LATHROP ENGINEERING

Name:

UNIT 7: ARRAY LISTS

AP Computer Science A Unit Due Date: **January 31, 2020**

Welcome to the seventh unit of *AP Computer Science*! This unit gives us our last method for storing data in a program: the *ArrayList*! Arrays were great tools, but in some ways they weren't always as flexible as we had hoped they would be… this is where ArrayLists will really shine! In the end, the expectation is that you learn the following:

-
- How to use the add(), remove(), size(), and get() methods with ArrayLists
- How to use a for loop to traverse an ArrayList
- The main differences between arrays and ArrayLists, and when we would want to use each

As we move through this unit, you are responsible for making adequate progress through the assignments, and for being done by the Unit Due Date (**January 31, 2020**). You are also responsible for completing each part before moving on to the next. Our unit is broken up into three main parts:

Part 1: **Creating ArrayLists** *(20 pts) Approx. 3 days*		
This unit starts with an introduction to a new way of storing information: the *ArrayList*. ArrayLists will feel very much like Arrays, but will allow more flexibility for how to work with them. We'll have to get used to a few different methods for adding or removing elements, but in general the use of ArrayLists should follow pretty smoothly if we feel okay about arrays.		ArrayList Notes
		Complete 8 ArrayList Tasks
		Check-off from Mr. Benshoof

Part 2: **Methods for ArrayLists** *(30 pts) Approx. 3 days*		
Next, it's important for us to acknowledge the difference between arrays and ArrayLists. In this part of the unit we'll actually only make 4 programs, but we'll make each of them twice. For each program you'll make it work once with standard arrays, and then recreate the program to work with ArrayLists. In doing so, we'll gain appreciation for the benefits of each type of data storage!		Array vs ArrayList Notes
		Complete 4 Array vs ArrayList Comparison Tasks
		Check-off from Mr. Benshoof

Part 3: **ArrayLists in Action!** *(30 pts) Approx. 2 days*		
Finally, we'll get some more robust practice with ArrayLists. In this part of the unit we'll complete another four challenges using only ArrayLists to really get to understand the methods that come along with ArrayList structures. This will be a good chance to make sure that we feel comfortable with ArrayLists and that we're ready to tackle some more complex problems with them in the future!		ArrayList Notes & Program Plans
		Complete 4 ArrayList Tasks
		Write Log 6: *ArrayLists*
		Check-off from Mr. Benshoof

(20 pts) Approx. 3 days

This unit is an overview of all-things ArrayList! *ArrayLists* are in many ways very similar to arrays: they store information in a list, and you can add/remove/work with those elements using various methods. There are a few differences in how ArrayLists work compared to arrays though. For example, you might make an ArrayList of numbers as shown below:

```
ArrayList<Integer> primes = new ArrayList<>();
```

Note that in this instantiation, I didn't get to say exactly what the prime numbers would be. I also couldn't use the primitive type 'int' as the parameter, I had to use the class name Integer. It's also not a type that at the end of the line I have angle brackets <> followed by parenthesis (). There are a few small differences in the setup of ArrayLists that we'll need to get used to, but eventually I think we'll learn to like ArrayLists almost more than arrays!

1. Take a full page of good notes on the presentations *ArrayList Intro*, *Storing Data in ArrayLists*, *ArrayLists and For Loops,* and *Arrays v ArrayLists*. These videos do a good job of introducing the ideas of ArrayLists, pointing out some of the technicalities we need to be aware of.
2. Now, complete the next 8 Java Tasks below that are all about working with ArrayLists!
   a. JAVA TASK 46: Make a program that creates an ArrayList able to hold Strings. Then use the add() method to put 4 strings of your choosing into the ArrayList. Then have the program print that ArrayList of Strings to the screen.
   b. JAVA TASK 47: Make a program that creates an ArrayList able to hold Integers. Then use the add() method to put 8 integers into the ArrayList before printing it to the screen. Then use the add(x,y) method to add 2 more integers into the very middle of the ArrayList and reprint it.
   c. JAVA TASK 48: Make a program that creates an ArrayList capable of holding Double values. Then use the add() method to put 6 doubles into that ArrayList. Print the ArrayList on the screen. Then use the remove() method to remove the *middle two* numbers from the ArrayList before printing it on the screen again.
   d. JAVA TASK 49: Make a program that creates an array of 9 random integers between the values 5 and 9 inclusive. Use a for loop to print the array nicely on the screen.
   e. JAVA TASK 50: Make a program that creates an ArrayList capable of holding Strings. Let the user enter as many Strings as they want through the terminal (use a Scanner), and add each of those Strings to the ArrayList. Each time they add a new String, the entire ArrayList should reprint to the screen (getting longer and longer each time).
   f. JAVA TASK 51: Make a program that creates an ArrayList capable of holding Integers. Use the add() method to fill the ArrayList with 8 integers. Then print the ArrayList to the screen. Now, let the user enter an index value and have the ArrayList remove() the integer at that index location before reprinting the entire ArrayList to the screen.
   g. JAVA TASK 52: Make a program that creates an ArrayList capable of holding Integers. Use the add() method to put 12 different integers into the ArrayList. Then, use a ***for*** loop to traverse the ArrayList and count how many of those integers were even. Print the ArrayList as well as the number of evens and odds to the screen.
   h. JAVA TASK 53: Make a program that creates an ArrayList capable of holding Strings. Use the add() method to put 10 different Strings into the ArrayList. Then, use a ***for*** loop to traverse the ArrayList and count how many of those strings are longer than 3 letters. Print the ArrayList as well as the total count of long Strings to the screen.

Part 1: Tasks	*10-8 points*	*7-4 points*	*3-0 points*
ArrayList Notes	+ Watch all presentations + You took a full page of notes on ArrayLists and their methods + Your notes include details about instantiation, editing, and traversing ArrayLists	- Less than a full page of notes on ArrayLists - Your notes are missing important parts	- Very brief or no notes in your notebook
Java Tasks 46-53	+ You completed all 8 Java Tasks from this section	- You did not complete all 8 tasks	- You did not complete any tasks
Checkoff from Benshoof	+ Mr. Benshoof got to see your array programs run successfully	N/A	N/A

(30 pts) Approx. 3 days

The next part of our unit asks us to make a variety of programs using both arrays and ArrayLists. The goal is to help reinforce the difference between these two data storage structures, and also to give us a little reminder about arrays. Each of the four programs listed below has some challenge to them, but shouldn't be too difficult.

The real challenge with these will be to write each program twice. The first time you write it, you should make it with arrays. The second time you write it you'll make it with ArrayLists. This way you can think through the similarities and differences between the two different tools.

1. Watch the videos on arrays and ArrayLists provided in section 2 of the unit webpage. These give interesting insights into the differences, but shouldn't be giving any real *new* information about methods. They're just trying to remind and emphasize the differences between the two tools. Take a page of notes on these ideas, and add to your notes as you continue to brainstorm and work on the next four challenges:
2. Now, complete the following 4 challenges TWICE. The first time you complete the challenge, do it with arrays ONLY! The second time you complete the challenge, do it with ArrayLists ONLY! You'll find that each of those tools has places where it is easier to work with than other:
    a. JAVA TASK 54: Make a program that stores the following values in a list: {1, 5, 9, 51, 12, -4, 0, 23, 35, 18, 12} Have your program use a 'for' loop to count how many of those numbers are even, and how many are odd. Have your program print the total counts as well as the list in order on the screen.
    b. JAVA TASK 55: Make a program that lets the user enter 8 different Strings. The program should store all 8 Strings in a list. Have your program use a 'for' loop to check every String to see which ones are longer than 5 characters. Have your program print back the list to the user, ***but it should only print the Strings that are longer than 5 characters.***
    c. JAVA TASK 56: Make a program that creates a list of 100 boolean values. The values should alternate true-false-true-false for the entire list. Then print your list to the screen for the user.
    d. JAVA TASK 57: Make a program that creates a list of 10 random integers less than 20 (use Math.random() to generate the numbers). The user should then be able to replace any value in the random list with their own custom value. The list should then be reprinted to the user in its updated form.
3. Take a moment and think about the differences and similarities between arrays and ArrayLists. Take some time to make a large Venn Diagram in your engineering notebook. Fill your Venn Diagram with descriptions of how they are similar and different. This will be useful in the future when we find places where we want to make *some kind* of list and we need to remember ways those tools are most effective.

Part 2: Tasks	*10-7 points*	*6-4 points*	*3-0 points*
Array vs ArrayList Notes	+ Watch the three presentations on Arrays and ArrayLists + Take a page of notes on the differences between these different tools + Add to your notes as you think through this part's Java Tasks	- Less than a full page of array/ArrayList notes - No brainstorming present	- Very brief or no notes in your notebook
	20-10 points	*9-5 points*	*4-0 points*
Java Tasks 54-57	+ You completed all 4 Java Tasks from this section + You completed each task twice: once using ONLYL arrays and once using ONLY ArrayLists	- You did not complete all 4 tasks - You did not complete each task twice	- You did not complete any tasks
Checkoff from Benshoof	+ Mr. Benshoof got to see your Java programs run successfully	N/A	N/A

(30 pts) Approx. 2 days

By now, you've seen the four most important methods in ArrayLists: *.add(), .remove(), .size(),* and *.get()*. Together, these methods let us work with ArrayLists and do all sorts of cool things. In addition, methods like *.set()*, *.indexOf(),* and *.clear()* can give us other smooth ways to work with them. In this part of the unit, we'll get more insights into using ArrayLists and more practice creating ArrayLists of different kinds. Work through each of the programs listed below and see what you can do!

1. JAVA TASK 55: Create a class called Student that keeps track of 3 different student-related variables. Your Student class should have all the proper private variables, constructors, toString() and setters/getters that we always use with classes. THEN, make a runner class that makes an ArrayList full of Student objects. Use the .add() method to add 6 new Students to the ArrayList. Then, print the ArrayList so that it shows the names of all the students in the ArrayList.
2. JAVA TASK 56: Create a class called ZooAnimal that keeps track of 3 different zoo-animal-related variables. Your ZooAnimal class should have all the proper private variables, constructors, toString(), and setters/getters that we always use with classes. THEN, make a runner class that makes an ArrayList full of ZooAnimal objects. You should use let the user provide information to create new ZooAnimals in the program to create new ZooAnimals. The program should then print the ArrayList of ZooAnimals back to the user with the updated values.
3. JAVA TASK 57: Create an ArrayList of Integers that lets the user enter their own integers. The ArrayList should *sort* the values as they are plugged into the ArrayList so that they are always in increasing order. The program should then print the new ArrayList back to the user each time they enter a new number.
4. JAVA TASK 58: Create an ArrayList for Strings that lets the user enter their own Strings into it. The program should print the ArrayList back to the user as they add new Strings. The user should be able to change any String in the ArrayList to a new one, and the updated ArrayList should then be reprinted to the user.

5. *Log 6: ArrayLists* - The last part of our unit is to write a full-page response about ArrayLists. Write a page in which you share the positives and negatives of AraryLists and their use in programming. What aspects of ArrayLists do you want to always remember? What parts are trivial and not really a big deal? How might ArrayLists serve as a cool tool in future programs?

Part 3: Tasks	*5 points*	*4-3 points*	*2-1-0 points*
ArrayList Program Notes & Brainstorm	+ You wrote a full page of brainstorms, ideas, and notes about how to structure your programs	- You wrote less than a page - Your notes do not outline a coherent plan	- Your notes are lacking or missing - There is no plan for your program
	15-10 points	*9-5 points*	*4-0 points*
Complete Java Tasks 55-58	+ You completed all 4 Java Tasks from this section	- You did not complete all 4 tasks	- You did not complete any tasks
	10-8 points	*7-4 points*	*3-0 points*
Log 6: *ArrayLists*	+ You wrote a complete page in your engineering notebook	- You wrote less than a full page	- You wrote less than half a page