Lab 7: More on While Loops and Nested For Loops

Objectives:

1. To use while loops with more complex conditions.
2. To use nested for loops that change only part of a picture.
3. Using arrays and nested for loops to create a short slide show

Preparation:

1. Go over the Lecture Notes: Topic 7, Topic 8

Exercise 1: More on Using While Loops

a) While loops can be used when checking that a value entered by a user is in the correct range, and asking the user to enter the number again if it is not. The program in this exercise will prompt the user to enter a number between 1 and 100 inclusive. If the number entered is not in this range, the program will ask the user to enter the number again. The almost completed program is provided below. Fill in the blank space so that the body of the while loop is repeated until the value entered is in the correct range, i.e. between 1 and 100 inclusive. Compile and run your program, entering several invalid numbers such as 0, 200, -5, etc. before finally entering a valid number.

```
public class EnterCorrectInput
{
  public static void main(String[] args)
  {
     System.out.println("We want a number between 1 and 100
inclusive.");
          int num = 0;

     while ( _______________________ )
          {
          num = SimpleInput.getIntNumber("Enter number: ");
          }
       System.out.println("The number you entered is " + num);
  }
 }
```

b) Change the program so that it asks and checks for a number between 0 and 100 inclusive. Hint: you will need to change the initial value of `num`. Compile and run your program, entering several invalid numbers before finally entering a valid number. Note that it is always a good idea to check the "endpoints" of your loop by entering valid numbers that are right on the boundary, for example 0 and 100, and invalid numbers that are right on the boundary, for example –1 and 101.

Exercise 2: Manipulating Part of an Image

a) In Topic 7 we discussed vertical mirroring. Provided below is the code for the method `mirrorVertical()`. Add it to your Picture class, compile it and fix any errors.

```
public void mirrorVertical()
{
  int mirrorPoint = this.getWidth()/2;
  Pixel leftPixel = null;
  Pixel rightPixel = null;

  // loop through the rows
  for (int y = 0; y < this.getHeight(); y++)
  {
    // loop from column 0 to just before the mirror point
    for (int x = 0; x < mirrorPoint; x++)
    {  leftPixel = this.getPixel(x,y);
       rightPixel = this.getPixel(this.getWidth()-1-x, y);
       rightPixel.setColor(leftPixel.getColor());
    }
  }
 }
```

b) Type the following test program into the Definitions pane, saving it in the file *TestMirroring.java*. Compile and run it with your new Picture class, choosing the file *blueMotorcyle.jpg* in the mediaSources folder. Is the changed picture mirrored left to right, or right to left?

```
public class TestMirroring
{
  public static void main(String[] args)
  {
    String fileName = FileChooser.pickAFile() ;
    Picture pictureObj = new Picture(fileName);
    pictureObj.explore();
    pictureObj.mirrorVertical();
    pictureObj.explore();
  }
}
```

c) Now add a new method to your Picture class that does vertical mirroring **right to left**. The header for your method should be
`public void mirrorVerticalRightToLeft()`
(Hint: You do not need to change the `for` loop variables! Your new method should be exactly like the method mirrorVertical, except that the colour of the *left pixel* is now set to be the colour of the *right pixel*.) Compile your changed Picture.java and fix any errors.

d) Add the following statements to your TestMirroring program right after the statements that are already there:
`pictureObj.mirrorVerticalRightToLeft();`
`pictureObj.explore();`
Compile and run it, again choosing the file *blueMotorcyle.jpg* in the mediaSources folder. Did you see the *original* picture mirrored left to right, and then right to left? Why not?

e) Change your TestMirroring program so that it shows the result of invoking `mirrorVerticalRightToLeft()` on the *original* picture. Hint: you will need to create a new Picture object from the original file again.

Optional Challenge Exercise

Write a method called `mirrorDiagonal` that mirrors a square picture on the diagonal. Choose the diagonal from top left to bottom right . To decide on an algorithm, it's a good idea to start by taking a small example (say a 5 by 5 grid) and writing out the coordinates of all the source pixels and the coordinates of the target pixels. This will allow you to see the pattern to use for the nested for loop. (Hint: see Lab 6 Exercise 1 step 4 for one idea for a nested loop structure. But you may want to do it differently.) Test your method either in the Interactions pane or using a variation of the TestMirroring program of Exercise 4, using the image in *flower1.jpg* in the mediasources folder.

Exercise 3: Making a slide show

You are going to make a short movie which displays 4 different images for 2 seconds each. Movies are essentially sequences of pictures that are typically displayed at 24 "frames" per second, each frame being a single picture. In order to store a series of images, you will be make use of arrays of image objects.

a) Type the following code into the definitions pane. In order to store the pictures to use for the slide show, we are first going to create the array for five image objects, and prompt the user to select files for each of these images.

```
public class SlideShow
{
  public static void main(String[] args)
  {
     Picture[] pictures = new Picture[4];

     /* Add code here to do the following
      * for each picture:
      * Prompt user for an image file
      * Create a new picture object
      * Show the image using the show() method.
      */
  }
}
```

b) Next, you will make use of the FrameSequencer and MoviePlayer classes provided with our textbook in order to create and display a movie. The FrameSequencer class has methods that will take a sequence of pictures and store them in a directory; the pictures can then be displayed in rapid succession, thereby creating a “movie”. The MoviePlayer class has methods to play back a sequence of pictures already created (e.g. by FrameSequencer). Add this code to your program to create a FrameSequencer object that will store the movie frames in the directory referenced by a String variable directoryName, created from an array of pictures referenced by the Picture[] variable pictures:

```
FrameSequencer frameSequencer = new
FrameSequencer(directoryName);
for (int i = 0; i < pictures.length; i++)
{
    frameSequencer.addFrame(pictures[i]);
}
```

c) Once a movie has been made with FrameSequencer, we can use the MoviePlayer class to show the movie. The constructor for a MoviePlayer object takes as a parameter a String that is the name of a directory containing the images to be displayed in the movie. The playMovie() method opens a window in which it then displays the movie. So to play a movie, add the following code which creates a MoviePlayer object and invokes the playMovie() method.

```
MoviePlayer movie = new MoviePlayer(directoryName);
movie.playMovie();
```

Question: What is wrong with the movie when it plays?

d) Fix the movie by adding a nested for loop to the code from step (b)