Final Exam Draft

The final exam is designed to be administered in two parts. The first part is an offline testing of programming fundamentals and concepts. These questions will be mostly Java based, with several questions requiring students to answer memorized aspects of Android. It is expected a student can complete this portion in 1.5 hours, but they are given 3 hours.

The second portion of the exam is a practical component. Students will be required to implement a functional Android application, incorporating several mandatory features. This half of the exam should emulate a take home portion of an interview, which verifies that a candidate is familiar with Android. It is expected a student can complete this portion in 3 hours.

```
package nyc.c4q;
public class FunctionsEatingFunctions {
    public static class StringBuilder{
        private String value;

        public StringBuilder(String start){
            value = start;
        }

        public StringBuilder append(String other){
            return new StringBuilder(value + other);
        }

        public String get(){
            return value;
        }
    }

    public void function1() {
        StringBuilder builder = new StringBuilder("start");
        builder.append("1");
        builder.append("2");
        builder.append("3");
        System.out.println(builder.get());
    }

    public void function2() {
        StringBuilder b1 = new StringBuilder("start").append("1");
        StringBuilder builder2 = new StringBuilder("end");
        builder2.append("2");
        System.out.println(builder.get());
        System.out.println(builder2.get());
    }
```

Question 1: What is the output of function1()?
Question 2: What is the output of function2()?

```
private static final int SIZE_ONE = 5;
private static final int SIZE_TWO = 2;
private static final int SIZE_OF_ALPHABET = 26;
private static final int ASCII_CAPITAL_A = 65;

private final Random random = new Random();

private String randomString() {
    int tempA = random.nextInt(SIZE_ONE) + SIZE_TWO;
    String value = "";
    for (int index = 0; index < tempA; index++){
       value += (char)(random.nextInt(SIZE_OF_ALPHABET) + ASCII_CAPITAL_A);
    }
    return value;
}
```

Class Random

An instance of this class is used to generate a stream of pseudorandom numbers.

int nextInt(int n)
Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Question 3:

A) **What is the minimum size of a String returned by randomString()?**
B) **What variable or constant determines the minimum size?**
C) **What is the maximum size of a String returned by randomString()?**
D) **In one sentence, describe how the char being added to the value string is calculated. Be sure to mention the smallest and largest integer that will be cast to a char.**

```java
private List<String> list1 = Arrays.asList("hello");
private List<String> list2 = Arrays.asList("hello" , "goodbye");

private List<String> expandList(
              List<String> stringList,
              int startPoint) {
    List<String> result = new ArrayList<>();
    for (int index = 0; index < startPoint; index++){
       result.add(stringList.get(index));
    }
    String stringToExpand = stringList.get(startPoint);
    result.add(stringToExpand);
    for (int index = 0; index < stringToExpand.length(); index++){
        result.add("" + stringToExpand.charAt(index));
    }
    for (int i = startPoint+1; i < stringList.size(); i++){
        result.add(stringList.get(i));
    }
    return result;
}
```

Question 4: What is the output of expandList(list1, 0)?
Question 5: What is the output of expandList(list1, 3)?
Question 6: What is the output of expandList(list2, 0)?
Question 7: What is the output of expandList(list2, 5)?

```java
public int popThenAdd(Stack<Integer> stack){
    if (stack.size() == 0){
        return 0;
    } else if (stack.size() == 1) {
        return stack.pop();
    } else {
        int value1 = stack.pop();
        int value2 = stack.pop();
        return value1 + value2 + popThenAdd(stack);
    }
}

public void popThenAppend(Stack<Integer> stack){
    while (stack.size() != 1){
        int value1 = stack.pop();
        int value2 = stack.pop();
        String result = value1 + "" + value2;
        stack.push(Integer.valueOf(result));
    }
}
```

Question 8: After initializing a stack with 5 items, and passing it to popThenAdd, how many times will popThenAdd be called? Include the first call with 5 items as call 1.

Question 9: After initializing a stack with the following code:

```java
Stack<Integer> stack = new Stack<>();
stack.push(2);
stack.push(4);
stack.push(6);
stack.push(8);
popThenAppend(stack);
```

What would be the output of System.out.println(stack.pop())?

```java
package nyc.c4q;
```

```
public class StateMachine {

    public enum Input {
        TURN_KEY,
        SHIFT_UP,
        SHIFT_DOWN,
        NONE
    }

    public enum State {
        OFF,
        IGNITION_TURNED,
        ENGINE_STARTING,
        ENGINE_IDLE,
        FIRST_GEAR,
        SECOND_GEAR,
        REVERSE,
        ENGINE_STOPPING
    }

    public static State getNextState(State currentState) {
        switch (currentState) {
            case OFF:
                break;
            case IGNITION_TURNED:
                return State.ENGINE_STARTING;
            case ENGINE_STARTING:
                return State.ENGINE_IDLE;
            case ENGINE_IDLE:
                return State.FIRST_GEAR;
            case FIRST_GEAR:
                return State.FIRST_GEAR;
            case SECOND_GEAR:
                return State.SECOND_GEAR;
            case REVERSE:
                return State.REVERSE;
            case ENGINE_STOPPING:
                break;
            default:
                throw new IllegalArgumentException("invalid state ");
        }
        return State.OFF;
    }
```

```
public static void handleState(State currentState) {
     switch (currentState) {
         case OFF:
             break;
         case IGNITION_TURNED:
             System.out.println("Ignition");
             break;
         case ENGINE_STARTING:
             System.out.println("Starting");
             break;
         case ENGINE_IDLE:
             System.out.println("Idle");
             break;
         case FIRST_GEAR:
             System.out.println("1st");
             break;
         case SECOND_GEAR:
             System.out.println("2nd");
             break;
         case REVERSE:
             System.out.println("Reversing");
             break;
         case ENGINE_STOPPING:
             System.out.println("Stopping");
     }
 }

 public void run(State initialState, Input[] inputs){
     State currentState = initialState;
     for (int index = 0; index < inputs.length; index++) {
         State nextState = getNextState(currentState);
         nextState = handleInput(currentState,
                                 nextState,
                                 inputs[index]);
         handleState(currentState);
         currentState = nextState;
     }
 }




  private State handleInput(State currentState,
```

```
                        State nextState,
                        Input input) {
        if (currentState == State.OFF && input == Input.TURN_KEY){
            return State.IGNITION_TURNED;
        } else if (currentState == State.IGNITION_TURNED &&
                   input != Input.NONE){
            return State.IGNITION_TURNED;
        } else if (currentState == State.FIRST_GEAR &&
                   input == Input.TURN_KEY){
            return State.ENGINE_STOPPING;
        } else if (currentState == State.FIRST_GEAR &&
                   input == Input.SHIFT_DOWN){
            return State.REVERSE;
        } else if (currentState == State.FIRST_GEAR &&
                   input == Input.SHIFT_UP){
            return State.SECOND_GEAR;
        }  else if (currentState == State.SECOND_GEAR &&
                    input == Input.TURN_KEY){
            return State.ENGINE_STOPPING;
        } else if (currentState == State.SECOND_GEAR &&
                   input == Input.SHIFT_DOWN){
            return State.FIRST_GEAR;
        } else if (currentState == State.REVERSE &&
                   input == Input.TURN_KEY){
            return State.ENGINE_STOPPING;
        } else if (currentState == State.REVERSE &&
                   input == Input.SHIFT_UP){
            return State.FIRST_GEAR;
        }
        return nextState;
    }

    public static void main(String[] args) {
        Input[] inputs = // Problems 10 - 12
        new StateMachine().run(State.OFF, inputs);
    }
}
```

For questions 10 - 12, assume the function output is used to initialize the inputs array. For each problem, you must document the transitions through the state machine

```
private static INPUT[] problem10() {
    return new INPUT[]{
            INPUT.NONE,
            INPUT.NONE,
            INPUT.TURN_KEY,
            INPUT.NONE,
            INPUT.NONE,
            INPUT.NONE,
            INPUT.TURN_KEY,
            INPUT.NONE,
    };
}
```

Problem 10:

Current State	**Input**	**Next State**	**Output**
Off			

```
private static INPUT[] problem11() {
    return new INPUT[]{
            INPUT.TURN_KEY,
            INPUT.SHIFT_DOWN,
            INPUT.SHIFT_UP,
            INPUT.TURN_KEY,
            INPUT.NONE,
    };
}
```

Problem 11:

Current State	**Input**	**Next State**	**Output**
Off			

```
private static INPUT[] problem12() {
    return new INPUT[]{
            INPUT.TURN_KEY,
            INPUT.NONE,
            INPUT.SHIFT_DOWN,
            INPUT.SHIFT_UP,
            INPUT.TURN_KEY,
            INPUT.NONE,
    };
}
```

Problem 12:

Current State	Input	Next State	Output
Off			

```
public String printStack(Stack<Integer> stack){
    ArrayList<Integer> output = new ArrayList<>();
    while (!stack.isEmpty()) {
        output.add(stack.pop());
    }
    String result = “”;
    for (int index = output.size()-1; index >= 0; index--){
        result += output.get(index);
    }
    return result;
}
```

The printStack function is supposed to return a string that is the concatenation of all the values in the stack, without modifying the stack. The first value in the string should be the first value pushed onto the stack.

Problem 13:
Write a function that will test whether or not printStack returns the correct value.

Problem 14:
Write a function that will test whether or not the stack passed into printStack is modified.

Practical Exam - Building an Android App from Scratch

Write an Android app that connects to your instructors machine (Ask for the ip address).

The app should have two Activities, one named HomeScreen, and one named DetailScreen.

The HomeScreen activity should have 1 fragment. This fragment will have a text view and a recycler view. The content should be populated from {instructor_website}/homescreen

The display json object will hold the text and text color to display. The text color will be defined by its name only, and the value will be in the colorPalette list in the son response.

All the colors in the colorPalette list must be displayed in the recycler view.

Clicking on an item in the recycler view will navigate to the DetailScreen activity.

The DetailScreen activity should have 1 fragment. This fragment will have a TextView with the background color set to the color that was selected on the Homescreen. The text in the TextView should be populated from {instructor_website}/{color_name}.