AP® Computer Science A Picture Lab Teacher's Guide

The AP Program wishes to acknowledge and thank Barbara Ericson of the Georgia Institute of Technology, who developed this lab and the accompanying documentation.

Picture Lab: Teacher's Guide

Overview

The goal of this lab is to use the context of manipulating digital pictures to introduce two-dimensional arrays. These labs are based on Media Computation, which was originally developed by Dr. Mark Guzdial at Georgia Tech. Barbara Ericson, the author of these labs, and Dr. Guzdial have written several books using Media Computation. Dr. Guzdial developed Media Computation because many students use computers to communicate rather than calculate, and find the standard examples like sales tax calculation and temperature conversion uninteresting. Students are digital natives and have large collections of digital pictures, songs, and movies. Media Computation teaches computing concepts by having students write programs that manipulate digital media: pictures, sounds, text, and movies. Many students have found writing methods that manipulate pictures interesting, and these activities allow students to do open-ended and creative assignments.

Learning Objectives

In this lab students will practice traversing all and part of a two-dimensional array of objects. In addition, students will be introduced to program analysis, binary numbers, inheritance, and interfaces.

Correlation with AP Computer Science A Course Description

Topic Outline: The array topics covered are IV-D and V-A-1. Program analysis is III-C. Binary numbers are III-H-1. Inheritance and interfaces are part of III-D.

Language Features: Arrays are listed in item 14. Inheritance and interfaces are listed in items 27 and 28.

Prerequisites

Students should already be familiar with programming fundamentals such as variables, operators, conditionals, simple loops, methods, and parameters. It is also helpful if students have covered binary numbers and UML class diagrams. Two-dimensional arrays of integers are used in activity A4; you may want to introduce them before this lab or let this lab be your students' first experience with them.

What Is Provided

This lab provides two folders:

- `pixLab`: Supply each student with a copy of this folder, which contains two folders:
  - `classes`: the Java classes to which your students will add methods and some small pictures used by the `PictureExplorer` (`leftArrow.gif` and `rightArrow.gif`) as well as the javadoc documentation for the classes in a folder called `doc`.
  - `images`: images that your students can use.
- `teacherMaterials`: For your use only, this also contains two folders:
  - `finalClasses`: a copy of the classes folder but with the completed versions of the classes.
  - `images`: (identical to the contents of the `pixLab images` folder).

Do not give the `finalClasses` folder to your students! It is provided for teachers to help them guide the students.

Optional materials
A1: digital camera
A2: a CD (to demonstrate the prism effect)
A3: egg cartons and small candies (like Skittles or M&Ms)
A5: a photo negative (film)
A6: a rectangular mirror to help students understand the mirroring activity

Students should be encouraged to use their own pictures in these labs and add pictures to their `images` folder. If the pictures are very large they can be scaled using the `Picture` method `scale(double xPercent, double yPercent)`. For example, a picture can be scaled to one-fourth size using:

```
Picture p = new Picture("myPicture.jpg"); // create a picture called p
Picture smallP = p.scale(0.25,0.25); // creates a new picture a quarter as big
smallP.write("smallMyPicture.jpg"); // writes it to the images folder
```

Installation/Setup

Each student will need a copy of the `pixLab` folder. Students should have permission to read and modify the contents of that folder. You will need a Java Development Kit, also known as a JDK (see http://www.oracle.com/technetwork/java/javase/downloads/index.html). Students should also use a development environment. DrJava is a free development environment for Java that allows students to try out code in an interactions pane; it also has a debugger. It can be downloaded from http://drjava.org. However, you can use any development environment with this lab. Just open the files in the `classes` folder in your development environment and compile them. Please note that there are two small pictures in the `classes` folder that need to remain there: `leftArrow.gif` and `rightArrow.gif`. If you copy the Java source files to another folder, you must copy these gif files as well.

Students should keep the `images` folder and the `classes` folder together in the `pixLab` folder. The `FileChooser` expects the images to be in a folder called `images`, at the same level as the `classes` folder. If it does not find the images there it also looks in the same folder as the class files that are executing. If you wish to modify this, change the `FileChooser.java` class to specify the folder where the pictures are stored. For example, if you want to store the images in "`r://student/images/`," change the following line in the method `getMediaDirectory()` in `FileChooser.java`:

```
URL fileURL = new URL(classURL,"../images/");
```

And modify it to

```
URL fileURL = new URL("r://student/images/");
```

and then recompile `FileChooser.java`. Try running the `main` method in `PictureTester.java` and if the `fileURL` is correct, two pictures should appear.

Activity Overview

The lab consists of nine activities. The table below briefly describes each activity, lists its prerequisites, and suggests how much time to spend on it. **It is important to note that these activities can all be done consecutively, or they can be spread out over the year, as you cover the prerequisite topics.**

Activity #	Description	Prerequisites	Java files	Time
1	Intro to digital pictures and color	Binary numbers		20-60 minutes
2	Picking a color		`ColorChooser.java`	20-60 minutes
3	Exploring a picture		`PictureExplorer.java`	20-60 minutes
4 (optional)	2D arrays in Java	`for` loops, arrays	`IntArrayWorker.java` `IntArrayWorkerTester`	30-60 minutes
5	Modifiying a picture		`Picture.java` `SimplePicture.java` `DigitalPicture.java` `PictureTester.java`	1-2 hours
6	Mirroring pictures		`PictureTester.java`	1-2 hours
7	Mirroring part of a picture		`PictureTester.java`	1 hour
8	Creating a collage		`PictureTester.java`	1-3 hours
9	Simple edge detection		`PictureTester.java`	1-3 hours

Activity Outline

A1: Introduction to digital pictures and color

- Summary: This activity introduces students to how digital pictures are represented on a computer. It introduces the following concepts: megapixels, pixels, the RGB color model, binary numbers, bit, and byte.
- Time Estimate: 20 minutes to 1 hour (if you want to spend more time on binary numbers during this activity)
- Suggested Lesson Plan: Bring in a digital camera and/or have students look at an advertisement for a digital camera. Ask the students how they think a digital camera

works. What do they think is recorded when a picture is taken and how is it represented on a computer? Have the students answer the questions in the activity. Students can practice converting binary to decimal and decimal to binary at http://forums.cisco.com/CertCom/game/binary_game_page.htm.

A2: Picking a color

- Summary: Students practice making different colors using combinations of red, green, and blue light (using a `JColorChooser`). This activity introduces packages and the `java.awt.Color` class.
- Time Estimate: 20 minutes to 1 hour
- Suggested Lesson Plan: Use a CD to show the colors in white light. Have students follow the directions in the activity to use a `ColorChooser` object to make several colors. Show them a color chart that displays the hex code for each color (see http://www.webmonkey.com/2010/02/color_charts/). Have them convert the hex codes to decimal.

A3: Exploring a picture

- Summary: Students explore pictures using the Picture explorer tool to see the red, green, and blue values at each row and column location. Students are introduced to pixelation by setting the zoom to 500%. This activity also introduces (0,0) as the first pixel location in a picture.
- Time Estimate: 20 minutes to 1 hour
- Suggested Lesson Plan: Have students follow the directions in the activity to explore several pictures (including ones they download). Discuss fair use of copyrighted materials. Bring in egg cartons and small candies (Skittles or M&Ms) and have the students put the candies into different rows and columns in an egg carton to give them more practice with how rows and columns are numbered. For example, have them work in groups to place a red candy in row=3, col=2, a green candy in row=2, col=3, and a yellow candy in row=0, col=0, etc.

A4: Two-dimensional arrays in Java (Optional activity)

- Summary: Students practice writing methods for working with a two-dimensional array of integers. This activity shows how to use a nested loop to traverse all the rows and columns in a two-dimensional array. This activity introduces the concepts: row-major order, column-major order, array of arrays, and nested loops. It also gives examples of how to declare a 2D array, create a 2D array, get or set an element in a 2D array, and get the number of rows and columns in a 2D array.
- Time Estimate: 30 minutes to 1 hour
- Suggested Lesson Plan: Have students work in groups to write several methods for the `IntArrayWorker` class.

A5: Modifying a picture

- Summary: Students learn about UML class diagrams and the classes `DigitalPicture`, `SimplePicture`, `Picture`, `Pixel`, and `Color`. Students are introduced to interfaces, abstract methods, constants, and inheritance. Students write methods to modify all the pixels in a picture.
- Time Estimate: 1 to 2 hours
- Suggested Lesson Plan: Bring in an old negative to show the students what a negative was. If you haven't talked about UML much before now, this would be a good time to show several UML class diagrams. Have students try to create some class diagrams for a given context, like an online shopping cart of items or a trip to the movies. You might also want students to research better algorithms for creating a grayscale image.

A6: Mirroring pictures

- Summary: Students will write methods that loop through part of a two-dimensional array. This activity introduces the concept of an algorithm and the problem solving technique of simplifying a problem to make it easier to solve.
- Time Estimate: 1 to 2 hours
- Suggested Lesson Plan: Bring in a rectangular mirror to help students see what mirroring means. Have students walk through this algorithm on a small rectangle of numbers on the board. Give them several two-dimensional arrays of numbers and have them write what these arrays would look like after the method has run. Have the students work in groups to write up the algorithm for diagonal mirroring.

A7: Mirroring part of a picture

- Summary: Students learn how to determine the number of times statements inside of a nested loop are executed. Students also learn how to loop through just a range of rows and columns.
- Time Estimate: 1 hour
- Suggested Lesson Plan: Have the students do the activity. Give the students several examples of nested loops with bounds and have them calculate the number of times the body of the nested loop executes.

A8: Creating a collage

- Summary: Students will overload a method by adding additional parameters. Students will create an image collage by copying several pictures to a larger picture and modifying the pictures in some way. They will mirror the resulting collage.
- Time Estimate: 1 to 3 hours
- Suggested Lesson Plan: Have students complete the exercises. Encourage students to use pictures that are meaningful to them in the collage. Print the collages and display them around the classroom or have each student display their final collage and explain how he or she created it.

A9: Simple edge detection

- Summary: Students will work on methods to do simple edge detection.
- Time Estimate: 1 to 3 hours
- Suggested Lesson Plan: You can show videos from robot competitions and encourage students to discuss how robots can find and follow a ball or face using a color camera.

Answers to Questions and Exercises

A1Q1	8 bits
A1Q2	3 bytes for RGB, 4 if you include Alpha (transparency)
A1Q3	307,200 pixels

A2Q1–5 You can just pick the swatches to see what the red, green, and blue values are for each color. These are example values:
Pink (255,153,153)
Yellow (255,255,51)
Purple (204,51,255)
White (255,255,255)
Dark Gray (51,51,51) (all shades of gray have equal values for red, green, and blue)

A3Q1 0

A3Q2 0

A3Q3 639

A3Q4 479

A3Q5 top to bottom

A3Q6 left to right

A3Q7 Yes, you should be able to see the pixels since the zoom level is set so high.

A3E1–2 Check this by having the students show the pictures they are exploring or by having them turn in their changed `PictureExplorer.java` file and check the `main` method.

A4E1 This method is in `IntArrayWorker.java` and a test method is in `IntArrayWorkerTester.java` in the `finalClasses` folder.

```
public int getCount(int target)
{
  int count = 0;
  int current = 0;
  for (int row = 0; row < matrix.length; row++)
  {
    for(int col = 0; col < matrix[0].length; col++)
    {
      current = matrix[row][col];
      if (current == target)
      {
        count++;
      }
    }
  }
  return count;
}
```

A4E2 This method is in `IntArrayWorker.java` and a test method is in `IntArrayWorkerTester.java` in the `finalClasses` folder.

```
public int getLargest()
{
  int largest = Integer.MIN_VALUE;
  for (int row =0; row < matrix.length; row++)
  {
    for (int col = 0; col < matrix[0].length; col++)
    {
      if (matrix[row][col] > largest)
      {
        largest = matrix[row][col];
      }
    }
  }
  return largest;
}
```

A4E3 This method is in `IntArrayWorker.java` and a test method is in `IntArrayWorkerTester.java` in the `finalClasses` folder.

```
public int getColTotal(int col)
{
  int total = 0;
  for (int row = 0; row < matrix.length; row++)
  {
    total = total + matrix[row][col];
  }
  return total;
}
```

A5Q1 No, this class inherits it from `SimplePicture`.

A5Q2 Yes, `SimplePicture` implements the interface `DigitalPicture` and provides the method bodies for all the methods defined in `DigitalPicture`.

A5Q3 No, because `DigitalPicture` is an interface; an object of that type cannot be created.

A5Q4 Yes, a `SimplePicture` is a kind of `DigitalPicture`.

A5Q5 Yes, a `Picture` is a kind of `DigitalPicture`.

A5Q6 Yes, a `Picture` is a type of `SimplePicture`.

A5Q7 No, a `SimplePicture` is not a type of `Picture`.

A5E1 You should see the beach picture and another picture of the beach that looks yellow.

A5E2 Students should try out some of the other tests.

A5E3 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
public void keepOnlyBlue()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel pixel = null;
  for (int row = 0; row < pixels.length; row++)
  {
    for (int col = 0; col < pixels[0].length; col++)
    {
      pixel = pixels[row][col];
      pixel.setRed(0);
      pixel.setGreen(0);
    }
  }
}
```

A5E4 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
public void negate()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel pixel = null;
  for (int row = 0; row < pixels.length; row++)
  {
    for (int col = 0; col < pixels[0].length; col++)
    {
      pixel = pixels[row][col];
      pixel.setRed(255-pixel.getRed());
      pixel.setGreen(255-pixel.getGreen());
      pixel.setBlue(255-pixel.getBlue());
    }
  }
```

```
}
```

A5E5 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
public void grayscale()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel pixel = null;
  int total = 0;
  int average = 0;
  for (int row = 0; row < pixels.length; row++)
  {
    for (int col = 0; col < pixels[0].length; col++)
    {
      total = 0;
      pixel = pixels[row][col];
      total = total + pixel.getRed();
      total = total + pixel.getGreen();
      total = total + pixel.getBlue();
      average = total / 3;
      pixel.setColor(new Color(average, average, average));
    }
  }
}
```

A5E6 Water filters out red, so try increasing the red in the picture. This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
public void fixUnderwater()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel pixel = null;
  for (int row = 0; row < pixels.length; row++)
  {
    for (int col = 0; col < pixels[0].length; col++)
    {
      pixel = pixels[row][col];
      pixel.setRed(pixel.getRed() * 3);
```

```
      }
    }
  }
```

A6E1 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
  public void mirrorVerticalRightToLeft()
  {
    Pixel[][] pixels = this.getPixels2D();
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    int width = pixels[0].length;
    for (int row = 0; row < pixels.length; row++)
    {
      for (int col = 0; col < width / 2; col++)
      {
        leftPixel = pixels[row][col];
        rightPixel = pixels[row][width - col - 1];
        leftPixel.setColor(rightPixel.getColor());
      }
    }
  }
```

A6E2 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
  public void mirrorHorizontal()
  {
    Pixel[][] pixels = this.getPixels2D();
    Pixel topPixel = null;
    Pixel botPixel = null;
    int height = pixels.length;
    for (int row = 0; row < height / 2; row++)
    {
      for (int col = 0; col < pixels[0].length; col++)
      {
        topPixel = pixels[row][col];
        botPixel = pixels[height - row - 1][col];
        botPixel.setColor(topPixel.getColor());
```

```
      }
    }
  }
```

A6E3 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
  public void mirrorHorizontalBotToTop()
  {
    Pixel[][] pixels = this.getPixels2D();
    Pixel topPixel = null;
    Pixel botPixel = null;
    int height = pixels.length;
    for (int row = 0; row < height / 2; row++)
    {
      for (int col = 0; col < pixels[0].length; col++)
      {
        topPixel = pixels[row][col];
        botPixel = pixels[height - row - 1][col];
        topPixel.setColor(botPixel.getColor());
      }
    }
  }
```

A6E4 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
  public void mirrorDiagonal()
  {
    Pixel[][] pixels = this.getPixels2D();
    Pixel leftPixel = null;
    Pixel rightPixel = null;

    // calculate the max area to mirror (min of width or height)
    int max = pixels.length;
    if (pixels[0].length < max)
      max = pixels[0].length;

    // loop through to the left of the diagonal line (row=col)
    for (int row = 1; row < max; row++)
```

```
    {
      for (int col = 0; col < row; col++)
      {
        leftPixel = pixels[row][col];
        rightPixel = pixels[col][row];
        rightPixel.setColor(leftPixel.getColor());
      }
    }
  }
```

A7Q1 90

A7Q2 112

A7E1 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
  public void mirrorTemple()
  {
    int mirrorPoint = 276;
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    int count = 0;
    Pixel[][] pixels = this.getPixels2D();

    // loop through the rows
    for (int row = 27; row < 97; row++)
    {
      // loop from 13 to just before the mirror point
      for (int col = 13; col < mirrorPoint; col++)
      {
        count++;
        leftPixel = pixels[row][col];
        rightPixel = pixels[row][mirrorPoint - col +
                                                mirrorPoint];
        rightPixel.setColor(leftPixel.getColor());
      }
    }
    System.out.println(count);
  }
```

A7E2 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
public void mirrorArms()
{
  Pixel topPixel = null;
  Pixel botPixel = null;
  Pixel[][] pixels = this.getPixels2D();

  // loop through the rows
  for (int row = 155; row < 191; row++)
  {
    // loop through the columns
    for (int col = 98; col < 169; col++)
    {
      topPixel = pixels[row][col];
      botPixel = pixels[191-row+191][col];
      botPixel.setColor(topPixel.getColor());
    }
  }

  // loop through the rows
  for (int row = 155; row < 191; row++)
  {
    // loop through the columns
    for (int col = 238; col < 296; col++)
    {
      topPixel = pixels[row][col];
      botPixel = pixels[191-row+191][col];
      botPixel.setColor(topPixel.getColor());
    }
  }
}
```

A7E3 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
public void mirrorGull()
{
  int mirrorPoint = 350;
```

```
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    Pixel[][] pixels = this.getPixels2D();

    // loop through the rows
    for (int row = 225; row < 332; row++)
    {
      // loop from 13 to just before the mirror point
      for (int col = 219; col < mirrorPoint; col++)
      {
        leftPixel = pixels[row][col];
        rightPixel = pixels[row][mirrorPoint - col +
                                                 mirrorPoint];
        rightPixel.setColor(leftPixel.getColor());
      }
    }
  }
```

A8E1 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
  public void copy(Picture fromPic,
                   int fromStartRow,
                   int fromStartCol,
                   int fromEndRow,
                   int fromEndCol,
                   int toStartRow,
                   int toStartCol)
  {
    Pixel fromPixel = null;
    Pixel toPixel = null;
    Pixel[][] toPixels = this.getPixels2D();
    Pixel[][] fromPixels = fromPic.getPixels2D();
    for (int fromRow = fromStartRow, toRow = toStartRow;
         fromRow <= fromEndRow && toRow < toPixels.length;
         fromRow++, toRow++)
    {
      for (int fromCol = fromStartCol, toCol = toStartCol;
           fromCol <= fromEndCol && toCol < toPixels[0].length;
           fromCol++, toCol++)
      {
```

```
        fromPixel = fromPixels[fromRow][fromCol];
        toPixel = toPixels[toRow][toCol];
        toPixel.setColor(fromPixel.getColor());
      }
    }
  }
```

A8E2 Students can do all sorts of collages. Check that a given collage meets the requirements.

A9E1 This method is in `Picture.java` and a test method is in `PictureTester.java` in the `finalClasses` folder.

```
  public void edgeDetection2(int edgeDist)
  {
    Picture copy = new Picture(this);
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    Pixel[][] pixels = this.getPixels2D();
    Color rightColor = null;

    // compare a pixel with one to the right of it
    for (int row = 0; row < pixels.length; row++)
    {
      for (int col = 0; col < pixels[0].length-1; col++)
      {
        leftPixel = pixels[row][col];
        rightPixel = pixels[row][col+1];
        rightColor = rightPixel.getColor();
        if (leftPixel.colorDistance(rightColor) > edgeDist)
        {
          leftPixel.setColor(Color.BLACK);
        }
        else
        {
          leftPixel.setColor(Color.WHITE);
        }
      }
    }

    // now compare a pixel with the one below it
```

```
    Pixel[][] copyPixels = copy.getPixels2D();
    Pixel topPixel = null;
    Pixel botPixel = null;
    Color botColor = null;
    for (int row = 0; row < copyPixels.length-1; row++)
    {
      for (int col = 0; col < copyPixels[0].length; col++)
      {
        topPixel = copyPixels[row][col];
        botPixel = copyPixels[row+1][col];
        botColor = botPixel.getColor();
        if (topPixel.colorDistance(botColor) > edgeDist)
        {
          pixels[row][col].setColor(Color.BLACK);
        }
      }
    }
  }
```

A9E2 There are many algorithms for doing edge detection. Maybe have a contest and see who comes up with the best algorithm that works on a set of different images.

Group Work

All of these activities can be done in groups. Students can discuss the questions in the activities and answer them in groups. The exercises can also be done in groups. The only activity where you may want to have students work by themselves is the collage exercise (A8). This is to enable each student to create a collage that is meaningful to them.

Assessment

Sample Free-Response Questions

1. Write two unrelated methods for the `Picture` class.
   a. Write a method `int getCountRedOverValue(int value)` that returns a count of the number of pixels in the current picture that have a red value more than the parameter `value`.
   b. Write a method `setRedToHalfValueInTopHalf()` that sets the red value for all pixels in the top half of the picture to half the current red value.
2. Write two unrelated methods for the `Picture` class.

a. Write a method `clearBlueOverValue(int value)` that sets the blue value to 0 for every pixel that has a current blue value great than the parameter `value`.
b. Write a method `int[] getAverageForColumn(int col)` that creates and returns an array of integers the size of the number of columns that contains the average of the red, green, and blue values for each pixel in the column at index `col`.

The code for each of these methods is in `Picture.java` in the `finalClasses` folder.

Sample Multiple-Choice Questions

1. Which of the following will compile without error?

   I. `DigitalPicture p = new Picture();`

   II. `DigitalPicture p = new SimplePicture();`

   III. `Picture p = new SimplePicture();`

   a. I, II, and III
   b. II only
   c. III only
   d. I and II
   e. II and III

   Answer d is correct.

2. If the following code is in a method in the `Picture` class, what will the value of `count` be after the following code executes?

```
int count = 0;
for (int row = 5; row < 12; row++)
{
  for (int col = 8; col < 13; col++)
  {
    count++;
  }
}
```

   a. 13
   b. 25

c. 35
d. 42
e. 48

Answer c is correct.

3. Which of the following sets the blue value to zero for all pixels in the bottom half of the picture?

I.

```
public void method1()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel pixelObj = null;
  int height = pixels.length;
  for (int row = 0; row < height / 2; row++)
  {
    for (int col = 0; col < pixels[0].length; col++)
    {
      pixelObj = pixels[row][col];
      pixelObj.setBlue(0);
    }
  }
}
```

II.

```
public void method2()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel pixelObj = null;
  int height = pixels.length;
  for (int row = height / 2; row < height; row++)
  {
    for (int col = 0; col < pixels[0].length; col++)
    {
      pixelObj = pixels[row][col];
      pixelObj.setBlue(0);
    }
  }
}
```

III.

```
public void method3()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel pixelObj = null;
  int height = pixels.length;
  for (int row = height-1; row >= height / 2; row--)
  {
    for (int col = 0; col < pixels[0].length; col++)
    {
      pixelObj = pixels[row][col];
      pixelObj.setBlue(0);
    }
  }
}
```

a. I, II, and III
b. II only
c. III only
d. I and II
e. II and III

Answer e is correct.

Extensions

E1: Steganography

Steganography is the science of hiding information in a picture. You can hide a black and white message inside a color picture by first changing all the red values in the original color picture to be an even value (by subtracting one if odd). Make a picture of the same size out of the message that will be hidden. Then loop through both the original picture and the message picture, setting the red value of a pixel in the original picture to odd (by adding one to it) if the corresponding pixel in the message picture is close to the color black. Write an `encode` method that takes the black and white picture message and changes the current picture to hide the message picture inside of it. Then also write a `decode` method that returns the picture hidden in the current picture. There is example code for `encode` and `decode` in the `Picture.java` class in the `finalClasses` folder.

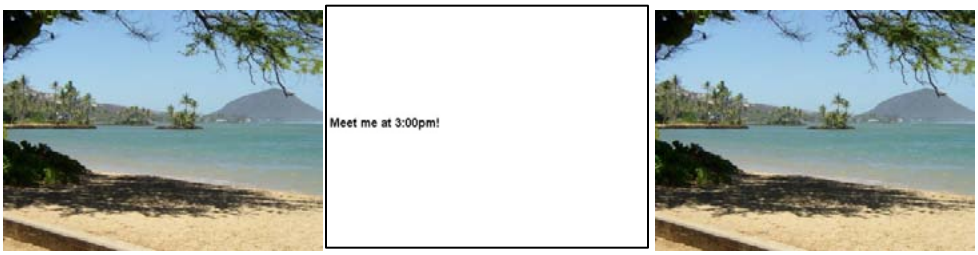

Figure 1: original (left), message (middle), beach with message hidden (right)

E2: Chromakey

Write a `chromakey` method that replaces the current pixel color with the color from another picture at the same row and column when the current pixel color is close to a specified color. In many movies, the actors are filmed in front of a green screen and then the green is replaced with a different background using a similar technique. There is sample code for the `chromakey` method in `Picture.java` in the `finalClasses` folder.

The picture in Figure 2 is of Dr. Mark Guzdial of Georgia Tech. Dr. Guzdial is the creator of the Media Computation approach to teaching computing concepts, which has students write programs that manipulate media: pictures, sounds, text, and movies. These labs are based on his work.

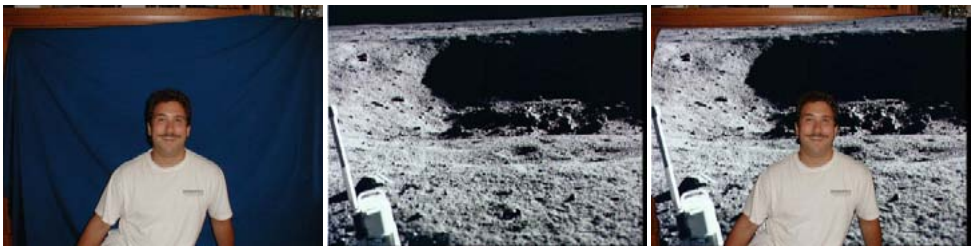

Figure 2: Dr. Guzdial (left), moon (middle), Dr. Guzdial on the moon (right)

References

Dann, W., Cooper, S., & Ericson, B. (2009) *Exploring Wonderland: Java Programming Using Alice and Media Computation*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson B. (2006) *Introduction to Computing and Programming in Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2009) *Introduction to Computing and Programming in Python: A Multimedia Approach*. (2nd ed.). Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2010) *Problem Solving with Data Structures using Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.