Be advised this is *not* an easy project that can be finished in an hour. Estimating a good, B-level student will spend an avg of 5-10 hrs. getting this project to work correctly, so do not wait until TUE to begin!

**CIS 200: Project 4 (50 pts + 10% Extra Credit)**

~~**Due Wed, Sep 20th by 11:59pm**~~

**Due date extended to Fri, Sep 22th by 11:59pm, but NOT accepted late.**

~~Programs submitted after the due date/time will be penalized 10% for each day the project is late. Projects are not accepted after 2 days after the due date – *no exceptions.*~~

As stated on the front page of the syllabus, ALL work submitted in this course is to be done BY YOU. Submitting *another's work as your own* is a violation of the K-State Honor Code. If you are unsure if you are crossing the line of *unauthorized collaboration* or *unauthorized assistance*, always ask the Professor.

If you use code that YOU did not develop (i.e., from another student, from the Internet source, etc.) *you must cite your source* in a comment above the line(s) of code that it is not your own work, otherwise it will be considered *plagiarism* (*representing others' work, whether copyrighted or not, as one's own*).

---

*As with previous Projects, it is best if you do some 'pre-planning' on this project BEFORE you start coding*.

Packages and *programmer defined objects* (i.e., *Card* class) can NOT be used. Additional methods *are* allowed but global variables declared outside of any method are NOT allowed. You must properly pass values between methods.

**Multiple Arrays must be properly declared / utilized to get credit on this Project. No 2D arrays or ArrayLists.**

**Assignment Description:** In this project, you will start the beginnings of a program that could be modified to simulate a game of *Five Card Draw Poker*.

A video on how to play five Card Draw Poker can be found at: https://www.youtube.com/watch?v=-OcfEkOeC-w&ab_channel=TripleSGames.

If you are unfamiliar with the possible "hands" of poker, go to: https://en.wikipedia.org/wiki/List_of_poker_hands

**Part 1:** PASSWORD PROTECT your game:

- Declare as a constant YOUR last name, another constant with YOUR 9-digit WID# and another constant with the password **CIS200$Fall23**. (If your last name is < 4 letters, pad at the end with 'z')
- *Username* will consists of the *first* 4 letters contained in the constant NAME + *last* 4 digits of constant WID. (Assume last name is at least 4 letters)

For the username, **do not hardcode the value**. Use various *String methods* to parse so constants can be changed, and the program will still work. Although *String concatenation is allowed*, I recommend the use of the more efficient *StringBuilder* class to *build* the username.

```
StringBuilder sb = new StringBuilder();          // create the object
sb.append (<use substring to pull off the first 4 chars of the current name>);
sb.append (<use substring to pull off the last 4 chars of the current SS#>);
String userName = sb.toString();
```

For GTA testing purposes, please add the following before having the user enter *username* and *password*.

```
System.out.println("Username generated: " + userName);
```

- Require the user to enter a *username* and *password* (only the password is case sensitive) before they are allowed to play the game that follows. If the username AND/OR the password is/are invalid, display an error message and ask the user to re-enter (OK to make your error *specific* to the error or *generic* such as "*User and/or Password is/are invalid*").

  If invalid 3 times, display another error message ("*Invalid Username and/or Password entered 3 times – Exiting*") and *lock out* the user from playing the game by exiting the program.

*Suggestion: Get this part working first, then work on the next portion of this program. Comment out this portion of the program so you don't need to keep entering this info every single time when testing.*

Here is a sample run of how the first part MUST appear if *invalid* username and/or password are entered

```
Username generated: Lang4629                    (Displayed for grading purposes)

Please enter your Username: Lang46              (Both invalid)
Please enter your Password: CIS200Fall23
User and/or Password is/are invalid

Please re-enter your Username: Lang56           (Username invalid)
Please re-enter your Password: CIS200$ Fall23
User and/or Password is/are invalid

Please re-enter your Username: lang4629         (Username valid…username is NOT case-specific)
Please re-enter your Password: CIS200$FALL23    (Password invalid…password IS case-specific)
Invalid Username and/or Password entered 3 times...EXITING
```

---

**Part 2:** The Simplified Game

***Reminder: You are NOT dealing TWO hands, like in normal poker. That is left as extra credit.***

Once the username and password have been validated, have the program "*deal*" a single hand containing five random playing cards that are unique. Each card will have a ***suit*** (*Spades, Clubs, Hearts*, or *Diamonds*) and a ***value*** (*2-10, Jack, Queen, King,* or *Ace*). You will need to add data validation code that includes a loop to make sure that the same card (i.e., same *value* and same *suit*) is not '*dealt'* twice.

Once 5 *unique* cards are '*dealt',* determine if you have any of the following, according to the rules of poker. The classifications or possible *hands* (in descending order) are:

1) **Royal Flush** (all the same suit and containing 10-J-Q-K-A in any order)
2) **Straight Flush** (all same suit containing ANY 5-card sequence…e.g., 4,5,6,7,8 or 9,10,J,Q,K in any order)
3) **Four of a Kind** (Ex: 4-Jacks or 4-five's)
4) **Full House** (3 of a kind/value and 2 of a kind/value (a pair) – Ex: 3-Jacks and 2-five's)
5) **Flush** (all the same suit)
6) **Straight** (ANY 5-card sequential numerical sequence, like 4,5,6,7,8 or 9,10,J,Q,K in any order)
7) **Three of a Kind** (Ex: 3-Jacks or 3-five's)
8) **Two pairs** (Ex: 2-Kings *and* 2-three's)
9) **A pair** (Ex: 2-Kings *or* 2-three's)
10) **High Card**

You will display the BEST (*highest*) classification for your poker hand (1-9). If none of the hands 1-9 are drawn, then simply display your highest card. (From low to high: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A).

After displaying the results of your hand, ask the user if they wish to '*Play Again?*' If so, repeat and 'deal' another hand of 5 cards, and display the result. Assume the cards are *reshuffled* and *dealt* from a ***new*** *deck*. In other words, duplicates from the previous hand could be dealt.

Continue until the user replies they no longer want to play. (Program should accept 'Y' or 'y' or 'N' or 'n' ONLY – so validate and error check user response to "*Play Again?*")

Here is a sample run of how your output MUST appear with *valid* username and/or password

(*Yours will NOT display 'Play again' validation in purple nor be side-by-side. Done here to save space*)

Username generated: Lang4629

Please enter your Username: Lang4629
Please enter your Password: CIS200$Fall23

** Welcome to the 2023 Las Vegas Poker Festival! **
(Application developed by <Your Name>)

Shuffling cards…
Dealing the cards…
Here are your five cards…
- 2 of Hearts
- Ace of Spades
- 2 of Clubs
- 2 of Diamonds
- Ace of Clubs

You were dealt a Full House.
Play Again (Y or N)? t
Please enter a 'Y' or 'N' only
Play Again (Y or N)? t
Please enter a 'Y' or 'N' only
Play Again (Y or N)? y

Shuffling cards…
Dealing the cards…
Here are your five cards…
- 2 of Hearts
- 6 of Hearts
- 3 of Hearts
- 9 of Hearts
- Queen of Hearts

You were dealt a Flush.
Play Again (Y or N)? Y

Shuffling cards…
Dealing the cards…
Here are your five cards…
- 5 of Clubs
- King of Spades
- 8 of Hearts
- Jack of Diamonds
- 9 of Clubs

High card is a(n) King.
Play Again (Y or N)? Y

Shuffling cards…
Dealing the cards…
Here are your five cards…
- 5 of Spades
- 8 of Spades
- 9 of Diamonds
- 5 of Diamonds
- 9 of Clubs

You were dealt two pair.
Play Again (Y or N)? y

Shuffling cards…
Dealing the cards…
Here are your five cards…
- 5 of Clubs
- 6 of Hearts
- 5 of Diamonds
- 7 of Hearts
- 4 of Spades

You were dealt a pair.
Play Again (Y or N)? y

Shuffling cards…
Dealing the cards…
Here are your five cards…
- 2 of Hearts
- 6 of Hearts
- 3 of Hearts
- 5 of Hearts
- 4 of Hearts

You were dealt a Straight Flush.
Play Again (Y or N)? n

**Requirements** (As with previous projects, *Programmer defined objects* can NOT be used for this project.)

- To better simulate playing cards as well as help simplify testing for GTA, you must use *two* separate 1D *integer* ARRAYS named *value* and *suit* (NO 2D arrays or ArrayLists) declared as follows:

```
int[] value = new int[5];
int[] suit = new int[5];
```

- Generate random numbers to simulate "drawing" cards. (Random numbers were covered previously in lecture and in Chapter 3 of your zyBook.)
- Each time you *deal* a card, generate a random number to represent the *suit* (*Spades, Clubs, Hearts, or Diamonds*) represented by 1-4 *and* a random card *value (2-10, Jack, Queen, King, or Ace*) using 11 to represent a *Jack*, 12 for a *Queen*, 13 for a *King*, and 14 for an *Ace*.
- You must use arrays throughout your program to be considered for credit. Programs that do not include arrays will NOT be considered for grading (i.e., zero), since arrays are the focus of this project.

I strongly suggest you *sort* your cards to make it easier to determine if any of the stated classifications (*hands*) exists in the cards you were dealt. To sort in *ascending* order (descending is more difficult), simply use the command: **Arrays.sort (arrayname);**

(FYI: Although two parallel arrays are used in the program, only ONE needs to be sorted, so the originally generated value/suit pair will be "undone". That's OK. Just display your cards AFTER sorting)

---

Once you get the program to properly generate random numbers and run properly, do this following…

- Since your program will rarely generate anything higher than a pair, a file named *Proj4_Include File.java* is provided in Canvas with code to be used to help test different possibilities, so you do not have to continually run your program in an attempt to get certain hands.
- To use this file, *copy and paste* the contents of this file into your project above where you declare your arrays and generate the random numbers for your arrays. Then, for testing, comment out your array declarations and random number generations and replace with hard-coded different hands.

For example, pseudocode might look like this…

```
// COMMENT OUT this section if using hard-code values
/*  int[] value = new int[5];
    int[] suit = new int[5];

        …section of your code that generates the random numbers and populates your arrays…
*/
```

To test for a *Royal Flush*, uncomment the following from "Proj4_Include File.java" and run your project.

```
int[] value = {10, 12, 14, 13, 11};
int[] suit = {1,1,1,1,1};
```

To test for a *Straight Flush*, comment out above and uncomment below and run your project.

```
int[] value = {9, 7, 8, 6, 5};
int[] suit = {1,1,1,1,1};
```

Use each array declaration in "Proj4_Include File.java" to test that all 9 hand possibilities shown on p.2 work in your program (since that is what the GTA will be doing when grading your project!)

---

**IMPORTANT:** To be considered for extra credit, you MUST add a comment in your top documentation "EXTRA CREDIT INCLUDED", otherwise the GTA will NOT test for extra credit.

**Extra Credit Challenge (+10%)**

(FYI: If doing the extra credit, your output will not exactly match the output on p.3 but should still function correctly according to the rules of poker.)

**Add** the following functionality to your application (do NOT write a separate application):

Modify the program so that it is closer to a game of poker between two *players*. It will deal **2** hands, display the same info as above for each player, then determine and display a winner – *Player 1* or *Player 2* or a *tie*. The winner is the one with the highest classification 1-9 (e.g., *full house* beats a *flush*) or the highest card (if 1-9 is not dealt to either). To simplify this extra credit, we will simply consider it a *tie* if both get the same classification (e.g., a pair of Aces or a pair of 2's will be a TIE in this implementation).

Unlike a real game of poker, the two *players* are the computer (not any human players) and you are not including the normal second step in Poker of discarding cards and drawing that number of cards. Also, you are NOT checking for uniqueness between the two players (i.e., BOTH players could have a 2 of clubs), since it's only 10% extra credit!

So that the GTA can still use the hard-coded values, please name the array for player one *value* and *suit* and player2 *value2* and *suit2*. Then, they should be able to easily comment out the portion of code in which you randomly populate these two arrays, and use hard-coded values to test the other hands.

---

**Running Requirements:**

This project will contain ONE file (called **Proj4**) containing a single class and a main method.
(Additional methods allowed but *global variables declared outside of any method* are NOT allowed)

Your program MUST COMPILE by command-line with the statement: **javac Proj4.java**
*(not project4, Project4, proj4, etc.*

It must then RUN by command-line with the command: **java Proj4**

****Make sure and test this before submitting, so points are not lost unnecessarily.**

---

**Documentation:** At the top of *EACH* class, add the following comment block, filling in the needed info:

```
/**
* <Full Filename>
* <Student Name / Lab Section Day and Time>
*
* <COMPLETE description of the project (at least 3 or 4 sentences) – i.e. What does the program do?
  Must be detailed enough so outside reader of your code can determine the specifics of the program>
*/
```

---

Submission – read these instructions carefully or you may lose points

To submit your project, log-in to Canvas and upload your *Projx.java* file. Only a *.java* file will be accepted for this assignment in Canvas. Again, make sure you program compiles AND runs at the command line.

**Important**: It is the *student's responsibility* to verify that the *correct* file is *properly* submitted. If you don't properly submit the *correct* file, *it will not be accepted after the 2-day late period*. No exceptions.

---

**Grading:** Only what it submitted to Canvas before the deadline will be considered for grading, so make sure you submit the CORRECT file. Files can be re-submitted until the deadline but only the LAST submission will be graded.

Programs that do not compile/run from the command line will receive a grade of ZERO, regardless of the simplicity of the error, so make sure you submit the *correct* file that *properly compiles AND runs from the command line.* Programs that compile/run are graded according to the following rubric:

| Requirement | Points |
|---|---|
| IMPORTANT: Program must declare and use two 1D arrays named *value* and *suit* (no 2-D or ArrayLists) to hold cards values and the arrays are used throughout code to be considered for credit | - |
| Correct Documentation Heading with full description | **2** |
| 3 Constants properly declared | **2** |
| Username/Password validation works properly | **6** |
| Code included to check for duplicate cards (i.e., won't deal two *Ace of Spades*) | **3** |
| Correctly deals/displays five randomly generated cards with given heading ("*Welcome to…"*) | **7** |
| Correctly determines the classification dealt (*flush, straight, high card,* etc.) using randomly generated cards (GTA will test by drawing 5 hands / 3 pts each) | **15** |
| Ask user if they want to play again, error checks input and correctly loops and repeats program | **3** |
| Correctly determines the classification dealt (*flush, straight, high card,* etc.) with randomly generated section COMMENTED OUT and HARD-CODED VALUES from the starter file provided (GTA will test 3 different classifications, which will vary between students / 3 pts each) | **9** |
| Part 1 still runs correctly if ANY of the 3 constants are changed (no credit for this criterion if missing constants or if username is hard-coded) | **3** |
| **OPTIONAL Extra Credit (+10%)**: Deals 2 hands, displays the same info as above for each player, then displays a winner (*Player 1* or *Player 2*) or TIE. | **5** |
| **Minus Late Penalty (10% per day)** | |
| **Total** | **50 + 5** |