Unit Data Structures – *Arrays, ArrayList*

I. **Theory review**- lessons arrays, array list, Scanner use, for each loop use

Searching (AP)	**Sorting (AP)**
Examine and be able to independently implement algorithms for **searching** data in arrays: linear search, binary search 1. Sequential and Binary	Examine and be able to independently implement algorithms for **sorting** data in arrays: selection sort, insertion sort, merge sort 1. Selection, Insertion and Mergesort

II. **Assignments** – work periods **March 25 –March 29 (Ultimate due date April 1)**

A. Array Assignment 1 (Counting Cards)
B. Sorting assignment 2 (Student marks)
C. Sorting Assignment 3 (Country population)
D. Bonus (Phone Speed dal)
E. In Group-create own demonstration of sort algorithm. Submit a proposal of your idea for approval.

III. **Additional Instructions:**

1. **All assignments are to be done with in <u>pairs or max group of 3</u>.**
2. Each of the following programs is to be fully documented and properly indented.
3. You must submit each program (only the .java files), using a folder with your names and the name of the assignment.
4. Appropriately formatted output is expected for all questions.
5. The efficiency of the program and the overall design structure will be taken into consideration during the marking***. You may choose to create different versions using arrays, array lists or two –dimensional arrays.*** You are responsible for testing your programs under all possible conditions. You can assume that all inputs are valid unless it says differently in the question.
6. **To keep track of work done and individual contribution a participation record (1 per group) will be filled out DAILY. The record will be submitted along with your assignments.**

A. Array Assignment 1 Counting Cards

In the card game called **Bridge,** each of the four players is dealt thirteen cards. Numerical values correspond to each card.. One widely used system assigns values as follows:

• 4 points for each ace
• 3 points for each king
• 2 points for each queen
• 1 point for each jack
• 3 points for a void (no cards in a suit)
• 2 points for a singleton (one card in a suit)
• 1 point for a doubleton (two cards in a suit)

Write a program that will read from a file into an array any number of sets of 4 strings **each** representing a bridge hand. The string should consist of 26 characters**, two characters for each card**. For each card, the first character will represent the denomination (using A for ace, K for king, Q for queen, J for jack, T for ten, and 2 to 9 for the other denominations) and the second character will represent the suit (S for spades, H for hearts, D for diamonds, and C for clubs).

For example, the string **4DTCKC5H7CQSAH4H6SJHAC9H2H** would represent a hand containing the four of diamonds, ten of clubs, king of clubs, and so on.

Your program should repeatedly evaluate each of the 4 hands and If the input is valid, the program should evaluate each hand and print a display of the hand together with its total value. A valid hand is one which has the correct number of characters as well as valid card values.

The display should be arranged according to the following rules:

1. Spades should be listed on the first line, hearts on the second line, diamonds on the third line, and clubs on the fourth line.
2. Within each line, the cards of a suit should be arranged in descending order of value (ace, king, queen, jack, 10, 9, ... , 2) with each value separated by one space. ***Drop off in a folder named CountingCards.***

Array assignment # 2- Student marks

1. Write a program using methods and parameters to do all of the following:
   a) Input a list of names and corresponding marks (1 mark per name) from the file **A7-1.txt** from the Pickup folder.Assign data to arrays. Output the original data to the console in 2 columns with 5 spaces apart.

   b) Sort the data alphabetically by name, using insertion, selection, or merge sort, and output the data to the console in columns. *Make sure that the name and the corresponding mark stay together.*

   c) Sort the data numerically, from largest to smallest, by mark, using insertion, selection, or merge sort, and output the data to the console in columns. *Make sure that the name and the corresponding mark stay together.*

Assignment 2 reads from a file and does not output to a file. Create different files for each of a,b,c. Name each class StudentMarksA, StudentMarksB, StudentMarksC. ***Drop off in a folder named StudentMarks.***

Array Assignment 3 Sorting Countries (by name and by population)

You are given a text file ***Countries-Populations.txt***, containing a list of countries of the world and their populations. Each line in the files contains information for one country, the country's name, the country's capital, area in square miles and population.

A. Input data from the files:

1. Read the lines in the file ***countries-populations.txt***
2. Declare 4 Arrays / ArrayLists –one for country name, one for the capital, one for the area and one for the population.
3. Using a while loop, fill in these arrays with the values read from the files.

B. Process the data and output, using FileWriter:

4. Sort the array list of *countryNames* alphabetically and save the sorted list of countries with their population to a file with the name ***sortedByCountry.txt*.** Use this format for each line:

```
Afghanistan                  31,056,997
Albania                       3,581,655
Algeria                      32,930,091
```

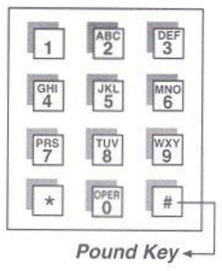
   The space between the country and its population should be coded with the TAB key: "Afghanistan **\t\t\t** 31,056,997"

5. Sort the countries by their population and save the sorted list to a file with the name ***sortedByPopulation.txt***. Use the same format:

```
China                         1,313,973,713
India                         1,095,351,995
United States                   298,444,215
```

6. Include proper documentation and in-line comments

Challenge (Bonus) assignment

 1 ABC 2 DEF 3 GHI 4 JKL 5 MNO 6 PRS 7 TUV 8 WXY 9 * OPER 0 # Pound Key	It is common to convert a phone number into a word by using the letters that each number converts to on a phone pad. For example the number 2 is represented by the letters a, b, or c. Write a program that will continually prompt the user for a phone number, of length 1 to 4, and will return all possible words that could represent that number. Your program should check for valid input (using a try/catch) and allow the user to reenter if an error occurs. If no words can be made from a given number then an appropriate message should be given to the user. The dictionary file, with about 60 thousand words in it, is called "**dict.txt**" and is in the pickup folder. You are going to use a HashSet object to give you the capability to search the words in the dictionary file, see "HashLookupEx.java" for example code using the HashSet object. Note: Most phone pads also include the letter Q for the number 7, and the letter Z for the number 9. For simplicity, if you want you can use the phone pad shown above which only has a maximum of 3 letters for each number. For example if the number 2665 was entered one of the words to represent that number is "cool". To receive a Level 4 also output the amount of time the search took, from the time the user hit Enter until all possible words were found.

Criteria		Level 1 50 – 59%	Level 2 60 – 69%	Level 3 70 – 79%	Level 4 80 – 100%	Mark
Knowledge & Understanding	Understanding of sorts	Demonstrates limited understanding of how and when to use sorting	Demonstrates some understanding of how and when to use sorting	Demonstrates a considerable understanding of how and when to use sorting	Demonstrates an extensive understanding of how and when to use sorting	
	Identification of data	Rarely identifies data best represented as lists	Identifies some data best represented as lists	Usually identifies data best represented as lists	Clearly differentiates data best represented as lists	
	Use of programming constructs for sequence, selection, repetition, input & output, subprograms	Program structures are not complete and/or improvement needed in use	Most program structures are complete; some improvement needed in use	All program structures are complete; minor improvement in use required	All program structures are complete and make sense	
Applications	Use of indices	Rarely uses indices and improperly declares arrays	Partially implements use of indices in programming structures and declares arrays and uses some appropriate variable types	Consistently uses indices correctly in programming structures and declares arrays using appropriate data types and index values	Uses indices correctly with a high degree of programming structures and code reflects efficient use of arrays	
	Completion of required components in the questions	Few components of the problems are complete and/or work correctly.	Some components of the problems are complete and/or work correctly.	Most components of the problems are complete and/or work correctly.	All components of the problems are complete and work correctly.	
	Applies subprograms appropriately in problems.	Demonstrates limited ability to use subprograms in problems.	Demonstrates some ability to use subprograms in problems.	Demonstrates considerable ability to use subprograms in problems.	Demonstrates extensive ability to use subprograms in problems.	
	Verifying a program	Demonstrates limited ability to adequately verify a program	Demonstrates some ability to adequately verify a program	Demonstrates considerable ability to adequately verify a program	Demonstrates extensive ability to adequately verify a program	
Thinking & Inquiry	Problem analysis	Demonstrates limited ability to analyze a problem	Demonstrates some ability to analyze a problem	Demonstrates considerable ability to analyze a problem	Demonstrates extensive ability to analyze a problem	
	Algorithm design	Demonstrates limited ability to design an algorithm as theprogram rarely solves problem	Demonstrates some ability to design an algorithm as the program is functional but fails limit testing	Demonstrates considerable ability to design an algorithm as the program adequately solves problem	Demonstrates extensive ability to design an algorithm as the program provides exemplary solution	
Communication	Descriptive naming conventions	Demonstrates limited ability to use descriptive naming conventions and rarely uses proper style (indentation, separation of program structures)	Demonstrates some ability to use descriptive naming conventions and usually uses proper style	Demonstrates considerable ability to use descriptive naming conventions and consistently uses proper style	Demonstrates extensive ability to use descriptive naming conventions and consistently uses proper conventions and style	
	Program documentation	Demonstrates limited ability to document a program clearly and adequately	Demonstrates some ability to document a program clearly and adequately	Demonstrates considerable ability to document a program clearly and adequately	Demonstrates extensive ability to document a program clearly and adequately	
	Presentation of program on the screen or printer	Demonstrates limited ability to clearly present the information on the console/file . Inadequate spacing/ headings/ text organization	Demonstrates some ability to clearly present the information on the console /file. Somewhat adequate spacing / headings/ text organization	Demonstrates considerable ability to clearly present the information on the console / file. Adequate spacing/ headings / text organization.	Demonstrates extensive ability to clearly present the information on the console/ file. Excellent spacing/ headings / text organization.	