CS 110 Lab Sheet 6

Lab cycle starting 6th November 2019, Normal Deadline, two weeks from the start of this lab cycle.

Stage 1: Simple Arrays

The following program creates an array of 10 integers and stores the squares of the numbers 0..9 in it. Extend it so that it prints out the array using a second loop - that is, don't put the 'println' statement inside the existing loop but add another one after it.

```
public class Squares {
     public static void main(String[] args) {

          int[] squares = new int[10];

          for (int i = 0; i < squares.length; i++) {
               squares[i] = i*i;
          }
     }
}
```

Stage 2: Fibonacci Numbers

Fibonacci numbers are defined as follows: The first two are both 1; and the ones after that are the sum of the previous two. That is, the sequence is 1, 1, 2 (because 1+1=2), 3 (because 1+2=3), 5 (2+3=5), 8, 13, 21 etc.

Write a program that uses a loop to put the first 10 Fibonacci numbers in an array. Hint: if you use a for loop (and you should!) with a loop counter called i, then then line:

```
fib[i] = fib[i-1] + fib[i-2];
```

will calculate the next Fibonacci number in the sequence. The tricky bit is to make sure the loop goes around the right number of times. You are allowed to set the first two Fibonacci numbers in the array to be 1 before the loop starts. That is, you can write:

```
fib[0] = 1;

fib[1] = 1;
```

Your program should then print out the Fibonacci numbers in the array.

Stage 3: Finding the Largest and Smallest in an Array

Suppose you are given the array:

```
int[] numbers = {1, 7, -3, 14, 19, 0, 2, -8, 6, 11, 3};
```

write a program that searches the array and prints out the smallest and largest numbers stored in it. Your program should not depend on the exact length of the array. That is, you should be able to add more numbers to the array (or remove numbers) and your program should still work without any other changes. Hint: you will need to use `numbers.length` to do this. HINT: if you start off with the program in Stage 1 you basically just have to replace the body of the for loop.

Stage 4: ArrayLists

Recreate the example in Stage 1 using an ArrayList - you will need to use the line:

```
import java.util.ArrayList;
```

to access the ArrayList library.

Challenge Task

Suppose you start out with an array like this:

```
int[] gaps = {1,2,4,5,8};
```

write a program that (a) converts the array into an `ArrayList` and then (b) inserts all the 'missing' numbers into the right place - that is, in this case, your program should insert 3, 6 and 7. Finally, you should print out the updated `ArrayList` to show that it has no 'gaps'. Your program should work for any array of integers - and not just the one in the example above.

Assessed Task: Person Name and Age.

Write a program that defines two arrays - one of strings and one of integers, both of size 10. Your program should then ask the user to enter the a string representing a persons name, and an integer representing their age. It should continue to do this until either the user enters 'done' instead of a name, or until the array is full (that is, 10 pairs of names and ages have been entered). It should then print out the names and ages as well as the names of the youngest and oldest.

Hints: One tricky (deliberately) part is making sure that once you've typed 'done' to finish entering names, your program does not then ask you for the age of the person with name 'done' - be careful about this. This is one of the few cases where, if you are careful, you can sensibly use a 'break' statement outside of a switch statement.

Challenge Task

Write a program that uses an `ArrayList` to sort numbers into order as they are entered by the user. Your program should read numbers from the keyboard and insert them into your ArrayList in the correct position so that when the `ArrayList` is printed the entered numbers are in order. The user should enter the number -99 to indicate that they have finished entering data.