CS252 Object-Oriented Programing with Java (Zaring)
Fall 2022
Project Phase 3
Due by 11:59pm on Friday, November 4
No late assignments accepted

This phase is a group assignment, not an individual assignment.

The groups are the same as they were for Phase 2 (and will be for the rest of the semester)

- Group 1: Anirudh Chauhan, Owen Gruenwald, Phat Tu
- Group 2: Reece Flynn, Ethan Grunewald, Son Nguyen
- Group 3: Ian Baker, Soren Basnet, Laura Miro Rodrigo, Shulei Wang
- Group 4: Hy Dang, Keagan Larson, Cody Pierce, Samuel Vue

Description:
The overall project is to develop a GUI application that provides as many of the capabilities of the `VM252dbg` debugger as possible. At a minimum, your final version must provide the capabilities provided by the `aa`, `ap`, `amb`, `ba`, `h`, `mb`, `n`, `ob`, `q`, `r`, `s`, and `z` commands along with the capability of loading an object file for execution.

Note that

- The capabilities must be provided in a GUI fashion wherever possible. Note that
  - Providing little or nothing more than an application that has a single text field that lets users type in commands as text and has a single text area that lets users see the results of the commands as text is completely unacceptable and will result in a project score of zero.
  - Providing little or nothing more than an application that has a single panel of buttons (one per command), has a single text field that lets users enter command parameters, and has a single text area that lets users see the results of the commands as text will result in only a minimal project score.
- Some of the capabilities will be provided simply by the GUI nature of your application. For example, providing views of the current contents of the accumulator, program counter, and memory as part of the basic appearance of the application would provide the capabilities of the `s` and `mb` commands without requiring an additional button, menu item, or similar.

Additionally providing the capabilities of the `bl`, `mi`, and `oi` commands is especially desirable.

The capabilities of the `r` command require special mention. When running a program, your application

- Must make it possible to pause and resume the execution of the program without having to terminate the application

- Must keep views of the simulated machine up to date at all times, including any/all views of the contents of the accumulator, program counter, and memory bytes
- Should visually highlight the memory byte(s) holding the instruction currently being executed (in a manner similar to the way that, say, Wing 101 and NetBeans highlight lines of code when you run code under the control of their debuggers)
- Should permit increasing and decreasing the rate at which instructions are executed (e.g., slow the rate of execution down so that the user can watch the program in action without having to single-step through the program)

Add additional features as you see fit and have time: sounds, graphics, user-controlled preferences, etc.

You are free to interpret the above requirements as you will, with the understanding that you need to strike a balance between what is achievable given the time and resources available and what is satisfying and impressive. A working, more-fully-featured application is preferable to a working, minimal application. A working, more-minimal application is preferable to a non-functional, more-fully-featured application.

For this phase, you must produce an initial version of the GUI for your application consistent with your plans for your completed application (realizing that your plans will no doubt change as you go along). Designed using the model-view-controller architecture, this proto-version of your application must be compilable and runnable, although it needn't have any actual functionality (e.g., it needn't exhibit any debugger behavior and needn't respond to button-presses, text entry, scroll-bar manipulation, etc.). However, it should be possible to close the application by closing its window(s) and you must include a skeletal version of a model object implemented as a class (again, realizing that it will certainly change as the project progresses).

Borrowing code from open-source applications of a related nature is utterly, completely, and strictly forbidden, even if you "just want to get some ideas". Your job is to create your own application, not to borrow/steal/glue together application code other people have designed and written.

What to Hand in:

- A zip/gzip/tgz bundle containing all its all associated `.java` files via the `Phase 4` item on the `Project` page of the CS252 Katie course
- Each group must hand in a single PDF of a professional-quality document containing

  - Your first thoughts about how you interpret and plan to meet the minimal requirements
  - Your first thoughts concerning how/if you might go beyond the minimal requirements
  - An annotated screenshot of your GUI indicating what the various components will be used for
  - Your first thoughts about how responsibilities will be divided among the members of the group in an equitable manner
  - Your selection of a primary contact person for your group (the person I'll contact when I need to ask your group questions)