CSE 8A Lecture 8

• Check all your grades (class, psa, lab …) in grade-source. Contact grader ASAP if you find a mistake.

• PSA4: deadline tonight. Interview Friday noon.
  – Careful! No more “submission” mistakes.

• Start early PSA5.
  – Tuesday no interviews (usually).
  – Tutors can’t give you solutions. Don’t ask for that.

• Tomorrow: LAB 4

• Reading for next class: None. Prepare for In-term 2
Exam topics TO-DO

• Up to Thursday last week (so up to “style”)
  – Methods and parameters
  – Loops (For, For each, While)
Some comments on style

Meaningful variable names (generally more than 1 character)

Pixel pix;
for (int xpos = 0; xpos < getWidth();  xpos++)
{
    for (int ypos = 0; ypos < getHeight(); ypos++)
    {
        pix = getPixel(xpos, ypos);
        pix.setColor(Color.BLACK);
    }
}

Proper indentation (Dr. Java will help with this)

One statement per line

Lines not longer than 80 characters
Fill in the code below to make the right (approximately) half of the picture \textit{pure blue}.

```java
Pixel p;
for (int x = ___________; x < ___________; x = ______) {
    for (int y = _____; y < ____________; y = _________ ) {
        p = getPixel( x, y )
        p.setBlue( ________ );
        p.setRed( _______ );
        p.setGreen( ________ );
    }
}
```
Mirroring Around Vertical Axis

Mirror left to right

Vertical axis
Mirroring Around Vertical Axis

Mirror right to left

Vertical axis
Mirroring Around Vertical Axis: Left to Right

• What are the parameter values (coordinates) we use to index leftPixel and rightPixel for the first three iterations of the inner loop? (assume picture has a height = 50 and width = 100)

```
int mirrorPt = getWidth()/2;
Pixel leftP, rightP;
for (int y = 0; y < getHeight(); y++)
{
    for (int x = 0; x < mirrorPt; x++)
    {
        leftP = getPixel(x,y);
        rightP = getPixel(getWidth()-1-x,y);
        rightP.setColor(leftP.getColor());
    }
}
```

<table>
<thead>
<tr>
<th></th>
<th>Left Pixel</th>
<th>Right Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0, 99</td>
<td>99, 0</td>
</tr>
<tr>
<td></td>
<td>0, 98</td>
<td>98, 0</td>
</tr>
<tr>
<td></td>
<td>0, 97</td>
<td>97, 0</td>
</tr>
<tr>
<td>B</td>
<td>0, 0</td>
<td>99, 0</td>
</tr>
<tr>
<td></td>
<td>1, 0</td>
<td>98, 0</td>
</tr>
<tr>
<td></td>
<td>2, 0</td>
<td>97, 0</td>
</tr>
<tr>
<td>C</td>
<td>0, 49</td>
<td>49, 0</td>
</tr>
<tr>
<td></td>
<td>0, 48</td>
<td>48, 0</td>
</tr>
<tr>
<td></td>
<td>0, 47</td>
<td>47, 0</td>
</tr>
<tr>
<td>D</td>
<td>0, 0</td>
<td>49, 0</td>
</tr>
<tr>
<td></td>
<td>1, 0</td>
<td>48, 0</td>
</tr>
<tr>
<td></td>
<td>2, 0</td>
<td>47, 0</td>
</tr>
<tr>
<td>E</td>
<td>None of the previous options</td>
<td></td>
</tr>
</tbody>
</table>
How do you figure these kinds of questions out?

- Answer: Draw a diagram
  - imagine “beginning” and “answer” (goal)
  - Draw arrows to show how to get from beginning to answer
  - Then fill in numbers in order, write loops to create those numbers

```
<table>
<thead>
<tr>
<th>Left_X</th>
<th>Left_Y</th>
<th>Right_X</th>
<th>Right_Y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Copying pixels

The key to (almost) all of the **image manipulation problems** in lab is to copy the color value across pixels in an image. The key is figuring out *which pixels* to copy and *where to copy them to*.

Here is the generic template that you will use for almost all of these problems:

```java
Pixel sourcePixel, targetPixel;
for ( int sourceX = [start of source region (horiz)];
    sourceX < [end of source region]; [move sourceX]) {
    for ( int sourceY = [start of source region (vert)];
        sourceY < [end of source region]; [move sourceY]) {
        sourcePixel = getPixel( sourceX, sourceY )

        targetPixel = getPixel( _________, __________ )
        targetPixel.setColor( sourcePixel.getColor() );
    }
}
```
The key to (almost) all of the image manipulation problems in lab is to copy the color value across pixels in an image. The key is figuring out which pixels to copy and where to copy them to.

Here is the generic template that you will use for almost all of these problems:

```java
Pixel sourcePixel, targetPixel;
for ( int sourceX = [start of source region (horiz)]; sourceX < [end of source region]; [move sourceX] ) {
for ( int sourceY = [start of source region (vert)]; sourceY < [end of source region]; [move sourceY] ) {
    sourcePixel = getPixel( sourceX, sourceY )
    targetPixel = getPixel( __________, __________ )
    targetPixel.setColor( sourcePixel.getColor() );
}
}
```

WARNING! This is tricky! It takes lots of practice and careful reasoning, so don’t be alarmed if you don’t get it at first. You will, so stick with it.
Mirroring Even Width versus Odd Width

int mirrorPt = getWidth() / 2;
...
for (int x = 0; x < mirrorPt; x++)
Mirroring Odd-width Pictures

- What happens when this code attempts to mirror a Picture around the **vertical axis** when the Picture’s width is odd (e.g. 101)?

```java
int mirrorPt = getWidth()/2;
Pixel leftP, rightP;
for (int y = 0; y < getHeight();  y++)
{
    for (int x = 0; x < mirrorPt; x++)
    {
        leftP = getPixel(x,y);
        rightP = getPixel(getWidth()-1-x,y);
        rightP.setColor(leftP.getColor());
    }
}
```

A. It will work fine
B. It will run, but it won’t mirror correctly
C. I won’t run, there will be an index out of bounds exception
D. It won’t even compile if getWidth() is odd
Mirror versus “flip” (PSA!) (around vertical axis)
What are the first \((x,y)\) coords for \(\text{topP}\) and \(\text{bottomP}\) to mirror around horizontal axis?

- **topP**
  - A. \((0,0)\) \((0,3)\)
  - \((0,1)\) \((0,2)\)
  - \((1,0)\) \((1,3)\)

- **bottomP**
  - B. \((0,0)\) \((0,3)\)
  - \((1,0)\) \((1,3)\)
  - \((2,0)\) \((2,3)\)

C. either A or B will work

D. none of the above
Challenge: Complete the code that mirrors in the order specified by answer B

```
int height = getHeight();
int width = getWidth();
int mid = height/2;
Pixel topP, botP;
for (                                    ) {
    for(                                    ) {
        topP = getPixel(                        );
        botP = getPixel(                        );
        botP.setColor(topP.getColor());
    }
}
```
Order of copying pixels

- When mirroring, we need to copy certain pixels to certain other pixels

- It doesn’t matter what order we copy in, as long as when we are done, pixels have been copied correctly

- Two most common orders:
  - **Row major order**: copy all the pixels in one row, then go on to the next row
  - **Column major order**: copy all the pixels in one column, then go on to the next column
Mirroring around horizontal axis

column-major order

```java
int height = getHeight();
int width = getWidth();
int mid = height/2;
Pixel topP, botP;
for(int x=0; x<width; x++) {
    for(int y=0; y<mid; y++) {
        topP = getPixel(x,y);
        botP = getPixel(x,height-1-y);
        // copy one to the other...
    }
}
```

row-major order

```java
int height = getHeight();
int width = getWidth();
int mid = height/2;
Pixel topP, botP;
for(int y=0; y<mid; y++) {
    for(int x=0; x<width; x++) {
        topP = getPixel(x,y);
        botP = getPixel(x,height-1-y);
        // copy one to the other...
    }
}
```
Challenge: What does this code do?

• Hint: trace some of the getPixel index values.

```java
int magic = getWidth()/2;
Pixel foo, bar;
for(int y = 0; y < getHeight(); y++)
{
    int countingDown = getWidth()-1;

    for(int x = 0; x < magic; x++)
    {
        foo = getPixel(x,y);
        bar = getPixel(countingDown,y);
        bar.setColor(foo.getColor());
        countingDown--;
    }
}
```

A. Copies top half into bottom half not mirrored.
B. Copies left half into right half not mirrored.
C. Mirrors around vertical axis, left into right
D. Mirrors around horizontal axis, top into bottom
E. Some other bizarre transformation
Challenge: What does this code do?

- Hint: trace some of the getPixel index values.

```java
int magic = getWidth()/2;
Pixel foo, bar;
for(int y = 0; y < getHeight(); y++)
{
    int countingDown = getWidth()-1;

    for(int x = 0; x < magic; x++)
    {
        foo = getPixel(x,y);
        bar = getPixel(countingDown,y);
        bar.setColor(foo.getColor());
        countingDown--;
    }
}
```
By what variable name do we refer to *collage* inside *makeC* in *Picture.java*?

A. collage  
B. callingObject  
C. Object  
D. Picture  
E. this
Match the scenario to the constructor call (we’ll vote for each scenario)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Create a picture from a specific file</td>
<td>A. Picture p = new Picture();</td>
</tr>
<tr>
<td>2) Create a picture that is a copy of another picture</td>
<td>B. Picture p = new Picture(&quot;filename.jpg&quot;);</td>
</tr>
<tr>
<td>3) Create a picture of a given width and height</td>
<td>C. Picture p = new Picture(other);</td>
</tr>
<tr>
<td>4) Create a picture of the same width and height as another picture</td>
<td>D. Picture p = new Picture(aNum,bNum);</td>
</tr>
</tbody>
</table>
What does this code do?

```java
Pixel foo;
for(int y = 40; y < 50; y++)
{
    for(int x = 1; x < 5; x++)
    {
        Pixel foo = getPixel(x, y);
        foo.setColor(Color.RED);
    }
}
```

Makes red box of width height

A. 10 5
B. 9 4
C. 5 10
D. 4 9
E. None of the above
What does this code do?

```
Pixel foo;
for(int y = 40; y < 50; y++)
{
    for(int x = 1 ; x <= 5; x++)
    {
        foo = getPixel(x,y);
        foo.setColor(Color.RED);
    }
}
```

Makes red box of
width  height

A. 10  5
B. 9  4
C. 5  10
D. 4  9
E. None of the above
What are correct loops to make a black box of width $x$ and height $y$?

```java
public void foo(int x, int y) {
    Pixel foo;
    """LOOP HEADER 1""
    { """LOOP HEADER 2""
        foo = getPixel(w, h);
        foo.setColor(Color.BLACK);
    }
}
```

A) for (int $w = 0; w <= x; w++)
    for (int $h = 0; h <= y; h++)
B) for (int $w = 10; w < x+10; w++)
    for (int $h = 20; h < y+20; h++)

C) for (int $w = 0; w < y; w++)
    for (int $h = 0; h < x; h++)
D) for (int $w = 10; w <= x+10; w++)
    for (int $h = 20; h <= y+20; h++)
```
From the book: Cropping A Picture (page 147-148) – we’ll change a bit

• Example of:
  – Working with both the calling object and a parameter object in a method
    • Extra information is passed to methods through parameters. The calling object is something like an extra parameter, named **this**
  – Doing something to a subset of the possible pixels in a picture
What part of Katie are we copying?
(slight mod from the book)

public void copyKatieSXXX(Picture sourcePic)
{
    Pixel sPixel = null, tPixel = null;
    for (int sX = 40, tX = 100; sX < 110; sX++, tX++)
    {
        for (int sY = 350, tY = 100; sY < 400; sY++, tY++)
        {
            sPixel = sourcePic.getPixel(sX, sY);
            tPixel = this.getPixel(tX, tY);
            tPixel.setColor(sPixel.getColor);
        }
    }
}

A. Feet
B. Part of dress
C. Hands
D. Part of Couch
E. Face
Parameters: getting information into methods

• It’s nice to have code that is “user controllable”…

• We have been hard-coding constants (40, 3, 100, for example) a lot, but we can write more flexible code using PARAMETERS

• This lets us write code to do things like “cropping and pasting into a blank canvas”, but letting the user specify what part of the source picture to crop, and where to place it in the canvas.
Underline the values you would change into parameters and write a new method header

```java
public void copyKtiesXXX(
```

```java
{ 
    Pixel sPixel, tPixel = null; 
    for (int sX = 40, tX = 100; sX < 110; sX++, tX++) 
    { 
        for (int sY = 350, tY = 100; sY < 400; sY++, tY++) 
        { 
            sPixel = sourcePic.getPixel(sX,sY); 
            tPixel = this.getPixel(tX,tY); 
            tPixel.setColor(sPixel.getColor(); 
        } 
    } 
}
```
TODO

• Study for In-term Exam 2

• Finish PSA4

• Lab tomorrow!