CPS209 Project #2

Choose your own adventure: Swing & 2D Graphics Edition!

Preamble:

Our first CPS209 project had you come up with a project or game or problem yourselves and then solve that problem using the full object-oriented arsenal provided by the Java language. For our second project, you will do much the same thing, but this time with a focus on interactive Swing components, 2D graphics, and event listeners.

Technical Requirements:

In this project you must demonstrate proficiency in three main categories:

1) Creating a graphical user interface in Swing
2) Drawing 2D shapes as part of that interface
3) Implementing event listeners (keyboard/mouse/action) to make your interface interactive.

Note that animation is not required, but you are welcome to include it if you wish. The full requirements for each category are below. These requirements are considered a baseline, or minimum. You are welcome to exceed them and go above and beyond.

Swing Requirements:

- Your program must include a JFrame and any three unique Swing components. These may include buttons, labels, etc. This may include radio buttons, or other such components, that we did not cover in class.

2D Graphics Requirements:

- Your program must add at least 1 JPanel to your JFrame and use these panels as a canvas to draw something meaningful using 2D shapes.

Event Listener Requirements:

- You must include at least one action listener, attached to one of your Swing components. This action listener should result in some kind of meaningful update to your 2D graphics/JPanel. In class, we only covered action listeners for Jbutton components, but if you use other components, this requirement could be covered by one of those instead.
- You must also have one mouse listener or one keyboard listener (or both! Or more than one of each!). Like the event listener above, this listener, when triggered, must result in some kind of meaningful graphical update to the shapes on your JPanel.

Non-Technical Requirements:

- **Program size:** Your solution should require at least 100 lines of meaningful code. Do not write inefficient code or needlessly expand your syntax for the sake of hitting 100 lines. If you're having trouble hitting the 100-line requirement, consider expanding the scope of your problem.
- **Comments:** You don't have to comment every line of code, but you should include a block comment before each method giving a brief description of what that method does and why it exists. Using the Javadoc format is suggested but not required.
- **Main method:** Your main method shouldn't do much. Just create an instance of the class that drives your program and begin any animation loop that you might have. Nothing more. You can refer to the examples we've done in class as a demonstration of this.
- **OOP?** There are no formal OOP requirements in this project, but you should still practice good object-oriented programming principles when structuring your code. For example, using nested classes for defining shapes to be drawn on your JPanel (see examples from the slides).

A note on the word *meaningful*: "Meaningful" implies that this construct plays some useful role in your program. Simple declaring an ArrayList in one of your classes and then not using it is not meaningful and will not satisfy the Collections requirement. Overriding toString() and then simply returning "Hello" arbitrarily is not meaningful, and will not satisfy the toString() requirement. There is some subjectivity here, but as long as you're not trying to game or fake the requirement, you're probably fine.

What to do:

1) Describe your problem/program in a paragraph or two, just like you did for the first project. Your description should be technical and precise – no vague handwaving. This should also include a brief description of how you satisfy the three technical requirements – What event listeners did you use? What is drawn on your JPanel? Which swing components did you use, and why?

You should also include brief instructions for using your program. If there is something the marking TAs should know, make sure it is described.

2) Solve your problem, satisfying all the requirements outlined above.

3) A file called "Project2Runner.java" is included with this project. You can use the main method in this file, and fill out the comment sections asking you to describe how your program meets the requirements. Paste your program description from part 1) into a block comment directly above your main() method.

Marking Scheme:

Out of 25 marks:

2 marks	Clear and concrete problem description
2 marks	Code is (meaningfully) commented
5 marks	100 lines of (meaningful) code
3 mark	Three different Swing components used (meaningfully)
3 marks	2D shapes drawn on a JPanel
5 marks	Action listener present, causes some kind of update to JPanel contents
5 marks	Mouse/keyboard listener present, causes some kind of update to JPanel contents

What to Submit:

Submit all your Java files on D2L under Project #2. Ensure your program works when all files are in the same directory, as this is how we will test your code. In addition, if you are using an IDE other than VSCode, ensure there are no "package" statements left in your submitted code (ie. "package mypack;"). Again, this might only happen if you are using something other than VSCode.

Plagiarism Disclaimer:

You are to work **alone** when writing your code. Students may not copy problem descriptions, and if two students independently come up with the same or similar problems, their programs must be sufficiently unique. You can discuss general ideas with your classmates, but you cannot copy code or develop code together nor take code from the web (this includes asking ChatGPT, CoPilot, or other Generative AI systems to solve your problem for you).

The Department of Computer Science takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned. Please see TMU's Policy 60 for possible penalties and consequences. If you are unsure what constitutes plagiarism, please see your instructor.