

Computer Sciences 302

Final Exam 3

Tuesday 12/16, Fall, 2014

I certify that:
- I will keep my answers covered and will not allow my exam paper or answer sheet to be viewed by others.
- I will not view or in any way use another's work in completing my answers.
- I will not take any photo or copy any portion of the exam or my work on the exam.

I understand that being caught doing any of these are violations will result in automatic failure of the course and academic misconduct charges for all individuals involved.

Print last [Family] name: _____ first [Given] name: _____

Signature: _____ CS login: _____

Circle Your Lecture	Lec 001 Deppeler	Lec 002 Deppeler	Lec 003 Hobbes	Lec 004 Hobbes

Before you Begin: (You may lose points if you do not complete these steps correctly)

(1) Scan your UW student ID as directed by Exam Proctor.
(2) Use a #2 pencil and complete the following on the red scantron (J5 ANSWER SHEET):
   - Write your last name (family name) from left to right int the LAST NAME field.
   - Write your first name (given name) from left to right in the FIRST NAME field.
   - Write your UW CAMPUS ID number from left to right in the IDENTIFICATION NUMBER field.
   - In the "Special Codes" section:
     - Write your lecture number as a three digit number **00x** under the letters A B C
     - Write the letter P for primary under the letter J and fill in the number (1) bubble.
(3) Fill in the bubbles for each letter and digit on your scantron form (J5 ANSWER SHEET).
(4) On this examination booklet:
   - Print and sign your name above.
   - Print your CS login and circle your lecture above.
(5) Check that there is a total of 20 pages in this exam.
(6) You may not use notes, books, calculators (or any electronic devices), or neighbors on this exam. Turn off and put away all electronic devices now. The time will be kept on the board.
(7) The exam is intended to take 90 minutes, but we will give you 2 hours to complete the exam.
(8) 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:

(9) Double check that you have correctly marked the bubbles on your answer sheet. **Only answers correctly marked on your answer sheet matter.** Marks in this booklet do not count.
(10) Scan your ID, and turn in your ANSWER SHEET, this examination booklet, and all loose pages.

Taking the Exam

There is one Multiple Choice part to this exam which contains 34 questions each worth 3 points, but max score on exam is 100 points. Not all questions are of the same difficulty. Check the time occasionally and pace yourself so that you have time to complete all questions to the best of your ability. If you get stuck, mark the question on the exam questions paper and work on another question and come back if there is time remaining.
Do not mark scantron sheet except with answers.
Be sure to review all reference information.

Operator Precedence Table:

level	operator	description
higher	(<expression>)	grouping with parentheses
	[] () .	array index, method call, member access (dot operator)
	+ - (<type>) ++ -- ! new	unary plus/minus type casting dec/increment logical not object creation
	* / %	multiplicative
	+ -	additive
	< <= > >= instanceof	relational
	== !=	equality
	&&	logical and
lower	||	logical or
	= += -= *= /= %=	assignment and compound assignments

The `java.lang.Comparable` interface:

```
int compareTo(Object ob)// Returns a negative if this object is less than ob,
                        // 0 if they are equal, and a positive if this object
                        // is greater than ob
```

The `java.lang.Comparable<T>` interface:

```
int compareTo(T t)      // Returns a negative if this T object is less than t,
                        // 0 if they are equal, and a positive if this object
                        // is greater than t
```

Methods from the `java.lang.Object` class:

```
String toString()       //returns a String representation of the class.
                        //This is the hashcode of the instance
                        //unless this method is overridden by the class..
boolean equals(Object o)//returns true if the object reference o is
                        //is the same as this. Often overridden,
                        //and redefined by the class.
```

Constant and Methods from the `java.lang.Math` class:

```
PI                      //field that represents the constant π
double random()         //Returns a random value between 0 (inclusive)
                        //and 1 (exclusive).
double pow(double x, double n) //Returns x raised to the n-th power
double sqrt(double n)   //Returns the square root n.
double abs(double n)    //Returns the absolute value of n.
double ceil(double n)   //Returns the value of n rounded up to the nearest
                        //whole number.
```

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
```

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

```
int length()            //Returns # of characters in this String.
char charAt(int index)  //Returns the character at the index.
int indexOf(String s)   //Returns the index within this string of the first
                        // character of the first occurrence of the specified
                        // string s.
boolean contains(String s)// Returns true iff string s is in this string,
                          // otherwise false.
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 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.
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.

boolean startsWith(String prefix)
                        //Returns true if this string starts with the specified
                        // prefix, false otherwise.
boolean startsWith(String prefix, int offset)
                        //Returns true if this string start at the specified
                        // offset with the specified prefix, false otherwise.
```

Constructors from the `java.awt.Color` class: r,g,b (0-255), a (0-10)

```
Color(float r, float g, float b)           // Creates a Color instance
Color(float r, float g, float b, float a)  // Creates a Color instance
Color(int rbg)                             // Creates a Color instance
Color(int r, int g, int b)                 // Creates a Color instance
Color(int r, int g, int b, int a)          // Creates a Color instance
```

Methods from the `java.io.File` class:

```
File(String filename)   // Creates a File object for the given file name
boolean exists()        // true if the specified file already exists
boolean canRead()       // true if the specified file can be opened for reading
boolean canWrite()      // true if the specified file can modified (written to)
```

Methods from the `java.io.PrintWriter` class:

```
PrintWriter(String filename) throws FileNotFoundException
                        // Creates a PrintWriter for the given file name
PrintWriter(File out) throws FileNotFoundException
                        // Creates a PrintWriter from out
void close()            // Closes the stream and any associated file
void print(String s)    // Prints given string
void println(String s)  // Prints given string followed by a newline
```

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

```
Scanner(String s)       //Creates a Scanner to read the String "s"
Scanner(System.in)      //Creates a Scanner object that reads from the keyboard.
Scanner(File fn) throws FileNotFoundException //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.Random` class:

```
Random()                //Creates a new random number generator.
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 0.0d (inclusive) and 1.0d (exclusive).
```

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

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

static <T> T[] copyOf(T[] original, int newLength)
// Copies the specified array, truncating or padding with nulls (if necessary)
// so the copy has the specified length.
```

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.
                        // throws IndexOutBoundsException if invalid index.
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.
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.
int size()              // Returns the number of used elements in this list.
E[] toArray()       // Returns an array of the specified type of this ArrayList
```