Computer Science 302

Final Exam
Monday, December 19th, 2016
100 points (20% of final grade)
Instructors: Gary Dahl and Jim Williams

(Family) Last Name: ________________________ (Given) First Name: ________________________

CS Login Name: ________________________ NetID (email): ________________________@wisc.edu

Circle your Lecture:
Lec001 TR 8:00 (Jim)  Lec002 TR 1:00 (Jim)  Lec003 MWF 1:20 (Gary)  Lec004 MWF 9:55 (Gary)

SCANTRON: Fill in these fields (left to right) on the scantron form using a #2 pencil.
1. LAST NAME (family name) and FIRST NAME (given name), as much as there is space to enter.
2. IDENTIFICATION NUMBER is your Campus ID number.
3. SPECIAL CODES ABC: enter your three-digit lecture number: 001, 002, 003, or 004.
4. SPECIAL CODES F: write the letter P for Primary and fill in the bubble for #1.

This exam contains two parts and is worth a total of 100 points.
Part I contains 20 Simple Choice Questions worth 2 points each, for a total of 40 points possible.
Part II contains 20 Multiple Choice Questions worth 3 points each, for a total of 60 points possible.
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 that may result in failure of the course. Any penalty will also be reported to the Dean's 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:

<table>
<thead>
<tr>
<th>Level</th>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>higher</td>
<td>( &lt;expression&gt; )</td>
<td>grouping with parentheses</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>[ ] ( ) .</td>
<td>array index, method call, member access</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>++ -- ++ -- + - !</td>
<td>post-increment, post-decrement</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>(type) new</td>
<td>pre-increment, unary plus/minus, logical negation</td>
<td>right to left</td>
</tr>
<tr>
<td></td>
<td>* / % + - +</td>
<td>casting and creating object</td>
<td>right to left</td>
</tr>
<tr>
<td></td>
<td>&lt; &lt;= &gt; &gt;= instanceof</td>
<td>multiplication, division, modulus, addition, subtraction, concatenation relational and Java’s instanceof operator</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>== != &amp; &amp;</td>
<td>equality</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>? :</td>
<td>conditional AND (short-circuits)</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>= += -= *= /= %=</td>
<td>conditional OR (short-circuits)</td>
<td>right to left</td>
</tr>
<tr>
<td>lower</td>
<td>= += -= *= /= %=</td>
<td>assignment</td>
<td>right to left</td>
</tr>
</tbody>
</table>

Constants and Methods from the java.lang.Math class:

- static Math.PI: Field that represents the constant π
- static double abs(double n): Returns the absolute value of n
- static double min (double x, double n): Returns the smaller of the two values as a double
- static int min(int x, int n): Returns the smaller of the two values as an int
- double pow(double x, double n): Returns $x^n$
- static double random(): Returns a random value between 0 (inc.) and 1 (exclusive)
- static double sqrt(double n): Returns $\sqrt{n}$

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

- String toString(): Returns a String representation of the object. This is the hash code 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: (which implements Comparable)

- static int parseInt(String s): Converts s into the corresponding int value. Throws a NumberFormatException if s cannot be converted into an int value.
- int intValue(): Return the int value of this Integer instance.
Methods from the java.lang.Object class:

- public boolean equals(Object obj)
  Indicates whether some other object is "equal to" this one.

- public String toString()
  Returns a string representation of the object.

Methods from the java.lang.String class: (which implements Comparable)

- int length()
  Returns number of characters in the String

- char charAt(int index)
  Returns character at the specified index of the 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.

- String concat(String s)
  Returns a new string that is the result of concatenating the String s to the end of this String.

- boolean equals(String s)
  Returns true if the contents of this String is the same as the contents of String s.

- String substring(int begin)
  Returns a new string that is a substring of this string starting at begin to the end of this string.

The java.lang.Comparable interface:

- int compareTo(Object obj)
  Returns a negative value if this is less than obj, zero if they are equal, and a positive value if this is greater than obj.

The java.lang.Comparable<T> interface:

- int compareTo(T obj)
  Returns a negative value if this is less than obj, zero if they are equal, and a positive value if this is greater than obj.

Methods from java.io.PrintStream class:
(called using System.out.print or System.out.println)

- void print( String s )
  Prints a String.

- Void print( Object o )
  Prints String representation of Object using String.valueOf(o)

- void println( boolean b )
  Prints a Boolean value and then terminate the line.

- void println( String s )
  Prints a String and then terminate the line.

- void println( Object o )
  Prints String representation of Object using String.valueOf(o) and then terminates the line.

Methods 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>

- String toString()
  Calls toString on each element in the list & returns a single comma separated String of these results.
Methods from the java.util.ArrayList<E> class (*REMEMBER 0-based indexing): Note the E’s below are replaced with the particular ArrayList’s element type.

```java
ArrayList<E>() Constructs an empty list.
ArrayList<E>(int cap) Constructs an empty list with cap initial capacity.
int size() Returns the number of used elements in this list.
E get(int index) Returns the item at the specified index in this list. throws IndexOutOfBoundsException 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.
```

Methods from java.io.PrintWriter class:

```java
PrintWriter(String filename) throws FileNotFoundException Creates a PrintWriter for the given filename.
PrintWriter(File out) throws FileNotFoundException Creates a PrintWriter from out.
void close() Closes the stream and 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:

```java
Scanner(System.in) Creates a Scanner that reads from the keyboard.
Scanner(String s) Creates a Scanner to read the String s.
Scanner(File fn) throws FileNotFoundException Create a Scanner to read from file.
void close() throws IOException Closes the stream and any associated file.
boolean hasNextInt() Returns true if the next input is an int 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 int.
double nextDouble() Returns the next word only, as a double.
String nextLine() Returns the next line as a String.
```

Exception Class Inheritance Hierarchy

```java
public class Throwable extends Object
public class Exception extends Throwable
  public class RuntimeException extends Exception
    public class ArithmeticException extends RuntimeException
    public class IndexOutOfBoundsException extends RuntimeException
    public class ClassCastException extends RuntimeException
    public class NullPointerException extends RuntimeException
  public class IOException extends Exception
    public class FileNotFoundException extends IOException
    public class EOFException extends IOException
```