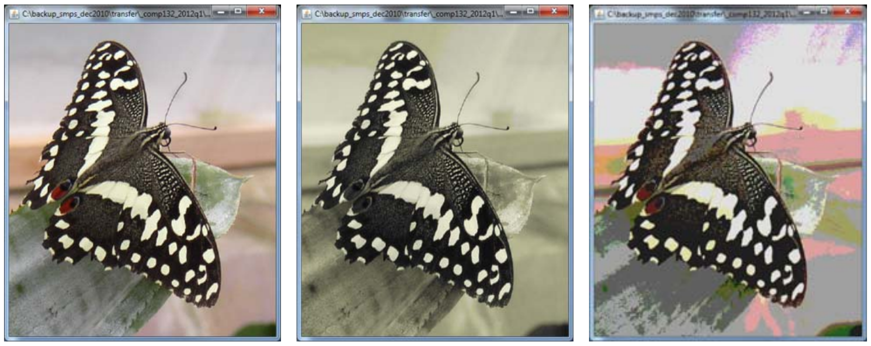
Module IV: Conditionals and Special Image Effects

Learning Objectives

- How to use conditionals with 2 possibilities.
- How to sepia-tint a picture.
- How to test for values in ranges.
- How to posterize an image.

Introduction

Sepia-toned pictures are interesting. They have a yellowish tint, and are used to make images looks old and "western". The image on the left shows the original picture the one on the right shows the sepia-toned one and the one on the right posterized.



The idea behind sepia-toning an image is to give a more yellowish tinge. A sepia-toned image is like a grayscale image, except that it is more of a "yellow-brown" scale. Here's a simple algorithm for making an image sepia-toned.

1. First, make the image grayscale
2. Loop through the pixels in the image
    a. Change the shadows (darkest grays) to be even darker (0 <= red < 60)
    b. Make the middle grays a brown color (60 <= red < 190)
    c. Make the highlights (lightest grays) a bit yellow (190 <= red)
        a. Increase red and green
        b. Or decrease blue

Hmm... There are 3 separate cases that we need to consider here. We've covered the if statement, but that only let us execute some statements if one condition was met. How do we deal with multiple conditions? Well, the next section will show you.

Multiple If Statements and Sepia-Toning

If we are doing different things based on a set of ranges, we can use multiple if statements. For example, if we want to do one action when (0 <= x <= 5), another set of actions when (5 < x <= 10) and yet another set of actions when (10 < x), this is best done with multiple if statements. The table below shows pseudocode and how it compares with the actual syntax for using multiple if statements.

Pseudocode	Actual Code
if x lies between 0 and 5 inclusive	**if (0 <= x && x <= 5) {**
do action 1	**action_1();**
	}
if x lies between 5 (non-inclusive) and 10 inclusive	**if (5 < x && x <= 10) {**
do action 2	**action_2();**
	}
if x is greater than 10	**if (10 < x) {**
do action 3	**action_3();**
	}

Notation... **Inclusive and exclusive ranges**

When a range of values is defined as "between 5 and 10, inclusive", what this means is that any value between 5 and 10, *including* 5 and 10 is valid. In Java this would be written as (5 <= x && x <= 10)

Similarly, if a range of values is defined as "between 5 and 10, exclusive", this means that any value between 5 and 10, but not including 5 and 10 is valid (i.e. 6, 7, 8, 9 are valid). In Java this would be written as (5 < x && x < 10)

So, a simple way to remember how to do inclusive and exclusive ranges in Java is to add an equal (=) sign if it is inclusive, and not add it if it is exclusive.

Now, the code above works great, but if our ranges are *disjoint*, that is, they do not overlap, then we spend extra time going through all the if statements, even if we already found our range and executed the correct statements.

The ranges that we talked about above are disjoint, and so, if x was 4, then we wouldn't need to check if x was greater than 5 (the second range) or greater than 10 (the third range). Since the ranges are disjoint, if we executed the statements in the first range, we would not need to care about the rest of the ranges, as we know that there is no way one value can fall between two or more disjoint ranges.

So, is there a way to "short-circuit" our evaluations of the conditionals? What we want is to be able to check each conditional, and if it evaluates to true, execute the statements in that block and not evaluate any more conditionals relating to that range. As it turns out, we can use the else statement, coupled with the if statement to form the else if statement.

The syntax for an else if statement is as follows:

```
if (expression1)
  statement1
else if (expression2)
  statement2
else if (expression3)
  statement3
  :
  :
else
  code that does not fit any other range
```

Let's also take a look at our previous example, now with the else if statement:

Pseudocode	Actual Code
if x lies between 0 and 5 inclusive	**if (0 <= x && x <= 5) {**
do action 1	**action_1();**
if x lies between 5 (non-inclusive) and 10 inclusive	**} else if (x <= 10) {**
do action 2	**action_2();**
if x is greater than 10	**} else {**
do action 3	**action_3();**
	}

If you look at the actual code, you'll notice an interesting side effect of using else if. Where before, we needed to check that x was greater than 5 AND less or equal to 10 (if (5 < x && x <= 10)), now we can leave out the part that checks if x is greater than 5, as if x was NOT greater than 5, then the code in the first if block would have executed. Similarly, we can leave out the conditional completely for the case where x is greater than 10, and replace that with an else statement, since if x was NOT greater than 10, one of the previous two statement blocks would have executed.

did I get this

Given the following code fragment, what is printed on screen?

```
int x = 6;

if (x < 5)
      System.out.println("A");
else if (x < 10)
      System.out.println("B");
else
      System.out.println("C");
```

solution

The first if block checks if x is less than 5, of which x is not. Then the second block executes, if x is less than 10, which it is. Thus, B is printed to the screen.

Given the following code fragment, what is printed on screen?

```
int x = 6;

if (x < 10)
      System.out.println("A");
if (x < 20)
      System.out.println("B");
else
      System.out.println("C");
```

solution

The first if block checks if x is less than 10, which it is. Thus 'A' is printed to the screen. Subsequently, since there isn't an else if block, the second if block is executed when x is less than 20, thus printing 'B' to the screen. Thus the overall output is 'AB'.

Back to sepia-toning

Now that we know how to deal with multiple conditionals and execute different statements for each, let's build the sepia-toned method!

The code below shows the completed code for the sepiaTint method.

```
/**
 * This method applies a sepia tint to an image
 */
public void sepiaTint() {
   Pixel pixelObj = null;
   double redValue = 0;
   double greenValue = 0;
   double blueValue = 0;

   //first, change the picture to grayscale
   this.grayscale();

   //loop through the pixels
   for (int x = 0; x < this.getWidth(); x++) {
      for (int y = 0; y < this.getHeight(); y++) {
         //get the current pixel and color values
         pixelObj = this.getPixel(x, y);
         redValue = pixelObj.getRed();
         greenValue = pixelObj.getGreen();
         blueValue = pixelObj.getBlue();

         //tint the shadows darker
         if (redValue < 60) {
            redValue = redValue * 0.9;
            greenValue = greenValue * 0.9;
            blueValue = blueValue * 0.9;
         }
        //tint the midtones light brown by reducing blue
         else if (redValue < 190) {
            blueValue = blueValue * 0.8;
         }
        //tint the highlights a light yellow by reducing blue
         else {
            blueValue = blueValue * 0.9;
         }

         //set the colors
         pixelObj.setRed((int) redValue);
         pixelObj.setGreen((int) greenValue);
         pixelObj.setBlue((int) blueValue);
      }
   }
}
```

Now, to test our method out, we can do the following:

```
Picture p = new Picture("<directory>");
p.sepiaTint();
p.repaint();
```

Posterizing an image

We've covered one image effect, now, let's work on another one. Posterizing an image involves reducing the number of different colors in an image. What we want to do is to set all values in a range to one value (the midpoint of the range). For example, we set all values that are less than 64 to 31, all values between 64 and 128 to 95, all values between 128 and 192 to 159, and every other value is set to 223.

Here's how the algorithm looks in pseudocode:

```
1. Loop through all the pixels in an image
    a. Get the red value for the pixel
        a. Find the right range and set the new red value
    b. Get the green value for the pixel
        a. Find the right range and set the new green value
    c. Get the blue value for the pixel
        a. Find the right range and set the new blue value
```

learn by doing

Given the algorithm for posterization, write a Java method that implements it. Use the following table to decide what color values to use.

Range	Color Value to Use
0 to 63	31
64 to 127	95
128 to 191	159
192 to 255	223

solution

```
/**
 * this method posterizes an image
 */
public void posterize() {
   Pixel pixelObj = null;
   int redValue = 0, greenValue = 0, blueValue = 0;

   for (int x = 0; x < this.getWidth(); x++) {
      for (int y = 0; y < this.getHeight(); y++) {
         //get the current pixel and color
         pixelObj = this.getPixel(x, y);
         redValue = pixelObj.getRed();
         greenValue = pixelObj.getGreen();
         blueValue = pixelObj.getBlue();

         //check that red values are set
         if (redValue < 64) {
            redValue = 31;
         } else if (redValue < 128) {
            redValue = 95;
         } else if (redValue < 192) {
            redValue = 159;
         } else {
            redValue = 223;
         }

         //check that green values are set
         if (greenValue < 64) {
            greenValue = 31;
         } else if (greenValue < 128) {
            greenValue = 95;
         } else if (greenValue < 192) {
            greenValue = 159;
         } else {
            greenValue = 223;
         }

         //check that blue values are set
         if (blueValue < 64) {
            blueValue = 31;
         } else if (blueValue < 128) {
            blueValue = 95;
         } else if (blueValue < 192) {
            blueValue = 159;
         } else {
            blueValue = 223;
         }

         //set the new colors
         pixelObj.setRed(redValue);
         pixelObj.setGreen(greenValue);
         pixelObj.setBlue(blueValue);
      }
   }
}
```

Once you are done, test it out with a picture of your choice!

Summary

In this module, we learnt:

- How to use if, else if and else to deal with more than 2 possibilities
    - We can add as many else if's as necessary
    - The syntax for if-else if-else is:

```
if (expression1)
   statement1
else if (expression2)
   statement2
else if (expression3)
   statement3
   :
   :
else
   code that does not fit any other range
```

- How to sepia-tint a picture
- How to posterize a picture