Yu-Ching Ho

CSC 204
Test 4
April 20, 2017

15

This is a closed book, closed notes test. Answer all of the questions in the space provided. Make sure that all of your answers are legible so that they can be graded. This is a 100 point test and points for each question are given in []. Enjoy!!

String favColors[] = {"red", "green", "blue"};

1. In <u>one line</u>, declare an array of Strings named *favColors* and initialize it to contain your three favorite colors. Remember, this is an array of Strings – not Colors. **[5]**

String favColors [] = new String [] {"red", "blue", "green"};

-2

2. What is the <u>index</u> of your favorite color in the last array slot? **[2]**

favColors [1];

-2

3. What is the output of the following code? **[3]**

```
System.out.println(favColors[favColors.length/2]+ "\n");
```

blue

for(int i=0; i < favColors.length; i+2){
favColors[i] = "orange";
}

4. Write a two-line piece of Java code that will change all of the "even indexed" colors in *favColors* to be "Orange" (remember *favColors* is just an array of Strings). **[5]**

```
for(int i=0; i < favColors.length ; i++) {
    if( favColors[i] %2 == 0) System.out.print("Orange");
}
```

i%2==0

-3

5. All of a sudden I have five favorite colors that I want to hold in my array of colors. Help me make the array *favColors* hold five colors instead of just three. You can assign any "colors" you want as the new Strings. You should insert the code suggested by the comments. **[10]**

```
// Declare a new array named tmpColors of Strings with room for 5 Strings.
String tmpColors [] = new String [5];
// Assign a new color to the first slot of tmpColors
tmpColors [0] = "yellow";
// Assign a new color to the last slot of tmpColors
tmpColors [4] = "black";
// Copy the original colors of favColors into the middle of tmpColors.
for (int i= 0 ; i < favColors.length ; i++ )
    tmpColors [ i+1 ] = favColors[ i ];
// Now make favColors refer to the new array of 5 colors.
favColors [] = favColors [];
favColors = tempColors;
```

-1

-8

~~ArrayList<String> favToEat[] = new ArrayList<String>();~~

ArrayList<String> favToEat[] = new ArrayList<String>();
favToEat.add("pasta", "pizza", "coke");

6. Declare an ArrayList of Strings named *favToEat* and fill it with your three favorite things to eat. **[5]**

String favToEat[] = new ArrayList<String>;
favToEat[] = ("pasta", "pizza", "coke");

-4

7. What is the index of your favorite thing to eat in the first "slot"? **[2]**

favToEat[0];

8. Give the Java code that will print out your list of favorite things to eat one item per line of output. Use the enhanced `for` loop. **[4]**

```
for(String f : favToEat[]) {
    System.out.println(f);
}
```

-1

9. All of a sudden you decide that you have two more favorite things to eat. Show all the code necessary to add your newly found favorite things to eat to *favToEat*. Place one at the beginning of the list and one at the end. **[4]**

```
favToEat.add(0, "cake");
favToEat.add(4, "rice");
```

↖ don't need last index cause with no index, will add to bottom of list

10. Now, give the Java code that will print out your new and improved list of favorite things to eat one item per line of output, but this time print the list in reverse order. **[5]**

```
for( int i = favToEat.size()-1; i >= favToEat.size(); i--;)
    System.out.println(favToEat[i]);
}
```

-1

-6

11. Create a new array of Strings named *myFavs*, large enough to hold all of your *favColors* and all of your *favToEat* Strings. Do not use integer numbers as you create this… get the length/size from the Array and ArrayList. **[5]**

```
int i = favColors.length;
int j = favToEat.size();
Array myFavs[] = new String(i+j);
```

or `String myFavs[] = new String[favColors. + favToEat.si`

12. Copy your *favToEat* into the first part of *myFavs*. **[5]**

```
for(int i = 0; i <= favToEat.size(); i++) {
    myFavs[i] = favToEat[i]; array list
                favToEat.get(i);
}
```

-1

13. Copy your *favColors* into the rest of *myFavs*. **[5]**

```
for(int i=0; i < favColors.length; i++) {
    myFavs[i + (favToEat.size + 1)] = favColors.length[i];
}
```

or `myFavs[i + favToEat.size()] = favColors[i];`

14. Write an entire helper method named *countM*, that is passed an Array of Strings, like *myFavs*, and returns the integer count of how many of those Strings start with the letter "M" or "m". Recall that String objects have the `charAt(i)` method that returns the char at the position i. **[10]**

```
public static int countM(Array myFavs[]) {
    int count = 1;0
    for(int i = 0; i < myFavs.length; i++) {
        if(myFavs[i].charAt(0) == "M" || "m") {
            count++;
        }
    }
    return count;
}
```

String

'M'

-1

|| myFavs[i].charAt(0 == 'm'

-2

15. What is the output of the following program? **[10]**

```
public class sweets
{
  final static int GUESTS = 7;

  public static void main(String[] args)
  {
        String platter[]  = new String[GUESTS];
        fillUp (platter);
        eatFrom (platter);
  }

  private static void fillUp (String[] contents)
  {
        String bakery[] = {"Cookies","Cake","Brownies","Eclairs","Pudding",
                           "Pie","Tarts","Donuts"};
        for (int i= contents.length; i>0; i--)
        {
          contents [i-1] = bakery[i];
        }
  }
  private static void eatFrom (String[] contents)
  {
        for (int i= contents.length-1; i >= 0; i-=2)
        {
          System.out.println("eat: " + contents[i]);
        }
  }
}
```

contents[6] = Donuts
5 = Tarts
4 = Pie
3 = Pudding
2 = Eclairs
1 = Brownies
0

↑ correct

contents.length = 7

contents[4]
2
0

eat : Tarts
eat : Pudding
eat : Brownies

off by 1

+ only 3 of 4

thus Donuts
Pie
Eclairs
Cake

(-5)

-5

16. Write an entire helper method named `evenCount`, that is passed an ArrayList of Integers, and returns the integer count of how many of the Integers in the ArrayList are even. **[10]**

```
public static int evenCount(ArrayList<Integers> ArrayInt[]) {
    int count = 0;
    for(int i = 0; i < ArrayInt.size(); i++) {
        if(ArrayInt[i].intValue() % 2 == 0) {
            count++;
        }
    }
    return count;
}
```

ArrayInt.intValue()

-1

17. Help me write a method that is passed two arrays of doubles, and returns a new array of doubles that holds all of the values from the two passed in arrays of doubles. **[10]**

```
public static double[] merge2doubles(double []d1, double d2[])
{
    int a = d1.length;
    int b = d2.length;
    int c = a + b;
    new merge2doubles[c];
    for(int i = 0; i < c; i++) {
        merge2doubles[i] = d1[i];
        merge2doubles[i+a] = d2[i];
    }
    return merge2doubles;
}
```

PTO

out of bands.

use 2 loops

-3

BONUS [10]: Given an ArrayList of ArrayLists of Integers named `crazy`, sum all the Integers and print the sum. You may use the back of this sheet.

```
int sum = 0;
for(int i = 0; i <= crazy.size; i++) {
    sum = crazy[i] + sum;
}
return sum;
System.out.println(sum);
```

-4

17) double d3[] = new double[d1.length + d2.length];

```
for(int i = 0; i < d1.length; i++) {
    d3[i] = d1[i];
}
for(int i = 0; i < d2.length; i++) {
    d3[i + d1.length] = d2[i];
}
return d3;
```

Bonus)

```
int sum = 0;
for (ArrayList<Integer> al : crazy) {
    for(Integer i : al) {
        sum += i.intValue();
    }
}
System.out.println(sum);
```