Computer Sciences 302

Midterm Exam 2

Thursday, November 19th, 2015

100 points (15% of final grade)

Instructors: Deb Deppeler, Jim Williams, and Gary Dahl

(Family) Last Name: _____ **(Given) First Name:** _____

Circle your Lecture: (Deb) Lec 001 8:00 TR (Deb) Lec 002 1:00 TR

(Jim) Lec 003 1:20 MWF (Jim) Lec 004 9:55 MWF (Gary) Lec 006 9:55 MWF

Fill in these fields (left to right) on the blue scantron form (use #2 pencil).

1. LAST NAME (family name) and first five letters of your FIRST NAME (given name).
2. IDENTIFICATION NUMBER is your Campus ID number.
3. Under ABC of SPECIAL CODES, write your lecture number as a three digit value 001, 002, or 003.
4. Under F of SPECIAL CODES, write the letter P for Primary and fill in the number (1) bubble.

FILL IN THE BUBBLES FOR EACH ITEM.

The exam has two parts and is worth a total of 100 points.
Part I has 20 Simple Choice questions worth 2 points each for 40 points possible for Part I.
Part II has 20 Multiple Choice questions worth 3 points each for 60 points possible for Part II.
You will have 120 minutes to complete the exam.
Be sure to read through every question completely.

I certify that I will keep my answers covered so that they may not be viewed by another student during the exam or prior to completion of their exam. I also certify that I will not view or in any way use another's work or any unauthorized devices. I understand that I may not make any type of copy of any portion of this exam without express permission from my instructor. I understand that being caught allowing another to view my work or being caught viewing another's work are both violations of this agreement and either will result in automatic failure of the course. Any penalty will also be reported to the Deans Office for all involved.

Signature: _____

1. Be sure to review the reference pages as needed during the exam.
2. **Turn off and put away** your cell phone, calculator, Inspector Gadget (watches, glasses, pencils, etc.) now and wait for the proctor to signal the start of the exam.

Disclaimer: the following are provided for your reference only, and the inclusion of information here does not guarantee it will be used on the exam.

Operator Precedence Table:

level	operator	description
higher	`( <expression> )`	grouping with parentheses
	`[ ] ( ) .`	array index, method call, member access (dot operator)
	`++ --`	post-increment, post-decrement
	`++ -- + - !`	pre-increment, unary plus/minus, logical negation
	`(type) new`	casting and creating object
↕	`* / %`	multiplication, division, modulus
↕	`+ - +`	addition, subtraction, concatenation
	`< <= > >=`	relational
	`== !=`	equality
	`&&`	conditional AND (short-circuits)
	`||`	conditional OR (short-circuits)
	`? :`	ternary conditional
lower	`= += -= *= /= %=`	assignment, arithmetic (compound) assignment

Public methods from the java.lang.Object class: (all members are public methods)

`String toString()`	Returns a `String` representation of the object. This is the hashcode of the instance unless `toString()` has been overridden.
`boolean equals(Object o)`	Returns `true` if the object referenced as `o` is the same as `this`. It is often overridden (redefined) by instantiable classes.

Methods from the java.lang.Integer class:

`static int parseInt(String s)`	converts `String s` into its integer value. throws a `NumberFormatException` if `s` cannot be converted into an integer value
`int intValue()`	Returns the value of *this* `Integer` instance as an `int`.

Class (static) Constant(s) and Methods from the java.lang.Math class:

`static dobule Math.PI`	Field that represents the constant π
`static double random()`	Returns a random value between 0 (inclusive) and 1 (exclusive)
`static double pow(double x, double n)`	Returns x^n
`static double sqrt(double n)`	Returns $\sqrt{n}$
`static double abs(double n)`	Returns the absolute value of n
`static double ceil(double n)`	Returns the value of n rounded up to the nearest whole number.
`static double sin(double theta)`	Returns the sine of the angle θ (θ is in radians) Other trig methods also available.

Methods from the java.util.Random class:

`Random()`	Creates a new random number generator instance.
`Random(int s)`	Creates a new random number generator seeded with `s`.
`int nextInt()`	Returns the next pseudo-random integer value.
`int nextInt(int n)`	Returns the next pseudo-random integer value between 0 (inclusive) and n (exclusive).
`double nextDouble()`	Returns the next pseudo-random double value between 0.0 (inclusive) and 1.0 (exclusive).

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

`int length()`	Returns number of characters in the `String`
`char charAt(int index)`	Returns the character at the specified `index` of the `String`
`boolean contains(String s)`	Returns `true` iff string `s` is in this string, otherwise `false`
`String toLowerCase()`	Returns a *new* string that is the lowercase version of this string.
`String toUpperCase()`	Returns a *new* string that is the UPPERCASE version of this string.
`int indexOf(String s)`	Returns the index within this string of the first character of the first occurrence of the specified string `s` or -1 if not found.
`boolean equals(String s)`	Returns `true` if the contents of this String is the same as the contents of 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.
`String substring(int begin)`	Returns a *new* string that is a substring of this string starting at `begin` to the end of this string.
`String substring(int begin, int end)`	Returns a *new* string that is a substring of this string starting at index `begin` up to but not including `end`.
`boolean startsWith(String prefix)`	Returns `true` iff this string starts with the specified prefix `prefix`, `false` otherwise.
`boolean startsWith(String pre, int off)`	Returns `true` iff `this` string starts at the specified offset `off` with the specified prefix `pre`, `false` otherwise.

Methods from the java.util.Scanner class:

`Scanner(String s)`	Creates a `Scanner` to read the `String s`
`Scanner(System.in)`	Creates a `Scanner` that reads from the keyboard.
`Scanner(File fn) throws FileNotFoundException`	Creates a `Scanner` to read from file
`void close() throws IOException`	Closes the stream and any associated file
`boolean hasNext()`	Returns `true` if there's another token of input.
`boolean hasNextInt()`	Returns `true` if the next input is an int value.
`boolean hasNextDouble()`	Returns `true` if the next input is a double value.
`boolean hasNextLine()`	Returns `true` if there's another line of input.
`String next()`	Returns the next word only, as a String.
`int nextInt()`	Returns the next word only, as an integer.
`double nextDouble()`	Returns the next word only, as a double.
`String nextLine()`	Returns the next line as a String.

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.

`new ArrayList()`	Constructs and returns an empty array list where elements are type `Object`.
`new ArrayList<E>()`	Constructs and returns an empty array list of the specified element type E.
`int size()`	Returns the number of used elements in this list.
`E[] toArray()`	Returns an array of the specified type of this ArrayList
`boolean contains(E item)`	Returns `true` iff the specified item is in this list, otherwise `false`.
`int indexOf(E item)`	Returns the index of the specified item if it is in this list, otherwise `-1`.
`E get(int index)`	Returns the item at the specified index in this list. `throws IndexOutBoundsException` if invalid index.
`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.
`E remove(int index)`	Removes and returns the item from the specified index.
`boolean remove(E item)`	Returns `true` iff the specified item was removed from this list, otherwise `false`.

Method from the java.util.Arrays class:

`static String toString(E[] array)`	Returns a `String` representation of any type (`E[]`) array.
`static void sort(E[] array)`	sorts the specified array in memory type E must be Comparable or Comparable`<E>`
`static int[] copyOf(int[] orig, int newLength)`	Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.
`static <E> E[] copyOf(E[] orig, int newLength)`	Copies the specified array, truncating or padding with `null`s (if necessary) so the copy has the specified length.