Program #2 - FunkyList

Prof Bill - Jan 2017

Tue Jan 24 - my Program #2 changes are in red.

Program #2 logistics:

- Emphasis: linked lists, generics, Java Collections, has-a, is-a, JavaFX gui
- Due: **Mon Feb 6, 2017** at the beginning of class
- Worth: **7 points** (7% of your grade)

1. Description

William Tee of the Ye Olde Video Shoppe is very happy with his Program #1 solutions. Wilhelmina Tee III, his granddaughter, is now looking for a more flexible and modern implementation.

In Program #2, we'll return to the 21st century and use Java. There are two parts:

- ➔ Create a `FunkyList` that implements those unusual list commands from Program #1. FunkyList is **generic**, so it can be used for lists of many things.
- ➔ Implement a gui using **javaFX** that displays the lists and makes all the funky commands available graphically.

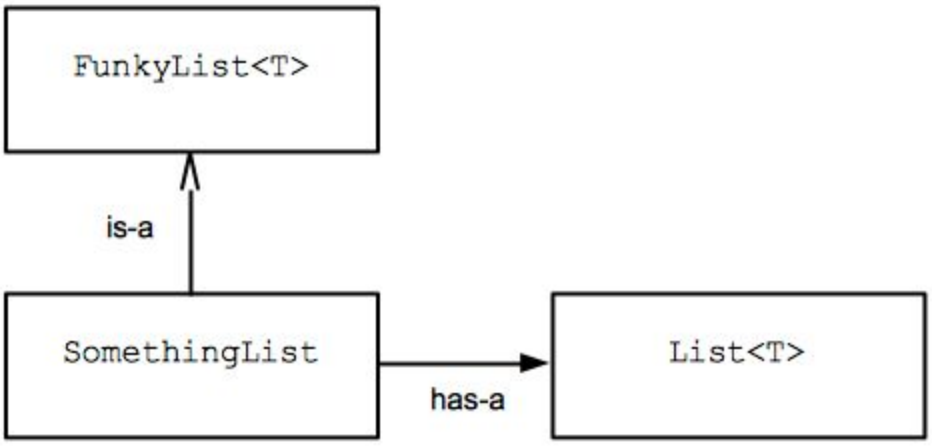
This assignment is less intense and maybe a little more creative as you get to define what your list holds and what your gui looks like. Have fun!
thanks… yow, bill

2. Requirements

Program #2 requirements are:

- Use **Java**. (applause? boo?) Semi-seriously, I hope this gives you a feel for the contrast between an old language in a command-line environment versus a modern OOP language with a hearty system library in a fancy IDE.
- We'll implement the **Program #1 commands**: Program #1 Video List. Remember, we changed "remove" to "delete". **Please reinstate the avalanche command in your Program #2 gui.**
- We'll define a Java interface in class: **FunkyList**. It will be generic. And it will define the methods to implement our commands.
- You class design will be:
  - SomethingList is-a FunkyList<T>.
  - SomethingList has-a List<T>.
  - See the Program #2 UML below.
  - And of course, you'll design some class to list: Video, Prof, Candy, whatev.
- Flex **your creativity**! Since FunkyList is generic, you can make lists of whatever you like: Videos, Professors, Ballerinas, etc. **Also, add 1 BIG or 2 small creative gui things to your solution.**
- Keep flexing! Your **gui** has to show your list and modify it using our commands, but how it looks is up to you.
- Please follow our class **coding guidelines**. Prof Bill's Java Coding Guidelines.

Program #2 UML Class Diagram



3. Details

I'll copy the Program #1 commands here for you.

Command <args>	Description
`add <video>`	Add video to the list in a *random* position; the new video is new selected
`select <num>`	Select video at position <num>
`print`	Print video list
`up`	Move selected video up 1 position
`down`	Move selected video down 1 position
`first`	Move selected video to first position
`last`	Move selected video to last position
`reverse`	Reverse the order of videos in the list
`delete`	Delete the selected video
`clear`	Clear all videos from the list
`exit`	Exit the program

Please submit a drawing of your gui before you start coding it up.

The Muganda Java text has a nice intro to JavaFX, Chapter 15.
You can also google to find plenty of javaFX examples online as well.

Please code up your solution, rather than using a gui builder or FXML. Thanks!

Grading

Create a **program2** folder on your k: drive.This folder should contain:

- All your Java source files
- Your program2 executable
- A **README.txt** file where you describe the status of your program and the creative command that you added

All your code must follow our class **Coding Guidelines**. Ugly code will be severely penalized. A program that doesn't even compile is probably worth 0 points.

How to **succeed** in coding: 1. Start early! 2. Don't be shy. Ask a question in class. Email me. Come to office hours. 3. Work in small bites. Divide your program into small, manageable tasks. Knock them down one by one. 4. Always be working. Your program should always compile and run. Never leave your work in disarray.

Remember our **plagiarism** guidelines as well. Getting help from google or stackoverflow or a friend is OK, but:

1. You must acknowledge any help you receive with a comment in your code
2. You must understand any code in your solution
3. Get help on program components, not the primary assignment (the tic tac toe philosophy)
4. If you have any questions in this area, contact me **before** you turn in your work, not after (when it's too late)

thanks… yow, bill