**CSC 1051 Algorithms and Data Structures I**

**Final Examination**

**December 20, 2016**

Name: ______________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>Value</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Please answer questions in the spaces provided. Please be legible. If you make a mistake or need more space, use backs of pages - clearly indicate where the answer can be found.

*Good luck and best wishes for the holidays!*
1. (_____ / 10) What gets printed? Please show output as it will appear, or indicate “NO OUTPUT”. If there is an infinite loop, be sure to show some lines of the output followed by “... INFINITE LOOP”.

```java
int a = 4;
do {
    a--;
    System.out.println(a);
} while (a > 4)
```

```
String[] n = {"one", "two"};
for (String word : n)
    for (String otherWord : n)
        System.out.println("Tra la la " + word + " and " + otherWord);
```

```java
int size = 10;
do {
    System.out.print(size + " => ");
    int category = size / 4;
    switch(category) {
        case 0:
            System.out.println("S");
            break;
        case 1:
            System.out.println("M");
            break;
        default:
            System.out.println("L");
    }
    size = size - 2;
} while (size > 0);
```

```java
int a = 4;
for (int a = 4; a <= 4; a++)
    System.out.println(a);
```
2. [ /10] The following program is supposed to determine whether a positive integer myNum is prime (i.e., has no divisors other than 1 and itself). It does this by checking if any values n are divisors, using a boolean variable gotIt to keep track of whether a divisor has been found.

```java
import java.util.Scanner;
public class PrimeTester
{
    public static void main (String[] args)
    {
        Scanner scan = new Scanner (System.in);
        System.out.println("Please enter a positive integer.");
        int myNum = scan.nextInt();

        int n = 2;
        boolean gotIt = false;

        System.out.print ("This number is ");
        if (gotIt)
            System.out.print ("NOT ");
        System.out.println ("prime.")
    }
}
```

a) Which of these versions of the code should go in the box above? (circle correct one)

```java
while (n < myNum)
{
    if (myNum % n == 0)
        gotIt = true;
    else  gotIt = false;
    n++;
}
```

```java
while (n <= myNum)
{
    if (myNum % n == 0)
        gotIt = true;
    else  gotIt = false;
    n++;
}
```

b) Show the output produced by the program, given to the following inputs:

- 23
- 15
- 2

Note: The differences are the "<" vs. "<=" and the `if/else` vs. just if (no else)
3. (_____/ 10)
a) Write a method with one parameter, a *String*, that returns that String reversed. For example, if the given String is “abcde”, the method should return the String “edcba”. Note that the method should not print anything.

b) Write a method with one parameter, a *String*, that returns true or false, depending on whether the given String begins and ends with the char “*”. For example, if the given String is “ab” or “a**d*”, the method should return false; if it is “*bb*” or “*”, it should return true. Note that the method should not print anything.
4. (10/10)

a) Write a method that adds \( n \) (an int) to the value of each element in an array of type int[].
Note that the method should not print or return anything.

\[
\text{public void addN(int[] a, int n)}
\]

b) Suppose the method addN() is defined in a class that extends JPanel, with a
paintComponent() method given below. Sketch the image produced when the
panel is painted in a graphics context.

\[
\begin{align*}
\text{public void } & \text{ paintComponent(Graphics page)} \\
& \{ \\
& \quad \text{super.paintComponent(page);} \\
& \quad \text{int[]} xPoints = \{50, 100, 150\}; \\
& \quad \text{int[]} yPoints = \{20, 100, 20\}; \\
& \quad \text{page.drawPolyline(xPoints, yPoints, xPoints.length);} \\
& \quad \text{addN(yPoints, -20);} \\
& \quad \text{page.drawPolyline(xPoints, yPoints, xPoints.length);} \\
& \}
\end{align*}
\]
5. (_____/ 10) Consider the following program:

```java
import java.util.Scanner;
import java.io.*;

public class FileOutputFinalF16
{
    public static void main(String[] args) throws IOException
    {
        Scanner inFile;
        PrintWriter outFile;

        inFile = new Scanner (new File("data-in.txt"));
        outFile = new PrintWriter("data-out.txt");

        while (inFile.hasNext())
        {
            String token = inFile.next();
            outFile.print(token.length() + " ");
        }
        outFile.close();
    }
}
```

a) Suppose the file `data-in.txt` is used as the input file. Show the contents of the file `data-out.txt` after execution of the program.

```
data-out.txt

```

b) List two examples of situations that could cause IOExceptions in the above code.

c) Suppose you want to catch and handle the IOExceptions using the following catch clause:

```java
catch (IOException e)
{
    System.out.println("Problem with file IO. Running interactively.");
    inFile = new Scanner(System.in);
    outFile = new PrintWriter(System.out);
}
```

(i.e., keep running, but issue a warning and do interactive I/O instead)

Show how to incorporate this in the above program:
1) Show the statements in the above code that need to be included in the try block
2) mark the position where you would insert the catch code.
6. (_____/ 10) Consider the following program:

```java
public class TwoDArray {
    public static void main (String[] args) {
        int[][] table = new int[5][10];

        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                table[row][col] = row * 10 + col;

        System.out.print ("# | \\	");
        for (int col=0; col < table[0].length; col++)
            System.out.print (col + "\\t");
        System.out.println();

        System.out.print ("-+-------");
        for (int col=0; col < table[0].length; col++)
            System.out.print ("-+----------------");
        System.out.println();

        for (int row=0; row < table.length; row++)
        {
            System.out.print (row + " | \\	");
            for (int col=0; col < table[row].length; col++)
                System.out.print (table[row][col] + "\\t");
            System.out.println();
        }
    }
}
```

We would like to modify the program to: (1) change array size; and (2) label rows/columns so that the output would now look EXACTLY like this:

<table>
<thead>
<tr>
<th>#</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
</tbody>
</table>

a) In the above output, circle entries corresponding to table[0][2] and table[2][3].

b) Annotate the code so that it works as described in (1) and (2) above.
7. (_____/ 10) Fill in code for an Employee class, following guidelines in comments.

```java
public class Employee {
    // instance variables
    String name;
    String position;
    double hourly; // hourly wages
    int yearHired

    // constructor: Construct object with w, x, y, and z as
    //              name, position, hourly pay rate, and
    //              year hired, respectively.

    // toString(): Returns a String corresponding to object.

    // getYearHired(): Accessor for yearHired

    // wages(): Given the number of hours worked (a value of type double,
    // returns the wages of this employee, calculated based on
    // hourly rate, for up to 40 hours and 1.5 overtime of
    // hourly rate for hours over 40.
}
```
8. (_____/ 10) Suppose you look up a class in the Java API and find something that looks like the following (NOTE: this is a made-up class):

```java
java.exam
Class Mystery
    java.lang.Object
        java.exam.Mystery
    public class Mystery
        extends Object

This is a made-up class. It does not matter what it actually does, I am just trying to see if you know how to use it.

Constructor Summary

Mystery(double x)
    Creates a new Mystery object.

Method Summary

boolean decider(int x, String y)
    Mystery method 1.

void updator(double x)
    Mystery method 2.
```

a) What import statement do you include in your program in order to use this class?

b) Write some code to declare variables for two objects of this class, named thing1 and thing2 (use any values of the appropriate type in the constructor).

c) Consider the following **client code**:

<table>
<thead>
<tr>
<th>Valid Java?</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing1.updator(5.3);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thing1.updator(double x);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mystery.updator(7.2);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thing2.deci(d)(int x, String y);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System.out.println(&quot;answer = &quot; + thing2.deci(d)(5, &quot;d&quot;));</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System.out.println(&quot;answer = &quot; + thing2.updator(7.2));</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if (thing2.updator(0) == 2.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System.out.println(&quot;ok&quot;);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if (thing2.deci(d)(5, &quot;d&quot;))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System.out.println(&quot;ok&quot;);</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
9. (______/ 10) Refer to the UML class diagram below. Recall that the `addDVD()` method in the `DVDCollection` class adds a `DVD` to the array representing the collection of DVD’s, increasing the size of the array, if necessary.

a) What is the name of the instance variable representing the array of DVD’s?

b) Write the code for the method `addDVD()`, implementing it as shown in the UML class diagram.

```
// -----------------------------------------------------------------
// Adds a DVD to the collection, increasing the size of the
// collection array if necessary.
// -----------------------------------------------------------------
```
10. (____/ 10) Refer to the UML class diagram from the previous problem.

a) We now wish to add a method to shuffle the DVD's in a DVDCollection. Do this by exchanging the positions of random items in the collection. How many exchanges to make? You can assume that if you perform twice as many exchanges as there are items in the collection, that would be sufficient to render the collection shuffled. Complete the code for the shuffle() method:

```java
public void shuffle()
{
    Random rand = new Random();
}
```

b) Complete the code of the Movies client, as directed by the comments.

```java
public class Movies
{
    public static void main (String[] args) throws IOException
    {
        DVDCollection movies = new DVDCollection();

        movies.addDVD("The Godfather", "Coppola", 1972, 24.95, true);
        movies.addDVD("District 9", "Blomkamp", 2009, 19.95, false);
        movies.addDVD("Iron Man", "Favreau", 2008, 15.95, false);
        movies.addDVD("Casablanca", "Curtiz", 1942, 19.95, false);
        System.out.println(movies);

        // shuffle the movies so far

        // add another movie (your choice of values for the parameters)

        // print out the movies in the collection
    }
}
```
**Boolean Expressions**

The reserved words `true` and `false` are the only valid values for a boolean type.

Example: boolean variables:

```java
boolean aboveAgeLimit = false;
boolean usePlural = hours > 1;
```

**Java Conditional statements** alter the linear flow of control. They use boolean expressions to determine what to do next.

Example:

```java
if (credits == 0)
    System.out.println( "CPA: None" );
else
    System.out.println( "\n\tGPA: “ + gpa + " \n\tCPA: " );
```

**The Conditional Operator Syntax**

- If the `condition` is true, `expression1` is evaluated; if it is false, `expression2` is evaluated.
- The value of the entire conditional operator is the value of the selected expression.

Example: Rewrite this using conditional operator:

```java
if (n > 0) System.out.println("positive");
else System.out.println("negative");
```

**The switch Statement in general**

```java
switch ( expression )
{
    case value1 :
        statement-list1
        break;
    case value2 :
        statement-list2
        break;
    case value3 :
        statement-list3
        break;
    .
    .
    .
    default:
        statement-list-n
}
```

If expression matches value2, control jumps to here.

**while Loop**

```java
while ( condition )
{
    statement
}
```

The body of a `do` loop executes at least once.

**do Loop**

```java
do
{
    statement
} while ( condition )
```

**for: a loop with built in “counter”**

```java
for ( int count = 0; count < 5; count++ )
    System.out.println( count );
```

```
for ( int count = 0; count < 5; count++ )
    System.out.println( count );
```

```
for ( int count = 0; count < 5; count++ )
    System.out.println( count );
```

```java
while (condition)
{
    System.out.println( count );
    count++;
}
```

```
while (count < 5)
{
    System.out.println( count );
    count++;
}
```
Arrays

- Declaration: `double[] scores = new double[10];`

  The entire array has a single name.

- Initialization:
  ```java
  scores[0] = 7.9;
  scores[1] = 8.7;
  scores[2] = 9.4;
  scores[3] = 8.2;
  scores[4] = 6.7;
  scores[5] = 9.8;
  scores[6] = 8.7;
  scores[7] = 8.1;
  scores[8] = 7.6;
  scores[9] = 9.1;
  ```

  Size of array: `scores.length = 10`.

Arrays as Parameters

- Example:
  ```java
  int[] ratings = {4, 3, 3, 1, 4, 3, 1, 0, 3, 4};
  System.out.println(average(ratings));
  ```

  Accumulate a definition for method `average()`:
  ```java
  public static double average(int[] a)
  {
    for (int num : a)
      sum += num;
    return (double)sum/a.length;
  }
  ```

UML Class Diagrams

UML = Unified Modelling Language
- Example: A UML class diagram for the RollingDice program:

  ```java
  public class stars {
    // Prints a triangle shape using asterisk (star) characters.
    public static void main(String[] args) {
      final int MAX_ROWS = 10;
      for (int row = 1; row <= MAX_ROWS; row++) {
        for (int star = 1; star <= row; star++)
          System.out.print (**);
      }
      System.out.println();
    }
  }
  ```

  Output:
  ```
  * 
  ** 
  *** 
  **** 
  ***** 
  ****** 
  ******* 
  ******** 
  ********* 
  ********** 
  ```

toString() method

- Example:
  ```java
  public String toString() {
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    return (acctNumber + \\
        name + \\
        fmt.format(balance));
  }
  ```

Account object:
- acct1
- Account number: 72354
- Balance: 102.56
- Name: "Ted Murphy"
try / catch

- Create a try block surrounding code that we think may cause an exception

- catch clause has code to tell it what to do
  - the “exception handler”
  - Can have multiple catch clauses
    - One for each type of exception thrown by try block

- If no exception is thrown, processing continues following the try statement (skips catch clauses)

---

Zero.java -- updated

```java
public class Zero
{
    public static void main(String[] args)
    {
        int numer = 10;
        int denom = 0;
        try
        {
            System.out.println (numer/denom);
        }
        catch (ArithmeticException problem)
        {
            System.out.println ("Bad division");
        }
        System.out.println ("this will not print");
    }
}
```

---

But is our trace gone???

- No
- Methods exist to get the trace and system error message
  - Method: getMessage()
    - Returns a string explaining the reason the exception was thrown
  - Method: printStackTrace()
    - Prints the call stack trace indicating where the error occurred

---

Unchecked Exceptions

![Unchecked Exceptions Diagram]