Java 1D Arrays - Practice Problems

All problems in this set should be completed in a project called Java11_Arrays. Each problem should get its own class and must be named starting with a letter. For example, we could name the second problem in level 1: level 1_2.

Level 1

1. Create an array of strings of your classes. Assign each class to a space in the array individually after the creation of the array (e.g. courses[0] = "CS1"). Then, use a loop to show all of the elements of the array.
2. Write a program that uses an array of strings. The user should input 5 names and then, after all input is finished, see the five names output.
3. Write a program that uses an array of doubles. The user should input 6 test scores and then, after all input is finished, see the six scores output.
4. Create an array of ints. Fill the array with 10 random values between 1 and 100. Then, show the values in the array, on the same line, separated by commas.
5. Create an empty array of ints. Use a loop to ask the user to enter numbers into the array. Input should continue until the user enters 20 numbers or a value of -1. When finished, output the number of non-negative values that they entered.

Level 2

1. Write a program that stores the scores of 10 tests for a student. The starting scores should be 80, 76, 98, 65, 79, 82, 85, 90, 99, 85. Then, use a for each loop to cycle through the list and output the average of the scores.
2. Create an integer array of size 50. Allow the user to enter numbers into the array as long as the numbers are positive and they don't exceed the available storage space of the array. As soon as they enter a negative number, the loop should terminate and the program should output the physical and logical sizes of the array.
3. Create a character array of size 35. Allow the user to enter letters into the array as long as the letters are uppercase and they don't exceed the available storage space of the array. As soon as they enter a lowercase letter, the loop should terminate and the program should output the physical and logical sizes of the array.
4. Create a program that asks the user how many names they want to enter, creates an array of that size, and then takes the names from the user. Finally, output the list after all information has been entered. Use a for each loop.
5. The Java String class has a method called *split* that takes a string parameter. This parameter is called a delimiter and is used to separate items of data. So, if a user enters a set of numbers in one line, separated by commas: *1,3,5,2,4,6* and that data is stored in the string, input, the information can be separated into an array as follows: *int [ ] arr = input.split(",");*. Using this idea, take a list of doubles from a user (of any length) separated by commas, store them in an array, and then find the average of the numbers.
6. Create a program that fills an array with fifteen random numbers between 1 and 100. Use a loop to search the list for the smallest number. Display the list, the smallest number and its location.

Level 3

1. Create a program that fills an array with twenty random numbers between 1 and 30. Ask the user for a number that they want to search for. If the number is found, output its index. If the number is not found, output -1. Also, print the list so that the user can verify your results.
2. Write a program that acts as a personal phonebook. Users can enter names and phone numbers, and determine the size of the arrays. Names are stored in one array and the numbers in another. Allow users to enter all of their information, then search for a name. When the name is found, show the name and number on the screen. If the name is not in the list, display an appropriate message.
3. ~~Create a program with two parallel arrays - one to store x values and the other to store y values. The arrays can be pre-populated. Then, use TurtleGraphics to draw the shape determined by the points generated by each index of the arrays (x,y).~~
4. Create an array of size 26 that stores the alphabet (in order). Then, create a separate array that stores the alphabet in a random order. At the top of the program, ask the user to enter a string. Then ask if they want to encrypt or decrypt that string. If they choose to encrypt, find the location of each letter in the original array and output (in its place) the letter at the same position in the unordered array. Decrypting would involve working in the opposite direction.