

COSC 1P02 Fall 2012

Lab Exam Procedures

Date

Friday, Nov. 30, 2012

Time

Lab 1 (R, 10:00-12:00, Stephen)	2:30
Lab 2 (R, 14:00-16:00, Ethan)	2:30
Lab 3 (T, 10:00-12:00, Dave)	9:30
Lab 4 (W, 12:00-14:00, Stephen)	2:30
Lab 5 (F, 12:00-14:00, Kelly)	12:00
Lab 6 (F, 10:00-12:00, Kelly)	9:30
Lab 8 (R, 18:00-20:00, Earl)	12:00
Lab 9 (W, 18:00-20:00, Kyle)	12:00

Location

D 205 (note this is **not** the 1P02 lab room)

Instructions

You will be randomly assigned **two** of the lab exam questions during your exam. To prepare, you should become familiar with all problems by coding them, and subsequently executing them.

During the lab exam, you will not be allowed any electronic media (diskettes, USB drives etc), personal notes, or network access. You will log on to a special exam userid and you will not have access to your Z : drive. You may bring a pen or pencil and paper will be supplied if needed. Do not bring anything else to the lab room for the exam.

In the question script you will be given specifications, including methods to write and their parameters. These should be followed and implemented for full marks. Solutions that produce the required result but do not follow the directions will be penalized. Part marks are awarded for correct partial implementations. You do not need to comment your code. Execution efficiency is not a concern, however proper use of the Java constructs we have discussed is expected.

You will be using the DrJava development environment. To aid you we will provide both the template for Java classes and the method summary for the classes we have discussed as attached here.

COSC 1P02 Class Template

```
package PackageName;

import Media.*;           // for Picture, PictureDisplay, etc.
import BasicIO.*;         // for ASCIIDataFile, etc. & BasicForm
import java.awt.*;         // for Color class
import static BasicIO.Formats.*; // for getCurrencyInstance, etc.
import static java.lang.Math.*; // for math constants and functions & random
import static java.awt.Color.*; // for Color constants (e.g. red)

public class ClassName {

    // Instance variable declarations

    public ClassName ( ) { // constructor

    }; // constructor

    // Method declarations

    public static void main ( String[] args ) {
        new ClassName( ).methodName( );
    }
}
```

COSC 1P02 Method Summary

Math

Method/Value	meaning
<code>E</code>	<i>constant</i> : the mathematical constant e
<code>PI</code>	<i>constant</i> : the mathematical constant π
<code>r = abs(x)</code>	returns the absolute value of x
<code>r = acos(x)</code>	returns the arc cosine of x
<code>r = asin(x)</code>	returns the arc sine of x
<code>r = atan(x)</code>	returns the arc tangent of x
<code>r = cos(x)</code>	returns the cosine of x
<code>r = log(x)</code>	returns the natural logarithm of x
<code>r = pow(a,b)</code>	returns a^b
<code>r = random()</code>	returns a random value between 0.0 and 1.0
<code>r = sin(x)</code>	returns the sine of x
<code>r = sqrt(x)</code>	returns the square root of x
<code>r = tan(x)</code>	returns the tangent of x

PictureDisplayer

method	meaning
<code>d = new PictureDisplayer()</code>	<i>constructor</i> : creates a new displayer with canvas 200x200
<code>d = new PictureDisplayer(pic)</code>	<i>constructor</i> : creates a new displayer with canvas to fit <i>pic</i> and with <i>pic</i> placed on displayer
<code>d = new PictureDisplayer(width, height)</code>	<i>constructor</i> : creates a new displayer with canvas of specified <i>height</i> and <i>width</i>
<code>d.close()</code>	wait until user presses Close button and close displayer
<code>d.placePicture(pic)</code>	place <i>pic</i> on the displayer
<code>d.waitForUser()</code>	wait until user presses OK before continuing

Picture

method	meaning
<code>p = new Picture()</code>	<i>constructor</i> : creates a picture object loading pixels from a file selected via a file open dialog
<code>p = new Picture(width,height)</code>	<i>constructor</i> : creates a picture object with specified <i>height</i> and <i>width</i> with all pixels white
<code>i = p.getHeight()</code>	returns height (in pixels) of picture
<code>q = p.getPixel(x,y)</code>	returns pixel in column x of row y
<code>i = p.getWidth()</code>	returns width (in pixels) of picture
<code>b = p.hasNext()</code>	returns true if another pixel is available
<code>q = p.next()</code>	returns the next available pixel
<code>p.save()</code>	present file save dialog to allow user to save picture as modified

Pixel

method	meaning
<code>i = q.getBlue()</code>	obtain blue color channel of pixel
<code>c = q.getColor()</code>	obtain color of pixel
<code>r = q.getDistance(<i>color</i>)</code>	returns the color distance between this pixel's color and <i>color</i>
<code>i = q.getGreen()</code>	obtain green color channel of pixel
<code>i = q.getRed()</code>	obtain red color channel of pixel
<code>q.setBlue(<i>i</i>)</code>	change the blue color channel of pixel to <i>i</i>
<code>q.setColor(<i>color</i>)</code>	change color of pixel to <i>color</i>
<code>q.setGreen(<i>i</i>)</code>	change the green color channel of pixel to <i>i</i>
<code>q.setRed(<i>i</i>)</code>	change the red color channel of pixel to <i>i</i>

Color

method	meaning
<code>red, green, ..., RED, GREEN, ...</code>	<i>constant</i> : standard colors
<code>c = new Color(<i>r,g,b</i>)</code>	<i>constructor</i> : creates a new color object with specified <i>r</i> , <i>g</i> and <i>b</i> components
<code>c = new Color(<i>value</i>)</code>	<i>constructor</i> : creates a new color object with color value (0-16,777,215)
<code>i = c.getBlue()</code>	returns blue value of color
<code>i = c.getGreen()</code>	returns green value of color
<code>i = c.getRed()</code>	returns red value of color

ASCIIPrompter

method	meaning
<code>d = new ASCIIPrompter()</code>	<i>constructor</i> : creates a prompter with default label
<code>d.close()</code>	closes prompter
<code>r = d.readDouble()</code>	waits for user to enter data and press OK, then reads data as a double and returns value
<code>i = d.readInt()</code>	waits for user to enter data and press OK, then reads data as an int and returns value
<code>s = d.readString()</code>	waits for user to enter data and press OK, then reads data as a String and returns value
<code>d.setlabel(<i>label</i>)</code>	sets the prompt label to <i>label</i>

ASCIIDisplayer

method	meaning
<code>d = new ASCIIDisplayer()</code>	<i>constructor</i> : displays window with a text area to display text
<code>d.close()</code>	waits for user to press Close and then closes displayer
<code>d.newLine()</code>	writes a line marker to the display so next output begins on next line
<code>d.waitForUser()</code>	waits for user to press OK before continuing
<code>d.writeDouble(<i>r</i>)</code>	writes the double value <i>r</i> to the display
<code>d.writeInt(<i>i</i>)</code>	writes the int value <i>i</i> to the display
<code>d.writeLine(<i>s</i>)</code>	writes the String value <i>s</i> to the display and positions to next line
<code>d.writeString(<i>s</i>)</code>	writes the String value <i>s</i> to the display

ASCIIDataFile

method	meaning
<code>f = new ASCIIDataFile()</code>	<i>constructor</i> : presents a file open dialog and opens text (data) file for input
<code>f.close()</code>	closes data file
<code>b = f.isEOF()</code>	returns <code>true</code> if last read failed because of EOF
<code>f.nextLine()</code>	skips rest of data on line so next read reads first field of next line
<code>r = f.readDouble()</code>	reads next field as a <code>double</code> and returns value
<code>i = f.readInt()</code>	reads next field as an <code>int</code> and returns value
<code>s = f.readString()</code>	reads next field as a <code>String</code> and returns value

ASCIIOutputFile

method	meaning
<code>f = new ASCIIOutputFile()</code>	<i>constructor</i> : opens a new text output file presenting a File Save dialog.
<code>f.close()</code>	closes file
<code>f.newLine()</code>	writes a line marker to the file so next field begins on next line
<code>f.writeDouble(r)</code>	writes the <code>double</code> value <code>r</code> as a field
<code>f.writeInt(i)</code>	writes the <code>int</code> value <code>i</code> as a field
<code>f.writeLine(s)</code>	writes the <code>String</code> value <code>s</code> starting immediately followed by line marker
<code>f.writeString(s)</code>	writes the <code>String</code> value <code>s</code> as a field

BinaryDataFile

method	meaning
<code>f = new BinaryDataFile()</code>	<i>constructor</i> : presents a file open dialog and opens binary data file for input
<code>f.close()</code>	closes data file
<code>b = f.isEOF()</code>	returns <code>true</code> if last read failed because of EOF
<code>r = f.readDouble()</code>	reads next 8 bytes as a <code>double</code> and returns value
<code>i = f.readInt()</code>	reads next 4 bytes as an <code>int</code> and returns value
<code>o = f.readObject()</code>	reads some number of bytes as an <code>Object</code> and returns object reference
<code>s = f.readString()</code>	reads some number of bytes as a <code>String</code> and returns value

BinaryOutputFile

method	meaning
<code>f = new BinaryOutputFile()</code>	<i>constructor</i> : opens a new binary output file presenting a File Save dialog.
<code>f.close()</code>	closes file
<code>f.writeDouble(r)</code>	writes the <code>double</code> value <code>r</code> as 8 bytes
<code>f.writeInt(i)</code>	writes the <code>int</code> value <code>i</code> as 4 bytes
<code>f.writeObject(o)</code>	writes the <code>Object</code> <code>o</code> as a sequence of bytes
<code>f.writeString(s)</code>	writes the <code>String</code> value <code>s</code> as a sequence of bytes

BasicForm

method	meaning
<code>f = new BasicForm()</code>	<i>constructor</i> : creates a new form with one default button (OK) that will size to the layout of the fields added.
<code>f = new BasicForm(button₁, button₂, ...)</code>	<i>constructor</i> : creates a new form with default buttons <i>button₁</i> , <i>button₂</i> ... that will size to the layout of the fields added.
<code>f.accept()</code>	presents the default button(s) and awaits user pressing a button
<code>i = f.accept()</code>	presents the default button(s), awaits user pressing a button and returns button number (from 0) of button pressed
<code>i = f.accept(button₁,button₂, ...)</code>	presents the specified <i>button</i> (s), awaits user pressing a button and returns button number (from 0) of button pressed
<code>f.clear(name)</code>	clears the field <i>name</i>
<code>f.clearAll()</code>	clears all fields in the form
<code>f.close()</code>	closes the form so that it cannot be used
<code>f.placePicture(name,picture)</code>	places <i>picture</i> onto the canvas <i>name</i>
<code>r = f.readDouble(name)</code>	reads and returns the field <i>name</i> as a double
<code>i = f.readInt(name)</code>	reads and returns the field <i>name</i> as an int
<code>s = f.readString(name)</code>	reads and returns the field <i>name</i> as a String
<code>f.setEditable(name,editable)</code>	makes field <i>name</i> <i>editable</i> (true) by user
<code>f.setTitle(title)</code>	sets title in title bar of window to <i>title</i>
<code>f.writeDouble(name,r)</code>	writes the double value <i>r</i> to the field <i>name</i>
<code>f.writeInt(name,i)</code>	writes the int value <i>i</i> to the field <i>name</i>
<code>f.writeString(name,s)</code>	writes the String value <i>s</i> to the field <i>name</i>

BasicForm Widgets

method	meaning
<code>f.addCanvas(name,label,width, height,x,y)</code>	adds a labeled canvas with <i>name</i> , <i>width</i> , and <i>height</i> at (<i>x</i> , <i>y</i>)
<code>f.addCheckbox(name,label,x,y)</code>	adds a labeled checkbox with <i>name</i> at (<i>x</i> , <i>y</i>)
<code>f.addLabel(name,label,x,y)</code>	adds a label with <i>name</i> at (<i>x</i> , <i>y</i>)
<code>f.addRadioButtons(name,label, vertical,x,y,button₁,...)</code>	adds labeled radio buttons with <i>name</i> , vertical or horizontal at (<i>x</i> , <i>y</i>) with button names <i>button_i</i>
<code>f.addSlider(name,label,min,max, size,x,y)</code>	adds a labeled slider with <i>name</i> and <i>size</i> over the range <i>min</i> to <i>max</i> at (<i>x</i> , <i>y</i>)
<code>f.addSound(name,label,width, height,x,y)</code>	adds a labeled sound play button with <i>name</i> , <i>width</i> and <i>height</i> at (<i>x</i> , <i>y</i>)
<code>f.addTextArea(name,label,rows, columns,x,y)</code>	adds a labeled text area with <i>name</i> , <i>rows</i> and <i>columns</i> at (<i>x</i> , <i>y</i>)
<code>f.addTextField(name,label, format,width,x,y)</code>	adds a labeled text field with <i>name</i> , <i>format</i> and <i>width</i> at (<i>x</i> , <i>y</i>)

ReportPrinter

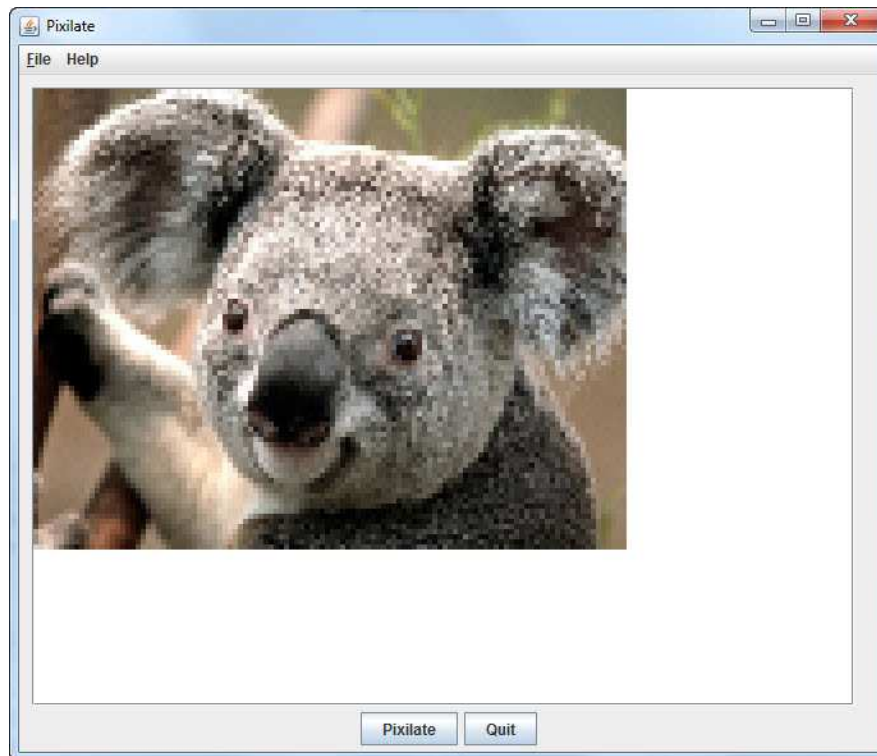
method	meaning
<code>p = new ReportPrinter()</code>	<i>constructor</i> : creates a new report
<code>p.addField(name, label, format, width)</code>	adds (L-R) a field to the report with <i>name</i> , <i>label</i> <i>format</i> and <i>width</i>
<code>p.close()</code>	closes report and prints it (print dialog)
<code>p.newLine()</code>	writes a line marker to the display so next output begins on next line
<code>p.newPage()</code>	forces next write to new page, printing header
<code>p.setTitle(line₁, line₂, ...)</code>	Sets title for page to specified <i>lines</i>
<code>p.writeDouble(name, r)</code>	writes the double value <i>d</i> to field <i>name</i>
<code>p.writeInt(name, i)</code>	writes the int value <i>i</i> to field <i>name</i>
<code>p.writeLine(s)</code>	writes the String value <i>s</i> starting at the next field and moves to next line
<code>p.writeString(name, s)</code>	writes the String value <i>s</i> to field <i>name</i>

Formats

method	meaning
<code>f = getCurrencyInstance()</code>	returns a format for currency (dollars)
<code>f = getDateInstance()</code>	returns a format for dates
<code>f = getDateTimeInstance()</code>	returns a format for date and time
<code>f = getDecimalInstance(n)</code>	returns a format for decimal number with <i>n</i> decimal places
<code>f = getIntegerInstance()</code>	returns a format for integer number
<code>f = getPercentInstance()</code>	returns a format for percentages
<code>f = getTimeInstance()</code>	returns a format for time

Question 1

Write a Java program that “pixilates” a picture. Pixilation occurs when a square block of pixels in the image is painted the same color. The color is the color of the pixel in the top left corner of the block. The process is performed on all blocks in the image. For example, with a block size of 2, each 2x2 block of pixels across and down the image are changed so that all 4 pixels are the same color as the pixel in the top left corner of the block. So pixels (0,0), (1,0), (0,1) and (1,1) are all painted the color of pixel (0,0); pixels (2,0), (3,0), (2,1) and (3,1) are all painted the color of pixel (2,0) and so on. In the following, the picture is pixilated with a block size of 4.



The program should load a picture and present it on a form (use a canvas size of 640x480). The first time the user presses `Pixilate`, the picture will be pixilated with a block size of 2. If the user presses `Pixilate` again, the image will be pixilated with a block size of 4, then 8 and so on. When the user presses `Quit`, the pixilated image is saved.

Use the following methods in your solution:

```
private void pixilate ( Picture aPic, int blockSize ) {  
    which pixilates the picture aPic using a block size of blockSize.
```

```
private void paintBlock ( Picture aPic, int x, int y,  
                           int blockSize, Color c ) {
```

which, in the picture `aPic`, paints the block of pixels of size `blockSize` and whose top left corner is `(x,y)` the color `c`.

For submission, use the file `koala.jpg` as input. Run the program pressing `Pixilate` twice (block size 4). Select `File/Print Image of Window...` and print the form as your output for submission. Press `Quit` and save the image as `Q1.jpg`.

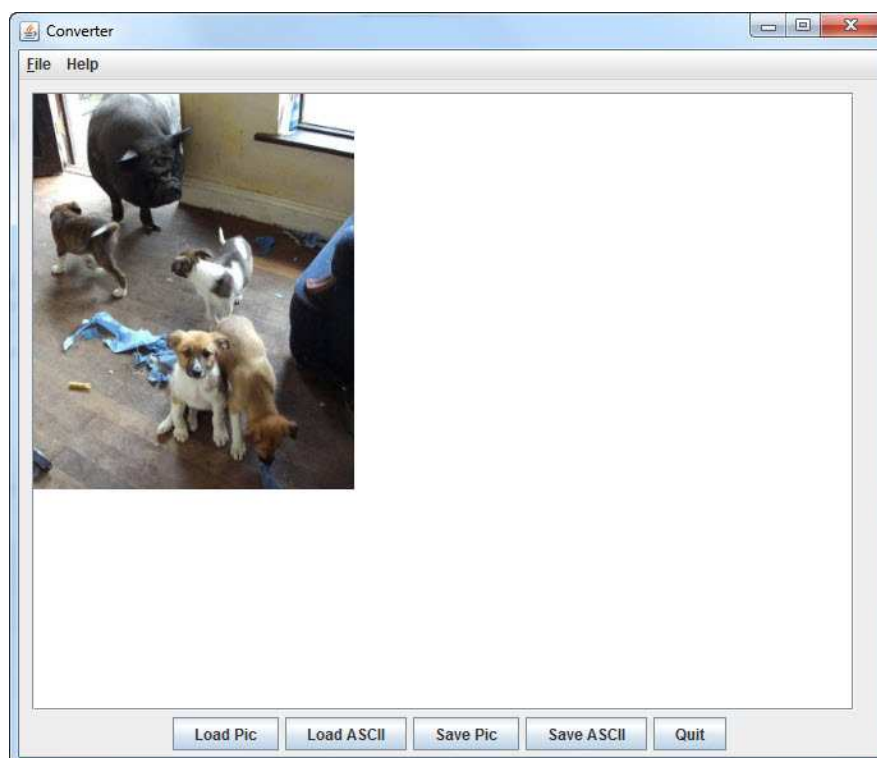
Question 2

Write a Java program that can store a picture as a text file (ASCIIOutputFile) and also load a text file (ASCIIDataFile) as a picture. A picture can be represented as text with one line containing the width and height (in pixels) and then one line per pixel, containing the red, green and blue color channel values of each pixel. For example, such a file might begin:

```
251  309
38   36   37
38   36   39
38   36   41
```

being a 251x309 pixel picture with the first three pixels on the top row being quite dark.

The program should display a form (with a canvass 640x480) such as:



When the user presses Load Pic, the program loads a picture (.jpg) file and displays it. When the user presses Save ASCII, the program converts the Picture on the display to text storing it to an ASCIIOutputFile. When the user presses Load ASCII, the program reads an ASCIIDataFile creating a Picture from it and displays the result. When the user presses Save Pic, the program saves the Picture on the display as a picture (.jpg) file. When the user presses Quit, the program terminates.

Use the following methods in your solution:

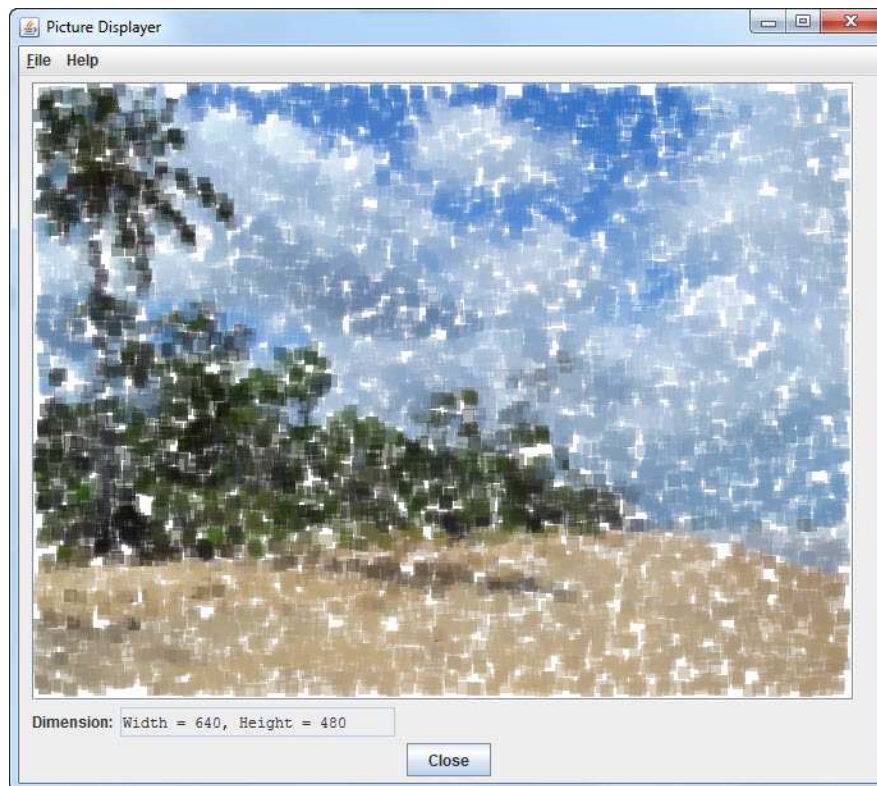
```
private Picture load ( ) {  
which opens an ASCIIDataFile loading and converting the text into a picture.
```

```
private void store ( Picture aPic ) {  
which opens an ASCIIOutputFile and converts aPic into text storing it to the file.
```

For submission, run the program, load the file animals.jpg and save it as ASCII as file Q2.txt. Now load the ASCII file jenny.txt and select File/Print Image of Window... and print the form as your output for submission.

Question 3

Write a program to convert a picture into an “impressionist” painting. For example, the mission beach image as an impressionist painting would look like:



The program should load a picture and then start with a blank painting (picture) of the same size. It should repeatedly (10,000 times) randomly select a pixel position (x, y) in the original picture. It should then paint the 10x10 block of pixels whose top left corner is (x, y) on the new picture with the color of the pixel chosen from the original picture. In painting the pixels in the new picture, the paint color should be blended with the color already on the pixel by averaging the two red, green and blue channel values, respectively. Care should be taken to avoid trying to paint beyond the boundaries of the new picture. When the user presses Close, the painting should be saved.

Use the following methods in your solution:

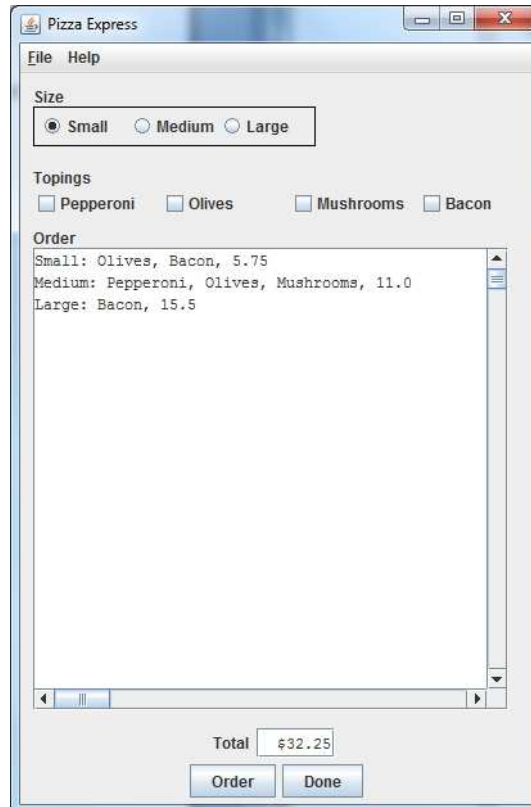
```
private Picture impression ( Picture aPic ) {  
    which produces an impressionist painting based on the picture aPic.  
  
    private void paintBlock ( Picture aPic, int x, int y, Color c ) {  
        which paints, within the picture aPic, the 10x10 block of pixels having position  $(x, y)$  as the  
        top left corner. The pixel colors should individually be a blend of their pre-existing color and the  
        color c.
```

For submission, load the mission_beach.jpg image and convert it to impressionism. Select File/Print Image of Window... and print the display as your output for submission. Press Close and save the picture as Q3.jpg.

Question 4

Pizza Express is a pizza restaurant. It sells pizzas in three sizes with a selection of 4 toppings. They need an application to allow customers to place orders for pizza.

The application should present a form as follows:



The screenshot shows a Java Swing window titled "Pizza Express". It has a menu bar with "File" and "Help". The main content area is divided into three sections: "Size" with three radio buttons (Small, Medium, Large), "Toppings" with four checkboxes (Pepperoni, Olives, Mushrooms, Bacon), and "Order" which is a text area containing three lines of text: "Small: Olives, Bacon, 5.75", "Medium: Pepperoni, Olives, Mushrooms, 11.0", and "Large: Bacon, 15.5". At the bottom, there is a "Total" label next to a text field showing "\$32.25", and two buttons labeled "Order" and "Done".

The user selects the Size and Toppings by clicking the radio buttons and checkboxes and then presses Order. The information about the pizza is written to the Order text area and the total price so far is displayed in the Total text field. Ordering continues, adding entries to the Order text area and displaying the updated total in the Total text field, until the user presses Done at which point the application terminates.

Use at the following method in your solution:

```
private double compute ( ) {
```

which computes the cost of the pizza as selected by reading the radio buttons and checkboxes.

The base price for a small pizza is \$5.00, a medium is \$10.00 and a large is \$15.00. The toppings cost: pepperoni: \$0.25, olives: \$0.25, mushrooms: \$0.50 and bacon: \$0.50.

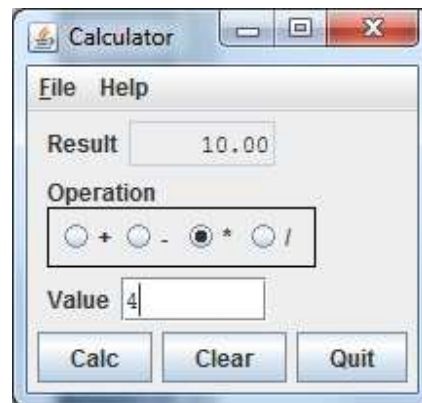
For submission, run your program to order a small with olives and bacon, a medium with pepperoni, olives and mushrooms and a large with bacon. Select

File/Print Image of Window... and print the display as your output for submission.

Press Done to quit.

Question 5

Write a program to provide a basic calculator. The calculator form would look like:



The calculator maintains a current result (initially 0 . 0) that is displayed in the (non-editable) `Result` text field. The user selects an operation using the `Operation` radio buttons (initially the + button is selected) and can enter a value in the `Value` text field. Pressing the `Calc` button performs the calculation based on the operation, displaying the result in the `Result` text area. In the above situation, the computation would be $10.00 * 4 = 40.00$ which after `Calc` is presses, would display 40 . 00 in the `Result` text field. If the `Clear` button is pressed, the result is reset to 0 . 0 (and displayed in the `Result` text area). If the `Quit` button is pressed, the application quits. Regardless of the button pressed, the `Value` text field is cleared before the form is redisplayed.

For submission, run the program and compute $10 * 4 + 13$ (to get the 10 . 00 initially, press `Clear`, enter 10 in the `Value` field, select + and press `Calc`). Select `File/Print Image of Window...` and print the display as your output for submission. Press `Quit`.

Question 6

Write a Java program to compute royalties due to authors of books published by Vanity Press for the book sales over the past year. The program should read an `ASCIIDataFile` of author data that contains one line per author giving: author name (string), price of the book (double) royalty rate (double), number books sold (integer) and number of books returned (integer). After computing the royalties, it produces a printed report (`ReportPrinter`), such as the following, with one detail line per author and two summary lines giving the total number of authors and the total royalties due to all authors:

Vanity Press Royalties Report						
Author	Price	Rate	Sold	Returned	Payable	Owed
Ima Brilliant	\$75.00	5%	120	15	\$393.75	
Douglas Knuth	\$125.00	6%	200	10	\$1,425.00	
Joe Author	\$65.00	5%	25	35		-\$32.50
Fred Flintstone	\$100.00	2%	300	200	\$200.00	
Number of Authors			4			
Total Royalties Due					\$1,986.25	

Books sold are the number of books shipped to bookstores (i.e. this is a wholesale business). However, bookstores may return unsold stock for a credit, this is the books returned. The royalties are based on the difference between the books “sold” and the books “returned”. Since the books may be “returned” in a different year than they were “sold” to the bookstore, it is possible for this difference to be negative (see Joe Author above). The royalties owed the author is the difference between the “sold” and “returned” books times the book price times the royalty rate. If this value is positive, it is written in the `Payable` field of the report, if it is negative, it is written in the `Owed` field. The number of authors is the number of authors processed regardless of the royalty amounts. The total royalties due is the straight sum of the royalties (positive or negative) owed the authors.

For submission, run the program using the file `authors.txt` as input. Print the report file generated as output for submission.