Part 1

The first step in the question is just to print out the welcome to the first part, which is done with a system.out.println. same with asking for user input for the number of asterisks.

Next we are pulling in a scanner called stars that will look for the user input

We are then assigning the number that is inputted into a variable called numOfStars.

Now, we need to run a loop to get the system to print out multiple rows of stars, decreasing by one each time. We know that the amount of loops we need to run is equal to the amount of stars, or numOfStars. We will keep running the loop as long as numOfStars is equal to or greater than 0 as we can’t have negative stars. We will decrease the number of stars, or I, by 1 each time.

We then run another loop inside it called j. j starts at 0, must be less than I, and is increased each time. J prints the \* character. Once it has fully iterated through j for an I, it will systemoutprintln so that it drops to the next level.

When we run the code, we can enter 10 for example

Part 2

The first thing we will do is identify a one dimensional array called stMarks with a length of 10 elements.

System out printline is giving us the length of my one d array

Then using sout to print both the entry and the user input line

Introducing a new scanner called Marks

Now we need a loop to ask us to input the mark 10 times. We want to run the loop starting at 0, 10 times, so I is less than or equal to 9 due to starting at index 0, and I increases each iteration. We want the for loop to print out the prompt every time, increasing the students number with i+1, then we want it to accept the double entered and store it in stMarks.

I used the commented out portion here to confirm that they were stored in stMarks.

Next we print the line explaining that it’s the final report

In order to get the total, we have to start it 0 and then loop through the array to add the values. Using integer j, but with the same restrictions that j starts at 0, iterates 10 times, and increments by 1 each iteration. This loop is adding each element to the total. We then use sout to give us the total.

We can then use system out to give us the average by including (total/length of stmarks)

To find the max, we are creating a new double called myMax assigned to the first element in my array so I have something to check it against. Then I run my for loop, starting at an I value of 1, run as long as I is less than the length of my array, and increment by 1 each time. There is an if statement here that if the element we’re on is greater than my assigned value of stMarks[0], then that becomes the new value of myMax. After running through the iterations, it should be saved as the highest number.

To find the mix, we create a new double called myMin also assigned to the first element in my array. Then I run the for loop starting a 1, running as long as I is less than the length, incrementing by 1 each time. If the element we’re on is less than what I’ve assigned to myMin, then it becomes myMin.

Both of these for loops have a sout print line to give the max and min value of the array.

Part 3

The first step was to declare the 2 dimensional array of myArray

Then soutprintline the welcome and listing of the elements

In order to list them in the grid fashion, I need nested for loops. The outer for loop deals with the rows and the inner for loop deals with the columns. For the outer for loop, I is starting at 0, must be less than the length of the array (3) and increments by 1 each time. For the inner loop, j is also starting at 0, but must be less than the length of myArray[i] which is actually 4, and increment by 1 each time. As it does that, it should print a value and tab over without moving to a new line. As it completes a j loop, it should skip to the next line with the soutprint for the next I loop to start.

In order to add the arrays, I need somewhere to add them too, so created a double called total equal to 0. Here I will need a for loop to iterate through the values of myArray[0][0] to myArray[0][3]. Using variable j equal to 0, must remain less than the length of myArray[0], incrementing j each time. Then we add the value of myarray[0][j] to total. Then we have a soutprintline to list the total.

Very similar process for average except we are establishing two doubles, sum and count, both equal to 0. Here we need a for loop within a for loop to count through the 2d array, the I loop taking care of the outer loop, and the j loop on the inside. Each time the loop is ran the counter increases by 1 to signify each element we are working with, and the value of the element is added to the running sum total.

We then introduce another double called average equal to the sum / count and sout print line the value

Lastly, to convert this 2d array to a 1d array we need to establish a 3rd variable that can account for positions such as myArray[0][0], [0][1], etc. first we will create a new array to store the 1d array in which will be made up of 12 integers called MyNewArray and establish the new int k valued at 0 to start. Again, the for loop for I and j to access the two layers of the array are opened. Within the loop, we are sending values of myArray[i][j] into MyNewArray at position k, and then incrementing k to ensure it’s a new position every time.

In order to print out the new array in one line, we want to run a loop starting at 0, that must be less than the length of the array, incrementing each time. This for loop will print out the value and a space and repeat through all of the values.