CS 1331 Homework 5

Due Friday October 5, 2012 8:00PM

Introduction

This homework will cover images, buttons, and event handling.
Be sure to name your classes as required by the instructions. Also be sure to use good coding style and indentation, and to use appropriate and descriptive variable names.
Do not forget about the collaboration policies detailed in homework 1, or about javadocing and commenting detailed in homework 2 and 3.
Your finished homework will look something like this:

In this assignment, you'll be programming a pet that a user can interact with. As the user plays with the pet, different circumstances will put the pet into different *states*. We've drawn sample images for each state, and have included them with this guide. You may also draw your own.

You can choose how you want to render the images on your GUI, but we recommend using the ImageIcon + Jlabel method (link). You can change the image on a label using the setIcon(Icon) method in JLabel.

5.1: Pet.java

Create a class called `Pet`, which will represent our pet. The pet will have 5 states: normal, sleeping, eating, laughing, and dead.

1. Define an enumerated type called `State`, which contains the following types: `SLEEPING, EATING, DEAD, LAUGHING,` and `NORMAL`. If you've forgotten how to make an enum, check Chapter 4 for more details.
2. Make instance variables for the Pet. It should have a variable of type `State` to hold the current state of the pet, an int to hold the pet's hunger, and variables for all the `ImageIcon`s of the pet.
3. The pet should also have a final instance variable to represent how full the pet can be. Call this variable `FULL` and set it initially to 10.
4. Create a constructor for the pet. The constructor should set the initial state to `NORMAL`, set hunger to `FULL`, and instantiate all the `ImageIcon` instance variables.
5. You should have getters for the status and hunger instance variables, following standard getter / setter naming convention.
6. Create the following methods:
    a. `getCurrentIcon:` returns the `ImageIcon` corresponding to the current state of the pet. For example, if the current state is `SLEEPING`, then it should return the sleeping `ImageIcon`.
    b. `poke:` sets the state of the pet to `LAUGHING`, and decrements the hunger variable. If the pet is dead, do nothing.
    c. `feed:` sets the state of the pet to `EATING`, and resets the hunger variable to `FULL`. If the pet is already eating, then the pet will overeat, so the state of the pet should be set to `DEAD`. If the pet is already dead, do nothing.
    d. `sleep`: sets the state of the pet to `SLEEP`. Do not modify the hunger variable. If the pet is dead, do nothing.
    e. `sit`: sets the state of the pet to `NORMAL`, and decrements the hunger variable. If the pet is dead, do nothing.
    f. `watch`: randomly calls the `poke`, `sleep`, or `sit` method. If the pet is dead, do nothing.
    g. `kill`: sets the state of the pet to `DEAD`.
    h. `checkForStarvation`: checks to see if the hunger value is zero. If it is, sets the state of the pet to `DEAD`. Should be called in the `poke` and `sit` methods.

Don't forget to javadoc the class, constructor, and methods, javadocing parameters where applicable (see the previous homeworks for an example of this).

5.2: PetPanel.java

Create a class called `PetPanel` that extends `JPanel`. This will display the pet and provide buttons for the user to interact with. It should have:

1. An instance variable to store the `Pet`, instance variables for the hunger display and picture of the pet (both of type `JLabel`), and instance variables for the buttons (feed, poke, watch, and kill). They should have the appropriate visibility / protection.
2. A constructor that takes a `Pet` parameter, and stores it into the instance variable.
3. Also in the constructor:
    a. Set the size of the `PetPanel`
    b. Instantiate the two `JLabel`s. The pet label should use the current icon of the pet, and the hunger label should use the current hunger of the pet.
    c. Instantiate the four buttons (feed, poke, watch, and kill).
    d. Add action listeners to the buttons (see step 5).
    e. Add the labels and the buttons to the `PetPanel`.
4. A method called `refresh` which updates the pet label to use the current `ImageIcon` of the pet, and updates the hunger label to use the current hunger of the pet.
5. An `ActionListener` to handle the event listening of the 4 buttons. This may be done with a private inner class, or any other method you have learned. Make sure to call `refresh` at the end of your `actionPerformed` method.

Don't forget to javadoc the class, constructor, and methods, javadocing parameters where applicable (see the previous homeworks for an example of this)

5.3: PetMain.java

Finally, create a class named `PetMain` with a main method that creates a `JFrame`, and brings together a `Pet` object and a `PetPanel` object:

1. Create a `JFrame`, and set the default close operation
2. Create a `Pet` object.
3. Create a `PetPanel` object, using the `Pet` above.
4. Add the `PetPanel` to the `JFrame`.
5. Pack, and set the visibility of the `JFrame`.

Turn-in Procedure

Turn in the following files on T-Square. When you're ready, double-check that you have *submitted* and not just saved as draft.

- Pet.java
- PetPanel.java
- PetMain.java
- Any other files needed to run your program. Please include the state photos!

All .java files should have a descriptive javadoc comment.

Don't forget your collaboration statement. You should include a statement with every homework you submit, even if you worked alone.

Verify the Success of Your HW Turn-In

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.

2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files.**
    (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    c. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework (past the grace period of 2 am) will not be accepted regardless of excuse. Treat the due date with respect. The real due date and time is 8 pm Friday. Do not wait until the last minute!