CS-141 Basic Array Problems with Solutions

1-Dimensional Arrays

1. Declare and create an array of 25 integers called `myNums`. Initialize the array by setting the i^{th} element to the value 2*i.
   ```
   int myNums[] = new int[25];
   for (int i=0;i<myNums.length;i++) {
        myNums[i] = 2*i;
   }
   ```
2. Print all the elements of `myNums` using each of the following approaches:
   a. the `Arrays` class

   ```
   System.out.println(Arrays.toString(myNums));
   ```

   b. a *for-loop*

   ```
   for (int i=0;i<myNums.length;i++) {
       System.out.println("i=" + i +"  value= " + myNums[i]);
   }
   ```

   c. an *enhanced for-loop*. (Note, don't have access to a loop variable i)

   ```
   for (int value: myNums) {
       System.out.println("value = " + value);
   }
   ```

3. Write a method (it needs to be static if you are to be able to call it from main – do you see why?) which prints the elements (one element per line) of `myNums` using either 2b or 2c. The array should be created in main and passed in as a parameter.   Test your method by calling it from main().
   Pay attention to the syntax used for the argument (in main) and the parameter (in printNums). Also note, that generally it is not a good idea to do lots of stuff in main and call static methods but we do it here for expediency and as an opportunity to see how static works.

```
public static void main(String[] args) {
     int myNums[] = new int[25];
     for (int i = 0; i < myNums.length; i++) {
         myNums[i] = 2 * i;
     }
     printNums(myNums);
 }


  public static void printNums(int[] nums) {
     for (int i = 0; i < nums.length; i++) {
         System.out.println("i=" + i + "  value= " + nums[i]);
     }
   }
}
```

4. Write a method which returns a String containing all the elements of `myNums` which are divisible by the number 3. Elements should be separated by spaces. Test your method by calling it from main().

```
public static void main(String[] args) {
    int myNums[] = new int[25];
    for (int i = 0; i < myNums.length; i++) {
        myNums[i] = 2 * i;
    }
    System.out.println(print3Divide(myNums));
}
public static String print3Divide(int[] nums) {
    String result="Numbers divisible by 3:\n";
    for (int i = 0; i < nums.length; i++) {
        if (nums[i] % 3 == 0) result = result + " " + nums[i];
    }
    return result;
}
```

5. Swap the 5th element of `myNums` with the 10th element.

```
int temp = myNums[5];
myNums[5] = myNums[10];
myNums[10] = temp;
```

6. Write a method which swaps specified elements, e.g. its header might look like:

```
   public static void swap(int[] myArray, int i, int j) {
           int temp = myArray[i];
           myArray[i] = myArray[j];
           myArray[j] = temp;
   }
```

Why won't it work to write the method as follows:

```
   public static void swap(int val1, int val2)
```

where the call to swap would look something like

```
  swap(myArray[1], myArray[2]);
```

This won't work because you are passing in copies of the array elements and not the actual array elements.

What is the difference between pass by value and pass by reference? *Pass by value will copy the value of the argument (as in the case of a primitive type) into the parameter variable. Pass by reference will copy the address (i.e. of an object) into the parameter variable so that the argument and parameter point to the same object.*

7. Declare and create an array of 20 numbers (doubles) called `nums`. Initialize each element to a random number. Compute their average. (*Note, this is done with 2 loops below but one could also do it all in a single loop*)

```
double nums[] = new double[20];
for (int i = 0; i < nums.length; i++) {
    nums[i] = Math.random();
    System.out.println("i = " + i + ": value = " + nums[i]);
}
double sum = 0;
for (int i = 0; i < nums.length; i++) {
    sum += nums[i];
}
double average = sum / nums.length;
System.out.println("Average = " + average);
```

8. Insert a new value into the 5th index of `nums`, shifting items to make room. Note, the last item will be lost. (*Note, this is tricky – do you see why it works best to work backwards? Try doing it going forward.)*

```
for (int i = nums.length-1; i > 4 ; i--) {
            nums[i] = nums[i-1];
        }
        nums[5]= Math.random();
```

9. Remove the 10th index in the array `nums`, shifting items to remove the hole. Place the removed item into the last position of the array.

```
double temp = nums[10];
for (int i = 10; i < nums.length-1; i++) {
    nums[i] = nums[i+1];
}
nums[nums.length-1]= temp;
```

10. Use the Arrays class to sort `nums`. Print the array before and after the sort. (*Note, the sorting will overwrite the original array so, if you want to keep the original ordering, you need to make a copy of the array*)

```
System.out.println(Arrays.toString(nums));
```

```
Arrays.sort(nums);
System.out.println(Arrays.toString(nums));
```

11. Copy `nums` into an new array using the following approaches:
    a. The Arrays.copyOf() method

    ```
    double[] numsCopy = Arrays.copyOf(nums, nums.length);
    System.out.println(Arrays.toString(nums));
    System.out.println(Arrays.toString(numsCopy));
    ```

    b. Create a second array of the same size and use a loop to set the values of the second array to be the same as `nums`.

    ```
    double [] numsCopy = new double[nums.length];
    for (int i = 0; i < nums.length;i++) {
        numsCopy[i] = nums[i];
    }
    ```

12. Write a method which creates and returns an array of 20 random integers in the range 0 to 99.

    ```
    public static int[] createArray(int n) {
        int[] myArray = new int[n];
        for (int i=0; i < n; i++) {
            myArray[i] = (int) (100 * Math.random());
        }
        return myArray;
    }
    ```

    The call to this method in main would look like:
    `int[] a = createArray(20);`

13. Write a method which takes an integer array as a parameter and doubles the value of each of its elements.

    ```
    public static void doubleArray(int[] arr) {
        for (int i = 0; i < arr.length; i++) {
            arr[i] = 2 * arr[i];
        }
    }
    ```

    In main, one could call it as follows:

    ```
    int[] a = createArray(20);
    doubleArray(a);
    ```

2-Dimensional Arrays

1. Declare and create (using curly brackets {..}) a 2D array of Strings called `names` with 2 rows and 3 columns. Use whatever values you want. Use loops to print the array out.

```
String [][] names = { { "firstRowFirstCol", "firstRowSecondCol",
     "firstRowThirdCol"}, { "secondRowFirstCol", "secondRowSecondCol",
      "secondRowThirdCol"}};

for (int row = 0; row < names.length; row++) {
    for (int col = 0; col < names[row].length; col++) {
        System.out.print(names[row][col] + " ");
    }
    System.out.println("");
}
```

2. Declare and create a 2D array of `ints` called `cells` with 4 rows and 8 columns. Using loops, set the value of each element of `cells` to be equal to the product of (one plus its row index) and (one plus its column index).

```
int [][] cells = new int[4][8];
for (int row = 0; row < cells.length; row++) {
    for (int col = 0; col < cells[row].length; col++) {
        cells[row][col] = (1+row)*(1+col);
    }
}
```

You can check your results by printing out the array:
```
for (int row = 0; row < cells.length; row++) {
    for (int col = 0; col < cells[row].length; col++) {
        System.out.print((1+row)+"*" + (1+col)+"=" + cells[row][col] + "  ");
    }
    System.out.println("");
}
```

3. Try to answer the following questions about `cells`:
    a. What type of variable is `cells`? 2D array of ints
    b. What type is `cells[2]`? 1D array of ints
    c. What type is `cells[1][2]`? int
    d. What is the value of `cells.length`? the number of rows = 4
    e. What is the value of `cells[2].length`? the number of columns = 8
    f. Does `cells[2][3].length` make any sense? Why or why not? This does not make sense since `cells[2][3]` is just an integer and this does not have a length.

4. Compute and print the average of the values in each row of `cells`.

```
for (int row = 0; row < cells.length; row++) {
    double rowSum = 0;
    for (int col = 0; col < cells[row].length; col++) {
        rowSum += cells[row][col];
    }
    double rowAverage = rowSum / cells[row].length;
    System.out.println("The average of row " + row + " is " + rowAverage);
}
```

5. Compute and print the average of the values in each column of `cells`.

```
// assumes each row has the same number of columns
for (int col = 0; col < cells[0].length; col++) {
    double colSum = 0;
    for (int row = 0; row < cells.length; row++) {
        colSum += cells[row][col];
    }
    double colAverage = colSum / cells.length;
    System.out.println("The average of col " + col + " is " + colAverage);
}
```

6. Compute and print the average over *all* the values in `cells`.

```
double sum=0;
for (int row = 0; row < cells.length; row++) {
    for (int col = 0; col < cells[row].length; col++) {
        sum += cells[row][col];
    }
}
// assumes each row has the same length
double average =  sum / (cells.length * cells[0].length);
System.out.println("The average is " + average);
```

7. Suppose you are writing a solitaire card game which begins with 7 piles of cards (piles 0 to 6). The i^{th} pile contains i+1 cards. Create a 2D array to represent these cards. Set the value of each card randomly. For example, the output might look like:

```
Cards:
pile 0:  4♣
pile 1:  8♣ 8♠
pile 2:  A♦ 5♠ A♦
pile 3:  9♣ A♦ 9♥ 7♦
pile 4:  A♦ 5♣ Q♦ J♦ K♦
pile 5:  4♣ Q♣ 6♥ 4♥ 9♠ 5♦
pile 6:  K♣ 4♥ 6♦ A♥ 5♦ A♣ Q♠
```

This problems differs from the earlier ones in that we are creating an array of objects, not primitive types. The 2D array is also not rectangular in shape, but rather is triangular.

```
Card[][] cards = new Card[7][];
for (int row = 0; row < 7; row++) {
    System.out.print("pile " + row + ": ");
    cards[row] = new Card[row+1];
    for (int col = 0; col <= row; col++) {
        cards[row][col] = new Card();
        System.out.print(" " + cards[row][col]);
    }
    System.out.println("");
}
```

ArrayLists

1. Declare and create an `ArrayList` of `Strings` called `animals`.
    a. Add animal names to the list (e.g. ant, aardvark, cat, crow, snake, dog, zebra, cheetah, coyote, duck, dingo, deer).
    b. Use a loop to print out the resulting elements in the `animals`.
    c. Use a loop to remove all animals whose names begin with `'a'` or `'c'`.

```
ArrayList<String> animals = new ArrayList<String>();
animals.add("deer");
animals.add("ant");
animals.add("aardvark");
animals.add("dog");
animals.add("cat");
animals.add("crow");
animals.add("snake");
animals.add("duck");

System.out.println("\nprint out list using for-loop:");
for (int i = 0; i < animals.size(); i++) {
    System.out.print(animals.get(i) + " ");
}
System.out.println("\n\nprint out list using an enhanced for-loop:");
for (String s : animals) {
    System.out.print(s  + " ");
}

// remove animals with names starting with a or c
// (do you see why you need to start from end of loop?)
for (int i = animals.size() - 1; i >= 0; i--) {
    String a = animals.get(i);
    if (a.charAt(0) == 'a' || a.charAt(0) == 'c') {
        animals.remove(i);
    }
}
```