Assignment 3 - Working with Arrays

Problem 1. What does the following code method do?

```
private static void mystery(int[] arr) {
    if (arr.length < 5) {
        throw new IllegalArgumentException("arr must have more than five elements");
    }

    int a = arr[1];
    arr[1] = arr[4];
    arr[4] = a;
}
```

Solution. This code swaps the positions of the 2nd and 5th element of the given integer array

Problem 2. Write a private static method concatReverse() that given a String array, concats the elements into a single string in reverse order. For example, `concatReverse("ab", "12", "cd")` should evaluate to `"cd12ab"`.

Hint: Define an String output variable and initialize it to “”. This is known as the *empty string*, the string with 0 characters. You can concatenate onto it.

Solution. One solution is:

```
private static concatReverse(String[] arr) {
    String output = "";
    for (i = arr.length - 1; i >= 0; i = i - 1) {
        output = output + arr[i];
    }
    return output;
}
```

Alternatively:

```
private static concatReverse(String[] arr) {
    String output = "";
    for (i = 1; i < arr.length; i++) {
        output = output + arr[(arr.length - 1) - i];
    }
    return output;
}
```

Problem 3. A 2D array is an array of arrays. For example, consider the following 2D array:

```
int[][] example = {{1, 2, 3, 4}, {5, 6, 7, 8}};
```

This is an array that contains two arrays, each containing 4 integers.
What do the following evaluate to? Which of these throw exceptions/won’t compile and why?

1. `example.length`
2. `example[0].length`
3. `example[0][3]`
4. `example[1][4]`
5. `example[3][2]`

6. `example[1][1]`
7. `2 < example[0]`
8. `9 < example[1][2]`

Solution.

1. `example.length`: 2
2. `example[0].length`: 4
3. `example[0][3]`: 4
4. `example[1][4]`: throws an ArrayIndexOutOfBounds exception since the second inner array only has 4 items, not 5
5. `example[2][2]`: throws an ArrayIndexOutOfBounds exception since the inner array only has 2 items, not 3
6. `example[1][1]`: 6
7. `2 < example[0]`: won't compile since example[0] is an array, not a number
8. `9 < example[1][2]`: false

Problem 4. Write a private static method sum2DRows() that given a $n \times m$ int array (i.e. int[n][m]) returns an array of length n, whose elements are the sums of the rows of the input. Using `example` defined in the problem above, `sum2DRows(example)` should return the array {`10, 26`} since $1+2+3+4=10$ and $5+6+7+8=26$. You will need to use two nested for loops, one to iterate over the rows, and one to iterate over the columns. To make your code easier to read, use the temporary variables `row` and `column` rather than `i` and `j`

Hint: Start with a simpler problem: how do you sum all the elements of a 1D integer array?

Solution.

```
public static int[] sum2DRows(int[][] arr) {
    int rows = arr.length;
    int columns = arr[0].length;

    int[] output = int[rows];

    for (int row = 0; row < rows; row++) {
        int sum = 0;

        for (int column = 0; column < columns; column++) {
            sum = sum + arr[row][column];
        }

        arr[row] = 0
    }
}
```

Problem 5. In one to two sentences, explain what you would change to your method above in order to write sum2DDown(), the method that sums the columns. For example, `sum2DRows(example)` should return the array {`6, 8, 10, 12`} since $1+5=6$, $2+6=8$, etc.

Solution. We would simply switch the rows and columns in our code above. Our output would have `columns` elements. Our outer loop would iterate over the columns and our inner loop would iterate over the rows.