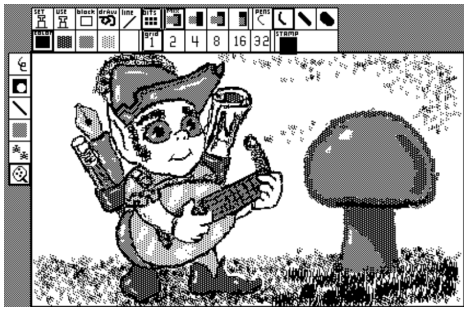
Homework #4, GUI-based Lizard Spock Yahtzee

CPSC 224



THE PROBLEM:

Migrate the text-based functionality of HW #3 to a GUI. If your solution to HW #3 did not achieve that level of functionality it should do so with this assignment.

The suggested approach for this assignment is to produce a succession of Minimally Viable Products (MVP's) that move functionality from the console to a frame. In the rubric for this assignment a "Complete and Correct Program" represents 50% of your score or 50 points. The table below represents one possible sequence of MVP's and the approximate points that would be awarded for each level of additionality functionality. Notice that to achieve all possible points your program should not require the console to play any part of a complete game.

Functionality	Points Awarded
Display the current 5 6-sided dice hand as images in a frame.	10
Click a button to display the current state of a scorecard including bonus and totals.	10
Click on a control widget (checkbox or image of a die) to select the dice to be kept and click a button to initiate a roll or reroll.	10
Display the possible scores of a hand in a frame and be able to click on a control to select a line to place the score. Possible scores should only include unused scorecard lines.	10
Be true to Lizard Spock Yahtzee by allowing the user to choose whether to play with 6, 8, or 12 sided dice, and whether there should be 5, 6, or 7 dice in a hand. Selections should be made via a dropdown box control.	10

Keep in mind that you can actually interact with the console while the GUI is running. It still accepts STDIN and STDOUT, so you can type input and see output statements, though output is easy while STDIN takes some work to still use, so assume you'll need to make GUI elements to generate events. This should help you transition between your terminal based application and the GUI based one as you keep transitioning your interface to the GUI.

With this assignment, I'm including a zip file with some additional media components to help you keep trucking. These include some images of dice and yahtzee imagery. Feel free to find or create your own as you desire. This material could be put in your project in a media/ directory for your application to load up. Remember to use relative paths in your program, not absolute ones or it won't run on other people's systems properly.

> Absolute path: c:/users/… or /home/crandall/apps/
> Relative path: ./media or ./../media

SUBMISSION:

When formatting and turning in your assignment be sure to follow: Dr. Sprint's Coding Standards which include the use of javadoc throughout your code.

Your code should be submitted via GitHub Classroom. The tag for this assignment is HW4. Use the same repository you used for HW2 and HW3. You should branch from your HW3 commit and start on this assignment. You'll probably want to overhaul your main function to start your GUI up and start working from there, or even create a new GUIMain.java and start using that to launch the GUI based game. Once you're done, merge to main and git tag your work has HW4.

After you've tagged your repo with the commit you're submitting, make sure to put a small file on Blackboard for me so I know that you're done.

Your assignment repository should also contain a .pdf document describing the major design and development considerations encountered in implementing your assignment. This document should be written in a narrative style and should include:

- A summary of the goal or purpose of the program in your own words.
- An overview of the general design you chose for your program.
- A UML Class Diagram for game functionality (*do not* include swing or AWT classes)

- UML Sequence Diagrams for game functionality
- A description of any major design and/or programming issues, why these were the major issues, how you addressed them, and why you addressed them the way you did.
- A retrospective of what you would have done differently if you had more time.

Your assignment repository should also include a folder (./doc) containing the HTML documentation generated via `javadoc`.

IMPORTANT:

This is still an individual assignment. Everything you turn in should be the result of keystrokes done by you. You should not consult nor use any existing classes related to Yahtzee or dice. You should not share your code nor look at the code of your classmates. It is ok to have generalized discussions about java and to have high-level design discussions about the classes, attributes, and methods necessary for the solution.

I highly recommend that you write some small throwaway GUI programs to test the individual Swing widgets and to practice with their behaviors. Working with the objects in isolation like that really makes a difference in understanding how they behave, and why they might not be working as expected in your main Yahtzee GUI game.