Sofia University
Department of Mathematics and Informatics

Course : Applied OO Programming part 1
Date: March 16, 2025
Student Name:

Lab No. 5

Problem 1a

Card Shuffling and Dealing) Modify the application of Fig. 7.9-11 (see Lab5SampleCode.rar) to deal a five-card poker hand. Then modify class DeckOfCards of Fig. 7.10 to include methods that determine whether a hand contains

a) a pair

b) two pairs

c) three of a kind (e.g., three jacks)

d) four of a kind (e.g., four aces)

e) a flush (i.e., all five cards of the same suit)

f) a straight (i.e., five cards of consecutive face values)

g) a full house (i.e., two cards of one face value and three cards of another face value)

Sample output expected as follows

```
Left hand:           Right hand:
Queen of Hearts      Three of Hearts
Four of Diamonds     King of Spades
Three of Spades      Queen of Spades
Jack of Hearts       Eight of Clubs
Eight of Spades      Queen of Diamonds

Hand Values:
none                 One Pair

Result: right hand is better
```

Problem 1b

(*Simulation: coupon collector's problem*) Coupon collector is a classic statistics problem with many practical applications. The problem is to pick objects from a set of objects repeatedly and find out how many picks are needed for all the objects to be picked at least once. A variation of the problem is to pick cards from a shuffled deck of 52 cards repeatedly and find out how many picks are needed before you see one of each suit. Assume a picked card is placed back in the deck before picking another. Write a program to simulate the number of picks needed to get four cards from each suit and display the four cards picked (it is possible a card may be picked twice). Here is a sample run of the program:

```
Queen of Spades
5 of Clubs
Queen of Hearts
4 of Diamonds
Number of picks: 12
```

Problem 1c

(*Game: pick four cards*) Write a program that picks four cards from a deck of 52 cards and computes their sum. An Ace, King, Queen, and Jack represent 1, 13, 12, and 11, respectively. Your program should display the number of picks that yields the sum of 24.

Problem 2a

Напишете Java конзолно приложение за шифроване на текстов низ от данни (***plaintext***) посредством алгоритъма на тъй наречения ***Caesar cipher***. При този алгоритъм всяка буква в текста за **шифроване** (***plaintext***) се **замества с буква отстояща на дясно от нея** на ***SHIFT_LENGTH*** **позиции** и се получава шифрования текст (***ciphertext)***

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Например, в оригиналната версия на този алгоритъм ***SHIFT_LENGTH=3*** и думата **TOY** (***plaintext***) се шифрова като **WRB** (***ciphertext)*- забележете,** *заместването с букви в края на азбуката води до* ***циклично продължение*** *от началото на азбуката*

Дешифриране се извършва **по обратния път** (*циклично заместване на вяка буква с тази отстояща от нея на* ***SHIFT_LENGTH позиции*****)** по зададени ***ciphertext***.

Напишете *конзолно* приложение на **Java**, която **кодира** даден ***plaintext*** и **декодира** даден (***ciphertext)*** в съответствие с по- горе **описания алгоритъм**.

Всички операции за обработка на низове да се сведат до работа с масиви като се **използват единствено** метода ***toCharArray***() на ***class String*** за преобразуване от ***String*** в масив от ***char.*** **Обратното преобразуване** от ***char[]*** в ***String*** става с използване на **конструктора** ***String(char[]).*** **Използване** на **други** ***String*** методи **не се допуска.**

При тестването на приложението използвайте меню в текстов режим за избиране на шифроване или дешифроване

Problem 2b

A **simple transposition cipher** works as follows to encrypt a given string (**plaintext**). The user enters phrase and **the number of characters of the phrase** are considered as the **cipher key**. Then the user supplies the string to encrypt- the plaintext. The encryption procedure cuts the plaintext into series of characters, whose number is equal to the cipher key, where the last portion of characters is eventually padded with spaces to the size of the cipher key. These series are stored by rows of a matrix and the encrypted text comprises the sequence of columns of the thus obtained matrix. Example of encryption:

plaintext: thisistheplaintext
key text **beauty** **→cipher key= 6**
encryption

thisis
thepla
intext

cipher text: ttihhnietspeilxsat

Once the user **provides the key text and the cipher text** the **decryption reconstructs the original plain text.**

Write a **Java application** to **implement the encryption and decryption process <u>using</u>** the ***String.toCharArray()*** method and arrays of ***char*** datatype as in the following example

```
string delimStr = " ,.:";
char [] delimiter = delimStr.toCharArray();// to char array
String text = new String(delimiter);        // to string
```

DO NOT USE ANY OTHER `string` METHODS!

Use **OO design** and define also a **class TransCipher** with respective data, constructors and methods, allowing you to implement the above encryption/ decryption functionality.

Test encryption / decryption of a string using different cipher keys.

<u>Problem No. 3</u>

За по- бързо сортиране на писма, Американската пощенска служба съветва бизнес компаниите, изпращащи големи количества от поща, да използват бар- код за означаване на ZIP кода на писмата. За целта се използва следната пет- цифрена схема за кодиране на **3- цифрен ZIP код**:

Цифра/базис на кодас	*7*	*4*	*2*	*1*	*0*
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

Така, **всяка от трите цифри на ZIP - кода** се представя като подреден набор от 5-те коефициента (`0` или `1`) пред базиса на кода според горната таблица.

Например, за представянето на **цифрата 1** в даден **3- цифрен ZIP код 123** използваме, че

`1 = 0*7 + 0*4+ 0* 2 + 1* 1 + 0*1`

и съответния й бар- код на тази цифра е `00011` **(сравнете с коефициентите за цифрата *1 в горната таблица*). Аналогично, баркодът на цифрите 2 и 3 в този цифрен ZIP код са съответно `00101` и `00110`.**

Напишете *class BarCode* , което има клас данна **code**, която е **статична константа**, реферираща **двумерен масив с 10 реда на 5 стълба** , чиито елементи са **инициализирани** с **0** или **1** **в съответствие с горната таблица**.

Напишете метод ***decode***() на ***class BarCode*** , който взима за аргумент трицифрено ***цяло число*** **(ZIP – кода)**, а връща ***String*** със **съответното бар- код представяне** на числото. **Използвайте** символа **':'** за **извеждане на нулите** в бар кода и символа **'|'** за **извеждане на единиците** в бар кода.Така, би следвало **ZIP – код** **111** да се представи със следния текст

: : : | | : : : | |: : : | |

Напишете приложение , което тества ***class BarCode*** **–**

- **Въвежда от стандартен вход** ZIP код (трицифрено ***цяло число***)
- Отпечатва **баркода на този** . ZIP код, чрез извикване на метод ***decode***() на обект от ***class BarCode***

Problem 4

(*Merge two sorted lists*) Write the following method that merges two sorted arrays into a new sorted array.

```
public static int[] merge(int[] list1, int[] list2)
```

Implement the method in a way that takes at most **list1.length + list2.length** comparisons. Write a test program that prompts the user to enter two sorted lists and displays the merged list. Here is a sample run. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list

```
Enter list1: 5 1 5 16 61 111 Enter
Enter list2: 4 2 4 5 6 Enter
The merged list is 1 2 4 5 5 6 16 61 111
```

Problem 5

(*Algebra: multiply two matrices*) Write a method to multiply two matrices. The definition of the method is:

```
public static double[][] multiplyMatrix(double[][] a, double[][] b)
```

To multiply matrix **a** by matrix **b**, the number of columns in **a** must be the same as the number of rows in **b**, and the two matrices must have elements of the same or compatible types. Let **c** be the result of the multiplication. Assume the column size of matrix a is **n**. Each element *cij* is

$$a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + \ldots + a_{in} \times b_{nj}.$$

For example, for two 3 * 3 matrices **a** and **b**, **c** is

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

where $c_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + a_{i3} \times b_{3j}$.

Write a test program that prompts the user to enter two 3 * 3 matrices and displays their product.

```
Enter matrix1: 1 2 3 4 5 6 7 8 9 Enter
Enter matrix2: 0 2 4 1 4.5 2.2 1.1 4.3 5.2 Enter
The multiplication of the matrices is
 1 2 3      0 2.0 4.0      5.3 23.9 24
 4 5 6   *  1 4.5 2.2  =   11.6 56.3 58.2
 7 8 9      1.1 4.3 5.2    17.9 88.7 92.4
```

Problem 6

(*Shuffle rows*) Write a method that shuffles the rows in a two-dimensional `int` array using the following header:

```
public static void shuffle(int[][] m)
```

Write a test program that shuffles the following matrix:

```
int[][] m = {{1, 2}, {3, 4}, {5, 6}, {7, 8}, {9, 10}};
```

Problem 7

Credit card numbers follow certain patterns. A credit card number must have between 13 and 16 digits. It must start with:

- 4 for Visa cards
- 5 for Master cards
- 37 for American Express cards
- 6 for Discover cards

In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine if a card number is entered correctly or if a credit card is scanned correctly by a scanner. Almost all credit card numbers are generated following this validity check, commonly known as the Luhn check or the Mod 10 check, which can be described as follows (for illustration, consider the card number 4388576018402626):

1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number.

2 * 2 = 4

2 * 2 = 4

4 * 2 = 8

1 * 2 = 2

6 * 2 = 12 (1 + 2 = 3)

5 * 2 = 10 (1 + 0 = 1)

8 * 2 = 16 (1 + 6 = 7)

4 * 2 = 8

2. Now add all single-digit numbers from Step 1.

4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37

3. Add all digits in the odd places from right to left in the card number.

6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38

4. Sum the results from Step 2 and Step 3.

37 + 38 = 75

5. If the result from Step 4 is divisible by 10, the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is invalid, but the number 4388576018410707 is valid.

Write a program that prompts the user to enter a credit card number as a long integer. Display whether the number is valid or invalid.

All the String processing must be avoided by transforming the String input into an array of characters by means of the method `toCharArray()` of class. The reverse transformation of a `char[]` into String is done by means of the constructor `String(char[])` of class `String`. **The use of other String methods is not allowed**

Here are sample runs of the program:

Sample 1:

```
Enter a credit card number as a long integer: 4246345689049834
4246345689049834 is invalid
```

Sample 2:

```
Enter a credit card number as a long integer: 4388576018410707
4388576018410707 is valid
```

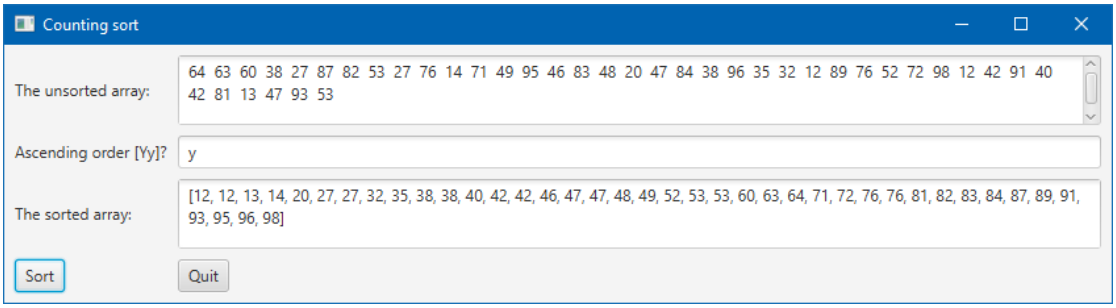
Problem 8

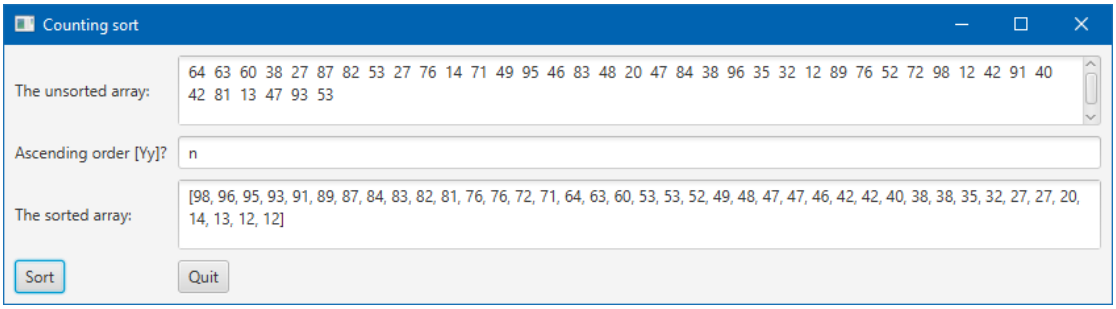
Implement the Counting sort algorithm in Java allowing to sort numbers in ascending and descending order. The method implementing the algorithm should be in the form:

```
public void sort(int[] arrayToSort, boolean sortOrder)
```

The method sorts `arrayToSort` in ascending or descending order when `sortOrder` is correspondingly `true` or `false`.

Implement the method in a JavaFXML application with the following GUI





Use class Random to generate the elements of `arrayToSort`

Problem 8

Write a class Horner with a static method

`public double computeByHorner(double[ ] digits)`

where digits are the coefficients of a polynomial of degree **`digits.length + 1`**

For example the coefficients $a_k, k = 0, 1, \ldots, n$ of the polynomial

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

can be stored in a one dimensional array **`digits`** with $n + 1$ elements. In practice, the evaluation of this polynomial for a given value of xДа can be used to compute the value of a number represented in a numeric system with base, 10, 2, 8, or 16, where the coefficients a_k are the digits in the number representation and x is the base in the numeric system i.e. $x = 10, 2, 8 \; or \; 16$

Implement the method of Horner to evaluate such a polynomial for $x = x_o$ making use of the following algorithm.

1. Initialize $k = n$, where n is the degree of the polynomial i.e. k `is digits.length - 1`
2. Denote $b_k = a_k$
3. Compute $b_{k-1} = a_{k-1} + b_k x_o$
4. Initialize $k = k - 1$
5. If $k \geq 0$ go to step 3, otherwise return the last computed value for b as the value $f(x_o)$

The algorithm can be illustrated as follows:

K	5	4	3	2	1	0
	$b_5 = a_5$	$b_4 = a_4 + x_0 b_5$	$b_3 = a_3 + x_0 b_4$	$b_2 = a_2 + x_0 b_3$	$b_1 = a_1 + x_0 b_2$	$b_0 = a_0 + x_0 b_1$

for a polynomial of degree 4 with coefficients {9, 7, 5, 3, 1} for $x = 2$ we obtain

4	3	2	1	0
$b_4 = 1$	$b_3 = 3 + 2 \cdot 1$	$b_2 = 5 + 2 \cdot 5$	$b_1 = 7 + 2 \cdot 15$	$b_0 = 9 + 2 \cdot 37$
1	5	15	37	83

1. Draw the UML activity diagram for the Horner algorithm using https://app.diagrams.net/?src=about
2. Write the software implementation of **class Horner** with a static method `public double computeByHorner(double[ ] digits)`
3. Test `computeByHorner(double[ ] digits)` with {9, 7, 5, 3, 1} for $x = 2$ and $x = 10$
4. Extend the application of **class Horner allowing** it evaluate hexadecimal numbers.