ASSIGNMENT 6    Data Structures
CSI310
Spring 2011

DUE: Early: Wed., March 2 at 11:59PM. Regular: Fri., Mar. 4, 11:59PM. Upload all your .java files for this project as SEPARATE attachments. Don’t submit an archive as this will delay the grading operation! Also, informatively named identifiers, nice indentation and appropriate use of methods like copyToFromWhere will count!

This assignment builds on build on your work for assignment 5. It must continue to use the linked list data structure internally. Specifically, the design should use 3 classes:

1. The class to contain the application; that class has the public static void main(String[]) method.

2. The PictureList class that is the blueprint for the whole list of Pictures and methods appropriate for working with the whole list.

3. The class that is the blueprint for zero or more linked list nodes or elements, each one referring to one Picture and also to either the next node or to nothing.

You must add the capacity for cutting, copying or pasting at any of the positions in the list, then redisplaying the combinations, and repeating this edit cycle any number of times. You must therefore also implement a “cursor”, familiar from word processing software, to indicate the specific place where your edit commands take place. Two commands to move the cursor must be implemented: Home, to move it to the left-most position; and forward, to move it one space to the right.

You need to add to the PictureList class two more instance variables (besides refToFirstElement and refToLastElement): One to implement the cursor and the other to implement a “copy buffer”. When one Picture is copied or cut, that Picture is kept available by having its location stored in the copy buffer instance variable so that it can be pasted one or more times into various list positions. As in familiar word processing software, when a new copy or cut operation is commanded, the old Picture in the copy buffer is replaced by the one that was just copied or cut. To clarify further, cut and copy do the same thing with putting the Picture at the cursor into the copy buffer; the difference is that the cut command also deletes it from the displayed combination.

The program should begin by working much like the Assignment 5 program: Interact with the user to import any number of images; where the user will not be asked how many images at the beginning. When the user indicates no more, the program should display ONE of the three top, centered or bottom justified combination required for Assignment 5.

When the combination Picture is on display, the user must be prompted as below to type one of the listed commands.

Command: home forward copy paste cut done ?

Notice that paste and cut are the only commands that can actually change the sequence of Pictures in the combination. Whenever the sequence changes, the program must call hide() on the old combination, make a new combination Picture and call show() on that new Picture. The edited combination appears instead of the previous one. TURN OVER!
There are three troublesome details\(^1\) and solving the problems caused by them is part of the assignment. They will be discussed in class and you are encouraged to discuss solution strategies on the discussion board, but remember: Don’t post code.

1. At the beginning, and until the first \texttt{cut} or \texttt{copy} command has been successfully performed, there is no \texttt{Picture} in the copy buffer to \texttt{paste}. Your implementation of the \texttt{paste} command must detect that situation, not crash, print a friendly and informative message, and continue handling new commands.

2. Suppose the list has say 3 items. There are then 4 positions for the cursor!

   When the cursor is at the beginning of the list, the \texttt{Picture} to be \texttt{cut} or \texttt{copied} lies just to its right. That one is the first \texttt{Picture} in the list. When a \texttt{Picture} is \texttt{pasted} at the beginning, it becomes the first and all the \texttt{Pictures} already in the list move forward one spot. (Of course, no real “moving” is done. The values of the \texttt{refToFirstElement} and some \texttt{nextElement} variables are changed to implement changing the order of the \texttt{Pictures}.)

   When the cursor is between two \texttt{Pictures} in the list, \texttt{copying} and \texttt{cutting} apply to the \texttt{Picture} to its right. \texttt{Pasting} will make that \texttt{Picture} to the right of the cursor “move one spot right” to make room for the copy pasted in!

   When the cursor is at the end of the list, there is no \texttt{Picture} to \texttt{cut} or \texttt{copy}. Your implementations of \texttt{cut} and \texttt{copy} must detect that situation, not crash, keep the original contents of the copy buffer (if any!), print a friendly and informative message, and make the program continue to handle new commands.

   The \texttt{paste} command given when the cursor is at the end of the list causes the \texttt{Picture} in the cut buffer to be copied to the end of the list, so the previous \texttt{Pictures} remain in place.

   (Clearly, 3 is just an example of a number. The program MUST work properly for ANY number of \texttt{Pictures}. Also, the same \texttt{Picture} might be in the list several times because of \texttt{pasting}.)

3. The list might become empty. The program must continue to work. In that case, the user can command \texttt{paste} one or more times and so see one or more of the same \texttt{Picture}.

   Note: You may use a GUI (Graphical User Interface) in addition to the text interface. No extra credit for GUIs though, but your friends might be more impressed! We will use the text interface to test and grade your program.

   \textbf{Real Extra Credit:} Don’t even think about this until you have all the regular credit functionality working and tested perfectly, in all cases including 1 \texttt{Picture}, 0 \texttt{Pictures} remaining in the list, etc.

   Make it happen that the \texttt{Picture} to the right of the cursor is somehow highlighted, bordered, marked with an X, etc. so the user can see where the cursor is at all times. Also, try to invent a way of indicating that the cursor is at the end. (If no mark is shown at all when the cursor is at the end, extra credit will still be awarded.)

   Two difficulties are that the program must restore the display of a \texttt{Picture} to the original when the cursor moves off it, and that the same \texttt{Picture} can be at several positions in the list at one time. The cursor will mark only one position.

\(^1\)Terminology that might be on a CSI310 exam or a job interview: Software designers call individual detailed situations in using a program separate \textbf{use cases}. Part of their job is to explicitly write out use cases. With the use cases written out, it is easier to design what the program should do in each case, and then check that the proposed design and eventually the code will actually do it. \textbf{WEB SITE designers TOO!}