Your Name:_____

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Georgia Tech community.

CS 1316 Exam 3
Fall 2009

Section/Problem	Points Earned	Points Possible
1. Terms & Concepts		33
2. Short Answers		12
3. Trees		9
4. Tree Question		5
5. Queue Operations		3
6. Code Comprehension GUI		10
7. Write Code: Musical Instrument		15
Total Points:		87

 Your Name:_____

1. Terms & Concepts (33 points)

For each of the terms below, write 1 or 2 sentences defining the term and proving you understand what your definition means. You may include an example if you think it will help your explanation. Be concise and precise.

1. abstract (super) class - A class meant to be subclassed that can not be instantiated

2. anonymous inner class - An unnamed class created inside of another class. Typically created as part of a GUI to handle user interface events.

3. binary search tree - A binary tree arranged so that all values to the left of the current node are smaller than it's value, and all values to the right of the current node are equal to or greater than the current node's value. Provides log2(N) search time.

4. final static variable - A constant variable that can't be changed.

5. interface - A listing of method signatures (without bodies!) that acts as a contract. If a class says that it implements an interface, it must define each of the specified methods.

6. layout manager - a java GUI object that handles the layout of sub-components within a java GUI Component. Examples are BoxLayout, FlowLayout, etc.

7. leaf node - Nodes in a tree that do not have any children of their own.

8. queue - An Abstract Data Type (ADT) that must implement two methods (enqueue and dequeue). Items placed in a queue using the enqueue method emerge from the queue in a FIFO (First In First Out) order when the dequeue method is called.

9. static method - A method that belongs to a class, and is shared by all objects instantiated from that class. Class methods can be called directly from the class, and do not require an object to be instantiated before they can be used. However, static methods can not access object variables.

10. this (keyword) - A reference to the current object. (How methods know which object they are associated with.)

11. void (keyword) - A keyword that specifies "nothing", used to indicate that a method does not take any parameters, or does not return anything.

 Your Name:_____

2. Short Answer (12 points)

a. What causes a null pointer exception? (2pts)

The Java runtime (JVM) throws a Null Pointer exception when the user's code tries to access something in an object (field or method) using a reference that is not pointing to an object. (Instead, the reference is actually pointing to nothing, or "null".

b. What is the difference between a compiler error and a runtime exception? Give an example of each. (3pts)

A compiler error (syntax error) is caused because something about your code is incorrect, and not valid Java syntax. A runtime error is caused when your code does something that is incorrect, such as dividing by zero, trying to open a file that does not exist, or trying to access an object field or method using a reference that points to null instead of an object.

c. Multiple Choice: Interfaces contain the following: (check all that apply) (1pt)

O method signatures <CORRECT ANSWER 1>
O method bodies
O static, final fields <CORRECT ANSWER 2>
O instance variables

d. What is the main difference between a continuous and a discrete event simulation? (3pts)

A continuous simulation evaluates the model at every timestep, while a discrete event simulation calculates the time of the next “interesting” event and skips to it.

e. How is sampled sound stored in a computer? What does each “sample” represent? (3pts)

The amplitude of a sound wave at each sample point in time is stored using a number between -32K and 32K. For example, 44100Hz sampling takes that many samples a second.

 Your Name:_____

3. Trees (9 points)

a. If the in-order traversal of the balanced binary search tree T is: M N O P Q R S, draw the tree:

```
                P
        N                R
M           O           Q           S
```

Grading: 3 if correct, 0 if not balanced.

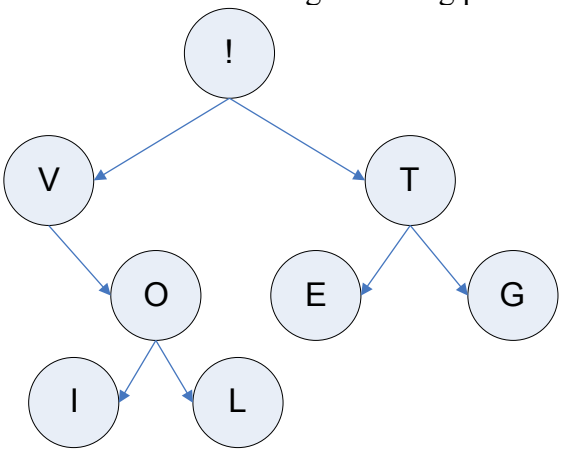
b. If the pre-order traversal of the binary search tree T is: F E D C B A, draw the tree:

One valid example is a tree that extends only to the left:

```
          F
        E
      D
    C
  B
A
```

Grading: 3 if correct, (-1 point for each node pairs that are out of order)

c. Traverse the following tree using post-order traversal. Write out each letter.



ILOVEGT!
Grading: 3 points if all correct. 1 point if "GT" are present in the correct order.

 Your Name:_____

4. Tree Question (5 points)

Consider the method X:

```
public static int X( TreeNode t) {
      if (t.left == null && t.right == null) {
            return t.data;
      } else if (t.left == null) {
             return Math.min(  t.data,   X(t.right)   );
      } else if (t.right == null) {
            return Math.min(  t.data,   X(t.left)  );
      } else {
            return Math.min(    Math.min( t.data, X(t.left) ),   X(t.right)   );
      }
} // end X
```

If you call X giving it a non-empty tree, what statement can you make about the value that X returns?

The code will return the minimum element in the tree. Grading: 5 points for the correct answer. 4 points if they say it returns the minimum of the leaf nodes. 3 points if they indicate they know what it will do, but don't make a correct statement about what the value X returns. 0 if they are wrong.

5. Queue operations (3 Points)

Examine the following series of code statements. Draw the contents of the queue after the following code executes. Clearly mark the front and back of the queue.

```
Queue x = new Queue();
x.enqueue("A");
x.enqueue("B");
x.enqueue("C");
String temp1 = x.dequeue();
String temp2 = x.dequeue();
x.enqueue("D");
```

Head: C D :Tail

(Not required, but if they specify that temp1=A and temp2=B that's fine)

1 point for each letter (C/D), 1 point for the correct ordering of C,D.

(-1 point for each other letter like A,B)

 Your Name:_____

6. Code Comprehension - GUI's (10 points)

Draw the GUI that will be made when the following code is run. If no text would be shown on the widget when the GUI is displayed, write the name of the type of widget instead (button, slider, etc)

```
import java.awt.BorderLayout;
import java.awt.GridLayout;
import javax.swing.*;
public class Test3GUI extends JFrame{
      public Test3GUI(){
            super("Test3GUI");
            this.getContentPane().setLayout(new BorderLayout());
            JLabel label1 = new JLabel("Lable1");
            JTextField text = new JTextField(20);
            JButton b1 = new JButton("Button1");
            JButton b2 = new JButton("Button2");
            JButton b3 = new JButton("Button3");
            JSlider slider = new JSlider();
            JComboBox cb = new JComboBox();
            JPanel p1 = new JPanel();
            p1.setLayout(new GridLayout(2,1));
            this.add(label1, BorderLayout.NORTH);
            p1.add(b1);
            p1.add(b2);
            this.add(p1, BorderLayout.EAST);
            this.add(cb, BorderLayout.WEST);
            this.add(slider, BorderLayout.SOUTH);
            this.add(text, BorderLayout.CENTER);
            this.add(b3, BorderLayout.CENTER);
            this.pack();
            this.setVisible(true);
      }
      public static void main(String[]args){
            Test3GUI gui = new Test3GUI();
      }
} // end class
```

Grading: one point for each of the following elements: Frame name (Test3GUI), border layout (everything being in the right spots), label1, combo box, slider, grid layout / panel, button 1, button 2, button 3, textfield (hidden behind button3)

 Your Name:_____

7. Create an Object: Musical Instrument (15 points)

You are given the following interface:

```
public interface MusicalInstrument {

// returns the sound of the instrument
public Sound getSound();

//returns the name of the musical instrument
public String getName();

}
```

Your task is to create a new class called Piano that fully implements the MusicalInstrument interface. Your new class should also contain the following:

- A public instance variable of type Sound
- A constructor that accepts a single parameter of type Sound and sets the sound instance variable to the input sound.
- An additional method called appendNPlay that takes in a sound "A" and a double "factor", and does not return anything. This method should append the input sound to the instance variable sound. After it does that, it must also modify the volume of the resulting sound by the factor parameter. (Multiply each sound sample by the factor, and put the new value back into the sample.) Once you have completed your manipulations, play the resulting sound!

Methods in the Sound class that you might find useful:

- Sound append (Sound appendSound)
- void play()
- SoundSample [] getSamples()

Methods in the SoundSample class that you may find useful:

- int getValue()
- void setValue(int)

Write your code on the back of this page:

 Your Name:_____

```
public class Piano implements MusicalInstrument {

public Sound mySound;

public Piano (Sound s) {
   mySound = s;
}

public Sound getSound() {
   return mySound;
}

public String getName() {
  return “Piano”;
}

public void appendNPlay(Sound A, double factor) {
 Sound newSound = mySound.append(A);

 SoundSample[] ss = newSound.getSamples();

 for (int i = 0; i < ss.length; i++) {
   int currentValue = ss[i].getValue();
   ss[i].setValue(currentValue * factor);
 }

  newSound.play();
}

}
```

Grading:
5 pts - Implementing the interface (1 for "implement", 2 for each function)
1 - Class variable mySound & associated constructor
9 - for Append & Play
1 - Header correct
2 - Appending input parameter to mySound
1- Storing result in a new Sound variable
2 - Getting an array of Sound Samples out of the new sound
2 - Apply the "factor" to the sound samples and saving it back into the samples
1 - play the sound.