CS 246	Software Design and Developent

Boldness

Threads of Glory — *Exploration*

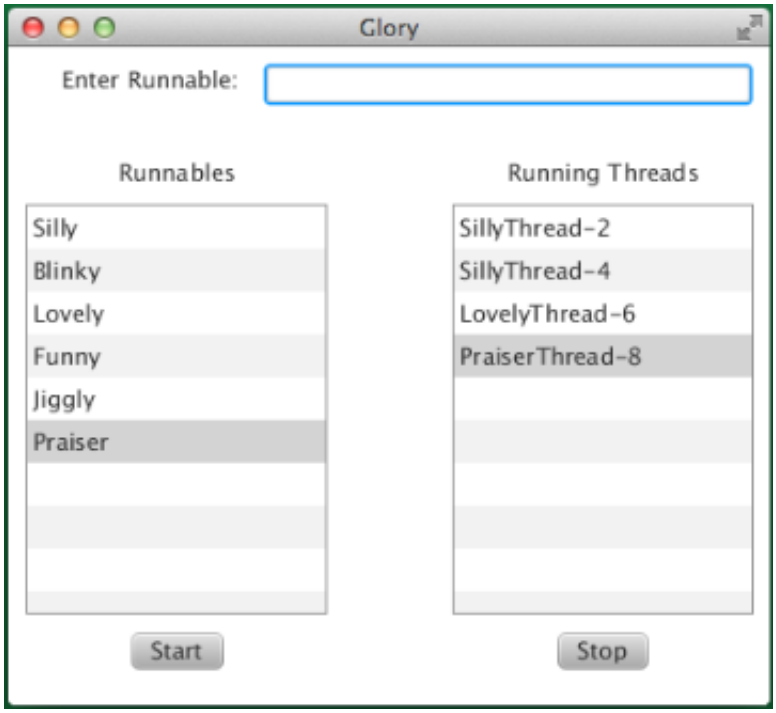
Goal

The purpose of this exploration is to give you a chance to shine — to show the heights of glory to which you can attain by your efforts, both creative and bold.

You should use this as a way to have some fun with while learning more about threads! You will also have the opportunity to learn more about dynamic class loading, and how to build a GUI using a modern toolkit.

Requirements

1. Create a stand-alone GUI application using JavaFX components having *at least* three labels, one textfield, two lists and two buttons, as shown (how it would appear on a Mac):



2. The "Enter Runnable:"-labeled text field will accept the name of a **Runnable** class. After you type text in that field and hit the 'Enter' key, *if that string of text actually names a valid* **Runnable** *class*, the text will vanish from the text field and will be added to the *Runnables* list (if not already there — i.e., no duplicates are allowed in the list). This list allows a single selection at a time — new entries should be "auto-selected" when added (you have to program that behavior, it doesn't happen automatically!) The action of selecting a list item will enable the *Start* button. Pushing the *Start* button will use this **String** class name and do the following:

(a) dynamically load the class,

(b) instantiate this loaded class,

(c) instantiate a **Thread** with that instance passed as the **Runnable** argument to the **Thread** constructor, start this **Thread** instance, and

(d) add the *name* of this **Thread** instance to the *Running Threads* list.

3. Each time you enter a string (the same as one previously entered, or else a different one) in the text field and/or select it from the *Runnables* list and push the *Start* button, create a new **Thread** and start it.
4. Figure out a good way to stop threads that have been started. The *Stop* button should be enabled when you have selected one or more items (yes, multiple selection is allowed) from the *Running Threads* list. Pressing *Stop* should cause each thread whose name is selected to exit normally from its **run** method.
5. Figure out a good way, if there is one, to *restart* threads whose **Runnable** classes have already been loaded.
6. Dynamically load, instantiate and start one or more **Thread** *subclasses* (not just **Runnable** implementations) of your own design.
7. Figure out how to have your **Runnable**s or **Thread** subclasses do something interesting — especially something that interacts with (or manipulates somehow) the user-visible components of the GUI.

Grading Criteria

	Exceptional 100%	**Good 90%**	**Acceptable 70%**	**Developing 50%**	**Missing 0%**
Correctness 30%	10 or more requirements met (see below).	Up to 9 requirements met (see below).	Up to 7 requirements met (see below).	Up to 5 requirements met (see below).	No requirements met.
Creativity/ Elegance 40%	**Good**, plus *all* BOOPOPs and BOODPAPs adhered to.	**Acceptable**, plus created two more Runnables/ Threads that when dynamically loaded and started interact with the Glory GUI somehow (same collaboration caveat). Most BOOPOPs and BOODPAPs adhered to.	**Developing**, plus created a java.lang.Thread subclass of own design (same collaboration caveat). Some but not all BOOPOPs and BOODPAPs adhered to.	Created a java.lang.Runnable implementation of own design (collaborated on ideas but not on implementation).	No plugin Runnables or Thread subclasses ceated.
Style 30%	Heeded all requirements in the Java Coding Standards document. Wrote excellent comments.	Paid attention to whitespace, line length, and brace alignment. Wrote good comments.	Paid attention to whitespace, line length, and brace alignment. Wrote some comments.	Paid attention to whitespace, brace alignment and line length. No comments to speak of.	No attention paid to formatting at all, no whitespace, too much on one line, braces misaligned, etc.

Correctness

Doing any of the following counts towards **Correctness** in the above rubric:

- Code compiles and runs without throwing NullPointer or other exceptions.
- The main window is resizable, and its components resize appropriately as it is resized.
- The "Enter Runnable:" label and companion text field are incorporated in the appropriate places.
- The "Runnables" label is in the appropriate place.
- The Runnables list in single-selection mode is in the appropriate place.
- The "Running Threads" label is in the appropriate place.
- The Running Threads list in multiple-selection mode is in the appropriate place.
- The two buttons, with correct labels, are in the appropriate places.
- The text field functions as it should.
- The Start button functions as it should.
- The Stop button functions as it should.

Comments

- Collaborating on this program was definitely a new experience for me. More than anything I learned how collaborating can both help and hinder the design process. Working with a classmate made this program take a long time because we both wanted to do things differently and had to reconcile some of the differences. Debugging went faster with two of us, but we also created more bugs. We pretty much ran out of time and so the code wasn't optimized completely. I liked learning about how to build a simple GUI and how to load classes into a program without recompiling. Even if we don't score the highest on this assignment, we're both better programmers for trying to work together on it. I've never really worked so closely with someone else like this before. I've learned more from this exploration about teamwork than any of the others and I think that this was good preparation for our class project.
- Your encouragement and method of teaching really has inspired me to give you my best efforts on these explorations. I'm afraid I may have disappointed you on this one. I believe I may have spent too much time reading all the different methods for JList, JFrame, JPanel, Collections, Color, Models, etc., and had many grand designs of really cool things to have this interface do. Alas, I am submitting my best effort that falls well short of these lofty designs. I believe I have learned to try and narrow the scope of my illusionary plans from now on, to provide first, the minimal requirements (which is the first and foremost requirement), and keep it to that. Then if time permits, to try and embellish a little bit, in this situation that would have been fine. Normally you only provide the level of complexity needed to ensure delivery of the required specifications to meet the customers expectations. I totally blew that here. Anyways, it has been an educational experience, and I think that is the main objective anyway. Hope you can find some good program ideas in my exploration and look past all of the deficiencies.
- This was a fun assignment, sadly, I only spent a few hours on it. I thought I knew how to make the [...] work as the break statement for each thread, but I couldn't figure it out. I liked how we are learning how to build up the UI exclusively in code. I think I like knowing "how things work" before I feel good about a tool doing things. I wished I would have started earlier on this assignment. I have a very visual default behavior and I found myself drawn toward the UI aspects of this exploration first and could have spent more time getting the guts of the Thread interactions working first and then going for the UI. That said, I'm not too proud of the UI. Not much polish there.

- I finally got to work with GUI!! I've been wanting an interface for a long time! I now understand threads a lot better than I used to, which was not hardly at all. I like that I was able to learn so much about the Java API docs as I searched and searched and searched for ways to implement [...] to make my code work. I didn't (and still don't) understand how to get instances of objects, or how to call/create objects dynamically with a string. This was practically all that I needed to finish my program, but I could not get it to work no matter how hard or what I tried. I spent the last four or five hours yesterday, and about eight or nine hours today until my brain was totally fried looking through the Java docs, source code examples, and Googling the Google out of Google searching for pages that would help me with this. This really frustrated me and now I have a headache. Other than that I really liked this program and the chance it gave me to get some experience learning Java.
- Well, I didn't fulfill all of the requirements, but I spent quite a few long hours working on this. I swapped a few ideas with a classmate, but we didn't really work together that much. I do have to say that I have learned quite a bit. I explored several different layout possibilities and ended up going with the flow layout because it seemed to have some of the simplest code. Until now I didn't really understand how to use an action listener, but I think that this exploration really helped me gain a better understanding of that. I still don't understand threads as well as I should, but I do have a better understanding of how to use them. With the knowledge that I have gained I can now create some simple GUI programs that can be of use. I liked the fact that we were able to create our own GUI in whatever process we decided and that we could also come up with our own thread design. I can't say that I really have any suggestions for improvement. I prefer a little bit more guidance on what is required and how to do it, but I understand that these explorations are to get us to problem solve and design on our own. This was a good project that I would have done better on had I had more time available and taken more time to study things out.