CS 1331 Exam 1

Jared Keenan Butler

TOTAL POINTS

80 / 100

QUESTION 1

Multiple Choice 30 pts

1.1 a **3 / 3**

✓ **- 0 pts** **A**

1.2 b **3 / 3**

✓ **- 0 pts** **C**

1.3 c **3 / 3**

✓ **- 0 pts** **D**

1.4 d **3 / 3**

✓ **- 0 pts** **C**

1.5 e **3 / 3**

✓ **- 0 pts** **D**

1.6 f **3 / 3**

✓ **- 0 pts** **A**

1.7 g **0 / 3**

✓ **- 3 pts** **Not A**

1.8 h **3 / 3**

✓ **- 0 pts** **B**

1.9 i **0 / 3**

✓ **- 3 pts** **Not B**

1.10 j **3 / 3**

✓ **- 0 pts** **Correct (free points)**

QUESTION 2

Multiple Choice 15 pts

2.1 a **3 / 3**

✓ **- 0 pts** **A**

2.2 b **3 / 3**

✓ **- 0 pts** **C**

2.3 c **3 / 3**

✓ **- 0 pts** **A**

2.4 d **3 / 3**

✓ **- 0 pts** **C**

2.5 e **3 / 3**

✓ **- 0 pts** **A**

QUESTION 3

Short Answer 25 pts

3.1 a **5 / 5**

✓ **- 0 pts** **Correct**

3.2 b **5 / 5**

✓ **- 0 pts** **Correct**

3.3 c **5 / 5**

✓ **- 0 pts** **"1331 1331 1331 " (quotation marks not required)**

3.4 d **5 / 5**

✓ **- 0 pts** **Correct**

3.5 e **5 / 5**

✓ **- 0 pts** **Correct**

QUESTION 4

Tracing and Programming 15 pts

4.1 a **1 / 5**

✓ **- 2 pts** Missing newlines

✓ **- 2 pts** Incorrect numbers

4.2 b **0 / 5**

✓ **- 5 pts** Not an Enum

4.3 c **0 / 5**

✓ **- 2.5 pts** Direction type is not Compass

✓ **- 2.5 pts** Assigned value incorrect (not Compass.value)

QUESTION 5

Programming 15 pts

5.1 a **10 / 10**

✓ **- 0 pts** Correct

5.2 b **5 / 5**

✓ **- 0 pts** Correct

CS 1331 Exam 1

Fall 2018

Name (print clearly): Jared Keenan Butler

9-Digit GTID: 903395728 Section (e.g., B1): B02

Signature: Jared Butler

- Failure to properly fill in the information on this page will result in a deduction of up to 5 points from your exam score.
- Signing signifies you are aware of and in accordance with the **Academic Honor Code of Georgia Tech** and that you will not discuss this exam with other students.
- Calculators and cell phones are NOT allowed.
- This is an object-oriented programming test. Java is the required language. Java is case-sensitive. DO NOT WRITE IN ALL CAPS. A Java program in all caps will not compile. Good variable names and style are required. Comments are not required.
- You have **45 minutes** to take the exam. If you keep writing after time is up or fail to turn in your exam immediately after time is up, you may receive a 0.

Question	Points per Page	Points Lost	Points Earned	Graded By
Page 1	30	-	=	
Page 2	15	-	=	
Page 3	25	-	=	
Page 4	15	-	=	
Page 5	15	-	=	
TOTAL	100	-	=	

1. Fill in the bubble next to the **best** answer.

[3] (a) Which of the following may appear on the **left hand side** of an assignment statement?

- ● a variable
- ○ an expression
- ○ an instance of a class which has overriden `operator=`
- ○ a previously defined constant.

[3] (b) Every variable has a _____.

- ○ name
- ○ type
- ● All of the above.

[3] (c) The length of an array ...

- ○ is determined at runtime.
- ○ is an `int` value.
- ○ doesn't change once the array is created.
- ● All of the above.

[3] (d) When is a value bound to a variable?

- ○ in an assignment statement
- ○ when an argument is passed to a method
- ● all of the above

[3] (e) Which of the following control structures is susceptible to unintentional fall-through?

- ○ `while`
- ○ `do-while`
- ○ `for`
- ● `switch`

[3] (f) When are types checked in a Java program?

- ● at compile time.
- ○ at run-time
- ○ never – Java is weakly typed.

[3] (g) Java classes provide

- ○ encapsulation
- ○ dynamic typing
- ● separate compilation

[3] (h) Given `int[] a = new int[5]`, what is the value of `a[1]`?

- ○ `null`
- ● `0`
- ○ `1`

[3] (i) Given `boolean shakeAndBake = true`, what is the value of `shakeAndBake ? "first" : "last"`?

- ○ `shakeAndBake`
- ○ `"first"`
- ● `"last"`

[3] (j) Meow!

- ● true

 Points available: 30 - points lost: _____ = points earned: _____. Graded by: _____

2. Fill in the bubble next to the **best** answer.

```
public class Coffee {
    public String name;
    public int sizeOunces;

    public Coffee(String name) {
        this(name, 16);
    }
    public Coffee(String name, int size) {
        name = name;
        sizeOunces = size;
    }
    public String getName() { return name; }
    public int getSizeOunces() { return sizeOunces; }
    public String toString() {
        return name + ": " + sizeOunces + " ounces";
    }
}
```

Assume the following statements have been executed:

```
final Coffee[] menu = {new Coffee("Mocha"), new Coffee("Espresso"),
                       new Coffee("Iced", 24)};
Coffee winter = menu[0];
menu[1].name = "Latte";
menu[2].name = "Macchiato";
```

[3] (a) What is the value of `menu.length`?

- ● 3
- ○ 2
- ○ null

[3] (b) What is printed on the console by `System.out.println(winter);`?

- ○ `Mocha: 16 ounces`
- ○ `Mocha: 24 ounces`
- ● `null: 16 ounces`
- ○ `null: 24 ounces`

[3] (c) Is the statement `menu[1] = null;` legal?

- ● Yes.
- ○ No.

null is default right?

[3] (d) Which of the following expressions is the correct way to test whether `menu[1]` and `menu[2]` have the same name?

- ⊖ `menu[1] == menu[2]`
- ○ `menu[1].getName() == menu[2].getName()`
- ● `menu[1].getName().equals(menu[2].getName())`
- ⊖ `menu[1] is menu[2]`

[3] (e) Which of the following expressions is the correct way to test whether `menu[0]` and `menu[1]` have the same sizeOunces?

- ● `menu[0].getSizeOunces() == menu[1].getSizeOunces()`
- ○ `menu[0].getSizeOunces().equals(menu[1].getSizeOunces())`
- ○ `menu[0] == menu[1]`
- ○ `menu[0].sizeOunces is menu[1].sizeOunces`

3. Short Answer

[5] (a) Assume you are at the command line in the directory of the file that contains the definition for a Java class `TicTacToe` . Write the command that you would execute on the command line to compile `TicTacToe` .

javac TicTacToe.java

[5] (b) Write the command that will execute the `TicTacToe` class you compiled above.

java TicTacToe

[5] (c) What will the following code print? 2 4 8

```
for (int j = 2; j <= 8; j *= 2) {
    System.out.print("1331 ");
}
```

"1331 1331 1331 "

(quotes will not be there, just to indicate where it starts and ends)

[5] (d) Convert the above for-loop into an equivalent while loop.

```
int j = 2;
while (j <= 8) {
    System.out.print("1331 ");
    j *= 2;
}
```

[5] (e) Consider the following 2d array: `int[] arr = {{0, 0}, {0, 0}};` Write three lines of code to change the array's content to `{{0, 1}, {2, 3}}`.

```
For (int i=0; i<arr.length; i++)
    For (int j=0; j<arr[i].length; j++)
        arr[i][j] = i + j + arr[0][1];
```

 Points available: 25 - points lost: _____ = points earned: _____. Graded by: _____

4. Tracing and Programming

[5] (a) What will be printed to the console when the class below is run from the command line?

```
public class FizzBuzz {
    public static void main(String[] args) {
        for (int i = 1; i <= 15; i++) {
            if ((i % 3 == 0) && (i % 5 == 0)) {
                System.out.println("FizzBuzz");
            } else if (i % 3 == 0) {
                System.out.println("Fizz");
            } else if (i % 5 ==0) {
                System.out.println("Buzz");
            } else {
                System.out.println(i);
            }
        }
    }
}
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

"1 2 Fizz 4 Buzz 6 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz"

(quotes just to indicate start & end of print)

[5] (b) Write an enum called `Compass` which has a value for each cardinal direction (north, south, east, west). Be sure to follow code conventions for enum constants.

```
public static final enum[] Compass = {NORTH, SOUTH, EAST, WEST};
```

[5] (c) In a single line of code, create a variable `direction` that holds one of the values of your enum.

```
enum direction = Compass[0];
```

 Points available: 15 - points lost: _____ = points earned: _____. Graded by: _____

5. Programming

[10] (a) Write a static method called `reverseString` that takes in a `String` parameter and returns a `String` which is the parameter reversed. For example, `reverseString("A banana")` should return the `String` "ananab A". No imports are allowed. Hint: The `String` class has a constructor that takes in a character array: `String(char[] input)` and a method `charAt(int i)` that returns the `char` at index `i` of the `String` instance on which it is called..

```
public static String reverseString(String str) {
    String result = str.substring(str.length()-1);
    for (int i=(str.length()-2); i>=0; i--) {
        result += str.substring(i, i+1);
    }
    return str;
}
```

[5] (b) Write a main method (including the header) that receives a `String` as the first command-line argument to the program, reverses it, and prints the reversed string to the console. You may assume that `main` will be in the same class as your `reverseString()` method above, and the user will only enter valid input.

```
public static void main (String[] args) {
    System.out.println(reverseString(args[0]));
}
```