SANTA CLARA UNIVERSITY

Computer Engineering Department

COEN 160: Lab Assignments 2 and 3

Classes and Objects; Inheritance

Assignment: Complete problems 1 to 3.

Objectives: To learn to write user-defined classes and create objects from them; create subclasses and use method overriding; use the Model View Controller architecture for animation.

Notes:

- ❖ Include comments in your code so that it is easy to understand the program you have written.
- ❖ Indent your programs properly.
- ❖ Each source code file must contain a title and a description of its purpose.
- ❖ Select identifiers that are meaningful.

Details:

USER-DEFINED CLASSES

Problem 1

Color histograms of images are frequently used in techniques for image search and retrieval. The color histogram of an image can be constructed by counting the number of pixels of each color in that image. A *distance metric* is used to determine the similarity between images. One example of a simple distance measure is the *histogram euclidean distance*. This measure considers the difference between corresponding bins in two histograms to determine the similarity between images.

For example, suppose that an image has 3 pixels with the following colors:

Pixel 1: r = 250, g = 100, b = 5

Pixel 2: r = 250, g = 10, b = 5

Pixel 3: r = 250, g = 0, b = 80

The color histogram consists of three histograms, one for each component red, green and blue. Each histogram will have 256 bins to represent the range of values (0-255) that each sample can take in a picture. As the image has three red samples with a value of 250, bin 250 in the red histogram contains the value 3. This pixel colors are stored in the histograms as follows:

$h_{\text{Red}}[250] = 3$

$h_{\text{Green}}[0] = 1$

$h_{\text{Green}}[10] = 1$

$h_{\text{Green}}[100] = 1$

$h_{\text{Blue}}[5] = 2$

$h_{\text{Blue}}[80] = 1$

The remaining bins will contain the value 0. Next, suppose that another image has the following non-zero color histogram values: $g_{\text{Red}}[250] = 1$, $g_{\text{Green}}[0] = 5$, $g_{\text{Blue}}[15] = 2$. The histogram euclidean distance is calculated as:

$d^2 = (h_{\text{Red}}[250] - g_{\text{Red}}[250])^2 + (h_{\text{Green}}[0] - g_{\text{Green}}[0])^2 + (h_{\text{Green}}[10])^2 + (g_{\text{Blue}}[15])^2 +$ etc.

Write a class called *ImageSimilarity* that contains the following methods:

- A method to calculate the color histogram of an image. You can model each histogram as an array of size 256 in your program. The method uses three arrays, each of size 256, to store the count of the red, green and blue pixels in an image.
- A method to calculate the histogram euclidean distance *d* between two images whose histograms are *h* and *g* using the following formula:

$$d^2(h, g) = \sum_{i=0}^{255} (h_{red}(i) - g_{red}(i))^2 + (h_{green}(i) - g_{green}(i))^2 + (h_{blue}(i) - g_{blue}(i))^2$$

Here, $h_{\text{red}}(i)$ represents the number of pixels in the *i*th bin of the red component of image *h*, and $g_{\text{red}}(i)$ represents the number of pixels in the *i*th bin of the red component of image *g*.

1. Add any fields that you think are necessary to this class.
2. Specify the visibility for all fields and methods.
3. Implement the methods in the class.
4. Test your class with different images inside a *main* method. (See Canvas for link to free media.)

INHERITANCE

Problem 2

a) Write an abstract class called *Fish* that contains the following two abstract methods:

```
abstract void displayInformation();

abstract void drawShape(Graphics2D g);
```

(a) Select any two fishes that you want to use in your program, and create a class for each of them as a subclass of *Fish*.

(b) Add fields such as *type*, *size*, *weight*, *interesting Facts*, and any other that you think are needed. In addition, include accessor and mutator methods for each field.

(c) In each subclass, override the *displayInformation* method of *Fish* to print out any interesting facts on the console.

(d) Implement the abstract *drawShape* method in the subclasses to draw a fish of a particular type in a window. You may use graphics shapes or load a picture from a file. (See Canvas for link to free media.)

(e) For example, suppose that you write classes *Shark* and *ClownFish*. Write a *main* method to test your program and verify polymorphic behavior as follows:

```
public static void main(String [] args) {

   DrawingKit dk = new DrawingKit();

   Graphics2D myGraphics  = dk.getGraphics();

   Fish f;

   f.drawShape(myGraphics);  // a graphics image or picture of a shark
is displayed

   f = new ClownFish();

   f.displayInformation();  // print information about clown fish

}
```

Problem 3

In this problem, you will use the Model, View, and Controller classes to do animation.

a) Recall that the *Vehicle* class was developed in Chapter 6. The code for this class is in the inheritance package on the CD-ROM. Create a new subclass of *Vehicle* called *Ship*. Add a constructor to this class.
b) Also, add a method called *drawShape*, with the following declaration:

```
public void drawShape(Graphics2D myGraphics);
```

This method draws the shape of a ship. Use your imagination to draw the ship using Java 2D classes or choose a picture of a ship.

c) Write a *main* method to test your class.

Part 2

d) Add animation to the *Ship* class. To do so, include a new method called *step*:

```
protected void step(Graphics2D g2)

   // your code to change the (x, y) coordinates of the ship

}
```

Implement this *step* method so that the ship will move right to left across the window.

e) Write a *main* method in which you create an instance of *Ship*, *Controller*, and *View* classes. Pass this instance of *Ship* to the *Controller* and *View* classes. Run your program along with the classes in the *src/com/programwithjava/animation* package.

Submission: You must submit the source code of your programs and the output when you run the program. Demonstrate that your program runs correctly to the teaching assistant.