CMSC131 Final Exam Practice Questions

Disclaimer: The following are questions that try to provide you with some practice material for the final exam. By no means do they represent the only material you should know for the final exam. We will not be providing answers to this set of questions, however, feel free to see the instructors or TAs for assistance during office hours. Some of the questions can be considered challenging.

A. Miscellaneous

1. What is the Java Virtual Machine?
2. What is byte code?
3. How can you generate bytecode for a program called Test.java?
4. What is the difference between a syntax error and a run time error?
5. Can every class in Java have a main method?
6. Write an output statement that prints the following string: ***the path to use is "home\games"***. The quotes and backslash must be printed.
7. Given the following code:

```
Scanner scanner = new Scanner(System.in);
String name = scanner.next();

if (name != null && name.length() >= 5) {
       System.out.println(name);
}
scanner.close();
```

   a. Does it makes a difference if the expression **name != null** appears after **name.length() >= 5**?
   b. Which Java feature associated with the **&&** and || operators is the above example relying on?

8. What is the default value for a local variable?
9. What is the default value for an instance variable?
10. Rewrite the following cascaded if statement using a switch statement.

```
Scanner scanner = new Scanner(System.in);
int age = scanner.nextInt();

if (age == 21) {
       System.out.println("Legal age");
} else if (age == 25) {
       System.out.println("rental possible");
} else if (age == 15) {
       System.out.println("Party");
} else {
       System.out.println("Unknown");
}
scanner.close();
```

11. Are the numeric comparisons in the following example valid?

```
Scanner scanner = new Scanner(System.in);
int a = scanner.nextInt(), b = scanner.nextInt();
double c = scanner.nextDouble(), d = scanner.nextDouble();

if (a == b) {
       System.out.println("Same integers");
}

if (c == d) {
       System.out.println("Same doubles");
}
```

12. Why the following **swapWrong** method does not swap the values? Would it make a difference if instead of integer parameters we were using **Integer** parameters?

```
public static void wrongSwap(int x, int y) {
        int temp = x;

        x = y;
        y = temp;
}
public static void main(String[] args) {
        int a = 10, b = 20;

        wrongSwap(a, b);
        System.out.println(a + " " + b);
}
```

13. Write pseudocode for a program that determines whether any two consecutive values in a sequence of values differ by at most a particular value. For example, for the sequence **9, 10, 18, 16** two consecutive values differ by at most two.
14. How do the **String** and **StringBuffer** classes differ?
15. What is the difference between method **overloading** and method **overriding**? Provide examples.
16. Provide an expression using the **Random** class that generates a random value in the range [4, 10].
17. What is the difference between **final int x** and **final static int x**?
18. What is the difference between a shallow copy and a deep copy?
19. Which two methods of the **Object** class do we often override?

B. Classes

1. What is the role the constructor plays in a class?
2. What is a copy constructor?
3. What is the current object (**this** reference)?
4. What is the heap?
5. What is the difference between a class and an object?
6. What is the value stored by a reference variable?
7. What is the difference between a **static** method and a **non-static** method?
8. When should you define a method as static?
9. Define an interface called **Road** with the following methods:
    a. String getName()
    b. double getDistance()

10. Define a class called **Highway** that has specifications below. The class implements the above **Road** interface.
    a. Instance variables
        i. **number** (integer)
        ii. **name** (String)
        iii. **distance** (double)
    b. Non-static methods
        i. **Constructor** – Takes a number, name and distance as parameters.
        ii. **Default constructor** – Initializes the object with the values 0, "NO_NAME", and 0 as the values for number, name and distance, respectively.
        iii. **get** and **set** methods for all instance variables.
        iv. **equals** method – Two roads are considered equal if they have the same number.
        v. **toString** – Returns a string with object information (number, name, distance).
    c. Static methods
        i. **getCount** – Returns the number of **Road** objects that have been created.
    d. This class implements the **Comparable** interface. Objects will be compared based on the number, with highways having a lower number appearing first, if we were to sort an ArrayList of them.

Feel free to add any other variables/methods you understand are needed.

11. Define a class called **Roadtrip** with the following specifications:

    a. Instance variables
        i. **name** (String)
        ii. **route** (ArrayList<Road>)
    b. Static methods
        i. **addRoad** – Static method that has as parameters a **Roadtrip** object and a **Road**. The Road will be added to the specified **Roadtrip** object.
    c. Non-static methods
        i. **Constructor -** Takes a name parameter representing the roadtrip's name.
        ii. **addRoad** – Appends a **Road** object to the ArrayList. You must implement this method using the **addRoad** static method above. **The method returns a reference to the current object.**
        iii. **removeRoad** – Removes the road from the ArrayList associated with a particular name.
        iv. **tripDistance** – Returns the total distance associated with the roadtrip.
        v. **equals** method – Two roadtrips are the same if they have the same name.
        vi. **toString** – Returns a string with the roadtrip name, followed by information about each road in the route.

    Feel free to add any other variables/methods you understand are needed.

12. Assume you also have a class named **LocalRoad** that implements the **Road** interface. Implement the method **printInfo** that has the following signature: `void printInfo(ArrayList<Road> roads)`
    The method will print the road number of highways that are part of roads ArrayList. Remember that not all roads are highways (**hint:** use instanceof).

C. Arrays

1. Define a method called **rotateLeft** that will rotate elements in an integer array by a particular number of positions. For example, given an array with the values **10, 4, 5, 70**, rotating the array by 2 will generate the array **5, 70, 10, 4**.

2. Define a method called **hasThreeOfAKind** that returns true if an array of length 5 has three values that are the same and false otherwise.

3. Define a method called **drawHourClass** that draws an hour glass with a specific number of lines. For example, calling drawHourClass(4) will draw the following diagram:

```
****
 **
 **
****
```

4. Define a method called **filter** that has the prototype below and that relies on the **Highway** class you defined above. The method returns a new array with **copies** of Highway objects that have a number in the specified range. The will throw an IllegalArgumentException if the list is null or **startRange** is greater than **endRange**.

```
public static Highway[] filter(Highway[] list, int startRange, int endRange)
```

5. Define a method called **includesAll** that has the prototype below and relies on the above **Highway** class. The method determines whether all the highways in list1 and part of list2.

```
public static boolean includesAll(Highway[] list1, Highway[] list2)
```

6. Define a method called **removeConsecutiveDuplicates** that has the prototype below and relies on the above **Highway** class. The method removes duplicate highways that appear one after another in the list. The method will return the number of highways removed.

```
public static int removeConsecutiveDuplicates(Highway[] list)
```

7. Define a static method called **firstRow** that given a 2-dimensional array of doubles, finds the first row that consists of an increasing sequence of values, that is, it finds the smallets i such that:

```
array[i][0] < array[i][1] < array[i][2] < ...
```

If such a row exists, a copy of the contents of that row are returned as a one dimensional array; otherwise null is returned.

8. The **Cell** class is defined as follows:

```
public class Cell {
   private char color;

   public Cell(char color) {
          setColor(color);
   }

   public char getColor() {
          return color;
   }

   public void setColor(char color) {
          if (color == 'R' || color == 'G' || color == 'B') {
                this.color = color;
          } else {
                this.color = 'Y';
          }
   }

   public String toString() {
          return Character.toString(color);
   }
}
```

   a. Is the above class mutable or immutable?
   b. Does the above class has a default constructor?
   c. Define a **recursive** static method named **colorRow** that will color a row with random colors.

```
public static void colorRow(Cell[][] cells, int rowIndex)
```

   d. Define a **recursive** static method named **flip** that will flip cells that surround the one associated with **rowIndex** and **colIndex**. Flipping a cell requires changing an 'R' cell to a 'B'cell; otherwise no change takes place.

```
public static int flip(Cell[][] cells, int rowIndex, int colIndex)
```

D. Recursion

Note: For these problems feel free to add at most one auxiliary method (if needed).

1. See previous problem.
2. Define a recursive static method called **readInteger()** that reads a string representing an integer value using JOptionPane.showInputDialog, and converts the value to an integer using Integer.parseInt(). If the provided string represented an integer value, the method will return the integer value; otherwise the method will print the message "Invalid value" and call itself to read another value. Hint: You must handle the NumberFormatException while implementing this method.
3. Given a string, define a recursive method that computes the number of vowels in the string.

4. Given a string, define a recursive method that returns a string where vowels have been replaced by the @ symbol.
5. Given an array, define a recursive method that computes the product of all the elements in the array.

E. Design

1. Write a Java program that simulates a guessing number game. The program will generate a random value the user needs to guess and keep asking the user for a guess until the user provides the expected value. If a value has already provided, the program will print the message "Value already provided". If the value is higher than the value to guess, the program will print "Too high"; if it is less, the program will print the message "Too low". Once the user enters the expected value, the program will print the number of attempts.
2. Define which classes / classes you might need in order to implement software that keeps track of students' submissions (similar to the submit server). You don't need to implement any code; just describe the classes / interfaces you might need and their relationship. For each student, the submit server keeps track of every submission and classifies them on-time, late or very-late. A submission is associated with a student and a project. Several projects can be associated with a class.

F. Memory Maps

Draw a memory diagram showing both the stack and the heap at the moment this program reaches the point marked **/* HERE */.**

```
public class MemMap {

	public static void proc1(Apple passed, int x, Apple[] container) {
		String newColor = "Green";

		passed.update(x, newColor);
		x += 7;
		passed = null;
		container[1] = container[0];
		container[0] = null;
		/* HERE */
	}

	public static void main(String[] args) {
		String color1 = "Red";
		int size1 = 10, size2 = 20;

		Apple a1 = new Apple(size1, color1);
		Apple[] basket = new Apple[2];
		basket[1] = new Apple(1, "Yellow");
		basket[0] = a1;
		proc1(a1, size2, basket);
	}
}

public class Cell {
	private char color;

	public Cell(char color) {
		setColor(color);
	}

	public char getColor() {
		return color;
	}

	public void setColor(char color) {
		if (color == 'R' || color == 'G' || color == 'B') {
			this.color = color;
		} else {
			this.color = 'Y';
		}
	}

	public String toString() {
		return Character.toString(color);
	}
}
```