CSE8A Lecture 6

- PSA3 due today midnight!
- PSA 4 will be released tomorrow: partner?
- Compile and run your code JUST BEFORE you submit. You will lose points (lots) if your code doesn’t run. (Turnin problems resolved. Sorry for the inconvenience!)
- LAB 3 TOMORROW. BE ON TIME to keep all your points!
- NEW! You can get one interview “for free” (a.k.a., you don’t miss those points if you don’t do it, JUST ONCE!)
- First two classes (reading quiz and participation) don’t count toward your grade (free drops)
- We start with ASSIGNED SEATING for the class: remember to check it before coming.
Your discussion groups

- Did you know…you were assigned not just a seat, but a group!
- Examine this layout to see who is in your group.
  - You **MUST** be discussing with the people shown in your group (same color) on this chart!
  - **NOT** just “anybody nearby to you”!
  - This prevents anybody from being left out (example: stuck between two groups neither of which really claims them) or other problems.

Pepper Canyon Hall 109 - CSE 8A - Fall 2013
Peer Instruction Layout:

<table>
<thead>
<tr>
<th>1 2 3 4</th>
<th>L</th>
<th>K</th>
<th>J</th>
<th>H</th>
<th>G</th>
<th>F</th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>5 6 7</td>
<td>8 9 10</td>
<td>11 12 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td></td>
<td>6 7 8</td>
<td>9 10 11</td>
<td>12 13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Your discussion groups

- Did you know…you were assigned not just a seat, but a group!
- Examine this layout to see who is in your group.
  - You **MUST** be discussing with the people shown in your group (same color) on this chart!
  - **NOT** just “anybody nearby to you”!
  - This prevents anybody from being left out (example: stuck between two groups neither of which really claims them) or other problems.

Solis 104 Layout

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

- You MUST be discussing with the people shown in your group (same color) on this chart!
- NOT just “anybody nearby to you”!
- This prevents anybody from being left out (example: stuck between two groups neither of which really claims them) or other problems.
Your discussion groups

- Did you know…you were assigned not just a seat, but a group!
- Examine this layout to see who is in your group.
  - You **MUST** be discussing with the people shown in your group (same color) on this chart!
  - **NOT** just “anybody nearby to you”!
  - This prevents anybody from being left out (example: stuck between two groups neither of which really claims them) or other problems.
Exam 1

• Median: 9.5 Great job everyone!

• High score: 10/10 (199 people scored this!)

• If you scored lower than 7.5:
  – Don’t panic, your lowest score is dropped, but…
  – Get help! Come talk to the office hours.
  – Do the reading and assignments with a partner (actually, everyone is encouraged to do this!)
World world1 = new World(200, 100);
Turtle avery = new Turtle(100, 50, world1);
Turtle colin = new Turtle(25, 25, world1);
Turtle sara = avery;
int first = 42;
int second = first;
first = 30;

avery.turn(90);
sara.forward(50);
colin.backward(50);
avery.turn(90);
avery.forward(25);
// Make the turtle draw a T-shape (NOT THE ONLY SOLUTION!)

public class TurtleTester {
    public static void main( String[] args )
    {
        World w = new World( 300, 300 );
        Turtle jane = new Turtle( 150, 250, w );
        // You will add code here. See below.
        jane.forward(120);
        jane.turn(90);
        jane.penUp();
        jane.forward(45);
        jane.turn(180);
        jane.turn(180);
        jane.penDown();
        jane.forward(90);
    }
}

TurtleTester.java
Review: The “return” value of a method

public int myMethod(int size) {
    // operations of
    // the method ...
    return variable_or_value_of_return_type;
}

Assign the result of a method ...
(unless the return type is void! Then you can’t do it, there is nothing to be assigned)

variable_of_return_type = myMethod(5);
While loops with pictures:
What does this code do?*

A. Removes all red from the picture
B. Changes ½ of the red pixels to not be red
C. Reduces the red component of ½ of the pixels
D. Reduces the red component of each pixel to ½ of its original value
E. Sets the red component of each pixel to 0.5

* Remember the meaning/effect of (int)!!
What pixels does this code modify?

Pixel[] pixelArray = this.getPixels();
int value = 0;
int index = 0;
while (index < pixelArray.length/4)
{
    value = pixelArray[index].getRed();
    value = (int) (value * 0.5);
    pixelArray[index].setRed(value);
    index = index + 1;
}

A) 
B) 
C) 
D) 
E) None of the above
A Picture is a 2-dimensional array of pixels
  - Each pixel in the Picture has (x,y) coordinates, with x==0, y==0 at the upper-left-hand corner of the picture

A Pixel[] pixel array is a 1-dimensional array of pixels
  - Each Pixel in the array has an integer index I, with I==0 indexing the first Pixel in the array

How do the two relate to each other…?
Pixel arrays and pixels in Pictures

• How do the two relate to each other…?

```java
Picture pic = new Picture("mypic.jpg");
Pixel[] pixArray = pic.getPixels();
```

![Image of a picture with pixel values and arrays](image.png)

Width 4
Height 3

Length 12
Pixel arrays and pixels in Pictures

• How do the two relate to each other…?

```java
Picture pic = new Picture("mypic.jpg");
Pixel[] pixArray = pic.getPixels();
```

What is the x coordinate of the pixel in the lower right corner of any picture?
A. `pic.getWidth()`  
B. `pic.getHeight()`  
C. `pic.getWidth()-1`  
D. `pic.getHeight()-1`  
E. 3
Pixel arrays and pixels in Pictures

- How do the two relate to each other…?

```java
Picture pic = new Picture("mypic.jpg");
Pixel[] pixArray = pic.getPixels();
```

What is position of this same pixel in `pixArray`, for *any* picture?

A. `pixArray[pic.getWidth()]`
B. `pixArray[pic.getWidth()-1]`
C. `pixArray[pixArray.length]`
D. `pixArray[pixArray.length-1]`
E. `pixArray[11]`
Pixel arrays and pixels in Pictures

• How do the two relate to each other…?

```java
Picture pic = new Picture("mypic.jpg");
Pixel[] pixArray = pic.getPixels();
```

Given an x,y position in pic, what is a formula to calculate the position, i, of that Pixel in pixArray?

A. \( i = \text{pic.getWidth()} \times y + x \)
B. \( i = \text{pic.getHeight()} \times y + x \)
C. \( i = \text{pic.getWidth()} \times x + y \)
D. \( i = \text{pic.getHeight()} \times x + y \)
E. None of these
This code is supposed to increase red by 1.5, but...

```java
Pixel[] pixelArray = this.getPixels();
int value = 0;
int index = 0;

while (index < pixelArray.length)
{
    value = pixelArray[index].getRed();
    value = (int)1.5 * value;

    pixelArray[index].setRed(value);
    index = index + 1;
}
```
Debugging: Tracing variables

Step 1: Have a mental model of what the variables are supposed to do
Step 2: You can add print statements to make sure variables are doing what you think are (in the future you will use a debugger)

```java
Pixel[] pixelArray = this.getPixels();
int value = 0;
int index = 0;

while (index < pixelArray.length)
{
    value = pixelArray[index].getRed();
    value = (int)1.5 * value;

    pixelArray[index].setRed(value);
    index = index + 1;
}
```
Debugging: Tracing variables

Where should you insert the following print statements in the code below?

```java
System.out.println("Old value of red is " + value);
System.out.println("New value of red is " + value);
```

```java
Pixel[] pixelArray = this.getPixels();
int value = 0;
int index = 0;
while (index < pixelArray.length)
{
    value = pixelArray[index].getRed();
    value = (int)1.5 * value;
    pixelArray[index].setRed(value);
    index = index + 1;
}
```

A. 1 & 2  
B. 1 & 3  
C. 2 & 3  
D. 3 & 4  
E. 2 & 4
DEBUGGING: This code should swap the red and blue components at each Pixel; what does it ACTUALLY do?

A. It has a compiler error
B. It sets the red value to be the same as blue
C. It sets the blue value to be the same as red
D. It really does swap them

Pixel[] pixelArray = this.getPixels();
int value = 0;
int index = 0;
while (index < pixelArray.length)
{
    Pixel pix = pixelArray[index];
    value = pix.getRed();
    value = pix.getBlue();
    pix.setRed(value);
    pixelArray[index].setBlue(value);
    index++;
}
Swapping (swap the red and blue components)

Pixel[] pixelArray = this.getPixels();
int value = 0;
int index = 0;
while (index < pixelArray.length)
{
    Pixel pix = pixelArray[index];
    <<CODE GOES HERE>>
    index++;
}

A
value = pix.getRed();
pix.setBlue(pix.getRed());
pix.setRed(value);

B
value = pix.getRed();
pix.setBlue(value);
pix.setRed(pix.getBlue());

C
value = pix.getRed();
pix.setRed(pix.getBlue());
pix.setBlue(value);

D
value = pix.getRed();
pix.setRed(value);
pix.setBlue(pix.getRed());
What picture most accurately describes what this code does?

```java
Pixel[] pixelArray = this.getPixels();
Pixel p;
Pixel q;
for(int index = 0; index < pixelArray.length-1; index++)
{
    p = pixelArray[index];
    q = pixelArray[index+1];
    p.setRed(q.getRed());
    p.setBlue(q.getBlue());
    p.setGreen(q.getGreen());
}
```

A. [Diagram A]
B. [Diagram B]
C. [Diagram C]
D. None of these

What if we change to ...
```java
p = pixelArray[index+1];
q = pixelArray[index];
p.setRed(q.getRed());
p.setBlue(q.getBlue());
p.setGreen(q.getGreen());
```
Why does this code have an error?

A. It tries to access pixelArray[-1]
B. It tries to access pixelArray[0]
C. It tries to access pixelArray[pixelArray.length]
D. It tries to access pixelArray[pixelArray.length+1]
E. None of the above

```java
Pixel[] pixelArray = this.getPixels();
int value = 0;
Pixel p = null;
for(int index = 0; index < pixelArray.length; index++)
{
    p = pixelArray[index];
    q = pixelArray[index+1];
    p.setRed(q.getRed());
    p.setBlue(q.getRed());
    p.setGreen(q.getGreen());
}
```
• True or False: The following code could be written using a for-loop.

```java
Pixel[] pixelArray = this.getPixels();
int index = 0;
while ( index < pixelArray.length )
{
    Pixel pix = pixelArray[index];
    pix.setGreen(255);
    index = index + 1;
}
```

A. True  
B. False
while

    Pixel[] pixelArray = this.getPixels();
    int index = 0;
    while ( index < pixelArray.length )
    {
        Pixel pix = pixelArray[index];
        pix.setGreen(255);
        index = index + 1;
    }

for

    Pixel[] pixelArray = this.getPixels();
    for ( int index = 0; index < pixelArray.length; index++ )
    {
        Pixel pix = pixelArray[index];
        pix.setGreen(255);
    }

for each

    Pixel[] pixelArray = this.getPixels();
    for ( Pixel pix: pixelArray )
    {
        pix.setGreen(255);
    }

Which do you prefer?
Why?
while vs. for vs. for each

• So when to use each? Sometimes it’s a matter of style, sometimes it’s ease of functionality

Choose the best loop to use in each of these situations

A. For
B. For each
C. While

1. You want to loop through a picture until you find a pixel that is all black. Then you want your loop to stop.

2. You want to loop through all the pixels in a picture and set each pixel red value equal to its green value

3. You want loop through the pixels in the first half of the picture and make them all black.
while vs. for vs. for each: Summary

Less chance of error
• for each
  – Use when you know you need to access and modify each pixel directly

• for
  – Use when you need to loop through a known number of pixels and need access to their index value

• while
  – Use when you are not sure how many pixels to loop through

Greater chance of error

Often, though, you can just choose the loop you like best
TODO

• Submit PSA3, plan the interview, tonight/tomorrow PSA4 will be released.
• Read next class: 5.1 and video