**HW03 - Challenge**

* [Description](https://labs.vocareum.com/web/1587618/507500.0/ASNLIB/public/docs/README.html#1)
* [Helpful Information](https://labs.vocareum.com/web/1587618/507500.0/ASNLIB/public/docs/README.html#5)
* [Instructions](https://labs.vocareum.com/web/1587618/507500.0/ASNLIB/public/docs/README.html#2)
* [Testing](https://labs.vocareum.com/web/1587618/507500.0/ASNLIB/public/docs/README.html#3)
* [Submit](https://labs.vocareum.com/web/1587618/507500.0/ASNLIB/public/docs/README.html#4)

**Description**

For this Challenge, you will be creating a class AccountGenerator.java that accepts input from a user and uses that data to generate a unique username and password. Both will then be printed for the user to review.

Note: This is not a secure way to generate usernames or passwords. We don't recommend using anything you create for this assignment for security purposes. You'll learn more about why it is a bad idea in future courses!

You may use the information we covered in the walkthrough and during lecture. We've also included a section in this document called "Helpful Information" that may be useful. However, if you find that you need additional methods to complete this challenge, feel free to research implementation strategies and use the [Java documentation](https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/lang/String.html) when creating your solution.

Note: 5 points of your Challenge grade is based on Coding Style.  Be sure to follow the standards described on Brightspace.  Use the "Run" button to check your Coding Style without using a submission.

**Helpful Information**

**ASCII Table\***

ASCII is a code for representing English characters and other symbols as numbers. Each character is assigned a number ranging from 0 to 127. For example, the ASCII code for uppercase P is 80.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 32: |SPACE| | 48: 0 | 64: @ | 80: P | 96: ` | 112: p |
| 33:  ! | 49: 1 | 65: A | 81: Q | 97: a | 113: q |
| 34:  “ | 50: 2 | 66: B | 82: R | 98: b | 114: r |
| 35: # | 51: 3 | 67: C | 83: S | 99: c | 115: s |
| 36: $ | 52: 4 | 68: D | 84: T | 100: d | 116: t |
| 37: % | 53: 5 | 69: E | 85: U | 101: e | 117: u |
| 38: & | 54: 6 | 70: F | 86: V | 102: f | 118: v |
| 39: ‘ | 55: 7 | 71: G | 87: W | 103: g | 119: w |
| 40: ( | 56: 8 | 72: H | 88: X | 104: h | 120: x |
| 41: ) | 57: 9 | 73: I | 89: Y | 105: i | 121: y |
| 42: \* | 58: : | 74: J | 90: Z | 106: j | 122: z |
| 43: + | 59: ; | 75: K | 91: [ | 107: k | 123: { |
| 44: , | 60: < | 76: L | 92:  \ | 108: l | 124: | |
| 45:  - | 61:  = | 77: M | 93: ] | 109: m | 125: } |
| 46: . | 62: > | 78: N | 94: ^ | 110: n | 126: ~ |
| 47: / | 63: ? | 79: O | 95: \_ | 111: o | 127: |DEL| |

In the table, the numbers / decimals on the left are the ASCII codes and the symbols on the right are the characters associated with them.

\* The first 32 entries in the table (0 - 31) are control codes. For this assignment, you do not need to worry about them.

**Type Casting**

Syntax: dataType variableName = (dataType) variableToConvert;

**Examples**

**double to int**

double calculatedMark = 87.6;

int finalGrade = (int)calculatedMark; // finalGrade is 87

**char to ASCII value**

char ch = 'P';

int asciiValue = (int)ch; // asciiValue is 80

Note: An explicit cast is not required in this case since casting from char to int is an Upcast. For more information, refer to the Primitive Types and Strings [lecture slides](https://purdue.brightspace.com/d2l/le/content/307113/viewContent/6489064/View).

**ASCII value to char**

int number = 97;

char ch = (char) number; // ch is 'a'

**String to int**

There are two ways to accomplish converting a String to an int

String s = "77";

// Primitive int is returned

int str = Integer.parseInt(s);

or

String s = "77";

// Integer object is returned

int str1 = Integer.valueOf(s);

**int to String**

There are a few different ways to accomplish converting an int to String, two popular methods are shown below

int number = 1234;

String str = String.valueOf(number);

or

int number = 1234;

String str = Integer.toString(number);

**Instructions**

**Accepting Input**

Begin by using a Scanner to read user input into the proper variables. Remember to prompt the user to enter each input.

The proper variable types are listed in the table below. The **Notes** section of the table describes the format of how the information will be entered by the user.

|  |  |  |
| --- | --- | --- |
| **Value** | **Type** | **Notes** |
| First Name | String | <first name> |
| Last Name | String | <last name> |
| Age | double | <any int value> |
| Birthday | String | <MM><space><DD><space> <YYYY> |

Note: You may assume that all input will be valid.

**Processing Input**

To generate the username, you'll process and combine a few of the inputs.

1. Retrieve the first letter from the first name.
   * For example, for the name "Kelly", retrieve "K".
2. Retrieve the entire last name.
   * For example, for the name "Thomas", retrieve "Thomas".
3. Retrieve the birth year.
   * For example, if the user enters "01 01 1900" for the birthday, retrieve "1900"
4. To generate the full username, combine each of the previously calculated values in order. For examples, see the next section.

To generate the password, you'll be processing each of the inputs in new ways.

1. Retrieve the first half of the first name. If the number of characters in the name is odd, round down.
   * For example, for the name "Logan", retrieve "Lo".
2. Retrieve the second half of the last name. If the number of characters in the name is odd, round up.
   * For example, for the name "Daniels", retrieve "iels".
3. Convert the age into a capital letter using the corresponding ASCII code. Reset the count when you run out of letters.
   * For example, for an age of 0, retrieve "A".  For an age of 25, retrieve "Z". For an age of 26, reset the count and retrieve "A". For an age of 29, retrieve "D".
4. Sum the month, day, and year entered by the user. Retrieve the remainder of the sum when it is divided by 12.
   * For example, for a birthday of 01 01 2000, the result is 10.
5. To generate the full password, combine each of the previously calculated values in order. For examples, see the next section.

**Print Output**

Be sure to use each of the prompts provided in the starter code.

**Testing**

Use the example outputs below to test your solution. Your output should match the examples exactly.

**Output Test 1**

What's your first name?  
[Purdue]  
What's your last name?  
[Pete]  
How old are you?  
[21]  
What is your birthday? Enter the month, the day and then the year (MM DD YYYY).  
[01 01 2000]  
The generated username is: PPete2000  
The generated password is: PurteV10

**Output Test 2**

What's your first name?  
[Mitch]  
What's your last name?  
[Daniels]  
How old are you?  
[71]  
What is your birthday? Enter the month, the day and then the year (MM DD YYYY).  
[04 07 1949]  
The generated username is: MDaniels1949  
The generated password is: MiielsT4

Note: Brackets [] indicate input.

Note:  Match this output exactly. Use the prompts provided in the starter code.

**Submit**

After testing your solution and verifying that it meets the requirements described in this document, you can submit on Vocareum.  You have 10 submission attempts to achieve full points.