Software Design Requirements (draft)

You must keep all your work for Project 2 in a single, self-contained directory (which might have subdirectories). Remember that the first thing the Project 2 program must do is ask your customer how many images he/she wants to import.

You must keep all your work for Projects 3 and 4 in a different, single, self-contained directory (which might have subdirectories). You should copy the LineCollageMaker.java and LineCollage.java file from the Project 2 directory to the Project 3-4 directory when you’ve finished Project 2 and move on to Project 3. Start Project 3 by modifying LineCollageMaker.java and LineCollage.java so the images are input until the user says to stop and the two classes work (TEST successfully!) together.

You must define two classes for Project 2 (the array version) and three classes for Projects 3 and 4 (the linked list version):

1. The application container class with the main() method. This should be named LineCollageMaker.java because the software enables your customer to make a linear collage.

2. One class whose one instance will manage the ENTIRE collection of Picture objects. Since the software manipulates a linear collage, name that class LineCollage, for the current collage that it is manipulating. The “singleton design pattern” means that a class is instantiated only once per program run.

3. Project 2 will also use a .java array of Pictures, and Project 3-4 require you write a third class beyond LineCollageMaker and LineCollage. In Java, arrays are a kind of class built into the language so you don’t define them explicitly.

   **Project 2:** The one instance (“singleton pattern”) of LineCollage will use (by “having” or “containing”) a Java array of G&E Picture objects. It will also have an int instance variable to track the number of Picture objects that are currently in the collage.

   **Projects 2-3:** The one instance (“singleton pattern”) of LineCollage will use the linked list data structure to store the Picture objects currently in the collage. The functional requirement in Project 3 that involves the image near the middle implies that LineCollage will also have an int instance variable to track the number of Picture objects that are currently in the collage.

To implement the linked list, Projects 2-3 must have a third class. Each instance of the third class will function as a linked list node. You can fashion this third class after the House class from my lecture on equipping the Houses in the Mad Ph.Ds neighborhood with blackboards on which to write the address of a friendly neighbor. You can, equivalently fashion it after Guzdial and Ericson’s PositionedSceneElement class of Chapter 7 or their SongNode class of Chapter 6. For simplicity’s sake, I will name this linked list node class Node.

(Another options are to (1) modify, or (2) extend G&E’s Picture class so each instance contains a reference to the the next instance. We didn’t yet cover extending a class, which is also called inheriting from a class. If you modify Picture.java, you MUST submit your modified version along with your other .java files when you upload your work for grading.)

**Details for Project 2–Array Version**

The LineCollageMaker class should be fashioned as follows:

```java
import java.util.Scanner;
class LineCollageMaker {
```
public static void main(String a[]) {
    LineCollage myCollage;
    Scanner uiInput = new Scanner(System.in);

    System.out.println("How many images?");
    int countdown = uiInput.nextInt();
    myCollage = new LineCollage(countdown);
    while( countdown > 0 )
    {
        Picture pin = new Picture(FileChooser.pickAFile());
        myCollage.addPictureAtEnd(pin);
        countdown--;
    }
    Picture firstToShow = myCollage.makeCollage();
    firstToShow.show();
    //YOU Code the user inteface loop and dispatch to methods
    //of myCollage here..
}

The LineCollage class should be fashioned as follows:

class LineCollage
{
    private Picture myArray[]; //To student: it starts out as a null reference.
    // No field for the array length is needed because
    // the length is available with myArray.length.
    private int nPictures;
    private Picture clipboard;
    public LineCollage(int n)
    {
        myArray = new Picture[n]; //To student: The array is CONSTRUCTED here.
        nPictures = 0;
    }
    public void addPictureAtEnd(Picture aPictureReference)
    {
        myArray[nPictures] = aPictureReference;
        nPictures++;}
    public Picture makeCollage()
    {
        int collageHeight;
        int collageWidth;
        // Figure out height and width..YOU code!!
        // This should be done HERE, not in addPictureAtEnd, because
        // code to be added will EDIT the Picture sequence, which changes
        // what the height and width should be!
        collageHeight = 100; //TEMPORARY CODE SO WE CAN COMPILE & TEST
        collageHeight = 100; //THE PROGRAM SO FAR
Picture retval = new Picture(collageHeight, collageWidth);
// compose the Pictures into retval..YOU code!!
    return retval;
}
  //add others yourself..
}

Note: You may use a GUI interface to ask the user yes or no, if perhaps you know how to program a GUI facility better than using the Scanner class. (In that case, omit my import statement but you might need to put in a different one.) No extra credit for GUIs though, but your friends might be more impressed!