# Object-Oriented Programming: Homework #3

## Time

- Assigned on 3/27/2023 (Monday); Due at 09:00 on 4/10/2023 (Monday)

## Submission

- Source Code Submit on iLearning.
- Your program code for this homework should be submitted as a single .java file named HW3_StudentID.java (**DO NOT** submit the entire Eclipse project, only the source files).
- In addition, please delete the package before uploading the file.

## Objectives

- Work with arrays.
- Iterate through arrays.
- Search arrays.
- Compare arrays.
- Validate user input.
- Familiarity with basic implementation of class and object.

## Program Descriptions

- Bulls and Cows is a guessing game where you try to guess a *n*-digit code. (Note: Each number in the code must be distinct.) When a number is guessed and a digit is in the correct location, it is a bull. When a digit is in the number, but in the incorrect location, then it is a cow. So, if a four-digit code is 2894, and the user guesses 4698. Then that is "1 bull and 2 cows" because '9' is in the correct location and '4' and '8' are in the code but are currently in the wrong location. The guessing continues until the code is guessed in the correct order. That is, "4 bulls!" in the case of four-digit code.
  The following website will give you a hint for the game.
  http://www.mathsisfun.com/games/bulls-and-cows.html
  To emulate our version, select the option on the website that creates codes from 0 to 9.

## Program Specifications

- Design an algorithm to extract individual digits from a single integer and put each digit in a list (e.g., array).
- Design an algorithm to create a random code and place the code in a list (e.g. array).
- Identify test cases, especially ones not covered in the sample run but indicated in the description. That is, create a test case for each of the following:
    1. How do you detect repeated digits in a user's input?
    2. How do you detect that a code is too long?
    3. How do you compare values in two arrays?
    4. How do you distinguish a bull from a cow?
- Program flow:
    1. Get the number of digits for the code.
        - Only accept inputs of 0, 3, 4, or 5. (Note: 0 is a cheat code)
        - Repeat until the input is valid.
    2. Load array with code digits.
        - If 0 is input as the number of digits for the code:
          (Note: This will allow TAs to enter a code to make testing easier. Since this is for testing purposes, you do not have to check these for correct input.)
            (1) Get code as a single integer (int).
            (2) Get the number of digits in the code. (e.g. 3, 4 or 5)
                - You need this so you can add a leading zero if needed, e.g., if a user typed in 29 for a 3-digit code, then you will need to create 029.
                - If you create an appropriate algorithm, the leading zeros should come automatically.
            (3) Extract individual digits from the single integer and put into the array for the code. Consider using **integer division** and **modulus**.
        - Else randomly create a valid code. (i.e. the number of digits was 3, 4 or 5)
            (1) Load array with the **required number** of random digits ensuring there are **no duplicates**.
    3. Regardless of the method used to create the code (entered from the keyboard or randomly generated), **output the code to be guessed**. (This is purely to help TA verify the correctness of your code. Of course, in a real published game, you would not output the code.)
    4. Get guess as a single integer (int).
        - Extract single digits from the single integer and put in an array for the guess.
            (1) If not enough digits, assume the missing digits to the left are zero.
                - For example, a 3-digit code with input code 35 is the same as 0-3-5 where zero is the first digit. For a 4-digit code, 35 is the same as 0-0-3-5.
            (2) Ensure the guess is a valid guess!
                - If it is invalid, report why the guess is not good and get it again.

- Invalid cases:
    1. The input has more digits than the number of digits in the code
    2. The input has repeated digits.
- If both problems occur, only report that the code has too many digits.

5. Determine the status of the guess, i.e., bulls and cows.
    - Count the number of bulls and cows by comparing the array for the guess with the array for the code.
        (1) Report the bulls and cows on separate lines
        $x$ **`bulls`**
        $y$ **`cows`**
        where $x$ is the number of bulls you counted and $y$ is the number of cows. Note that even if you have only one bull or cow, you still output the plural word bulls or cows.
    - When the guess matches the code (i.e. all bulls), output the result as shown in the sample run. Be sure to include a leading zero when outputting the correct guess if it has one. E.g. 0-2-3 and not 2-3.

## Requirements

- The code and guess must each be stored in an array of integers.
- You may not use vectors.
- Follow formatting as demonstrated in the "Sample Runs" below.
- You MAY NOT use string comparisons.
- Even though we only have required the program to work with codes of 3, 4 or 5 digits, your program should be easy to modify and scale to 2 - 9 digits.
- Do not use functions to find something that you could do by iterating through the array yourself. For example, DO NOT use something like `.find()` instead of practicing iterating through a compound data structure (i.e., array) element by element and determining if there is a match during the iteration.
- Define a class `BullsAndCows`, and implement all required functions under this class. A sample template for the class is shown as follows:

```
class BullsAndCows
{
    // Data Members
    int code;
    ...

    // Constructor
    public BullsAndCows(int input)
    {
        code = input;
        ...
    }

    // Function Members
    int generate_code(int num)
    {
        ...
        return ...;
    }

    ...
}
```

  Note that the above template code is just an example for your reference. You can do your own naming conventions except for the class name that must be named `BullsAndCows`.

## Example Program Run

- The input is expected to be as in the **green text** below, and the output of the results should precisely be as in the **red text** below. This is what things will look like when you are debugging and run your code in Eclipse.
- Note: This sample run is not complete testing, so it does not cover all test cases. You should test other cases to ensure your code works in different options.
  - Test Case #1:

```
Enter number of digits in code (3, 4 or 5): 0
Enter code: 2894
Enter number of digits in code: 4
Number to guess: 2-8-9-4↵
Enter Guess: 5555
Each number must be different.↵
Enter Guess: 59
Each number must be different.↵
Enter Guess: 12345
You can only enter 4 digits.↵
Enter Guess: 112233
You can only enter 4 digits.↵
Enter Guess: 4698
1 bulls↵
2 cows↵
Enter Guess: 9687
0 bulls↵
2 cows↵
Enter Guess: 2894
4 bulls ... 2-8-9-4 is correct!↵
```

  - Test Case #2:

```
Enter number of digits in code (3, 4 or 5): 0
Enter code: 29
Enter number of digits in code: 3
Number to guess: 0-2-9↵
Enter Guess: 89
2 bulls↵
0 cows↵
Enter Guess: 29
3 bulls ... 0-2-9 is correct!↵
```

- Test Case #3:

```
Enter number of digits in code (3, 4 or 5): 5
Number to guess: 4-6-9-7-3↲
Enter Guess: 49763
2 bulls↲
3 cows↲
Enter Guess: 46973
5 bulls ... 4-6-9-7-3 is correct!
```