Homework 07 – Animal Sanctuary

Authors: Nathaniel, Stephanie
Topics: Java FX, Event Driven Programming

Problem Description

Please make sure to read the document fully before starting!
You have just been hired by your local animal sanctuary to help keep track of all the animals! Your supervisor has told you they need an easy-to-use GUI that volunteers can use to enter animal's info as they come into the sanctuary and to see what animals are currently here. Luckily, you've taken CS 1331 and know exactly how to tackle this!

Solution Description

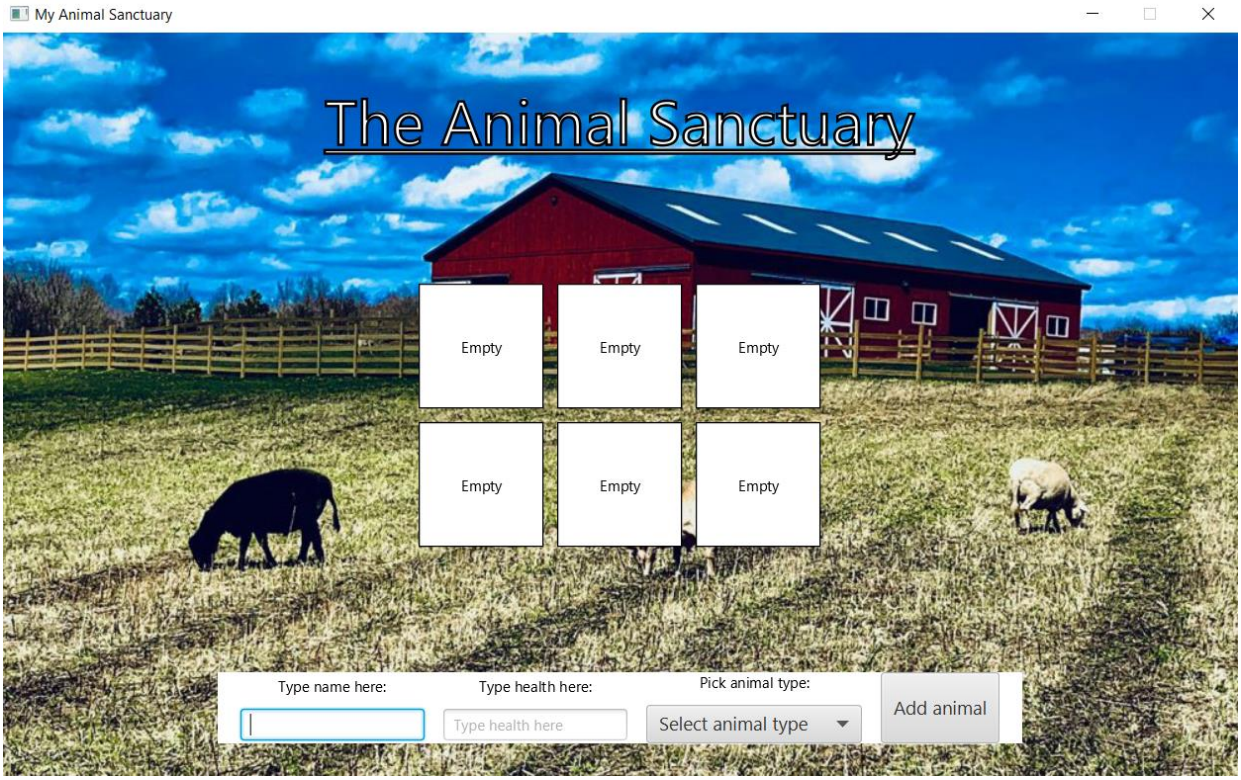
For this assignment, you will create a GUI that meets the requirements below:

`AnimalSanctuary.java`

Write a JavaFX Class called AnimalSanctuary with the following functionality:

- The title of the window should be "My Animal Sanctuary"
- The background image should be an image of animals (of your choice!) **Submit your image with your code as** `animalImage.jpg`.
    - If you want, an image has been provided for you
- There are three main components:
    - An area that shows all of the animals currently in the sanctuary
    - A place for the user to input new animals with their fields
    - A button to submit animal to the sanctuary
- Inputs for animals:
    - This area should allow the user to input information about animals being added to the sanctuary
    - Each new animal should take in a name, type of animal, and health (scale 1-5)
    - Name of Animal
        - A text input field to enter in the name
        - If no name is given, default to "No Name Yet"
    - Type of Animal
        - A dropdown menu to select the animal type
        - Use the provided `Animal.java` enum (Feel free to add to the enum!)
    - Health
        - User should input the health of the animal on an integer scale of 1 to 5.
        - Use a text input field to enter the animal health, if the health is invalid default to a health of 5.
- Button that places the inputted animal in any available space in the animal sanctuary
    - All input values are cleared after this button is pressed
    - If there is no space, send an alert (see Alert class in JavaFX API) to the user that says "There is no more room!"
- Sanctuary area

- The sanctuary should have 6 distinct regions for the animals inside the sanctuary (Think of what kind of pane easily splits the screen into distinct sections)
- If a space is occupied with an animal, it should:
  - Display the animal's name
  - Display the animal's type
  - Display the animal's health
  - Be visually distinct from an empty space (other than displaying basic information). For example, change the color.
- If a space is not occupied, it should:
  - Display "Empty"
- An animal cannot be placed into an occupied space
- If you click on an occupied space, that animal is released back to the wild and the space should revert to being unoccupied



Example:

Tips and Tricks

- **IMPORTANT: Do not use absolute path for your images/files. Use relative path as the autograder will not find your files if you use absolute path.**
- The Java API is your friend! Use it to your advantage.
- Make sure your are properly able to run JavaFX programs before starting this assignment. Do NOT wait until the last moment to configure your JavaFX!
- Work incrementally. Get a basic UI up and running. Add the buttons, cells for display, etc., piece by piece. Think of which type of layout manager(s) you want to use.
- You will be learning about event handling while you have the HW available to you.

Extra Credit: we will award extra credit based on the following grounds:

1. Adding a non-trivial addition to the program's graphics that clearly extends its look [This cannot be something like just changing the background image]
    - E.g. adding pictures for each animal to show when they are in the sanctuary
2. Adding a non-trivial additional to the program's controls or functionality that clearly extends its capabilities
    - E.g. Adding a queue for animals that can't fit in the sanctuary, so they can be added as animals are removed
3. An impressive, simply amazing submission that blows the TAs away (We've seen some students do really cool things before, just run away with the assignment and make it awesome)

- You can choose to do one, two, all the above, or none.
- **If you choose to go for extra credit**, attach a file **extra_credit.txt** to your submission that explains exactly what additional features you have added.
- **Extra credit features should not affect any of the base functionality!**

Checkstyle

You must run checkstyle on your submission (To learn more about Checkstyle, check out cs1331-style-guide.pdf under CheckStyle Resources in the Modules section of Canvas.) **The Checkstyle cap for this assignment is 35 points.** This means there is a maximum point deduction of 35. If you don't have Checkstyle yet, download it from Canvas -> Modules -> CheckStyle Resources -> checkstyle-8.28.jar. Place it in the same folder as the files you want to run Checkstyle on. Run checkstyle on your code like so:

```
$ java -jar checkstyle-8.28.jar yourFileName.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up a bove this message, and the number at the end would be the points we would take off (limited by the ch eckstyle cap mentioned in the Rubric section). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

Additionally, you must Javadoc your code.

Run the following to only check your Javadocs:

```
$ java -jar checkstyle-8.28.jar -j yourFileName.java
```

Run the following to check both Javadocs and Checkstyle:

```
$ java -jar checkstyle-8.28.jar -a yourFileName.java
```

For additional help with Checkstyle see the CS 1331 Style Guide.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `AnimalSanctuary.java`
- `Animal.java`
- `animalImage.jpg`
- **Any additional files needed for your code to compile**

Make sure you see the message stating the assignment was submitted successfully. From this point, Gradescope will run a basic autograder on your submission as discussed in the next

section. **Any autograder test are provided as a courtesy to help "sanity check" your work and you may not see all the test cases used to grade your work.** You are responsible for thoroughly testing your submission on your own to ensure you have fulfilled the requirements of this assignment. If you have questions about the requirements given, reach out to a TA or Professor via Piazza for clarification.
You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your latest submission. **Be sure to submit every file each time you resubmit**.

Gradescope Autograder

If an autograder is enabled for this assignment, you may be able to see the results of a few basic test cases on your code. Typically, tests will correspond to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.
The Gradescope tests serve two main purposes:

- Prevent upload mistakes (e.g. non-compiling code)
- Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or Checkstyle your code; you can do that locally on your machine. Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Allowed Imports

- Any imports from any package that starts with `java` or `javafx` that do not belong to the AWT or Swing APIs.
- Remember that JavaFX is different that AWT or Swing. If you are trying to import something from javax.swing or java.awt, you are probably importing the wrong class.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`
- `System.arraycopy`

Collaboration

Only discussion of the Homework (HW) at a conceptual high level is allowed. You can discuss course concepts and HW assignments broadly, that is, at a conceptual level to increase your understanding. If you find yourself dropping to a level where specific Java code is being discussed, that is going too far. Those discussions should be reserved for the instructor and TAs. To be clear, you should never exchange code related to an assignment with anyone other than the instructor and TAs.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files
- Test your code in addition to the basic checks on Gradescope

- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- **Check on Ed Discussion for a note containing all official clarifications and sample outputs**

It is expected that everyone will follow the Student-Faculty Expectations document, and the Student Code of Conduct. The professor expects a **positive, respectful, and engaged academic environment** inside the classroom, outside the classroom, in all electronic communications, on all file submissions, and on any document submitted throughout the duration of the course. No inappropriate language is to be used, and any assignment, deemed by the professor, to contain inappropriate, offensive language or threats will get a zero. You are to use professionalism in your work. Violations of this conduct policy will be turned over to the Office of Student Integrity for misconduct.