Use the following lines in order to generate a random integer:

```
import java.util.Random;
//Par exemple, générer un nombre dans l'intervalle [a,b]:
Random random = new Random();
int nb=a+random.nextInt(b-a);

```

Exercise 2.16 *Let us take exercise 2.5 again.*

① *Represent the coordinates of n points in a two-dimensional space, using a two-dimensional table, named $PointsT$, and fill it randomly;*

② *Let $P(x, y)$ be a point. Calculate and display the distance between P and all points in the table $PointsT$, as well as the minimum distance and the point in the table $PointsT$ that has the minimum distance.*

Coordinates of the n points are real numbers. For generating a real random number in a range $[a, b]$:

```
import java.util.Random;
Random random = new Random();
double nb = a+random.nextDouble(b-a);

```

Exercise 2.17 *Write a program that computes the standard deviation of N real numbers saved in an ArrayList (or array). The standard deviation S is computed as follows:*

$$S = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2}{N}}$$

The variable $\bar{x}$ is the average of N input values x_1 through x_N. In the case where an array is used, the program first prompts the user for N and then declares an array of size N. Noting that the N real values can be generated randomly as well as N.

Exercise 2.18 *Use an array-list of integers to represent the first n element of the Fibonacci sequence. Then, compute and display their sum.*

R *The Fibonacci sequence is defined as follow:*

$$\begin{aligned} U_0 &= 0, \\ U_1 &= 1, \\ U_n &= U_{n-1} + U_{n-2} \text{ for } n >= 2 \end{aligned}$$

Exercise 2.19

① *Display alphabet in uppercase (A to Z) and lowercase (a to z).*

② *Convert all characters of a given string st to uppercase.*

③ *Display the number of uppercase characters, lowercase characters, vowels, numeric characters in a given string St.*

R *Use the following statements for the second question:*

```
Character.isLowerCase('a')
Character.isUpperCase('a')
Character.isLetter('a')
Character.isDigit('2')

```

Exercise 2.20 ***Functions for array manipulation***

We want to define a collection of functions for manipulating tow-dimensional array. In order to do that, start by defining the following static functions:

① *A function named random, that returns as value a random integer $\in [1, 10]$;*

② *A function named $fill$, that fills randomly a table t of integer;*

③ *A function named $display$, that displays values of a table t of integer;*

④ *A function named minT, that returns as value the min value of a table t of integer;*

⑤ *A function named maxT, that returns as value the max value of a table t of integer;*

⑥ *A function named somT, that returns as value the sum of all values of a table t of integer;*

As second step, use the above functions to write the following functions:

① *A function named $fill$, that fills randomly a matrix m of integer;*

② *A function named $display$, that displays values of a matrix m of integer;*

③ *A function named minT, that returns as value the min value of a matrix m;*

④ *A function named maxT, that returns as value the max value of a matrix m;*

⑤ *A function named somT, that returns as value the sum of all values of a matrix m;*

Finally, in order to test the above defined functions, use these functions in a main program to fill randomly an irregular table T and display values of T, the min, the max, and the sum of all values of T. The size of each line of T must be also a random value.

R *An example of an irregular table is shown in the Exercise 2.26.*

Exercise 2.21 ***Amicable numbers***

In mathematics, two natural numbers n and m are said to be amicable (friendly or amicable) if the sum of the proper divisors of n equals m and the sum of the proper divisors of m equals n. The proper divisors of a number m, noted $div_p(m)$, are all divisors of m accept itself. For example, the proper divisors of 6, $div_p(6)$ = {1, 2, 3}. So, two numbers m and n are amiable if: $\sum div_p(m)$= n and $\sum div_p(n)$=m.

For example: 220 and 284 are amicable numbers, because:

$\sum div_p(220)= 1+2+4+5+10+11+20+22+44+55+110 = 284$

and

$\sum div_p(284)= 1+2+4+71+142 = 220$

An other definition (manner), the amicable numbers are two different natural numbers related in such a way that the sum of the divisors of each are equals, and equals to their sum. The divisors of a number m are all divisors of m, noted $\sigma(m)$. For example, $\sigma(6)$ = {1, 2, 3, 6}. So, two numbers m and n are amiable if: $\sum \sigma(m)= \sum \sigma(n)=n+m$.

For example: 220 and 284 are amicable numbers, because:

$\sum \sigma(220)= 1+2+4+5+10+11+20+22+44+55+110+220 =$ ***504***

and

$\sum \sigma(284)= 1+2+4+71+142+284 =$ ***504***

and $220+284 =$ ***504***

- ***Requested work**:*

  Modularize the code using auxiliary methods to write a method named $FriendNumbers$, that takes as input an array of integers and displays all pairs of friendly numbers within it. Each pair will be displayed only once.

R *The first seven amicable pairs are: (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368), (10744, 10856), (12285, 14595).*

2.10 Supplementary activities

Exercise 2.22 ***Closest number to zero***

Write a program that displays the temperature closest to 0 among input data. If two numbers are equally close to zero, positive integer has to be considered closest to zero (for instance, if the temperatures are -4 and 4, then display 4).

- *Input 1: N, the number of temperatures to analyze*
- *Input 2: N temperatures expressed as integers ranging from -273 to 5526*
- *Output: Display 0 (zero) if no temperatures are provided. Otherwise, display the temperature closest to 0.*
- *Constraints : $0 < N \leq 10$*

Exercise 2.23 *You need to figure out what to do by observing the following input/output:*

- *Input: 5 4*
- *Output:*

```
*****
*   *
*   *
*****
```

Exercise 2.24 ***Prime numbers***

Display the sum of the first N prime numbers. For example, if $N = 4$, we get four prime numbers: $2, 3, 5, 7$ and the sum of these numbers is 17.

Exercise 2.25 *Use an array-list of integers to represent the first n element of the Syracuse sequence. Then, compute and display their sum as well as the min and the max values.*

R *The Syracuse sequence is defined as follow:*

$$\begin{aligned} U_0 &\in N^*, \text{ we choose } U_0 = 7 \\ U_{n+1} &= \frac{U_n}{2}, \text{ if } u_n \text{ is even} \\ U_{n+1} &= 3U_n + 1, \text{ if } u_n \text{ is odd} \end{aligned}$$

Exercise 2.26 ***Irregular tables***

Irregular arrays are multidimensional arrays such that each row has different size. Write a Java program that declares, initializes, and displays the following irregular array:

1

2 3

4 5 6

7 8 9 10

Exercise 2.27 *The purpose of this exercise is to write a program to identify the most frequently occurring element in an array of integers. This program should also display the number of occurrences in the array of this most frequent element. For example, for the following array:*

{2, 7, 5, 6, 7, 1, 6, 2, 1, 7}

The program should indicate that the most frequent element is 7 and that its frequency of appearance is 3.

Exercise 2.28 ***ASCII Code***

Assuming a string represents the address of a website, for example: `St="www.univjijel.dz"`. *To calculate the IP address, we need to convert all the characters of St to ASCII code and then calculate the sum of the obtained values. For xample, "abc"* $\Longrightarrow 97 + 98 + 99 = 294$

The first IP number will be the result mod 256 (Because there is no IP number greater than 256). The second IP number will be double of the sum mod 256, the third will be triple of the sum mod 256 and the fourth will be quadruple of the sum, mod 256. For example:

- *input: "www.univjijel.dz"*
- *output: Ip adress: 111.222.77.188*

R *It is not a real algorithm, it is only an adaptation to the exercise.*