A1 Questions

1. Computer use digits from 0 to 9 and powers of 10 to represent values.
2. 3 bytes to represent a color in the RBG color model
3. 640\*480 = 307200 pixels

A2 Questions

1. Java.awt.Color[r=249 , g=107 ,b=141]
2. Java.awt.Color[r=249 , g= 255,b=0]
3. Java.awt.Color[r=255, g= 0,b=255]
4. Java.awt.Color[r=255, g= 255,b=255]

A3 Questions

1. 2
2. 0
3. The right most column index is 639
4. The bottom most row is 479
5. The row index increase to the bottom
6. The column index increase to the right
7. Yes, I can see the pixels

A5 Questions

1. No, but it is there are instances where it classes the method.
2. Yes, the method getPixels2D does exist.
3. Wouldn’t compile because it’s just an interface so it can’t make a new picture.
4. No because DigitalPicture is a interface and not a method
5. No because DigitalPicture is a interface and not a method
6. Yes because SimplePicture is a class where it could be passed
7. Yes because A non-abstract class provides bodies for all the methods declared in the interface, either directly or through inheritance.

A6

(Couldn’t get it to work)

A7 Questions

1. It would run through it 90 times
2. It would run through it 90 times

A4 Exercises

**public** **int** getCount(**int** num)

{

**int** total = 0;

**for** (**int** row = 0; row < matrix.length; row++)

{

**for** (**int** col = 0; col < matrix[0].length; col++)

{

**if** (matrix[row][col] == num)

total++;

}

}

**return** total;

}

**public** **int** getLargest(){

**int** largest = matrix[0][0];

**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];

}// end of if

}// end of inner loop

}// end of outer loop

**return** largest;

}// end of getLargest

**public** **int** getColTotal(**int** col){

**int** total= 0;

**for** (**int** row= 0; row < matrix.length; row++){

total = total + matrix[row][col];

}//end of for loop

**return** total;

}//end of getColTotal

A5 Exercises

**public** **void** keepOnlyBlue(){

Pixel[][] pixels = **this**.getPixels2D();

**for** (Pixel[] rowArray : pixels)

{

**for** (Pixel pixelObj : rowArray)

{

pixelObj.setRed(0);

pixelObj.setGreen(0);

}

}

}// end of keep only blue

**public** **void** negate()

{

Pixel [] pixelArray = **this**.getPixels();

**int** red,green,blue;

**for** (Pixel pixelObj : pixelArray)

{

red = pixelObj.getRed();

green = pixelObj.getGreen();

blue = pixelObj.getBlue();

pixelObj.setColor(**new** Color(255-red, 255-green, 255-blue));

}

}

A6 Exercises

**public** **void** mirrorVerticalRightToLeft()

{

Pixel[][] pixels = **this**.getPixels2D();

Pixel leftPixel = **null**;

Pixel rightPixel = **null**;

**int** length = pixels[0].length;

**for** (**int** row = 0; row < pixels.length; row++)

{

**for** (**int** col = 0; col < length / 2; col++)

{

leftPixel = pixels[row][col];

rightPixel = pixels[row][length - 1 - col];

leftPixel.setColor(rightPixel.getColor());

}

}

}// end of mirror

**public** **void** mirrorHorizontal(){

Pixel[][] pixels = **this**.getPixels2D();

Pixel topPixel = **null**;

Pixel bottomPixel = **null**;

**int** length = pixels.length;

**for** (**int** row = 0; row < length / 2 ; row++)

{

**for** (**int** col = 0; col < pixels[0].length; col++)

{

bottomPixel = pixels[length- 1 - row][col];

topPixel = pixels[row][col];

bottomPixel.setColor(topPixel.getColor());

}

}

}// end of mirror Horizontal

**public** **void** mirrorHorizontalBotToTop(){

Pixel[][] pixels = **this**.getPixels2D();

Pixel topPixel = **null**;

Pixel bottomPixel = **null**;

**int** length = pixels.length;

**for** (**int** row = 0; row < length / 2 ; row++)

{

**for** (**int** col = 0; col < pixels[0].length; col++)

{

bottomPixel = pixels[length- 1 - row][col];

topPixel = pixels[row][col];

topPixel.setColor(bottomPixel.getColor());

}

}

}// end of mirrorHorizontalBotToTop

A7 Exercises

**public** **void** mirrorArms()

{

**int** mirrorPoint = 192;

Pixel topPixel = **null**;

Pixel bottomPixel = **null**;

Pixel[][] pixels = **this**.getPixels2D();

// loop through the rows

**for** (**int** row = 164; row < mirrorPoint; row++)

{

// loop from 13 to just before the mirror point

**for** (**int** col = 100; col < 306; col++)

{

topPixel = pixels[row][col];

bottomPixel = pixels[mirrorPoint - row + mirrorPoint]

[col];

bottomPixel.setColor(topPixel.getColor());

}

}

}// end of mirror arms

**public** **void** mirrorGull()

{

**int** mirrorPoint = 338;

Pixel topPixel = **null**;

Pixel bottomPixel = **null**;

Pixel[][] pixels = **this**.getPixels2D();

// loop through the rows

**for** (**int** row = 164; row < 317; row++)

{

// loop from 13 to just before the mirror point

**for** (**int** col = 221; col < mirrorPoint; col++)

{

topPixel = pixels[row][col];

bottomPixel = pixels[row]

[mirrorPoint - col + mirrorPoint];

bottomPixel.setColor(topPixel.getColor());

}

}

}// end of mirrorGull

A8 Exercises

**public** **void** copy(Picture sourcePicture,

**int** startX, **int** startY,

**int** endX, **int** endY,

**int** targetStartX, **int** targetStartY)

{

Pixel sourcePixel = **null**;

Pixel targetPixel = **null**;

// loop through the x values

**for** (**int** x = startX, tx = targetStartX;

x < endX && x < sourcePicture.getWidth() &&

tx < **this**.getWidth();

x++, tx++)

{

// loop through the y values

**for** (**int** y = startY, ty = targetStartY;

y < endY && y < sourcePicture.getHeight() &&

ty < **this**.getHeight();

y++, ty++)

{

sourcePixel = sourcePicture.getPixel(x,y);

targetPixel = **this**.getPixel(tx,ty);

targetPixel.setColor(sourcePixel.getColor());

}

}

}

// i did it?

**public** **void** createCollage()

{

Picture flower1 = **new** Picture("D:/Documents/APProgramming2/pixLab/images/robot.jpg");

Picture flower2 = **new** Picture("D:/Documents/APProgramming2/pixLab/images/flower2.jpg");

Picture flower3 = **new** Picture("D:/Documents/APProgramming2/pixLab/images/kitten2.jpg");

**this**.copy(flower1,0,0,0,0);

**this**.copy(flower2,100,0,0,0);

**this**.copy(flower3,200,0,0,0);

Picture flowerNoBlue = **new** Picture(flower2);

flowerNoBlue.zeroBlue();

**this**.copy(flowerNoBlue,300,0);

**this**.copy(flower1,400,0);

**this**.copy(flower2,500,0);

**this**.mirrorVertical();

**this**.write("D:/Documents/APProgramming2/pixLab/images/collage.jpg");

}

A9 exercise

**public** **void** edgeDetection(**int** edgeDist)

{

Pixel leftPixel = **null**;

Pixel rightPixel = **null**;

Pixel topPixel = **null**;

Pixel bottomPixel = **null**;

Pixel[][] pixels = **this**.getPixels2D();

Color rightColor = **null**;

Color bottomColor = **null**;

**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***);

}

}

**for** ( **int** row = 0; row < pixels.length - 1; row++)

{

**for** ( **int** col = 0; col < pixels[0].length; col++)

{

topPixel = pixels[row][col];

bottomPixel = pixels[row + 1][col];

bottomColor = topPixel.getColor();

**if** (topPixel.colorDistance(bottomColor) > edgeDist)

bottomPixel.setColor(Color.***BLACK***);

**else**

bottomPixel.setColor(Color.***WHITE***);

}

}

}// end of edgeDetction