GMU Spring 2023 – CS 211 – Exercise 1

Due Date: Thursday, February 2nd, 11:59pm

Changelog

- Nothing yet!

Description

The purpose of this assignment is to get practice with basic statements, expressions, boolean logic and branching. You will not use loops, helper functions or other "advanced" features. Think of this assignment as a refresher of your programming skills and a smooth transition to Java.

Instructions

- Honor Code: This assignment is individual work of each student, you're not allowed to collaborate in any form. Copying code from other sources (peers, websites, etc.) is a serious violation of the University's Honor Code. Your code will be examined for similarities with other sources.
- Documentation: Comments are not required in this assignment but it's a good practice to add comments to any piece of code that is not obvious what it does.
- Imports: You may not import any classes or packages
- You are not allowed to use loops, arrays, and any language construct that hasn't been covered in class yet

Submission

Submission instructions are as follows:

1. Name your file **`E1.java`** and upload it to Gradescope using the following link

   https://www.gradescope.com/courses/498183/assignments/2586369
2. Download the file you just uploaded to Gradescope and compile/run it to verify that the upload was correct
3. Make a backup of your file on OneDrive using your GMU account

If you skip steps 2 and 3 and your submission is missing the proper file(s), there won't be a way to verify your work and you will get zero points.

Grading

- Grading will be primarily automated.
- Manual grading will be used only for checking hard-coding, violations, comments, etc.
- When you upload your code to Gradescope, the built-in script will run some basic checks for validation purposes only. This is not the actual grading. The actual grading will take place afterwards and you can't see the tests that will be used.
- If your code doesn't execute successfully with the Gradescope's built-in script, it means that something fundamental is wrong and we won't be able to grade it unless you fix it. You will get zero points if you don't fix and reupload your code.

Testing

We won't be using testers in CS211. The goal is to help you put more focus on writing logically correct programs instead of trying to pass certain tests only. As a programmer, you must learn how to test your code on your own and you must also develop the ability to discover the corner cases that can potentially break the logical correctness or crash your code completely.

Task

You will implement a Java program that encodes the Three Card Poker rules and, given three cards, it can tell what rank they are. Three Card Poker is similar to the traditional poker but it's played with 3 instead of 5 cards, which makes it simpler as it has fewer hands to consider. All you need to know is that the game has the following six hands in descending order. No other knowledge is assumed or needed to do this assignment.

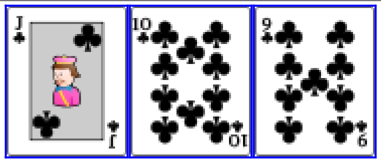
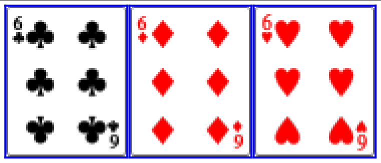
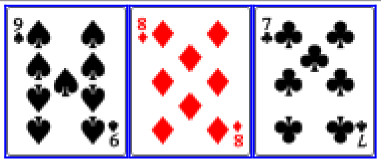
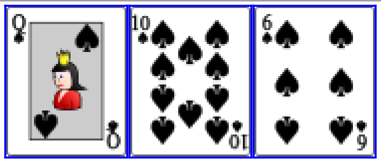
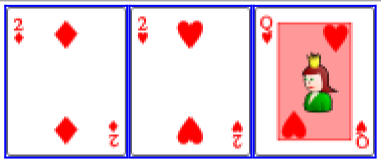
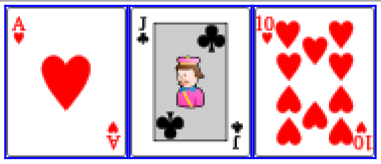
Rank	Description	Example	Output
Straight flush	Three suited cards in sequence		10
Three of a kind	Three cards of same rank		8
Straight	Three cards in sequence		7
Flush	Three suited cards		5
Pair	Two cards of same rank		3
High card	None of the above		1

Table 1

Since we haven't discussed the complete topic of Input/Output yet, and to avoid any errors due to misspelling and capitalization, your program will not read or print anything. All your code must be placed inside a method named **`calculate`** that accepts three parameters, one for each card, and returns a single number. A complete template for your code is provided at the end of this document.

The number returned from **`calculate`** is one of the values provided in the column "output" of Table 1. Think of it as the *score* of a certain hand.

The three input parameters of **`calculate`** represent the three cards of the hand. Each card is encoded with a positive integer. We will use the Table 2 mapping to represent the standard 52-card deck with integers.

Invalid inputs: If `calculate` is provided with syntactically correct but logically incorrect values (e.g. -1, 55, 1000, etc.), the method must return the value of zero.

Assumptions:

- The three parameters are guaranteed to be distinct. You don't have to validate this.
- If a hand satisfies more than one rank (e.g. a Straight flush is also a Flush), we select the higher one (e.g. Straight Flush).
- To simplify things, the Ace will always rank low (i.e. 1). This means that a hand like "Queen, King, Ace" is not a straight.

	Ace	2	3	4	5	6	7	8	9	10	Jack	Queen	King
Clubs	1	2	3	4	5	6	7	8	9	10	11	12	13
Diamonds	14	15	16	17	18	19	20	21	22	23	24	25	26
Hearts	27	28	29	30	31	32	33	34	35	36	37	38	39
Spades	40	41	42	43	44	45	46	47	48	49	50	51	52

Table 2

Provided code

```
// save this class in a file named E1.java
class E1
{
   // The main method will not be graded
   // You must use it for testing your code
   public static void main(String[] args)
   {
      ////////////////// testing approach 1 //////////////////////
      // use the following code and run your program             //
      // by providing the three numbers directly in the terminal //
      // e.g.    C:\>  java E1 34 2 50                            //
      int a = Integer.parseInt(args[0]);                         //
      int b = Integer.parseInt(args[1]);                         //
      int c = Integer.parseInt(args[2]);                         //
      System.out.println(calculate(a,b,c));                      //
      /////////////////////////////////////////////////////////////

      ////////////////// testing approach 2 //////////////////////
      // call the calculate method with any arguments you want   //
      // and print the returned value                            //
      System.out.println(calculate(34, 2, 50));                  //
      /////////////////////////////////////////////////////////////
   }

   public static int calculate(int card1, int card2, int card3)
   {
      // You must put all your code here
      // this method must return an integer, for example:
      // return 0;

   }
}
```