Answer Key

CSC 1051 Algorithms and Data Structures I

Final Examination
December 20, 2016

Name: KEY

<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>
</tbody>
</table>

TOTAL 100

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!

Villanova University    CSC 1051    www.csc.villanova.edu/~map/1051    Dr. Papalaskari
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”.

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

Output:
```
3
```

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

Output:
```
tra la la one and one
tra la la one and two
tra la la two and one
tra la la two and two
```

```
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);
```

Output:
```
10 => L
8 => L
6 => M
4 => M
2 => S
```

```
int a = 4;
for (int a = 4; a <= 4; a++)
   System.out.println(a);
```

Output:
```
4
```
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)

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

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

Note: The differences are the "<" vs. "<=" and the if/else vs. just if (no else)

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

- **23** ______ This number is prime
- **15** ______ This number is NOT prime
- **2** ______ This number is prime
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.

```java
public String reverse(String a)
{
    String m = "";
    for (int i = 0; i < a.length(); i++)
        m = a.charAt(i) + m;
    return m;
}
```

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.

```java
public boolean starry(String a)
{
    return (a.charAt(0) == "*" && a.charAt(a.length()-1) == "*");
}
```
4. (_____/ 10)

a) Write a method that adds \( n \) (an \texttt{int}) to the value of each element in an array of type \texttt{int[]}.

Note that the method should \textit{not} \texttt{print} or return anything.

```java
public void addN(int[] a, int n) {
    for (int i = 0; i < a.length; i++)
        a[i] += n;
}
```

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

```java
public void paintComponent(Graphics page) {
    super.paintComponent(page);
    int[] xPoints = {50, 100, 150};
    int[] yPoints = {20, 100, 20};

    page.drawPolyline(xPoints, yPoints, xPoints.length);
    addN(yPoints, -20);
    page.drawPolyline(xPoints, yPoints, xPoints.length);
}
```
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
8 1 8 4 9 2 7 8 3 5
```

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

- file not found for the Scanner – cannot do input
- bad file name for the PrintWriter – cannot create the file

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[3][9];

        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 + "	");
        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] + "	");
            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:

```
# | 0 1 2 3 4 5 6 7 8
---+-------------------
0 | 0 1  2  3  4  5  6  7  8
1 | 10 11 12 13 14 15 16 17 18
2 | 20 21 22 23 24 25 26 27 28
```

**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.
    public Employee(String w, String x, double y, int z) {
        name = w;
        position = x;
        hourly = y;
        yearHired = z;
    }

    // toString(): Returns a String corresponding to object.
    public String toString() {
        return (name + "\n, " + position + ", year hired" + year
            + ", hourly rate: " + hourly);" + hourly);
    }

    // getYearHired(): Accessor for yearHired
    public int getYearHired() {
        return 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.
    public double wages(double hours) {
        double w;
        if (hours <=40) w = hourly * hours;
        else w = 40 * hourly + (hours - 40) * 1.5 * hourly;
        return w;
    }
}
```
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**

```java
Mystery(double x)
    Creates a new Mystery object.
```

**Method Summary**

```java
    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?

```java
import java.exam.Mystery;
```

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).

```java
Mystery thing1 = new Mystery(3.14);
Mystery thing2 = new Mystery(42.0);
```

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.decider(int x, String y);</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>System.out.println(&quot;answer= &quot; + thing2.decider(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>
</tbody>
</table>
| if (thing2.updator(0) == 2.0)
    System.out.println("ok"); | ✓   |    |
| if (thing2.decider(5, "d");
    System.out.println("ok"); | ✓   |    |
```
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?

```java
___ collection ____
```

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

```java
public void addDVD(String t, String d, int y, double c, boolean b) {
    DVD temp = new DVD(t,d,y,c,b);
    if (count == collection.length)
        increaseSize();
    collection[count] = temp;
    count++;
}
```
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();

    for (int n = 1; n <= count * 2; n++)
    {
        int i = rand.nextInt(count);
        int j = rand.nextInt(count);

        DVD temp = collection[i];
        collection[i] = collection[j];
        collection[j] = temp;
    }
}
```

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
        movies.shuffle();

        // add another movie (your choice of values for the parameters)
        movies.addDVD("Legally Blonde", Luketic, 2001, 18.99, true);

        // print out the movies in the collection
        System.out.println(movies);
    }
}
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
    
    gpa = qp / credits;
    System.out.println("\n\nGPA: " + gpa);
```

The Conditional Operator Syntax:

```
condition ? expression1 : expression2
```

- 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:

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

using conditional operator:

```java
System.out.print(n > 0? "positive": "negative");
```

The switch Statement in general

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

If expression

- If expression matches value2, control jumps to here.
- If expression matches value3, control jumps to here.

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

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

  Index

  0 1 2 3 4 5 6 7 8 9

  Size of array: `scores.length = 10`

  Array element `scores[2]`

Arrays as Parameters

- Example:

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

  Declaration: `2D Arrays - Overview`

  ```java
  double[][] courseGrade = new double[3][10];
  ```

  2D array element `courseGrade[1][4]`

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

  Example:

  ```java
  public static double average(int[] a) {
      int sum = 0;
      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 RollingDice {
      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();
      }
  }
  ```

  ```java
  public class Die {
      public static String toString() {
          return (value == 1) ? "one" : (value == 2) ? "two" : (value == 3) ? "three" : (value == 4) ? "four" : (value == 5) ? "five" : "six";
      }
  }
  ```

  ```java
  public class Account {
      private String name;
      private double balance;
      public Account(String name, double balance) {
          this.name = name;
          this.balance = balance;
      }
      public String toString() {
          return "Account: " + name + " Balance: " + balance;
      }
  }
  ```

  ```java
  public class AccountClient {
      public static void main(String[] args) {
          Account acct1 = new Account("Ted Murphy", 102.56);
          Account acct2 = new Account("Sandra Sentine", 123.45);
          System.out.println(acct1.toString());
          System.out.println(acct2.toString());
      }
  }
  ```

Example: `Stars.java`

```java
// Stars.java
// Author: Lewis/Simpson
// Demonstrates the use of asterisk (star) characters.
//
public class Stars {
    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();
    }
}
```
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