Practice Final Examination

We have omitted the space for the exam in order to save trees, but please please please, **set aside 3 hours to do this exam before you look at the answers!**

Final Exam: Gates B01&B03, August 12, 2011 - 12:15pm-3:15pm

Problem 1 – Short Answer (11 points)

1a) Suppose that the char array `letters` has been declared and initialized as follows:

```java
char[] letters = { 'a', 'b', 'c', 'd', 'e', 'f' };
```

This declaration sets up an array of six elements with the initial values shown in the diagram below:

```
  letters
    'a' 'b' 'c' 'd' 'e' 'f'
```

Given this array, what is the effect of calling the method

```java
mystery(letters);
```

if `mystery` is defined below as:

```java
private String mystery(char[] array) {
    String result = "";
    for(int i = array.length-1; i >= 2; i--){
        if(i < array.length/2 + i%4) {
            result += (char)((i-1)*(i-1) + 'h');
        } else result += array[i];
    }
    return result;
}
```

Work through the method carefully, and indicate what is returned by the call to `mystery` below:

1b) Below is a method Osvaldo wrote when he was developing the Hangman solutions*. Unfortunately, as is often the case with Osvaldo’s code, the method is very buggy. Circle the bugs in the code and write a short sentence for each explaining the precise nature of the problem.

```java
public class HangmanLetterTester extends ConsoleProgram {
    //Instance Variables
    private char ch;
}
```

* No, really. This is actually Osvaldo’s original solution code.
//Lots more Hangman code (possibly)

/*
 * Method: revealLetter(word, display, ch)
 * --------------------
 * Replaces '-' with ch in display, wherever ch
 * would occur in word.
 * word     – hidden word not shown to player
 * display     – word displayed to player, with dashes
 * for letters not yet guessed
 * ch        - letter guessed by player
 * Example:
 * revealLetter("bookkeeper", "b--kk--p--")
 * should return the string "b--kkeepe--".
 * This means our run should print out:
 * Guessing e on b--kk--p-- gives us: b--kkeepe--
 */

public void run() {
    ch = 'e';
    String word = "bookkeeper";
    String soFar = "b--kk--p--";
    String answer = revealLetter(word, soFar);
    ch = 'f';
    println("Guessing e on " + soFar + " gives us: " + answer);
}

private String revealLetter(String word, String display) {
    String nuWord;
    for(int i = 1; i < word.length(); i++) {
        char ch = word.charAt(i);
        if(ch.equals(this.ch)) {
            nuWord += this.ch;
        }else{
            nuWord += display;
        }
    }
    return nuWord;
}

Problem 2 – Arrays (19 points)

Odometer

Since most of you have driven a car or at least have aspired to do so, we’re going to have
you write a program that simulates an odometer. What is an odometer and what does it look
like? Well it’s the reading that comes from how many miles you have driven a car, and
looks like this:
What you are going to do is write a program that is going to simulate an odometer by creating an array that will hold the 7 integers, with each element holding a digit from the odometer.

You are responsible for writing two methods, each its own subproblem. For both problems you can always assume that the array will always be an array of 7 ints.

**Problem 2A**

The first method is an `odometerString` method, which will create a `String` based on the contents of the integer array. Thus if we were to call `odometerString` with the contents from the array shown above, the string returned would be.

“5,339.9”

There are a couple of things to notice here: they are:

1) no leading zeroes were included in the string. (but if the odometer has less than a mile, the string formed should have at least two places, ie “0.0”
2) There is a decimal point inserted to correctly represent the tenths place
3) There is a comma to separate the thousands place, and that should be added as well.

```java
private String odometerString(int[] odometer) {
```
Problem 2B

The second method we would like you to implement is the `add1TenthMile` method, which given the 7 digit array, will add 1 tenth of a mile to the odometer. Thus in the previous picture, if we had 0 0 5 3 3 9 9 and we added a tenth, the odometer would read 0 0 5 3 4 0 0. Give it 1 1 1 1 1 1 2 and call `add1TenthMile`, the odometer should read 1 1 1 1 1 1 3. Also if you are familiar with cars, you should know what would happen in this scenario as well.

If we were to call the `add1TenthMile` method, the odometer should roll back to all zeroes. Remember that each element in the array is only supposed to represent 1 digit, so write your method accordingly.

```java
private void add1TenthMile(int[] odometer) {

}
```

Problem 3 – Graphics and Interactors (20 points)
(We went over this in class, so this should be easy, but try not to look at your notes!)

Draw a Progress Bar!

Your job is to write a GraphicsProgram that draws a progress bar on the screen that responds to user events.

The progress bar is simply 2 `GRects`: an outline and the filler. The outline is simply a `GRect` that is centered in the screen and is `WIDTH` pixels wide and `HEIGHT` pixels high. The filler rectangle is a filled, pink `GRect` that is the same height and starts at the same x,y as the outline, but that initially starts with a width of 0. As the bar progresses (when the user hits `Increment`) the filler will grow to the right, until it is the same size as the outline.

Initially the window will start out empty, and there will be 3 buttons in the `SOUTH` as such:
The buttons will work as follows:

**Start**  -- when the user presses `Start`, this simulates starting a long task. This is when our progress bar shows up on the screen, initially empty.

**Increment**  -- when the user presses `Increment`, this simulates the task progressing by a single step. We draw a little bit more of the progress bar filling up. Our program assumes that each task is broken up into `NSTEPS` steps. Here we show what the program looks like if `NSTEPS` is 5, as the user repeatedly presses the `Increment` button. Notice how the progress bar is full after `Incrementing` 5 times, and how on the 6th time we press `Increment` the progress bar disappears. We are simulating that the task has `Ended`. So pressing `Increment` once the progress bar is full is the same as pressing `End`.

**End**  -- when the user pressed `End`, this simulates the task finishing, and we remove the progress bar.
This can possibly happen early, before we’ve reached the full N STEPS. Here we show the user pressing Increment 3 times, and then pressing End.

Other things to keep in mind:
- Pressing Start at any time should reset the progress bar, so that you only see the outline, and the filler has width 0.
- Pressing End at any time should remove everything from the canvas.
- Pressing Increment once the bar is full should also remove everything from the canvas. Further presses of Increment should do nothing, until the user has pressed Start again to restart everything.

```java
public class ProgressBar extends GraphicsProgram {
    public static final int NSTEPS = 5;
    public static final int WIDTH = 200;
    public static final int HEIGHT = 20;

    // a) Put any IVARS you need here

    // b) write your program here
```
Problem 4 – Java Programming (20 points)

Sudden Death Reality TV Simulation

Alright all you American Idol fans out there...(all two of you), I'm sure all of you have watched some type of reality TV show where each week (round) they pick three people to be in the “bottom three” or “sudden death” and from those three one person is eliminated from the show. Since this process of picking the people to be nominated for elimination and the person that actually gets eliminated seems totally random what we are asking you to do is build a simulation of the contest that will help the TV executives script the whole show.

Write a program that will given a list of names, pick three to be nominated for sudden death, and from those eliminate one. This "pick 3" style elimination will keep going until one person is left and that person can be declared the winner. A sample run of the program is below. Notice how it picks the winner at the end. Your program is responsible for printing out the same type of statements the whole way through.
Here is a sample run with only five contestants.

Because this is a java programming problem you are responsible for making everything to get this to run through the whole simulation. All I have provided you with is some code to
get you started. You cannot declare anymore instance variables, but I will tell you that you must use at least 1 `ArrayList` to hold the names. Another word of caution is to think about how you can make sure not to pick the same person twice for sudden death. One way to solve that would be to use a second `ArrayList` that temporarily holds the people that are in sudden death. Also, remember that if you call `println` on a variable of type `ArrayList`, it will automatically print the list for you, as shown above with the brackets and commas. The order in which the names appear does not matter, so you can just pass the object to `println`. Make sure to declare the `ArrayList` objects you need and any helper methods that you need to get started. (In your code, please have an `EliminateOnePerson` method, that chooses the three and then takes one person out of the arraylist, or at least show us the area in your code that does this). Because you have to declare the `ArrayList`, you will also be responsible for putting the names into the `ArrayList`.

```java
public class SuddenDeath extends ConsoleProgram {
    private RandomGenerator rgen = new RandomGenerator();

    public void run() {
        String[] names = {"Aileen", "Rex", "Brandon", "Cheryl", 
                          "Jeff", "Nia", "Felicia", "Gale", "Sheila"};

        // Additional space for problem 4
    }
}
```

**Problem 5 – Strings (12 points)**

We've already seen how you can encode secret messages using Ciphers. Now we're going to examine another encoding strategy...one that you probably also used in 3rd grade. We might call this strategy **Acronym Encoding**. To decode a message that has been encoded with **Acronym Encoding**, like "go round on super saturday" you simply take the first letter of each word: "gross".

We're confident that you could write a program to decode a message. (In fact, the book already has an Acroynm example that does just this). **Instead**, your job is to write a method `encode()` to do the encoding.

Given a secret message as a `String`, `encode()` should replace each letter in the string with a word starting with that letter, and return the resulting `String`. ie encoding the message "monday, noon" might produce the result:

"mailman over night doesn't appear yummy, nevertheless onward on norwegians"

Assume you are given a function that you can call:

```java
public String randomWordStartingWith(char ch)
```

that, given a character, returns a random word that starts with that character.
Write the `encode` method below

```java
private String encode(String message) {
    // Additional space for problem 5
}
```

**Problem 6 – Data Structure Design (18 points)**

**Acronym Encoding, part 2**

Here you are going to finish our Acronym Encoding example by writing the `randomWordStartingWith()` method. To do this you will need to write a program that reads in a dictionary file of words, and stores the words in some kind of data structure. Your data structure should be designed so that it's easy to go from a letter, like 'g', to a list of all the words in the dictionary that start with 'g'.

Your `randomWordStartingWith(char ch)`, then, will get the list of words that start with the given char, and return a random word from that list.

Your job in this Problem will be to:

a) read in the dictionary file and store all the words in a convenient data structure, and

b) write `randomWordStartingWith(char ch)` that uses this data structure.

The hardest part is thinking about how you will store the words from the file so that it will be easy to write `randomWordStartingWith()`. Note: Keep in mind that it's not just as simple as storing all the words in one big `ArrayList`. If the letter is 'g', you can't just look through the list and find the first word that starts with 'g'. You have to pick a random word from the list of ALL words that start with the letter 'g'.

a) **Declare your ivars here, and write the `readFile()` method below.**

The `run()` is provided for you.

```java
public void run() {
    readFile("ShorterLexicon.txt");
    // This loop calls your encode() from Problem 3, which in turn
    // will use the randomWordStartingWith() that you write in (b)
    while(true) {
        String text = readLine("Message: ? ");
        String encodedText = encode(text);
        println("Encoded Message is: " + encodedText);
    }
}
```
private void readFile(String filename) {
}

b) write the randomWordStartingWith() method

private String randomWordStartingWith(char ch) {
}