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 2
Fall 2009

Section/Problem	Points Earned	Points Possible
1. Terms & Concepts		21
2. Short Answer		6
3. Program Comprehension		5
4. Sound Manipulation		15
5. List Question		14
6. Fix Replace		20
Total Points:		81

 Your Name:_____

1. Terms & Concepts (21 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:

1. array - homogeneous data stored together in memory. Easily indexed using the bracket operator (e.g. myArray[3]).

2. block - Section of code (statements) between brackets such as used by a for or while loop or conditional statement.

3. casting - Specifying to java that it should treat one data type/object as another or convert primitive types. e.g. int q = (int) 35.5

4. class - Definition of behaviors and fields as a blueprint from which objects are instantiated.

5. inheritance - The process in which a subclass “borrows” or “inherits” behaviors and fields from it's superclass.

6. object - Instantiation of a class, has it's own object fields that are different from all other objects of the same type/class.

7. static field (variable) - A variable that belongs to a class and not object instances. (It can be accessed and modified by all objects of that class, but is shared between them.)

2. Short Answers (6 points)

For each of the following questions, write a 2-4 sentence answer:

2a. How is sampled sound stored in a computer? What does each “sample” represent?

The amplitude of a sound wave at each sample point in time is stored using a number between approximately -32K and 32K. For example, 220,50 Hz sampling takes that many samples a second.

2b. How are images stored in a computer? (How are colors for each pixel represented, how much space does a pixel take up in bits and bytes?)

Each pixel in an image is represented by 3 eight bit integers (0-255) that represent the intensity of Red, Green and Blue light respectively.

 Your Name:_____

3. Program Comprehension (5 points)

The recursive function printIt is shown below, what is output as a result of the call `Print.printIt("rambling", 4);` ?

```
public class Print {
  public static void printIt(String s, int numChars) {
    if (numChars > 0 ) {
      System.out.print( s.charAt(numChars-1) );
      printIt(s, numChars-1);
    }
 }
}
```

Grading: bmar – 5 points (-1 point if missing a single letter, -1 points if it's vertical, otherwise, it's a zero)

4. Sound Manipulation (15 points)

Write a static method called **scaledMix** which takes in as parameters *two* Sounds (A and B). The scaledMix method must scale the second (B) sound to be the same length as the first sound (A) and then mix the two sounds together equally. The resulting sound should be returned. You do not need to write an enclosing class, just a static method.

Here is some information about three methods defined as part of the Sound class that you may want to use:

`int` `getLength()` - Method to return the length of the sound as the number of samples.

`Sound` `scale(double factor)` – Scale (in length) up or down myself by the given factor and return the result (1.0 returns the same, 2.0 doubles the length, and 0.5 halves the length)

`Sound` `mix(Sound mixIn, double ratio)` - Mix myself with the mixIn sound, with a percent ratio of input that can vary from 0.0 to 1.0.

 Your Name:_____

```
public static Sound scaledMix( Sound A, Sound B) {

        double lengthA = A.getLength();
        double lengthB = B.getLength();

        B = B.scale(   lengthA / lengthB );
        Sound retVal = A.mix(B, 0.5);
        return( retVal);
}
```

Grading:

4 points for header (static, Sound, Parameter types are Sound, A,B)

5 points for grading the correct factor (-1 if they didn't ensuring double division)

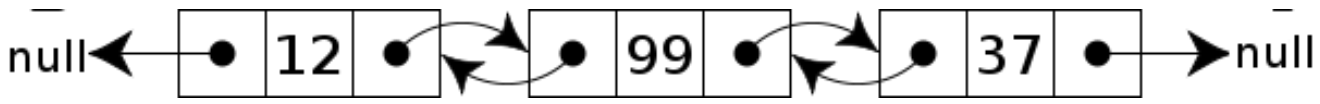
1 point for using scale

3 points for using mix (1 point for 0.5 ratio, other 2 for correct use)

2 points – returning the sound.

5. List Question (14 points)

Each element in a doubly linked list has a next pointer that holds a reference to the next node in the list, as well as a prev pointer that holds a reference to the previous node in the list.




```
// Remove node q from a doubly-linked list
private void doDelete( Node q) {
    q.getPrev().setNext( q.getNext() );
    q.getNext().setPrev(  q.getPrev() );
}
```

 Your Name:_____

5a. List 3 examples where the code above will fail to properly maintain the doubly linked list structure or throw an exception when removing node Q: (6 points total, 2 per part)

1- Removing the First Node (2pts)

2-Removing the Last Node (2pts)

3-If Q is null (2 pts) (Length of list < 3 only gets 1 point, as it's a restatement of the above two)

5 b. Re-write the doDelete method above to fix the problems, so that it will correctly remove the node Q in all cases. (8 points total)

```
// Remove node q from a doubly-linked list
private void doDelete( Node q) {
    // Make sure that q is not null. If it is, give up!
    if (q == null)
     { return();  }

    // Make sure we are not the first node in the list before
    // trying to affect the node in front of us!
    if( q.getPrev() != null) {
         q.getPrev().setNext( q.getNext() );
    }

    // Make sure we are not the last node in the list before
    // trying to affect the node behind us!
    if (q.getNext() != null) {
        q.getNext().setPrev(  q.getPrev() );
     }
}

Grading:
 +2 for doing nothing if Q is null
 +3 for checking to make sure we are not the first node before attempting
    to modify the node in front of us.
 +3 for checking to make sure we are not the last node before attempting to
    modify the node behind us.
```

 Your Name:_____

6. Fix Replace (20 points)

Review the replace method in the appendix at the end of this test. The replace method will currently replace the contents of the sound with the contents of a newSound, if it matches oldSound. Rewrite the replace method so that it actually compares the sounds (sample-by-sample) rather than simply comparing the filenames to determine if the current sound is one that needs to be replaced. See the last page of the test appendix for source code and API functions that will be helpful.

```
public  void replace( Sound oldSound, Sound newSound) {
  Boolean sameFlag;
  if  ( oldSound.getLength() != this.mySound.getLength()  ){
       sameFlag = false;
   } else {
        sameFlag = true;
         for ( int I = 0; I < this.mySound.getLength(); i++) {
              if (this.mySound.getSample(i).getValue() != oldSound.getSample(i).getValue() ) {
                  sameFlag = false;
                 } // end if
      } // end for
} // end else
if (sameFlagg) {
   this.mySound = newSound;
 }

} // end method.
```

Grading:

+3 Checks to see if the length is the same. If not, do nothing.

+9 (If length is the same, iterates over the array and compares this.mySound-> oldSound)

+5 detects whether or not they are equal

+3 If sounds are equal, replaces the song, if not equal, does nothing.

 Your Name:_____

Test Appendix:

```
/**
 * Replace my sound with the other sound if I match "oldSound"
 * Two sounds are equal if they have the same filename
 * @param oldSound sound to be replaced
 * @param newSound sound to put in its place
 */
public void replace(Sound oldSound, Sound newSound) {

    if (this.getFileName().equals(oldSound.getFileName())) {
      // Replace my sound Samples. TODO
      SoundSamples [] ss = this.getSamples();
      SoundSamples [] ssSource = newSound.getSamples();
      for(int i = 0; i < this.getLength(); i++ ) {
          ss[i]setValue(  ssSource[i].getValue());
      } // end for
    } // end if file names are the same.

} // end method replace.
```

API:

Here are a few methods you may want to know about that the Sound object supports:

SoundSample getSample(int index) - Method to create and return a SoundSample object for the given index.

SoundSample[] getSamples() - Method to create and return an array of all SoundSample objects contained in this Sound.

Here are a few methods you may want to know about that the SoundSample object supports:

int getValue() - Method to get the value of this sample as an int

void setValue(int value) - Method to set the value of this sample