Converting A String Into An Array Of Characters

The **toCharArray()** method turns a String into an array of characters. For example:

```
String str = "ABCDE";
char[ ] list = str.toCharArray();
```

results in an array of characters containing the values {'A', 'B', 'C', 'D', 'E'}.

This method can be combined with other methods (as long as the other methods return a String). For example, to read an array of characters from the keyboard:

```
char[ ] list;
Scanner reader = new Scanner(System.in);
list = reader.nextLine().toCharArray();
```

LAB 10.5 - ASSIGNMENT

Lab 10_5A - 60 points

OBJECTIVE

The Fibonacci numbers are the sequence of numbers defined by the linear recurrence equation

$$F_n = F_{n-1} + F_{n-2}$$

WAP that fills an array integers with the first 90 numbers in the Fibonacci sequence. Then display the numbers on the console screen.

FIELD SUMMARY

- **long[ ] fibonacci** – an array of integers.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to input, and output.
- **process** – place a zero in the first element of the array and a 1 in the second element. Then process the rest of the array filling in values using the formula shown above.
- **output** – The first 90 numbers in the Fibonacci sequence.

SAMPLE OUTPUT

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368

Lab 10_5B - 70 points

OBJECTIVE

WAP that reads a series of numbers from a data file (**"numbers.dat"**) and stores them in an array of integers. Output the *smallest* and *largest* value. Then output the *mean* (average) and the *median* values. The *median* can be found by arranging all the numbers from lowest value to highest value and picking the middle value if there are an odd number of values.

3	5	**8**	13	22

median = **8**

If there is an *even* number of values, then the median is defined to be the mean of the two middle values.

3	5	**8**	**13**	22	36

median = **(8 + 13) / 2**

HINT: You need a counting (for) loop in ***process*** to calculate the *mean.*

FIELD SUMMARY

- **int[ ] list** – an array of integers (make it 500)
- **int count** – an integer to keep track of the number of array elements used.
- **int mean** – the average of all the numbers in the array.
- **int median** – the median value of all the numbers in the array.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input`, `process`, and `output`.
- **input** – declare a `Scanner` object and read eight integers from a data file storing them in ***list***.
- **process** – calculate the mean and median values.
- **output** – display the smallest and largest value. Then display the mean and the median values. Format all doubles to display one decimal place with no trailing zeros.

SAMPLE DATA FILE INPUT (numbers1.dat)

NOT SHOWN - (An even number of elements - Open the file if you want to view it)

SAMPLE OUTPUT

The smallest value is 2
The largest value is 951
The mean value is 484.6
The median value is 506.5

SAMPLE DATA FILE INPUT (numbers2.dat)
NOT SHOWN - (An odd number of elements - Open the file if you want to view it)
SAMPLE OUTPUT
The smallest value is 2 **The largest value is 951** **The mean value is 484.3** **The median value is 459**

Lab 10_5C - 80 points

OBJECTIVE

WAP that reads the names of various animals from a data file (**"animals.dat"**) into an array of type String. Then prompt the user to enter a letter from A-Z. Sort the array and output all of the names that begin with the input character. Then output the number of animals whose names begin with the input character.

HINT: Read the character (A-Z) in as a String not as a char. The String class has a method **`startsWith`** that receives a String as an argument and returns *true* or *false* depending on whether the string calling the method starts with the String received as an argument.

FIELD SUMMARY

- **String[ ] list** – an array containing the names of animals.
- **int count** – an integer to keep track of the number of array elements used.
- **String letter** – a letter to be read from the keyboard.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input` and `output`.
- **fileInput** – declare a `Scanner` object and read a series of animal names from a data file storing them in **`list`**.
- **kybdInput** – prompt the user to enter a character from a to z (it should work with uppercase or lowercase letters). Read the character entered storing it in **`letter`**.
- **output** – Sort the array and display all the names of all the animals that starts with **`letter`**.

SAMPLE DATA FILE INPUT

EAGLE
TOUCAN
FERRET
….
GUPPY
HIPPOPOTAMUS
NIGHTHAWK

SAMPLE KEYBOARD INPUT

Enter a letter (A-Z): **L**

SAMPLE OUTPUT

```
LADYBUG
LAMB
LAMPREY
LARK
LEMMING
LEOPARD
LION
LOBSTER
LOON
LYNX

The names of 10 animals in the list begin with the letter L.
```

Lab 10_5D - 90 points

OBJECTIVE

WAP that reads a series of words from a data file <**"prefix.dat"**> and store them in an array of type String (prefix). Read a second series of words from a second data file <**"suffix.dat"**> storing them in a parallel array (suffix). Combine each of the prefixes and suffixes and save the combined words in a third array. Sort and output all the combined words.

FIELD SUMMARY

- **String[ ] prefix** – an array of words (the prefix of the combined words).
- **String[ ] suffix** – an array of words (the suffix of the combined words).
- **String[ ] words** – the combined words (prefix + suffix).
- **int count** – an integer to keep track of the number of array elements used.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input`, `process`, and `output`.
- **input** – declare a `Scanner` object and read all the words stored in **"prefix.dat"** into the array prefix. Then read all the words stored in "suffix.dat" and store them in the array suffix.
- **process** – combine all the words in ***prefix*** with all the words found in ***suffix***. Store the combined words in ***words***.
- **output** – Sort and display all the combined words.

SAMPLE DATA FILE INPUT

<"prefix.dat">

COW
APPLE
RIO
. . . . ..
HARRY
MOUNTAIN
BIG

<"suffix.dat">

BOY
PIE
GRANDE
. . . . .
HOUDINI
LION
FOOT

SAMPLE OUTPUT

```
APPLE PIE
BIG FOOT
COW BOY
CROSSWORD PUZZLE
DESK TOP
DOUBLE CROSS
GEORGE WASHINGTON
GOLDEN GATE
HARRY HOUDINI
HOLLY WOOD
HONG KONG
HOT DOG
JUNGLE JIM
LINCOLN TUNNEL
MOUNTAIN LION
NEW ORLEANS
NEW YORK
PIE CRUST
PIG PEN
RIO GRANDE
SAN FRANCISCO
SINGLE GIRL
SUPER MAN
TUNNEL VISION
WHIPPED CREAM
WOODEN DECK
ZIEGFELD GIRLS
```

Lab 10_5E - 100 points

OBJECTIVE

WAP that reads a series of characters from the keyboard and stores them in an array of type char. Sort and output the array.

HINT: Read the characters in as a string and then convert the string into an array of characters**.**

FIELD SUMMARY

- **char[ ] list** – an array of characters.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input`, `process`, and `output`.
- **input** – declare a `Scanner` object and read a series of characters from the keyboard.
- **output** – Sort and display the contents of the array.

SAMPLE KEYBOARD INPUT

Enter a word: **canasta**

SAMPLE OUTPUT

aaacnst