The master list will be manipulated by the user in several ways. First, the user may load an action list from the disk, either replacing the existing list or merging with it. Second, the user may manually add new actions or edit existing ones through a dialog box. Finally, the user may delete actions from the master list.

Milestones

1. Create a FinchData class that extends FinchDataAbstract and contains the master action list and provides a set of methods that will allow the GUI code to manipulate the list. (20 pts)

2. Create the GUI layout for the main window. This class, FinchDisplay, will extend the FinchDisplayAbstract class. (5 pts)

3. Add action listeners to the main window that allow the user to manipulate the action list. (20 pts)

4. Create the layout for the pop-up dialog box that allows the user to view/edit a single Finch Action. This class, FinchActionDialog, will extend FinchActionDialogAbstract. (5 pts)

5. Add the action listeners to the dialog box that allow the user to edit the Finch Action. In addition, add the necessary functionality that will allow the dialog box to create a new Finch Action object. (20 pts)

Other components:

- Develop and use a proper design: create an **instance diagram** that shows the parent-child relationships of the graphical component instances. (15 pts total: 10 pts for the design; 5 pts for the final version of the program)
- Use proper documentation and formatting (javadoc and in-line documentation) (15 pts)

Note: of the design and documentation components, a total of 10 points are available during the design phase. The remaining 90 points are obtainable for the final submission of the project.

Other notes:

- This project is **much** more complicated than any of the others to date (you have about 1000 new lines of code to write, not including documentation). Start early.
- You must extend the provided abstract classes (FinchDataAbstract, FinchDisplayAbstract and FinchActionDialogAbstract). These abstract classes may not be edited. (note: exceptions may be granted by the instructor if appropriate reasons are given)
- You must write your own code for the GUI component layout (no tools may be used to automatically generate this code).
- Implement and test incrementally. Doing the implementation in the order of the milestones will help you do this.
- The example executable may be used as a guideline for the look-and-feel of your GUI. However, you may make different choices.

This lab is due in two phases:

1. Thursday, November 5^{th} at 5:00pm: design
2. Thursday, November 19^{th} at 5:00pm: completed program and short demonstration.

More details for what to hand-in and when may be found below.

Resources

Main web page / projects / project4

- project4.pdf: this project description
- cs2334_project4.pdf: a copy of the lab section slides

- Abstract class definitions: FinchDisplayAbstract.java FinchActionDialogAbstract.java and FinchDataAbstract.java
- FinchDriver.java: driver class
- *.txt: example input files
- project4.jar: executable implementation of the assigned program

Example Executable

project4.jar contains an example implementation of the assigned program. To execute it, move it, finch.jar, finch_core.jar to the same folder. At the command line (unix), "cd" to this folder and execute:

```
java -cp project4.jar:finch_core.jar:finch.jar FinchDriver
```

In windows: you will also need local copies of the **dll** files. Then: "cd" to this folder and execute:

```
java -cp project4.jar;finch_core.jar;finch.jar FinchDriver
```

Note: there is also a project4_noFinch.jar file that you can use instead that does not attempt to connect to the Finch.

Input Files

We will keep the same input text file format as with the previous project.

Milestones

Milestone 1: FinchData Class

Your program will maintain a single "master" Finch Action list that will be manipulated by the user. Your FinchData class will provide this functionality and must extend the FinchDataAbstract class. This class will provide:

- A set of methods for manipulating the master action list. It is through these methods that your FinchDisplay class will manipulate the action list.
- A set of methods for interacting with the Finch. Note: the other classes **will not** touch the Finch directly.

Notes:

- You may need to introduce private helper methods (in particular, you have already implemented some of these in previous projects and labs).
- You should not have to introduce any other public methods.

Milestone 2: FinchDisplay Class Look-and-Feel

The primary GUI window will provide a view of the master action list and a means for the user to manipulate the list. This GUI will include the following components:

- A menu for file manipulation (read and merge both text and binary files, and write binary files) and for exiting the program.
- A name filter. If nothing is specified, then all actions in the master list will be displayed by the window. If a non-empty string is specified, then only actions whose names that match the string will be displayed.
- Buttons for executing the filtered action list in forward and reverse orders.
- A panel that contains the current (filtered) set of actions (one action per line). Hint: use a JList for this.
- A set of buttons for manipulating individual finch actions:
  - Creating new actions
  - Editing a selected action
  - Deleting one or more selected actions, and
  - cleaning the master action list (deleting duplicate entries).

For this milestone, complete the parts of the constructor that will create all of the graphical components. If it is helpful, feel free to insert "dummy" data into the graphical components.

Milestone 3: FinchDisplay Behavior

Provide the full implementation of FinchDisplay, including:

- Implementations of the abstract methods.
- Creation of event listeners that will handle the various events.

Note: you should have this milestone completely working before moving on to the next milestone.

Milestone 4: FinchActionDialog Class

The FinchActionDialog class will extend the FinchActionDialogAbstract class and will provide a pop-up dialog box that will allow a user to create a new Finch Action or to edit an existing one.

In particular, the behavior will be as follows:

- If the user clicks on the **New** button in the main window, then a dialog box with a reasonable set of default values will be opened. Once the user completes the specification of the action and clicks on the **Ok** button in the dialog box, then this action will be added to the master action list.
- If the user selects an action in the main window, and then clicks on the **Edit** button, the dialog box will be initialized with values that reflect the selected action. Once the user completes the specification of the action and clicks on the **Ok** button in the dialog box, then this action will be added to the master action list and the old action will be removed.
- If the user clicks on the **Edit** button without selecting an action, then the interface should behave as if the **New** button was pressed.
- If the user clicks on the **Cancel** button in the dialog box, then the dialog will close, but no changes will be made to the master action list.

For this milestone, complete the parts of the constructor that will create all of the graphical components.
Hints:

- The dialog box will contain components for all possible values for any action.
- Depending on the selected action type, a subset of these components will be displayed. What components are visible can be configured using the setVisible() method for any graphics component. Note that if a container is set to be not visible, then all of its children are not rendered.
- FinchActionDialog **should not** interact with FinchData in any way. Changes to the master action list should be handled only by FinchDisplay.

Milestone 5: FinchActionDialog Behavior

Provide the full implementation of FinchActionDialog, including:

- Implementations of the abstract methods.
- Creation of event listeners that will handle the various events.

Hand-In Procedures

Part 1: Design

Deadline: Thursday, November 5^{th} at 5:00pm

Each group must hand in one copy of each of the following:

1. A printed cover page that lists the group members, work contributed by each, and any outside citations.
2. An **instance diagram** that shows the parent-child relationships of the graphical component instances.

Part 2: Complete program and short demonstration.

Deadline: Thursday, November 19^{th} at 5:00pm

Each group must do the following:

1. Turn in a printed cover page that lists the group members, work contributed by each, and any outside citations.
2. An updated version of the **instance diagram**.
3. Turn in project4.zip to D2L. This is the zip file produced by Eclipse that contains:
   - Each of the class implementations with documentation. The author(s) of each class should be documented at the top of the java file.
   - html description of your project produced by javadoc (including private components).
4. A short demonstration. We will reserve time during your laboratory section for you to demonstrate your working program. You may also attend office hours or make appointments to perform the demonstrations.

General Hints and Notes

- Your program must be your own work. Do not discuss or look at the solutions of other groups in the class. However, you may discuss general issues (i.e., not directly related to the project requirements) with your classmates, as well as use the book and the resources available on the net.
- Start your work early. This is not a trivial programming assignment.
- Ask for help early. If you are stuck on something, talk to the TA or the instructor sooner than later (this is what we are here for).