CME1212 Algorithms and Programming II
Homework 2

Upload your source code files from *DEUZEM SAKAI* until **15 April 2021, 23:55.**

Write a Java program for the card game: ***"Matching Pairs"***.

It is a game that requires finding identical pairs.
In other words, it is a game where the user needs to match pairs of tiles.

Tiles should include <u>fruit</u> words such as apple, orange, kiwi, banana, etc.

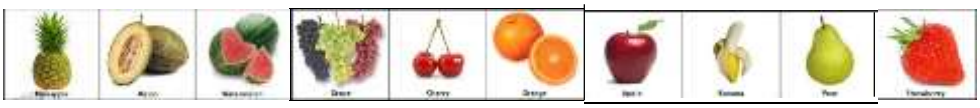
<u>A sample file:</u>
fruits.txt

Pineapple
Melon
Watermelon
Grape
Cherry
Orange
Apple
Banana
Pear
Strawberry

The Beginning of the Game

There is a text file (D:*fruits.txt*) that includes maximum 15 fruits. Read the file and insert them in a ***fruitStack***.

A sample *fruitStack*:

fruitStack 

top

There are two stacks with *x* in size. The second stack contains the matches of tiles in the first stack.
The default value of *x* is 5. If the user wants a different size, he/she can change the value of *x* in the source code.

The game boards (**stacks**) should be randomly filled with <u>distinct</u> pairs. You should randomly select *x* fruits from *fruitStack* to fill game boards. Each element in the stacks should be different from the others. For example, a stack doesn't contain two orange fruits.

Example:

Stack1 

Stack2 

Game Playing

Playing is very simple - the computer turns over two tiles randomly, one tile from the first stack and one tile from the second stack. If they are identical, the program deletes them from the game boards (***stacks***), if not, it tries again.

The End of the Game

When all pairs are identified (when all tiles are deleted from the game boards (***stacks***)), the game will be over.

The program must display all steps until the game is over.

The followings should also be printed.
- the number of tries (step) and
- the score.

Scoring
The scoring principle is as follows:

- Each time the computer makes a successful match, the score should be increased by 20 points.

- If the computer fails to match, the score should be decreased by 1 point.

Pelin
180
Kaan
160
Ali
150
Yeliz
140
Cem
120
Can
100
Ece
90
Sibel
80
Remzi
70
Nazan
65

High Score Table
Read the file "*D:\\highscoretable.txt*".

The high-score-table contains the top-10 game scores in descending order.
The high-score-table must be read from the text file and inserted into the *high-score-stack(s)*.

If he/she gets a score within the top-10 results, he/she must be inserted into the *high-score stack*(s).
If the same score exists in the table, the new score should be inserted to the next of them.

At the end of the game, the new score table should be written to the same file.

The program ends after printing the high score table on the screen.

Don't take any input from the user !!!

This homework will be graded by Res.Asst.Dr. Özge KART.
You can ask your questions her from the "**FORUM → Homework 2 - Questions**" part of the *DEUZEM SAKAI* software.

Sample output:

```
Fruit Stack: pineapple  melon  watermelon  grape  cherry  orange  apple  banana  pear strawberry

Stack1: melon  pear  orange  apple  banana                score=0
Stack2: banana  apple  pear  orange  melon

Randomly generated numbers: 2   5                        step=1

Stack1: melon  pear  orange  apple  banana                score=-1
Stack2: banana  apple  pear  orange  melon

Randomly generated numbers: 1   5                        step=2

Stack1: pear  orange  apple  banana                       score=19
Stack2: banana  apple  pear  orange

Randomly generated numbers: 2   2                        step=3

Stack1: pear  orange  apple  banana                       score=18
Stack2: banana  apple  pear  orange

Randomly generated numbers: 2   3                        step=4

Stack1: pear  orange  apple  banana                       score=17
Stack2: banana  apple  pear  orange

Randomly generated numbers: 4   1                        step=5

Stack1: pear  orange  apple                              score=37
Stack2: apple  pear  orange

Randomly generated numbers: 1   2                        step=6

Stack1: orange  apple                                    score=57
Stack2: apple   orange

Randomly generated numbers: 2   2                        step=7

Stack1: orange  apple                                    score=56
Stack2: apple   orange

Randomly generated numbers: 2   2                        step=8

Stack1: orange  apple                                    score=55
Stack2: apple   orange

Randomly generated numbers: 2   1                        step=9

Stack1: orange                                           score=75
Stack2: orange

Randomly generated numbers: 1   1                        step=10

Stack1:                                                  score=95
Stack2:
```

```
The game is over.

High Score table:
Pelin  180
Kaan   160
Ali    150
Yeliz  140
Cem    120
Can    100
You     95
Ece     90
Sibel   80
Remzi   70
```

Notes

1- In your program, you can use the stack data structure as you want, but you must use only stack.
Don't use other data structures such as an array or arraylist or list.
Don't use STRING data type in the main solution, instead of a stack.

2- The stack class has only the following methods: push, pop, peek, isFull, isEmpty, and size.
Don't add a new method into the stack class.

For example, don't write a *display* method in the Stack class.
For example, don't write a *search* method in the Stack class.

All other methods must be written in the ***main*** program.

3- Upload format

Step1: Create a new folder, named by your student number and name (without any space)
For example: 2015510012_Ali_Tas

Step2: Copy all java files into this folder


Step3: Compress the folder **2015510012_Ali_Tas.zip**

Step4: Upload the file **2015510012_Ali_Tas.zip** from *DEUZEM SAKAI*

4- Don't use **ENIGMA** or any other extra library.

5- If you are late, your grade will be decreased by 10 points for each day. After five days, your assignment will not be accepted.

6- Assignment must be your individual work.
Cheating is strictly prohibited.
All source codes will be automatically compared with each other by using a program.
If any cheating occurs, your assignment will be graded with **zero (0)**.

7- Your program must work correctly under all conditions. Try to control all possible errors.

8- You should use meaningful variable names, appropriate comments, and good prompting messages.