Programming questions will ask you to select or write code (and write methods) to make the Turtle graphics shown here.

INSTRUCTIONS

This is a closed book and note examination, with 18 1/2 x 11 inch paper sheet of notes allowed. No other written material and no interpersonal or other communication or discussion (except with the prof. or proctors) or sharing of information is allowed. Cell phones or any electronic devices (other than a calculator) are strictly forbidden! 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. Failure to abide by any of these terms will result in a grade of zero for the exam.

NAME ____________________________________________________________

NET ID ___________________________________________________________

WRITE THE NAME OF YOUR TA:______________________________________

------- For scoring use only. Do not write below this line --------------

Sample Exam published to help with Spring 2013 review and study. Some changes were made to adapt examples to the Spring 2013 version but the essentials of the questions are unchanged.
Section 1 – Multiple Choice and Short Answer (10 questions, 3–4 points each)
Select the best answer to the question from the choices provided.

1. (3) Which is not a method of the Turtle class? Hint: THINK of what Turtles have and can do, versus what they don’t have or can’t do.
   a. getPixel( )
   b. forward( )
   c. setPenWidth( )
   d. setPenColor( )
   e. turn()

2. (4) Key principles of Java syntax: (write in two answers):
   Every { must be matched by one ________
   Every ( must be matched by one ________

   Don’t worry about the picky exception: Except when these characters are within quotation marks, like in "zz{xy" or '{'.

3. (3) Key principle of Java meaning: = does not mean "equals" as it does in mathematics. What does it mean in Java?___________________________

4. (3) Here's a fragment of Java code. TRACE what the computer does line by line!! Show your thoughts by writing and overwriting in the boxes to demonstrate what the computer actually does.

   ```java
   int X;
   int Y;
   X = 5;
   Y = 7;
   X = Y;
   Y = X;
   System.out.print(X);
   System.out.print(Y);
   ```

5. (3) Then answer: What does it print?
   a. 5 7
   b. 7 5
   c. 5 5
   d. 7 7

6. (3) What is the result of (int)(16.2/2.0) ?
   a. 8
   b. 7
   c. 8.1
   d. 16
   e. The answer is undefined
7. (3) Why do Guzdial and Ericson's Turtles require an int parameter in the Turtle's forward method? (G&E Turtles required you to "cast" doubles to ints with (int). This question requires you to think. Some of the "wrong" choices might be false statements!) Spr13: also setRed(int)

   a. Turtles are confined to an (integer) pixel grid, so the distance between pixels is always an integer (whole number), no matter which direction, even slanted, does the Turtle go.
   b. They made programming their Turtles easier.
   c. The code they wrote in Turtle.java (and SimpleTurtle.java) requires it, even though distances really can be non–integer quantities, like √2
   d. Write in your own answer if you have a better choice

   ___________________________________________________________________
   ___________________________________________________________________
   ___________________________________________________________________

8. (3) When is TEST performed in

   while( TEST ) { BODY } ?

   a. Only before the first time BODY is run.
   b. Before the first time BODY is run and after the last time BODY is run, only.
   c. Before the first time BODY is run and after each time BODY is run.
   d. Only when the loop finishes.
   e. Only when the loop starts.

9. (3) What kind of data or information LOCATES a particular Pixel in the 2–dimensional array that comprises the digital Picture of Ms. Ada Byron, Lady Lovelace who programmed Babbage's Analytical Engine on the cover of the exam?

   a. The color stored in the Pixel, a combination of three integer 0–255 numeric intensities of red, green and blue light.
   b. The name of the Pixel
   c. Variables
   d. Two integers
   e. Approximate data

10. (3) Which straight line sequence of Java code makes the ArtisticTurtle draw 4?

   a. forward(100);turn(-90-45);forward(75);turn(-90-45);forward(75);
   b. forward(100);turn(-90+45);forward(75);turn(-90-45);forward(75);
   c. forward(100);turn( 90+45);forward(75);turn( 90+45);forward(75);
   d. forward(100);turn( 90+45);forward(75);turn( 90+45);forward(75);
   e. forward(100);turn(-90-45);forward(75);turn(-90+45);forward(75);

   (Notes: Unlike in math, positive angles mean clockwise turning. 90–45 is 45 90+45 is 135 –90–45 is –135 –90+45 is –45 )
11. (10 points) **CALLERs** initiate a method calls and **CALLEE**s get called, that is, receive calls. Check EXACTLY ONE box in each row.

<table>
<thead>
<tr>
<th>Activity</th>
<th>CALLER (initiator)</th>
<th>CALLEE (recipient, who runs the method body code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>initiates the call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>determines the parameter values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reads and uses the parameter values</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>
12.(10 points) Acting like a WW II human computer is handy for understanding what a given piece of code actually does, sensible or not. (Brief: Do what computer does.) Consider the follow code fragment:

```java
int count = 0; //HINT: Write a zero on the count ticket FIRST!
int index = 0; //HINT: Remember the previous hint.
    //Also index is a different variable from count.
while ( index < 2 )
{
    count = index + 3; //Hint: Remember all previous hints.
    count = 2 * count;
    index = index + 1;
}
System.out.println(count);
```

What is printed?(just one number) _____________________________

Use the boxes to simulate the ledger paper or tickets (i.e., variables) used by a computer who (or which) follows directions expressed by the Java code. Keep track of the number currently written in each variable (a) after the first two lines of code, and then, (b) during each and every line in every repetition of the code in the for loop’s body.

For (b), whenever the computer writes a number into a variable, demonstrate your awareness of that by writing the correct number in the box just below the box where you last wrote. Remember to heed the names (index and count) of the two variables. (There are more boxes than you need because the loop stops before you will use them up.)

```
<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>
</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 – Coding!

13. (12 points) Fill in the code below to compute (and then print) the sum of the whole numbers from 1 to 100, by (a) making the loop body add the next number to the sum so far and (b) making the loop body keep the next number up to date.

Then, clearly label with (a) and (b) the lines of code you wrote whose purposes are to given by (a) and (b) above, respectively.

```java
int sum;
sum = 0;
int next;
next = 1;
while ( next <= 100 )
{
    // Label for (a) here.

    // Label for (b) here.
}
System.out.println( sum );
```

14. (A1 8) Make your own `ArtisticTurtle` method for drawing a 200 pix. tall two-prong fork (or snake's tongue), similar to what's on the cover of this exam. (A2 8) Make sure that before returning, your method makes `this` (a `Turtle`) go back to its original location and head pointing direction.

Your method should begin:
```
public void drawFork( )
{
    this.forward( 200 ); //You write the rest!
}
```

(B 8) Code a main method that makes a `World`, then a `Turtle` in that `World`, and then uses a loop to draw a circle of 15 forks using the method you have made. (360 degrees/15=24 degrees)

15. Assume you solved the previous problem correctly. (6) Make a parametrized `ArtisticTurtle` method with one parameter `int nForks`. When called, (A 3) it should first print "Hey, I'm going to draw a number of forks on your World." (B 3) Second it should print what number of forks it should draw.

(C 11) Third, it should use your `drawFork` method in a loop to make `this` (an `Artisticturtle`) draw a circle of forks with `nForks` (number of) forks spaced at equal angles. Strategy hint: Program it to divide (with /) 360.0 degrees by `nForks` to calculate how many degrees the `Turtle` should turn after each arrow.

Notes: (1) Do NOT recode the body of `dofork()` again.
(2) Do NOT code another `main( )` method! Only code a new method that gives `ArtisticTurtle` the capability to draw a circle of ANY, unpredictable number

Page 6 of 8
of forks. The number is determined by the caller, NOT the callee code. You write the callee code.

**Answer Question 14 on this page (24 points of programming!)**

Both programming questions will be graded on bases of CLARITY, NEATNESS, UNDERSTANDABILITY, REASONABLE EFFICIENCY, INDENTATION, ETC, in addition to logical correctness. Minor syntax errors will be forgiven.
Answer Question 15 on this page (23 points of programming!)