Computer Sciences 302
Midterm Exam 2, 20%

Thursday 11/18, 2010

Print last name: ______________________________, first: ______________

Signature: ______________________________ CS login: ______________

Circle Your Lecture	**Lec 1**	**Lec 2**	**Lec 3**	**Lec 4**	**Lec 5**	**Lec 7**	**Lec 8**	**Lec 9**	**Lec 10**
	Skrentny	Skrentny	Dalibor	Dalibor	Finn	Dan	Finn	Alex	Alex

Before you Begin:
(1) Turn in your UW student ID.
(2) **On this examination booklet:**
- Print and sign your name above.
- Write your CS login and circle your lecture above.
(3) Check that there is a total of 10 pages in this exam.
(4) You may not use any notes, books, calculators (or any other electronic devices), or neighbors on this exam. Turn off and put away your cell phone, pager, pda, etc. now.
(5) The exam is intended to take about 100 minutes, but **we will give you 2 hours to complete the exam**.
(6) We can't provide hints but if you need an exam question clarified or feel that there is an error, please bring this to our attention. If needed, **corrections will be written on the board**.

When you've Finished:
(7) Turn in this examination booklet and make sure we return your ID.

Taking the Exam

Write your answers to the questions in this examination booklet. Write neatly and format your Java code by indenting and spacing for readability. Comments are not required, *but if something isn't specified, do something reasonable and state your assumptions*.

Note a REFERENCE is provided at the END OF THE EXAM, which you should review when the exam begins.

Parts	Number of Questions	Topic	Possible Points	Score
I	1	Coding a Complete Program	18	
II	2	Using Pre-built Classes	15	
III	1	Coding an Instantiable Class	18	
IV	2	Using Arrays and ArrayLists	15	
		Total	66	

Part I Coding a Complete Program *[1 question, 18 points]*

[/ 18 points] Use good programming practices to **write a complete Java program** that gives advice on what to use given the weather outside. When the program is run the user is asked for the temperature (-10 to 90 degrees Fahrenheit) followed by the wind speed (0 - 50 miles per hour). *If the user inputs a value that is out of range, the program displays a brief error message and requires the user to input a new value until one within range is entered.* Next, the program prompts the user to input whether or not there is precipitation of any form. If the user enters the character `y` or `Y` then there's precipitation. Any other input means there's no precipitation. After getting the input, the program then displays advice for what outerwear to use based on the first chart below and then displays additional advice based on the second chart below. After the advice is given the program ends.

	TEMPERATURE			
WIND SPEED	<= 32	> 32 to <= 55	> 55 to <= 70	> 70
0 to 11	winter coat	jacket	light jacket	nothing
> 11	winter coat with hood	jacket with hood	wind-breaker	nothing

	TEMPERATURE	
PRECIPITATION	<= 32	> 32
yes	hat	umbrella
no	nothing additional	nothing additional

You may assume that input of the correct type will be entered. It is not required, but you may use additional static methods if you wish, for example, to reduce the redundancy in your program.

Use this page if additional space is needed to answer Part I.

Part II }Using Pre-built Classes *[2 questions, 15 total points]*

1.) [/ 7 points] Write a complete method that is passed two strings representing phone numbers and returns true if the area codes of two phone numbers are the same, false otherwise. The phone numbers may be in one of these forms:

"`1-AAA-NNN-NNNN`" as in these examples "`1-123-456-7890`" or "`1-608-555-1212`"
"`AAA-NNN-NNNN`" as in these examples "`193-234-5678`" or "`608-123-4321`"

where the area codes are the sequence of digits indicated by AAA as underlined in the examples. If the second example of each form were passed to your method, it would return true since they both have the same area code "608". Any other combination of the examples would return false since they have different area codes. *You may assume that parameters are passed valid strings of the forms above. Good use of the* `String` *methods listed on the reference page at the end of the exam will reduce your work for this question and earn full credit.*

2.) **[/ 8 points] Complete the method below** used for routing the delivery of packages as is done by the postal service and similar companies. For each incoming truck, your method removes all of the packages in that truck that have the same destination zip code as the outgoing truck and adds them to the outgoing truck. The following classes have been implemented: a `Package` class, and a `Truck` class that has the following methods:

`Truck` Class methods:

```
void add(Package p)         //Adds Packages p onto this Truck.
Package remove(int zipcode)
   //Removes and returns one Package from this Truck that matches the
   //destination zipcode. If the Truck has no more packages with the
   //destination zipcode, null is returned
int getZipcode()            //Returns this Truck's destination zipcode.
```

You may assume the outgoing `Truck` *object and the array of incoming* `Truck` *objects both have been properly initialized, but it is possible for the array of* `Truck` *objects to have null elements.*

```
public static void fillTruck(Truck[] incoming, Truck outgoing) {
```

Part III Coding an Instantiable Class *[1 question, 18 points]*

[/ 18 points] Consider implementing a program that manages the videos for a video service such as YouTube. To do this you'll need a means to represent each video object in your program. **Implement the instantiable class**, named `Video`, used to represent a video recording as described below. Use good object-oriented programming practices when choosing public vs. private, instance vs. class members, and constants vs. variables.

- The number of seconds in a minute (i.e., 60)
- A count of the total views (partial or complete) for all of the videos
- The video's title
- The video's author's name
- The video's length in whole minutes and whole seconds
- A count of the times the video has been viewed (partially or completely)
- A count of the times the video has been completely viewed (i.e., beginning to end)

- A constructor that is passed the title, name, and the length of the video in whole seconds. The constructor uses this information to do the appropriate initializations. *You may assume the video's length is positive,* but you must convert the length in seconds to a length in minutes and seconds. For example, if 131 is passed in, then the video's minutes would be set to 2 and the seconds would be set to 11. (Hint: Use integer division `/` and remainder division `%`.)
- A means to safely access each piece of information above. *Note, if you use good object-oriented programming practices you can reduce your work for this part.*
- A method named, `changeTitle`, that allows the video's title to be changed.
- A method, named `view`, that is used to play the video and update the view counts. This method has no parameters and no return value. It calls the `play` method in the `VideoControl` class, *which you should assume already exists*. The `play` method is a *class* method that is passed the video's reference (Hint: Use the `this` reference.) and returns true if the video has been played completely, false if it was stopped before the end of the video.
- A method, named `toString`, that returns a string representing the video. Your string will have in this order the video's title, author's name, and time in the form `minutes:seconds`.

Use this page if additional space is needed to answer Part III.

Part IV Using Arrays and ArrayLists *[2 questions, 15 total points]*

1.) **[/ 7 points]** Complete the method below that is given an array of percentages (positive integers between 0 and 100 inclusive) and returns a new array containing the number of occurrences of each percentage in the given array. For example, if the given array contains 0, 1, 0, 2, 2, 0 then the new array would have 3 for the value of first element since the given array had 3 zeros, 1 for the second element since there was only a single one, 2 for the third element, and 0 for the remaining 98 elements.

```
public static int[] countOccurrences(int[] givenArray) {
```

2.) **[/ 8 points]** This question is divided into multiple parts. You may assume your answers in the prior parts are correct when answering the parts that follow afterward.

A. **Declare and initialize** an `ArrayList` of `String`s, named `products`:

B. **Add the strings** "HoneyCrisp Apples" and "Hungarian Rye Bread" to `products` list.

C. **Add the string** "Bay Scallops" between the two strings already in `products` list.

D. Assume that `products` has been properly initialized as an `ArrayList` and filled with grocery product names. Also assume that an integer, named `limit`, specifies the maximum number of characters that will fit on signs to be printed and used in a grocery store. **Write a code fragment** that displays the index and name of each item in the `products` list that is ***longer*** than the limit.

Exam Reference Page

Methods from the `java.util.Scanner` class:

```
Scanner(System.in)      // Creates a Scanner object that reads from the keyboard.
String next()           // Returns the next token of input.
double nextDouble()     // Scans the next token of the input as a double.
int    nextInt()        // Scans the next token of the input as an int.
String nextLine()       // Returns the next line of input as a String.
```

Methods from the `java.lang.String` class **(*REMEMBER 0-based indexing is used)**:

```
String(String str)      // Creates a String object given another String.
String toUpperCase()    // Returns a String that is the upper-case
                        // version of the String it is called with.
String toLowerCase()    // Returns a String that is the lower-case
                        // version of the String it is called with.
char charAt(int index)  // Returns the char value at the specified index.
boolean equals(String s)// Returns true iff the contents of this string
                        // is the same as that of the string s.
boolean equalsIgnoreCase(String s)// Returns true iff the contents of the
                                  // this string is the same as that of the
                                  // string s, ignoring differences in case.
int indexOf(int ch)     // Returns the index within this string of the first
                        // occurrence of the specified character.
int lastIndexOf(int ch) // Returns the index within this string of the last
                        // occurrence of the specified character.
int length()            // Returns the length of this string.
String substring(int beginIndex)
                        // Returns a new string that is a substring of this string
                        // starting at beginIndex to the end of the this string.
String substring(int beginIndx, int endIndx)
                        // Returns a new string that is a substring of this string
                        // starting at beginIndx up to but not including endIndx.
```

Methods from the `java.util.ArrayList` class **(*REMEMBER 0-based indexing is used)**:

```
Note the E's below are replaced with the particular ArrayList's element type.
void add(E item)        // Adds the specified item to the end of this list.
void add(int index, E item)// Adds the specified item by inserting it into this
                           // list at the specified index.
boolean contains(E item)// Returns true iff the specified item is in this list,
                        // otherwise false.
E get(int index)        // Returns the item at the specified index in this list.
int indexOf(E item)     // Returns the index of the specified item if it is in
                        // this list, or -1 if the item is not in this list.
int size()              // Returns the number of used elements in this list.
```