INSTRUCTIONS

This is an open book and note examination. You may use any written material you bring to the exam, but there is no interpersonal or other communication or discussion (except with the prof. or proctors) or sharing of information. Cell phones or any electronic devices (other than a calculator) are strictly forbidden! You must remain in the exam room at all times. If you need to leave the room, come to the front, leaving your paper and ask permission. You may take NOTHING with you to leave the room (and that includes cell phones!). Failure to abide by any of these terms will result in a grade of zero for the exam.

You must show a picture ID to the instructor or TA when you turn in your exam.

NAME _____________________________

NET ID _____________________________

LAB SECTION (circle one):

Mon 12:35    Mon 4:40    Tue 2:45    Wed 1:40    Wed 4:15
Mon 1:40     Tue 1:15    Wed 11:30   Wed 2:45    Thurs 2:45

------- For scoring use only. Do not write below this line ----------
Section 1 – Multiple Choice (10 questions, 3 points each)
Select the best answer to the question from the choices provided.

1. Which is not a method of the Picture class?
   a. `getPixels()`
   b. `getPixel()`
   c. `getHeight()`
   d. `setColor()`
   e. `show()`

2. What is the result of `5/2`?
   a. 5
   b. 3
   c. 2
   d. 2.55
   e. The answer is undefined

3. What is the result of `5.1/2.0`?
   a. 5
   b. 3
   c. 2
   d. 2.55
   e. The answer is undefined

4. What is the result of `(int)(5.1/2.0)`?
   a. 5
   b. 3
   c. 2
   d. 2.55
   e. The answer is undefined

5. When is `TEST` performed in
   
   ```
   for ( SETUP; TEST ; INCREMENT ) { BODY } 
   ```
   a. Only after `SETUP` is run.
   b. Just before each time `BODY` is run.
   c. Just before each time `INCREMENT` is run.
   d. Only when the loop finishes.
   e. Only when the loop starts.

6. When is `INCREMENT` performed in
   
   ```
   for ( SETUP; TEST ; INCREMENT ) { BODY } 
   ```
   a. Between `SETUP` and `TEST`.
   b. Between `TEST` and `BODY`.
   c. Between `BODY` and `TEST`.
   d. When the loop variable (like `int x` declared in `SETUP`) is used in `BODY`.

Page 2 of 8
7. What are the (x,y) coordinates of the Pixel in the lower left corner of the screen?
   a. (0,0)
   b. (0,639)
   c. (0,479)
   d. (639,0)
   e. (639,479)

8. What is the weighted average of these two numbers with given weights?
   0.0 with weight 2/3
   12.0 with weight 1/3
   a. 0.0
   b. 3.0
   c. 4.0
   d. 6.0
   e. 8.0

9. Which statement about arrays is FALSE?
   a. All tickets (elements) in an array have the same type.
   b. The first index is always 0.
   c. If \( A \) refers to the array, the index of the last element is
      \( A.length - 1 \)
   d. All tickets (elements) in an array have the same value.
   e. An array can never be made smaller or larger after it was first made.
10. Look at the code in Problem 11 below. Suppose the entire definition of \texttt{drawT} is the last thing coded into the \texttt{Turtle} class. How many \}’s (right, closing curly onion rings) must appear in the \texttt{Turtle.java} file after the single \} at the end of the code given in Problem 10?

a. 0
b. 1
c. 2
d. more than 2.
e. Cannot tell from the given information.

11. (10 points) Fill in the Java code that

a. makes \texttt{main} CALL method \texttt{drawT} with a reasonable and correct parameter of you own choice, so the T is visible and fits in the 640 by 480 pixel \texttt{World}.

b. makes the \texttt{drawT} method of \texttt{Turtle} accept one double precision parameter that is then used to control the size of the T that it draws.

All this code is within \texttt{Turtle.java}

\begin{verbatim}
public static void main(String a[])
{
    World w = new World();
    Turtle tur = new Turtle(w);
    //WRITE YOUR ANSWER TO a. HERE
}
//TO ANSWER b, FILL IN WHAT'S NEEDED BELOW, OR
//IF NO MORE CODE NEEDS TO BE WRITTEN, CHECK HERE________

public void drawT( )
{
    this.forward((int)(100*scale));
    this.turn( 90.0 );
    this.forward((int)(25*scale));
    this.turn( 180.0 )
    this.forward((int)(50*scale));
}
\end{verbatim}
12. (10 points) We dramatized CALLERS who initiate a method call and CALLEES who get called, that is, receive calls. Check EXACTLY ONE box in each row.

<table>
<thead>
<tr>
<th></th>
<th>CALLER</th>
<th>CALLEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>initiates the call</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>writes on the calling card</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reads numbers and other information from the calling card</td>
<td></td>
<td></td>
</tr>
<tr>
<td>does the work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>waits for the work to be completed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>returns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
13. (10 points) Code-tracing is especially handy for understanding the logic behind a given piece of code. Consider the following code fragment:

```java
int count = 0;
for ( int index = 0 ; index < 2 ; index = index + 1 )
{
    count = index - 1;
    count = 2 * count;
}
System.out.println(count);
```

What is printed? (Just one number) _____________________________

Use the following boxes to simulate the tickets (i.e., variables) used by a computer who (or which) follows directions expressed by the Java code. Keep track of the number currently on each ticket (a) after the first two lines of code, and then, (b) during each and every repetition of the code in the for loop's body. Remember to heed the names (index and count) of the two tickets. Hint: There might be more boxes than you need to use because the computation stops before you use them up. It's your job to figure out when the loop stops!

<table>
<thead>
<tr>
<th>index</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WARNING: Now that you simulated the computation, GO BACK and CHECK and probably FIX UP your answer! Simulating unfamiliar code is tricky and most beginners will get the prediction of what's printed WRONG before they do the full simulation! (On college exams, it's wise to go back and check earlier question subparts after reaching the end of a big problem.)
Section 3 – Filling in code
For each of the following questions, fill in the missing Java statement in the box provided:

14. (5 points) Fill in the code below so finds the sum of all of the red intensities of all the Pixels in a Picture’s array of Pixels. (Finding the sum of some numbers means adding them up.)

```java
int sum = 0;
for (Pixel pix : aPicture.getPixels())
{
    // Add code here
}
```

15. (5 points) The following method should (A) calculate and then (B) print the weighted average of number N1 with weight W1 averaged with number N2 with weight W2 (provided as parameter values). Add the code to make that happen.

```java
public void ave(double N1, double W1, double N1, double W2) {
    double average;
    // Write code to compute the answer.
    // Write code to print the answer.
}
```
16. (30 points: Write a method AND explain the purpose of each thing you write!) Your job is to add a method to the Picture class that will put centered black crosshairs into the Picture referred to by this. When myPict.crossHairs() is called, your code should change the completely white Picture labelled BEFORE to the one labelled AFTER.

BEFORE

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>

AFTER

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>

In order to do this, the computer must make several computations and it must be controlled by loops to repeat certain operations. The things you must program are listed by number below:

1. Get the Picture's height (with getHeight() ).
2. Get the Picture's width (with getWidth() ).
3. Compute half the height (approximately).
4. Compute half the width (approximately).

**Program two separate loops!**
5. One for loop to blacken the correct row of Pixels. (The horizontal crosshair should be only 1 Pixel wide.)
6. One for loop to blackens the correct of Pixels. (The vertical crosshair should be only 1 Pixel wide.)
7. Get a Pixel pix of the horizontal crosshair.
8. Get a Pixel pix of the vertical crosshair.
9. Actually blacken one Pixel in the horizontal crosshair (use pix.setColor( Color.black ); or equivalent.
10. Actually blacken one Pixel in the vertical crosshair.

Write Java code into the method, as complete and correct as you can make it; AND (for 20 of the 30 points!!!) LABEL WITH (1), (2), .... (10) exactly each piece of code whose purpose is the numbered item, to prove that you understand the purpose of everything you write!! ((1), (2), ... (10) will NOT appear in numeric order!)

```java
public void crossHairs() {
    // Code here
}
```