CSE 8A Lecture 9

• Reading for next class: 6.1-6.4 (conditional statements)

• PSA4 interview due tomorrow noon.

• PSA5: Collage and Picture Flip, DON’T WAIT (it’s longer than the previous PSAs)
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`
Examples of scenarios for different Picture constructors

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Calls to Picture constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Create a picture from a specific file</td>
<td>A. Picture p = new Picture(&quot;filename.jpg&quot;);</td>
</tr>
<tr>
<td>2) Create a picture that is a copy of another picture</td>
<td>B. Picture p = new Picture(other);</td>
</tr>
<tr>
<td>3) Create a picture of a given width and height</td>
<td>C. Picture p = new Picture(aNum,bNum);</td>
</tr>
<tr>
<td>4) Create a picture of the same width and height as another picture</td>
<td>D. Picture p = new Picture();</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++)
    {
        foo = getPixel(x,y);
        foo.setColor(Color.RED);
    }
}
```

Makes red box of

<table>
<thead>
<tr>
<th>width</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. 10</td>
<td>5</td>
</tr>
<tr>
<td>B. 9</td>
<td>4</td>
</tr>
<tr>
<td>C. 5</td>
<td>10</td>
</tr>
<tr>
<td>D. 4</td>
<td>9</td>
</tr>
<tr>
<td>E. None of the above</td>
<td></td>
</tr>
</tbody>
</table>

1) Solo: (1 min)
2) Discuss: (2 min)
3) Group: (30 sec)
What does this code do?

```java
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
public void copyKatieXXX(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
REVIEW: 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 copyKatiesXXX(
{
    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());
        }
    }
}
```
Using parameters

In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, 
    int ySource, int xTarget, int yTarget )
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = target.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
```

Write the code to copy the square at position (10, 50) in fish to the blank canvas (vote on next slide)
Using parameters

In Picture.java...

public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    // Body omitted to save space
}

In main...

Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();

A. fish.copyRegionTo(blank, 10, 50, 30, 30)
B. fish.copyRegionTo(blank)
C. fish.copyRegionTo()
Parameters and scope

In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    // Body omitted to save space
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
```

Variables only exist in the region they are defined. I.e. variables in main cannot be accessed in copyRegion and vice versa. The region where a variable exists is called its scope.
In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource,
    int ySource, int xTarget, int yTarget )
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = target.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
```

Main’s variables

- fish
- blank
Parameters and scope

In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = target.getPixel(tX,tY);
            tPixelsetColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
```
Parameters and scope

In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = target.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}

In main...
```

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
```

Main’s variables

- fish
- blank
Parameters and return values

In Picture.java...

```java
public Picture copyRegionToNew(int xSource, int ySource, int xTarget, int yTarget)
{
    Picture newCanvas = new Picture();
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = newCanvas.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture newCanvas = fish.copyRegionToNew(10, 30, 50, 50);
newCanvas.show();
```

What error will the following code produce?

A. This code will not compile
B. The line “Picture newCanvas = fish.copyRegionToNew…” in main will cause an error
C. The line newCanvas.show() will cause an error
Parameters and return values

In Picture.java...
public Picture copyRegionToNew(int xSource, int ySource, int xTarget, int yTarget )
{
    Picture newCanvas = new Picture();
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = newCanvas.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}

In main...
Picture fish = new Picture( "fish.jpg" );
Picture newCanvas = fish.copyRegionToNew(10, 30, 50, 50);
newCanvas.show();
Some comments on comments

/* A method to copy a 100x100 region of the calling object’s image to a blank canvas.
 * xSource, ySource: the upper left corner of the region to be copied.
 * xTarget, yTarget: the upper left corner where the region will appear in the new canvas.
 * returns a new canvas with the region copied into it. */

public Picture copyRegionToNew(int xSource, int ySource, int xTarget, int yTarget) {
    Picture newCanvas = new Picture();
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++) {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++) {
            sPixel = this.getPixel(sX, sY);
            tPixel = newCanvas.getPixel(tX, tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
    return newCanvas;
}

Header (method) comments required.

Inline comments if necessary
Challenge: Using methods (to do cool stuff)

```java
public void copyRegionTo (Picture target, int xSource,
    int ySource, int xTarget, int yTarget )
```

Using the method `copyRegionTo` (header above) write a method that will copy a pattern of 20x20 squares from a source image to a target image (see below). Assume `copyRegionTo` has been modified to copy a 20x20 square instead of 100x100.
TODO

• Reading for next class: 6.1-6.4
• Finish PSA5