Program #2 - GUI of Decision

Prof Bill - Apr 2018

Program #2 logistics:

- Due: **Tue Apr 24, 2018** at the beginning of class (2 weeks)
- Worth: **8 points** (8% of your grade)
- Learn: GUI design, JavaFX, Java Collections Framework (JCF), linked lists

1. Description

It's back to the 21st century with Program #2 with a new and improved GUI of Decision. We'll use Java, JavaFX graphics, and ready-made, freeze-dried linked lists out of a package (aka JCF). Finally, mix in some of your own creativity, and you win.

Read on!
thanks… yow, bill

Source: https://en.wikipedia.org/wiki/The_Jetsons

2. Commands

None.
(ha)

3. Gui

Here are some features I'd like to see in your new wheel GUI do (gui do-eee).

Feature	**Description**
Add item	Add an item to the current wheel (disallow while spinning?)
Name wheel	Set/change the name of the wheel (new wheel at the start of a session has a default name?)
Reverse items	Reverse the order of items in the wheel
Show wheel report	Report about the current wheel in a window somewhere: wheel name, size, first item, last item, and anything else you deem interesting
Spin wheel	Spin the wheel, report the chosen item, and then remove it from the wheel
Reset wheel	Reset (or reload) the wheel, either via an explicit gui action or after my wheel items are exhausted after spinning
Clear wheel	Clear all items from the wheel (reset name?)
Save wheel	Save wheel to your library (P2 folder of wheel files? same file format as P1?)
Load wheel	Load a wheel from your library, select from all the wheels in library (again, P2 folder of wheel files?)
Show wheels	Show all the wheels available in your library.

I got a lot of this from our friend, wheeldecide.com.

So, what exactly does your GUI look like? Well, that's up to you. You don't even need to have a spinning wheel… just a GUI that makes random decisions and the other features listed previously. We'll do some of this design work in class.

Some details:

- ❏ **Creativity** - As in P1, please add at least one unique feature to your P2. Creativity! Expression! You!

- ❏ **JCF** - Use the Java Collections Framework (JCF) for your lists. ArrayList?

- ❏ **JavaFX** - Use JavaFX for your gui code. Muganda Chapter 15 is a good start. Prof Google is very helpful here as well. Examples! Also, please don't use a gui builder or some XML thingie. Just code it up.

- ❏ **Gui** - Your program should have a graphical interface, aka a Gui. (duh) We'll have some team design time in class. I'll ask you to submit a sketch of your P2 GUI early on in this process, before you start coding.

- ❏ **Files** - We should be able to reuse our wheel file format from last time: first line the name, each remaining line is an item in the wheel.

- ❏ **Library** - Save a library of P2 wheel files in a designated folder, sort of like an iTunes library.

- ❏ **Wheel? Spinning?** - I'm not requiring that you implement a complex spinning wheel animation. If you do, great. If not, then figure out a compromise. As I said, I don't even care if it's a wheel. Maybe… the magic square of decision? I'm considering the Subaru of Decision myself. We'll see.

How to succeed (writing any program):

1. Start early!
2. Don't be shy. Ask a question in class. Email me. Come to office hours.
3. Small bites. Divide and conquer your program into small, manageable tasks.
4. ABW. Always be working. Your program should always compile and run. Use the debugger. Never leave your work in disarray.

4. Grading

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

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

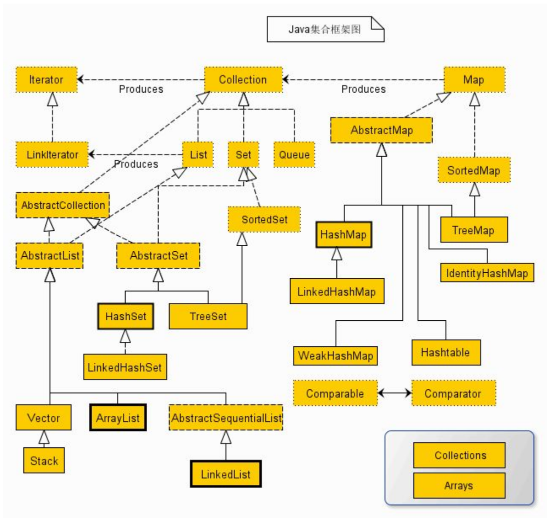
All your code must follow our class **Coding Guidelines**. Ugly code will be severely penalized. A program that doesn't even compile is probably worth 0 points.

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 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

PS - Here. This should help you understand the JCF. (not!)



Source: https://infinitescript.com/2014/10/java-collections-framework/