Happy Cinco de Mayo!!!!

Final Examination
May 5, 2018

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>20</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>100</strong></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 a great summer!
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 = 3;
while (a < 8)
{
    if ((a%2)==0)
        System.out.println(a);
    a++;
}

int[] b = {4, 5, 6, 7};
for (int i = 2; i < b.length; i++)
    System.out.println(b[i] * 3);

int a = 7;
do
{
    a--;
    System.out.println(a);
} while (a < 5);

String[] word = {"up", "down");
for (String w1 : word)
    for (String w2 : word)
        System.out.println("yes " + w1 + " " + w2 + " " + w1);

int[] c = {2, 5, 1, 3, 4};
for (int i = 0; i < c.length; i++)
{
    System.out.print(i + " ");
    System.out.println(c[i] > 3 ? " yes" : " no");
}
```

Output:

```
4
6

Output:
18
21

Output:
6

Output:
yes up up up
yes up down up
yes down up up
yes down down down

Output:
0 no
1 yes
2 no
3 no
4 yes
```
2. (_____/10)
We studied the problem of repeatedly obtaining input and performing a calculation, for example, computing the circumference of a circle given its radius, using the following algorithm:

*Rewrite this algorithm, modifying it so that it uses a while structure to repeat the processing of each input in two different ways.*

a) Keep computing circumferences and ask each time whether to keep going.
*Variables:* radius, circ, ans
*Algorithm:*

```java
ans = 1
while (ans equals 1)
    input radius
    circ = 2 * radius * PI
    print circ
    print “do another?”
    input ans
```

b) Keep computing circumferences until user inputs -1 for the radius (sentinel value)
*Variables:* radius, circ
*Algorithm:*

```java
input radius
while (radius != -1)
    circ = 2 * radius * PI
    print circ
    input radius
```

c) Compute the circumference of 5 circles (exact count).
*Variables:* radius, circ, count
*Algorithm:*

```java
count = 1
while (count <= 5)
    input radius
    circ = 2 * radius * PI
    print circ
    count = count + 1
```
3. (___/10) Complete the following code fragment following the hints given as comments. (This is similar to the project 5 problem of expanding the queue.)

String queue = ":JJJJJ:KKKKKK:AAAAAA"

int t1 = scan.nextInt();
int t2 = scan.nextInt();
Random rand = new Random();

//*** 1 numTimes is a number in the range [t1 .. t2]
int numTimes = rand.nextInt(t2 - t1) + t1;

//*** 2 personSymbol = a random char in the range ['A'.. 'Z']
char personSymbol = (char) ('A' + rand.nextInt(26));

//*** 3 updating the String representing the queue:
// add a colon followed by numTimes the personSymbol.
// For example, if numTimes = 3 and personSymbol = 'B',
// the queue will become ":JJJJJ:KKKKKK:AAAAAA:BBB"
queue += ":";
for (int j = 0; j < numTimes; j++)
    queue += personSymbol;
4. (____/10)

a) Write a Java method `maxOfThree()` with three parameters of type `double` that returns a value of type `double` that is the largest of the three given values. For example, `max(35.2, 45.7, 22.8)` should return 45.7. Note that the method should not print anything.

```java
dooble  max(dooble a, dooble b, dooble c)
{
    dooble max = a;
    if (b > max)
        max = b;
    if (c > max)
        max = c;
    return max;
}
```

b) Write a method `maxArray` with one parameter, an array of `double` that calculates and returns the maximum value stored in the array. For example, if the array contains the values {-3.5, -4.0, 5.4, 1.6}, the method should return the value 5.4. Note that the method should not print anything.

```java
dooble  maxArray(dooble[] a)
{
    dooble max = a[0];
    for (int j = 1; j < a.length; j++)
        if (a[j] > max)
            max = a[j];
    return max;
}
```
5. (___/10) Consider the code for the start method of a JavaFX application, below, which produces the graphic shown to the right.

a) Draw a group hierarchy diagram that includes all the graphical elements (line, circle, rect, ellipse, quote, message, root).

b) Using the grid on the next page sketch the graphic in the coordinate system.
- Be sure to draw and position all the shapes precisely in the grid.
- Mark the center of the circle and note its coordinates on the sketch
- Mark the endpoints of the line and note their coordinates on the sketch
Draw diagrams showing the contents of array table after execution of the following code fragments. (Be sure to include indices in your diagram).

```java
int[][] table = new int[2][3];
for (int i=0; i<2; i++)
    table[i][2] = i+1;

char[][] table = new char[4][2];
String sample = "Alternative facts."
int count = 0;
for (int i=0; i<4; i++)
    for (int j=0; j<2; j++)
        table[i][j] = sample.charAt(count);
        count++;
```

The diagram shows a grid with the following information:

- A point at (100, 65) labeled as (100,65).
- A point at (35, 60) labeled as (35, 60).
- A point at (150, 170) labeled as (150, 170).
- A rectangle labeled "CINCO DE MAYO!!!!".
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 +", "+ position +", year hired" + year
        + ", hourly rate: " + 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;
    }
}
Using the Employee class from the previous question:

a) Draw a UML diagram for the Employee class.

```
employee

name: String
position: String
hourly: double
yearHired: int

toString: String
getFullYearHired(): int
wages(hours: double): double
```

b) Write client code that uses the Employee class:
Instantiate an Employee object with name “Lucia Rodriguez”, with position “software engineer”, hourly rate $42.50, hired in 2013. Assign it to a variable named `coderBoss`.
```
Employee coderBoss = new Employee("Lucia Rodriguez",
                                    "Software engineer",
                                    42.50,
                                    2013);
```

c) Write client code that uses the Employee class:
Suppose you have three Employee objects `e1`, `e2`, and `e3` and that `e2` worked 44.5 hours last week, whereas `e1` and `e3` both worked 40 hours. Write some client code to calculate and print: 1) the wages for each employee and 2) the average of their wages. (Note: it is NOT necessary to format as currency).
```
double w1 = e1.wages(40);
double w2 = e2.wages(44.5);
double w3 = e3.wages(40);
System.out.println((w1 + w2 + w3) / 3);
```

d) Write client code that uses the Employee class:
Suppose you have two Employee objects `e1`, `e2`. Write some client code that uses the `getFullYearHired()` and `toString()` methods of the Employee class to print the information for the Employee who has been with the company the longest (i.e., hired earlier).
```
if (e1.getYearHired() < e2.getYearHired())
    System.out.println(e1);
else
    System.out.println(e2);
```
8. (_____/10) Consider the following program:

```java
//*****************************************************************
// TwoDArray.java       Author: Lewis/Loftus
// Demonstrates the use of a two-dimensional array.
//******************************************************************
public class TwoDArray {
    // Creates a 2D array of integers, fills it with increasing
    // integer values, then prints them out.
    public static void main(String[] args) {
        int[][] table = new int[3][9];
        // Load the table with values
        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                table[row][col] = row * 10 + col;
        // Print the table
        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                System.out.print (table[row][col] + "	");
        System.out.println();
    }
}
```

a) The output produced is shown below. Circle the entries for `table[1][2]` and for `table[2][3]`

<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>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>
<td></td>
</tr>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>

b) On the next page, rewrite the code of the main method, so that instead of a 2D array

of `int`, it creates a 2D array of 3x3 values of type `boolean` as follows:

- use an initializer list to instantiate and initialize the array `table` to represent the following configuration, where the asterisk indicates a `true` value at that position in the array (no asterisk means `false`). For example, `table[2][0]` is `false`, whereas `table[2][1] is true`.

- Modify the loop that prints the values of the array so that it prints an asterisk or blank, for each element in the table. The output does NOT have to show row and column labels, it should look like this:

```
# | 0 | 1 | 2
---|---|---|---
0 | * | * | *
1 | * | * | *
2 |   | * | *
```
//**************************************************************
// TwoDArray.java       Author: Lewis/Loftus
// Demonstrates the use of a two-dimensional array.
//**************************************************************
public class TwoDArray
{
    //---------------------------------------------
    // Creates a 2D array of integers, fills it with increasing
    // integer values, then prints them out.
    //---------------------------------------------
    public static void main (String[] args)
    {
        // Load the table with values
        boolean[][] = {
            {false, true, true},
            {true, false, true},
            {false, true, false}};

        // Print the table
        for (int row=0; row < table.length; row++)
        {
            for (int col=0; col < table[row].length; col++)
            {
                System.out.print((table[row][col] ? "*": "") + "	");
        }
        System.out.println();
    }
}
**Answer Key**

9. [_____/ 20] Consider the program `ProductCodesS18Final.java` on the last page. The code contains several errors and omissions that need to be corrected.

a. Line 26 is causing an error:

```java
26    System.out.println("\"products.txt\" + " not found");
```

- What kind of error is it? (Syntax, Runtime, or Logical) **Syntax**
- Show how to correct it

```java
System.out.println("\"products.txt\" + " not found");
```

What is the problem and how would you correct it?

b. The following elements are missing from the code, but the code has comments `//*1, //*2, ...` in their places. Note that there is one more number than there are elements (not all the placements are used). Write the number corresponding to each one for its correct placement.

```java
//*___4___ fileScan = new Scanner(System.in);
//*___2___ throws IOException
//*___6___ outfile.close();
//*___1___ import java.io.*;
//*___3___ PrintWriter outfile = new PrintWriter ("out.txt");
```

c. Suppose the code has been corrected according to a and b, above, so now you are to run it with a text file `products.txt`, (available in the same folder as the code) with the contents below. Some of the input from this file, however, causes the code to throw exceptions. For each line in this text file, mark it either as “ok” or specify which of the following exceptions will be thrown; choose from one of the following:

- NumberFormatException
- FileNotFoundException
- StringIndexOutOfBoundsException
- arrayIndexOutOfBoundsException

```
products.txt

AAA6AAAAA  ____________NumberFormatException _________
AAA3446AAR  ______________ ok
AAA1111AAR  ______________ ok
AAA1111111  ______________ ok
BBB2222AAR  ______________ ok
CCC11111111  ______________ ok
DDD1234  ____________StringIndexOutOfBoundsException _________
DDD123456R  ______________ ok
```

Villanova University    CSC 1051    www.csc.villanova.edu/~map/1051    Dr. Papalaskari
d. Suppose you would like to catch and handle NumberFormatException by skipping this input using the following catch clause:

```
catch (NumberFormatException exception)
{
    System.out.println("Bad code: " + code);
}
```

In order to do this, you need to include some lines in the try block and add the catch immediately after it. The lines included should be:

```
from line ___35___ to line ___45_____
```

e. Suppose the code has been corrected to handle all exceptions caused by ill-formed input in the file. What can you expect for the output from this program and what will be the contents of the file out.txt?

```
OUTPUT: out.txt

Bad code: AAAAAAAAAA
Bad code: DDD1234
# of valid codes entered: 6
# of banned codes entered: 2

AAA3446AAR B
AAA1111AAR
AAA1111111
BBB2222AAR B
CCC111111111
DDD123456R
```

Villanova University    CSC 1051    www.csc.villanova.edu/~map/1051    Dr. Papalaskari
// ProductCodesF18Final.java
import java.util.Scanner;
import java.io.*;

public class ProductCodesF18Final {
    public static void main(String[] args)
            throws IOException {
        String code;
        char zone;
        int district, valid = 0, banned = 0;

        File inFile;
        inFile = new File("products.txt");
        Scanner fileScan;

        PrintWriter outfile = new PrintWriter("out.txt");

        try {
            fileScan = new Scanner(inFile);
        } catch (IOException e) {
            System.out.println("products.txt" + " not found");
            System.out.println("Input from keyboard instead");
            fileScan = new Scanner(System.in);
        }

        while (fileScan.hasNext()) {
            code = fileScan.nextLine();
            zone = code.charAt(9);
            district = Integer.parseInt(code.substring(3, 7));
            valid++;
            outfile.print(code);

            if (zone == 'R' && district > 2000)
                banned++;
            outfile.print("\tB");

            outfile.println();
        }

        System.out.println("# of valid codes entered: "+ valid);
        System.out.println("# of banned codes entered: "+banned);
        outfile.close();
    }
}
The Conditional Operator Syntax
- The value of the entire conditional operator is the value of the boolean expression.
- Example: Rewrite this:
  
  ```java
  if (a > b) System.out.print("positive");
  else System.out.print("negative");
  ```

  Using conditional operator:
  
  ```java
  System.out.print((a > b) ? "positive" : "negative");
  ```

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

Example:
```java
if (credits == 0) System.out.print("Grades:");
else System.out.print("GPA = ");
```
Example: Stars.java

```java
public class Stars {
    public static void main(String[] args) {
        System.out.print("*****");
        // Print a triangle shape using asterisk (*) characters
        for (int row = 1; row <= 5; row++) {
            for (int star = 1; star <= row; star++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

**UML Class Diagrams**

UML = Unified Modelling Language

- Example: A UML class diagram for the RollingDice program.

**Arrays**

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

- Initialization:
  ```java
  double[] scores = {7.9, 8.7, 9.4, 8.2, 9.8, 8.7, 8.1, 7.4, 9.1, 9.1};
  ```

- Size of array: `scores.length`

- Arrays as Parameters

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

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

**Arrays as Parameters**

- 2D Arrays - Overview

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

- ```java
  int courseGrade[1][4] = [[7.9, 8.7, 9.4, 8.2], [6.7, 9.8, 8.7, 8.1], [7.4, 9.1, 8.2, 8.0]];
  ```

- ```java
  System.out.println(average(ratings));
  ```

- Assumee a defintion for method average:

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