A7. Quizit. CS1110 Fall 2008 Due date on the CMS

Where this assignment came from. The first programming course in computer science at the University of Toronto has a similar assignment this fall; we took the idea and much of the text below from it. Thanks, Paul Gries, for your help! Their assignment does not have a GUI and is done using Python instead of Java.

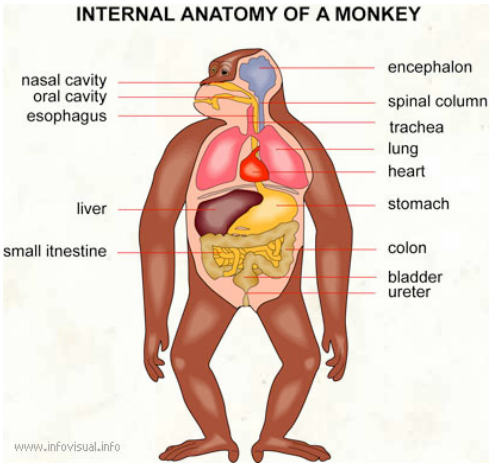
Partners. If you want to work with a partner, both of you should do what is necessary on the CMS several days before the submission deadline. It is dishonest and against academic integrity to split the work so that each does half of the work. You must work together.

Time spent. When you submit this program, we will ask you to tell us how much time you spent. Please keep track of this.

Overview. This assignment is quite different in nature from the previous ones. There is a fairly large program to complete, with more freedom in what you do than in other assignments. The assignment is designed to give you practice in the following areas:

- Program design
- Arrays and Vectors
- String manipulation

Quizit. It's almost time for that animal bio quiz. Or was it the bridge architecture quiz? Either way, it's obviously time to procrastinate and write a program to help you study. Your program will show a picture like the one on the right, but with numbers instead of terms (e.g. "liver"), and the player will try to match up terms on the picture.



Thus, the program you write will interact with the user --a player-- just like in a game.

The player will be given a list of topics (each with a picture) to choose from. The player chooses a topic, and the GUI shows the picture with numbers as labels to various parts of the picture. The player can pick one of the numbers and guess the real term associated with the picture part for that number. After each guess, the GUI provides feedback about whether the guess was right or wrong.

We provide a series of screenshots of this game being played. Please read that page carefully before you read any further!

Also, you can play the game on the web, because we have made an applet of it. Click here or access it from the assignments page of the course website.

Downloading files. From the course website, download file A7.zip and unzip it. It contains files A7.java, Item.java, A7GUI.java, A7Panel.java (which make up the program skeleton that we give you); two images files, shark.jpg and 076monkey.jpg; and file topics.txt, which contains the data that the program will read in. Put all files in a directory, open the .java files in DrJava, and compile.

Input to the program. File `topics.txt`, located in the same directory as your program, contains the information used by the program to draw lines and labels. The file contains no blank lines and has the following structure:

```
topic: topic-name file-name1
term1 x1 y1 x2 y2
term2 x1 y1 x2 y2
term3 x1 y1 x2 y2
...

topic: topic-name file-name1
term1 x1 y1 x2 y2
term2 x1 y1 x2 y2
...
```

Thus, the file is divided into "topic blocks": The first line of each block, starting with `"topic:"`, gives the name of the topic (e.g. Shark) and the name of the file containing an image (e.g. `shark.jpg`). This image file should be in the directory with program.

Each `"term"` line indicates the correct answer to a part of the image and contains the coordinates of the lines from that point. The first

pair, `(x1, y1)`, represents the point closest to the actual picture (and thus farthest from the term); the second pair `(x2, y2)` is the other endpoint of the line (closest to the term).

For the examples in this handout, we used this `topics.txt` file; the image files are shark.jpg. and 076monkey.jpg. Image 076monkey.jpg is a modified version of an image on website www.infovisual.info/02/076_en.html. Gries has written to infovisual.info to ask for permission to use the image this way and to tell them about the assignment --the resulting program might be a neat interactive addition to their website. Their reply was that they were thinking of adding this feature to their wesbite in 2009 and they wanted more information on how we did it. The website has hundreds of images for different topics.

Overview of communication between GUI and your code in A7. All the code you will write goes in file `A7.java`. The program is started either by using it as an *application* by executing `A7.main(null);` or by visiting a webpage that has it included as an applet. We will discuss bboth applications and applets later. In either case, file topics.txt is read in and stored and an instance of the GUI class is created and shown, with the topics filled in.

When the player chooses a topic and clicks button *topic*, the GUI calls an `A7` procedure that does two things: (1) Retrieve the information for the topic from file `inputFile` ---the topic image and the labeling information for the image--- so that it can then draw the appropriate lines and numbers before displaying the image. In the next section, we discuss how this information is kept in a "dictionary" of items. (2) Call a procedure of the GUI to initialize the drop-down list of topics.

When the player clicks button *OK*, the GUI calls an `A7` procedure to process the player's association of a number and a term. This requires two things to be done: (1) Update the item given by the number with the player's choice of a term and (2) if the choice was correct, call the GUI to remove the number and the term from the drop-down lists.

Keeping track of information: a dictionary. For a topic, your program need to keep track of the lines. To do this, use the coordinate pairs (x1, y1) and (x2, y2) found in `topics.txt`. Here's the spiracle coordinate pair for the shark picture: `(187, 180) (177, 131)`. The first point (x1, y1) is the **point on the picture**, not the coordinates next to the label. The lines will be drawn on the image by your program whenever the picture is displayed. (What is a spiracle?)

For each line, your program will also maintain the correct term, the number associated with the term, and the player's guess at the term ("" if none guessed yet).

We suggest that your program use class `Item` to contain the information for one line. We give you a skeleton for class `Item`. It need fields for:

> The points: `(x1, y1)` and `(x2, y2)`
> The correct term
> The player's guess at this term ("" if they haven't guessed this term yet)
> The number of this line

The `Items` for the current topic can be kept in Vector `items`. When a topic is chosen, a method in `A7` will build this dictionary from the information that is in Vector `inputFile`. This dictionary satisfy the following:

- the items are ordered by the points `(x1, y1)` and `(x2, y2)`. For example, if items `b` and `c` satisfy `b.x1 < c.x1`, then item `b` comes before `c` in `inputFile`, and if `b.x1 = c.x1` and `b.y1 < c.y1`, `b` comes before `c`.
- The number of each item is based on its placement in field `items` after the items have been sorted by their points. Thus, the first item in `items` has number 0, the second number 1, etc.
- You may maintain the points any way you wish. Two obvious choices are: (1) use four int variables and (2) use two instances of class `java.awt.Point`.

Here is the dictionary for the shark topic. Each line gives the information in an instance of class `Item`:

```
(87, 188), (75, 181), 'Snout', '', '0'
(113, 192), (99, 160), 'Nostril', '', '1'
(128, 221), (108, 247), 'Mouth', '', '2'
(146, 180), (127, 146),'Eye', '', '3'
(187, 180), (177, 131), 'Spiracle', '', '4'
(194, 229), (188, 289), 'Labial furrows', '', '5'
(267, 242), (248, 312), 'Gill openings', '', '6'
(412, 359), (435, 366), 'Pectoral fin', '', '7'
(431, 110), (393, 87), 'Dorsal fin spine', '', '8'
(479, 76), (466, 55), 'First dorsal fin', '', '9'
(603, 293), (608, 310), 'Pelvic fin', '', '10'
(664, 265), (724, 338), 'Clasper (males)', '', '11'
(739, 141), (764, 108), 'Second dorsal fin', '', '12'
```

```
(745, 273), (754, 297), 'Anal fin', '', '13'
(787, 207), (831, 318), 'Caudal keel', '', '14'
(795, 180), (810, 136), 'Precaudal pit', '', '15'
(953, 182), (984, 191), 'Caudal fin', '', '16'
```

More on interaction with the GUI. Your `A7` methods will have to interact with the GUI. The places of interaction are minimal. We explain them.

A7 procedures called by the GUI:

1. `startTopic(t)`. Called when button *topic* is clicked. This method has to (1) call a method in the GUI to give it the image for the topic, (2) build the dictionary for topic t, and (3) call methods in the GUI to construct the number and term drop-down lists.

2. `associateTerm(t, n)`. Called when button *OK* is clicked. This method has to process the player's request to associate term t with line number n.

3. `drawLinesLabels(g)`. Called when the window has to be repainted. This method has to draw all the lines and labels on them, with calls like `g.drawString(s, x, y);` Just what point `(x,y)` is depends on the line `(x1,y1)–(x2,y2)` that is being drawn. Take a look at the monkey and shark images and notice where the text is placed in relation to the line. You can be as simple or sophisticated as you like in placing the text, as long as it is readable. In developing *our* program, we considered the following:
    - if the line is more vertical than horizontal and y1 > y2, string s should appear centered above the line
    - if the line is more vertical than horizontal and y2 > y1, string s should appear centered below the line
    - if the line is more horizontal than vertical and x1 < x2, string s should appear to the right of the line
    - if the line is more horizontal than vertical and x2 < x1, string s should appear to the left of the line

GUI procedures called by A7:

1. `emptyComboBoxes()`. To make sure the drop-down menus are empty.

2. `setUpImage(imName)`. To place the image that is in file `imName` in the window.

3. `addToComboBox(n)`. To add integer `n` to the numbers drop-down menu.

4. `removeFromNumberList(n)`. To remove integer `n` from the numbers drop-down menu.

5. `addToComboBox(t)`. To add term `s` (a String) to the term drop-down menu.

6. `removeFromTermList(s)`. To remove term `s` (a String) from the term drop-down menu.

How to go about writing this program

As mentioned earlier, the code you write goes in classes `A7` and `Item`. Each has some code already, but basically the design and implementation is up to you. An instance of class `A7GUI` provides the `JFrame` on which everything is written, and class `A7Panel`, which is placed in the `JFrame`, is where the image for a topic is drawn. You don't have to know anything about `A7GUI` and `A7Panel` except how to use them, but we suggest you spend some time looking at them.

In developing this program try to make progress in a way that allows the program to get more and more functionality, little by little, making sure that everything is correct as you go. If you don't maintain correctness of what you have done, errors will be hard to find.

When debugging, use `System.out.println` statements at judiciously chosen places to help you figure out what is right and wrong. You can use a JUnit testing class, but you don't have to.

As you design and develop, create "helper" methods, to keep all parts of the program simple. No method should be more than 20-25 lines long, for example. If a method is getting too long, break off some functionality into another method. Be sure to put specifications on all your methods before you write their bodies! We will not be lenient in grading methods whose specifications are not precise, clear, and thorough.

Here is how to to proceed.

1. Play with the completed program that we make available, so that you know exactly how it works from a user's perspective.

2. Study this handout, as well as a series of screenshots of this game being played. Then begin implementing pieces of this program until it is done, as outlined below. Do *not* go on to another part until you know that this part works correctly.

3. The `A7` constructor does nothing. If this program is called using `A7.main(null);`, procedure `main` is called. If called from a webpage, the constructor is called and then procedure `A7.init` is called. Look at how similar `main` and `init` are. They (1) get a buffered reader for file topics.txt, (2) call method `readTopicsFile` to read the file into `inputFile` (we give you this method), and (3) create an instance of the JFrame A7GUI.

Your first step should be to complete function `fillItems`. After this is written, when you do `A7.main(null);` in the DrJava interactions pane, the GUI window should open and you should be able to select topics in the drop-down topics list.

4. Next, write the body of procedure `startTopic`, which is called to start a new topic. In the body, we have given you hints to indicate what it should do. Proceed in small steps --we don't list all of them here but just the more important ones.

4A. Work on class `Item`, putting in the necessary fields and getter/setter methods and writing function `toString`, so that you can use it when debugging. Write the class invariant, and put javadoc specs on all methods. Don't write everything at once, write just what you need to start building the dictionary, and complete more pieces when the need arises.

4B. You will need to write a method that retrieves the lines for the gives topic and stores the dictionary for the topic in field `items`. This is fairly complicated because you need to break each line into its pieces: the term and points. Here are important points:

- The term may consist of several words --all the words before the first one that starts with a digit.
- The elements of field `items` should be sorted based on their points, as explained elsewhere. Method `Item.sort` can be used for this. Don't fill in the number field of the items in until `items` has been sorted.
- **Do not proceed past this point until you are certain that this method is correct. A mistake here can be disastrous later on**.

4C. Once field `items` is correctly constructed, you can see about placing the numbers and terms in the two drop-down menus.

4D. Finally, when the image is drawn, you should also see the lines and numbers. This requires writing function `drawLinesLabels`.

5. As the last part to write, which is quite easy compared to the rest, write the body of `associateTerm`, which is called by the GUI when the player click buton *OK*.

Grading

Here are some general outlines and explanatory notes:

Correctness: Your program should perform as specified. Correctness, as measured by our tests, will count for the largest single portion of your grade.

Design and organization: While we have specified the methods that communicate between the GUI and class `A7`, and we have provided a few other methods, there is quite a lot for you to organize yourself. You will need to write several helper functions. Modularity and clarity counts. If your code is a mess, you will lose points.

Commenting: Each method you write must have a suitable javadoc specification. Within methods, the more complicated parts of your code should also be described using comments.

Reuse: If you find yourself repeating a task, add a helper method and call that method instead of duplicating the code.

Submitting your work

At the top of file `A7.java`, put a comment that gives your name(s), netid(s), and the time you spent on this assignment. Submit files `A7.java` and `Items.java` on the CMS.