



# University of Maryland College Park

## Department of Computer Science

### CMSC131 Spring 2019

### Exam #1Key

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE):

STUDENT ID (e.g. 123456789):

#### Instructions

- Please print your answers and use a pencil.
- Do not remove the staple from the exam. Removing it will interfere with the Gradescope scanning process.
- To make sure Gradescope can recognize your exam, print your name, write your directory id at the bottom of pages with the text DirectoryId, provide answers in the rectangular areas provided, and do not remove any exam pages. Even if you use the provided extra pages for scratch work, they must be returned with the rest of the exam.
- This exam is a closed-book, closed-notes exam, with a duration of 50 minutes and 200 total points.
- Your code must be efficient.
- Multiple choice questions have only one answer unless indicated otherwise.
- You don't need to use meaningful variable names; however, we expect good indentation.

#### Grader Use Only

#1	Problem #1 (Miscellaneous)	50	
#2	Problem #2 (Diagram)	50	
#3	Problem #3 (Cashier)	100	
<b>Total</b>	Total	200	

## **Problem #1 (Miscellaneous)**

1. (3 pts) Java compilers produce:
- Assembly code.
  - Byte Code.
  - Machine code (can run on the computer CPU).
  - None of the above.

Answer: b.

2. (3 pts) The Java Virtual Machine runs
- Java code
  - Java bytecode
  - HTML
  - None of the above.

Answer: b.

3. (3 pts) A class is:
- A blueprint for the structure of an object
  - An object
  - A Java Virtual Machine
  - None of the above.

Answer: a.

4. (3 pts) The following code fragment prints:

```
int y = 4;
int x = y++;
System.out.println(x);
```

- 4
- 5
- 0
- 45

Answer: a.

5. (3 pts) The following code fragment prints:

```
int x = -1, y = 7;
boolean answer = ((++x > 0) && (++y == 8));
System.out.println(answer + " " + y);
```

- false 0
- true 7
- false 8
- true 8

Answer: None.

Grading: Everyone gets credit for this one as no valid answer was provided.

6. (3 pts) The expression "Albert".compareTo("Bob") returns:
- A negative number
  - A positive number
  - Zero
  - None of the above.

Answer: a.

7. (3 pts) When is the body of a **do while** never executed?
- When the appropriate condition is provided.
  - Never

Answer: b.

8. (3 pts) Which of the following could be used to name variables in Java? We are not asking if they are good style, just whether or not they are permissible. Circle all that apply.

Dream#      34Warm      &flat      car\_component

Answer: car\_component

9. (4 pts) Name two primitive types used to store integers in addition to **int** and **long**.

Answer: byte and short.

10. (4 pts) Assuming **x**, **y** and **z** are integer variables, complete the following assignment that will initialize the **answer** variable with the value true if **x**, **y** and **z** are in increasing order and false otherwise. For example, 5, 10, 21 are in increasing order; 5, 3, 21 are not.

```
boolean answer = x < y
```

Answer: `x < y && y < z.`

11. (10 pts) Re-write (in the box) the following code fragment using a *for-loop*. The body of the for loop can only have the `System.out.println` statement (no other statement).

```
int y = 3, val = 10;
while (y >= 1) {
    System.out.println("val: " + val);
    val *= 2;
    y--;
}
```

Answer:

```
for (int y = 3, val = 10; y >= 1; val *= 2, y--)
{
    System.out.println("val: " + val);
}
```

12. (4 pts) Complete the following assignment so we are able to print the following message. Notice that double quotes surround the message.

```
"C:\home\tmp"
```

```
String path =  
System.out.println(path);
```

Answer: `"\"C:\\home\\tmp\""`

13. (4 pts) Define a String constant named BEST\_SCHOOL that has as value "UMCP".

Answer: `final String BEST_SCHOOL ="UMCP"`

## Problem #2 (Diagram)

Complete the program below that prints a rectangle. The program will read the size (an integer value). It will then generate a rectangle with a number of rows that corresponds to size and a number of columns that is three times the number of rows. One third of the rectangle columns (leftmost ones) use the # character; the rest will use the \* character. Use the message "Enter size:" to read data. You can assume users will provide correct data and a size value larger than or equal to 1. Below we have provided two examples of running the program. Underlined text represent input provided by the user. Remember, your program must work for other values. You will lose credit if you use more than two loops in your solution (a nested loop represents two loops).

```
Enter size: 2  
#####  
#####
```

```
Enter size: 5  
#####  
#####  
#####  
#####  
#####
```

### One Possible Solution

```
public class Diagram {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Enter size: ");  
        int size = scanner.nextInt();  
  
        for (int row = 1; row <= size; row++) {  
            for (int col = 1; col <= size * 3; col++) {  
                if (row <= size && col <= size) {  
                    System.out.print("#");  
                } else {  
                    System.out.print("*");  
                }  
            }  
            System.out.println();  
        }  
        scanner.close();  
    }  
}
```

## Problem #3 (Cashier)

Complete the implementation of the program below that represents a cashier system in a supermarket. This supermarket only sells two items: "bread" with a cost of \$2.00 and "milk" with a cost of \$3.00. Your program will read an item's name using the message "Enter item:". If the item entered is "bread" or "milk", you will increase the count of items seen by 1 and adjust the total amount due by the item's price. An item can be entered multiple times (each instance must be counted). If the user enters "done" as an item's name, the program will print the total number of items seen (using the message "Items:") and the total amount due (using the message "Total:\$"). The number after "Total: \$" does not need to have two decimal digits after the period (any format is fine). If the user enters as item's name something other than "bread", "milk" or "done", the program will print the message "Invalid item" and continue reading the next item. Invalid items do not change the current item's count nor total amount due. Below we are providing an example of running the program. Underlined text represents input provided by the user. Remember, your program must work for other values.

```
Enter item: milk
Enter item: paper
Invalid item
Enter item: bread
Enter item: milk
Enter item: done
Items: 3
Total: $8.0
```

### One Possible Solution

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    String item;
    double total = 0;
    int totalItems = 0;
    do {
        System.out.print("Enter item: ");
        item = scanner.next();
        if (!item.equals("done")) {
            if (item.equals("bread")) {
                total += 2.0;
                totalItems++;
            } else if (item.equals("milk")) {
                total += 3.0;
                totalItems++;
            } else {
                System.out.println("Invalid item");
            }
        }
    } while (!item.equals("done"));
    System.out.println("Items: " + totalItems);
    System.out.println("Total: $" + total);

    scanner.close();
}
```