Mav’s Animal Shelter Software

Due Tuesday, March 22 at 8 a.m.

CSE 1325 - Spring 2022 - Homework #7 - Sprint 3 - Revision 0 - 1

Assignment Background

This semester we will be developing the Mavs Animal Shelter Software (MASS), which manages a no-kill shelter aiming to help companion animals of various types to adopt humans. Yes, I have a soft spot for warm fuzzies. We need to keep track of the animals at our shelter, the human candidates for the animals to adopt, and the animal - candidate matches made in heaven.

You are encouraged to replace Dog and Cat with other families of animals such as guinea pigs, tarantulas, snakes, lizards, or parrots.

For the project, we will use a different class hierarchy, with the user interface at root and the data model in package shelter.

Sprint 3

In this sprint, you will add the ability to save and load your data.

Plan your work in Scrum.xlsx, and include your planning and status information at **cse1325/P07/docs/**.

In your git-managed cse1325 directory **cse1325/P07/**, with reference to the UML diagram below, update your JFrame subclass (that is, your main window) to include File > New, Open, Save, and Save As behaviors. Also add toolbar buttons for these 4 new features.

- **New** should create a new, empty animal shelter.
- **Open** should provide a file chooser dialog, showing only .mass files by default, and when a file is selected discard any existing data and load the data from the file. Cancel or file read errors should result in NO loss of existing data and clean recovery.
- **Save** should save existing data to the most recently opened or saved filename. (You'll need a new attribute to retain this filename. I recommend that you store the filename in a String field Shelter.filename.)
- **Save As** should provide a file chooser dialog, showing only .mass files by default, and when a file is selected or a new filename typed in, save existing data to that file. Remember the new filename for subsequent Save requests.

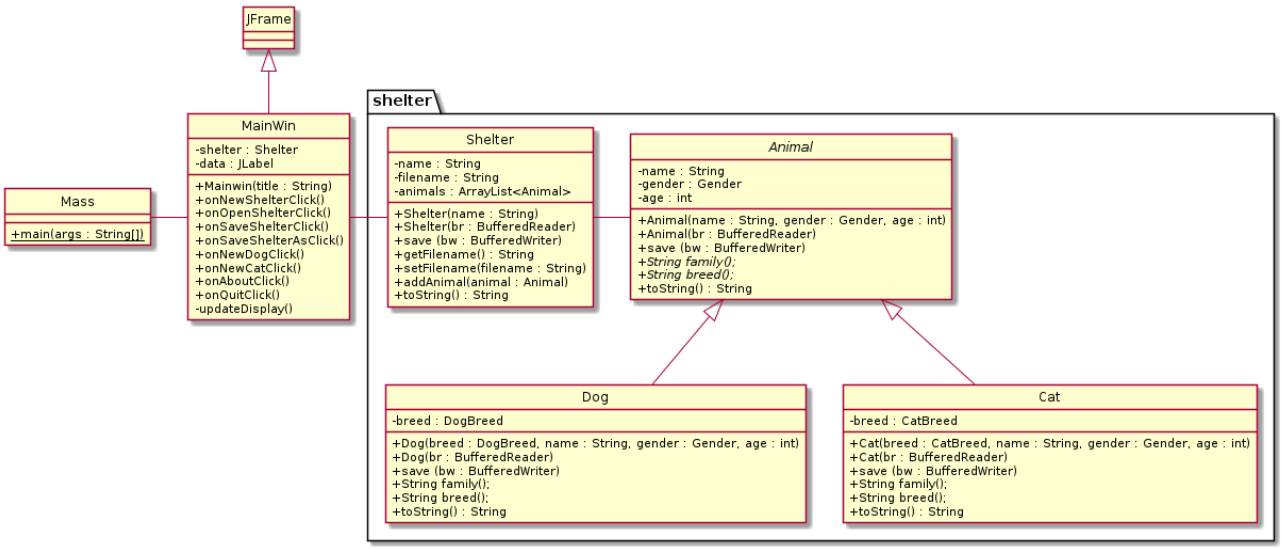
Update your display after each of these behaviors. While it is NOT required that you ensure no data loss (that is, discarding unsaved data without warning), it would be a nice touch to obtain user concurrence (think JOption.showConfirmDialog) before any loss of data.

Don't forget attributions for any toolbar icons you use!

No new classes are required, but new methods will be created. An updated class diagram is below.

You may *adapt* code from the suggested solution for Sprint 2, the Nim project from Lecture 14, and the example code provided in Lecture 14. If your project has serious issues, contact me about baselining the suggested solution for Sprint 2 so that you can get back on track.

Add, commit, and push all files to your private cse1325 GitHub repository.



Hints

Creating Dialogs

Use the JFileChooser dialog provided by Swing. Lecture 14's Nim provides a good example. The recommended file extension is .mass.

File Format

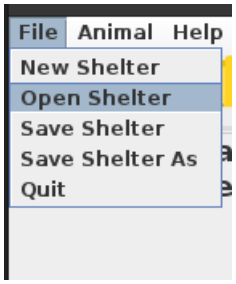
I *strongly* recommend that you follow the pattern taught in lecture, in which each class is responsible for saving (via a save(BufferedWriter)) and restoring (via a *constructor* (BufferedReader)) only its own attributes. This preserves encapsulation and simplifies your code.

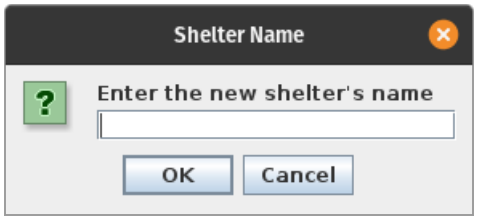
To handle containers such as ArrayLists where the number of elements may vary, first write out the *number of elements*, then tell each element to save itself. When reconstructing, read in the number of elements, and then construct and add that many elements back into the ArrayList.

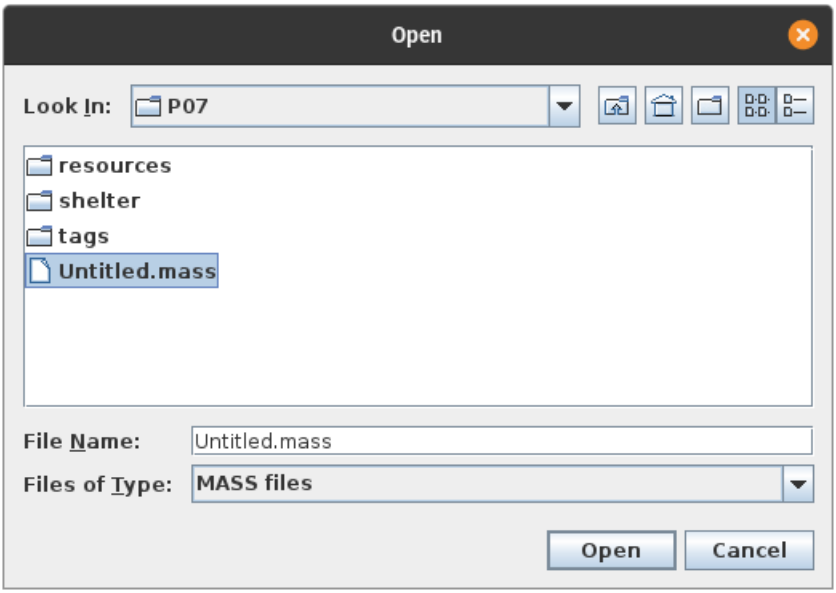
To handle inheritance, have your superclass save its own attributes. Each subclass should delegate to the superclass to write and read its own attributes, then follow with the subclass' attributes.

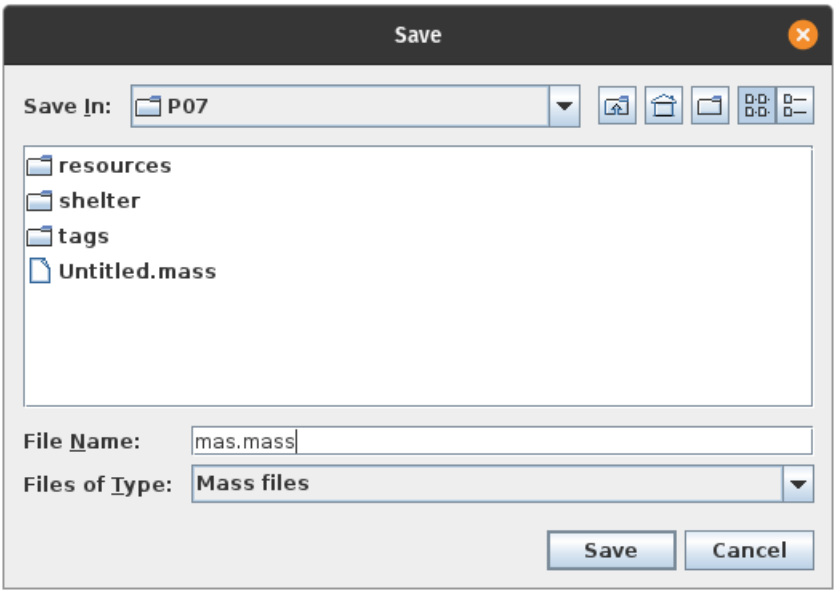
More complicated is the ArrayList of a superclass type (in our case, ArrayList<Animal> in class Shelter). As discussed in class, I recommend that you have each subclass' save method first write its own family (e.g., "dog" as provided by the family() method) out on its own line, followed by its attributes. The Shelter's constructor (that is, the superclass constructor) will read this identifier for each element from the BufferedReader while reconstructing the ArrayList, so that it knows which subclass constructor to call to create the next object to add to the ArrayList.

Screenshots










```
ricegf@antares:~/dev/202201/P07$ cat Untitled.mass
Mav's Animal Shelter
2
dog
Charlie
male
7
Mix
cat
Caramel
female
5
Mix
ricegf@antares:~/dev/202201/P07$
```