Lab 5: Using Arrays and While Loops

Objectives:

1. To practice using one-dimensional arrays in Java.
2. To practice using while loops in Java.
3. To practice changing pixel colours.
4. To learn about collages for Assignment 2.

Preparation:

1. Go over the Lecture Notes: Topic 5, Topic 6
2. Textbook reading (optional): Chapter 4

Exercise 1: Loops and Arrays of Integers

a) In this part of the exercise, you will practice writing while loops.

1. In the Interactions pane, type the following:

```
int counter = 1;
while (counter <= 10 )
{
   System.out.print("*");
   counter = counter + 1;
}
System.out.println();
```

   How many asterisks are printed?

2. Reset the Interactions pane. Type statements that print exactly the same thing as the code from step 1, except start the counter at 0.

b) In this part of the exercise, you will practice initializing and working with an array of integers.

1. In the Interactions pane, type the following:

```
int[] nums = {1,2,3,4,5};
System.out.println(nums[0]);
```

2. How would you now print the rest of the items in the array `nums`?

3. As you probably noticed, this is somewhat tedious (and would get worse if the array contained more items!) We can use a while loop to print all the items in this array. Type the following statements, filling in the blanks with the appropriate numeric values:

```
int index = _____;
while (index < _____ )
{
   System.out.println(nums[index]);
   index = index + 1;
}
```

4. Recall that array objects have a public variable called `length`, which contains the number of items in the array. Type the following statements, filling in the blanks so that it prints all the items in the array:

```
index = _____;
while (index ___ nums.length )
{
   System.out.println(nums[index]);
   index = index + 1;
}
```

5. Now type the following statement:

```
System.out.println(nums[nums.length]);
```

   What happens? Why?

6. Now suppose we want to initialize an array `moreNums` of size 100 that contains the integers 1 to 100. It is impractical to do so by using a statement that lists all the values in the array, for example:

```
int [] moreNums = {1, 2, 3, 4, 5, 6, …
```

   We can do this by creating the array, and then using a loop in which we initialize each element with an assignment statement.

   Reset the Interactions pane and type the following statements, filling in the blanks. Hint: What integer should be stored in `moreNums[0]`? in `moreNums[1]`?

```
int[] moreNums = new int[ ___ ];
int index = _____;
while (index < ______)
{
   moreNums[index] = __________;
   index = index + 1;
}
```

7. Type the statements to print only the first 10 items in the array `moreNums,` using a while loop. If you did not get the numbers 1 to 10 printed, go back to step 6 and fix your initialization, and then try printing again.

Exercise 2: Changing the Color of Pixels

In the Interactions pane, make a variable `pictureObj` refer to a picture that contains the image in the file *caterpillar.jpg* that is in your mediaSources folder. (Refer back to Lab 4 Exercise 1 if you need to).

We have seen the following methods:

- `getPixel(x,y)` of the Picture class, which allows you to access one of the pixels in a picture
- `setColor(aColor)` of the Color class, which can be used to change the colour of a pixel.

We can use these methods to change the colour of the following pixels of the caterpillar picture to black:

(100,100), (101,101), (102,102), (103,103), (104,104), (105,105), (106,106)

One way to do this is to type the following sequence of Java statements in the Interactions Pane (note that you can repeat the previous line in the Interactions pane by using the up-arrow key, and then edit the line before hitting Enter for that line – this saves a lot of typing!)

```
import java.awt.Color;
pictureObj.getPixel(100,100).setColor(Color.black);
pictureObj.getPixel(101,101).setColor(Color.black);
pictureObj.getPixel(102,102).setColor(Color.black);
pictureObj.getPixel(103,103).setColor(Color.black);
pictureObj.getPixel(104,104).setColor(Color.black);
pictureObj.getPixel(105,105).setColor(Color.black);
pictureObj.getPixel(106,106).setColor(Color.black);
pictureObj.repaint();
```

How does the above sequence change the image?

Exercise 3: Using a Loop to Change the Color of Pixels

Setting the colour for a series of pixels as in the previous exercise is tedious. It is also repetitive, so we have an opportunity to use a loop construct here!

In this exercise, you will write a complete program that draws a line in a picture. Enter the following code in the Definitions pane, filling in the missing statements as specified by the comments within the code.

```
import java.awt.Color;
public class DrawLine
{
  public static void main (String[] args) {

    /* add code here to choose a filename of an image,
    create a Picture object from the image (referenced by
    the variable pictureObj), and then display the picture
    on the screen */

    int i = 0;
    while(i<25)
    {
       pictureObj.getPixel(100+i,100+i).setColor(Color.black);
       i++;
    }
    pictureObj.explore();
   }
 }
```

Save it as the file *DrawLine.java*, compile it, fix any errors, and run the program, choosing the file *caterpillar.jpg*   How does it change the image you got from *caterpillar.jpg*?

Try using the *Zoom* feature of the Picture Explorer (button in top left corner). You will be able to see individual pixels in lines in the image.

Exercise 4: Adapting Your Program to Change More Pixels

Now, add code to your `DrawLine` program to also draw a magenta vertical line of 25 pixels and a white horizontal line of 25 pixels in the caterpillar image, both starting at (100,100). Compile and run your modified program.

Exercise 5: Creating a Collage

A *collage* is a collection of pictures copied to some background image. The purpose of this exercise is to have you create a simple collage, in preparation for Assignment 2.

a) Download the file *Picture.java* from the Labs link (under Lab 5), open it in DrJava and compile it. It is a modified version of the *Picture.java* provided by the textbook, with an added method called `copyPictureTo()` which has the following header:

   ```
   public copyPictureTo(Picture sourcePicture, int xStart, int yStart)
   ```

   This method will copy the *source picture* that is passed in as a parameter, into the *target picture* (the picture object on which the method is invoked). The parameters `xStart` and `yStart` are the (x,y) positions at which to start the copy into the target picture. You will be invoking this method in steps c) and d) below.

b) Reset the Interactions pane. To create and view a “blank” picture (which will be the background for our collage, and which we will call the “canvas”), type the following sequence of statements in the Interactions pane:

   ```
   Picture canvas = new Picture(640,480);
    canvas.show();
   ```

c) In your mediaSources folder there are two images of flowers, each 100 pixels per side. You will be copying them onto the canvas, one at a time, using the `copyPictureTo()` method. First type in the following statements, choosing the file *flower1.jpg*. (Note that sometimes the FileChooser window comes up hidden behind DrJava.)

```
String fileName1 = FileChooser.pickAFile();
Picture picFlower1 = new Picture(fileName1);
canvas.copyPictureTo(picFlower1,100,0);
canvas.repaint();
```

d) Now type in statements to copy the image in *flower2.jpg* onto the canvas, at the position (300,0). Hint: first create a Picture object, as in step c), perhaps using the variable names `fileName2` and `picFlower2` for your reference variables.

e) Save the canvas to a file by typing the following statement:

```
canvas.write("Z:/collageLab5.jpg");
```

f) Check the image in your new JPEG file by opening it with the Picture Explorer (the `explore` method):

```
String fileName = FileChooser.pickAFile();
Picture pictureObj = new Picture(fileName);
pictureObj.explore();
```