Exercises - Unit 7
Arrays: definition and applications

Group I1E

Year 2017/2018

1. Given an array of integers `v`, consider the following code:

```
int i;
for (i = 0; i < v.length && v[i] == 0; i++);
if (v[i] != 0) return i;
else return -1;
```

   What happens when this code is executed and all the elements of `v` are initially equal to 0?

2. What does the following code?

```
public static int f(int x, int [] a) {
    int found = -1, i = 0, j = a.length - 1;
    while (i <= j && found == -1) {
          if (a[i] == x) found = i;
          else if (a[j] == x) found = j;
          i++;
          j--;
    }
    return found;
}
```

3. What shows on the screen the following code?

```
class Trace{
  public static void main(String[] args) {
     int[] array1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
     int[] array2 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
     f(array1);
     f(array2);
  }
  public static void f(int[] a) {
     int i = 0, j = a.length-1;
     if (j%2 == 0) j--;
     while (i < a.length && j >= 0) {
        System.out.print(" " + a[i] + " " + a[j]);
        i+=2;
        j-=2;
     }
     System.out.println();
  }
}
```

4. Write a class method that receives an array of integers `v` and an integer `x` and returns the number of occurrences of `x` in `v` (number of times that `x` appears in `v`)

5. Write a class method that receives an array of integers `v` and two integers `b` and `e` ($0 \leq$ `b` $\leq$ `e` $\leq$ `v.length-1`), and then multiplies by two the elements of `v` between those two positions `b` and `e`

6. Write a class method that receives an array of integers `v` and two integers `b` and `e` ($0 \leq$ `b` $\leq$ `e` $\leq$ `v.length-1`), and then inverts the elements of the array between those two positions (i.e., element `v[b]` will be exchanged with `v[e]`, `v[b+1]` with `v[e-1]`, etc.)

7. Write a class method that receives an array of integers `v` and two integers `b` and `e` ($0 \leq$ `b` $\leq$ `e` $\leq$ `v.length-1`), and then moves one position to the right (i.e., from position i to position $i+1$) all the elements of the array between those two positions (both included); the movement must be circular (i.e., element in position `e` will be finally in position `b`)

8. Write a class method that receives an array of integers `v` and two integers `b` and `e` ($0 \leq$ `b` $\leq$ `e` $\leq$ `v.length-1`), and then moves one position to the left (i.e., from position i to position $i-1$) all the elements of the array between those two positions (both included); the movement must be circular (i.e., element in position `b` will be finally in position `e`)

9. Write a class method that receives an array of integers `v` and returns the maximum number stored in the array

10. Write a class method that receives an array of integers `v` and returns the second highest number stored in the array (suppose that the size of the array is at least 2)

11. Write a class method that receives an array of integers `v` and returns the number of odd elements in even positions

12. Write a class method that receives an array of integers `v` and two integers `x` and `n`, and returns the number of elements of `v` lower than `x` that are in positions previous to `n`. Tip: `n` can be any integer value, including negative values and values greater than `v.length`

13. Write a class method that receives an array of integers `v` and returns if the array is sorted in an ascendent way

14. Write a class method that receives an array of integers `v` and returns the position (if exists, otherwise return -1) of the first subsequence of the array with three consecutive values in three consecutive positions; e.g., for {`7, -3, 4, 5, 6, 9, -2, -4, -3, -2, -1, 11, 0`} it will return 2.

15. Write a class method that receives an array of integers `v` and a non-negative integer `x`, and returns if the sum of the elements of the array is greater than `x`

16. Write a class method that receives an array of *non-negative* integers `v` and a non-negative integer `x`, and returns if the sum of the elements of the array is greater than `x`; **check the least number of positions of `v` that are needed**

17. Write a class method that receives an array of integers `v` and two integers `x` and `n`, and returns the first position of the subsequence of `n` consecutive values greater than `x`, or -1 when that subsequence is not present

18. Write a class method that receives an array of integers `v` and returns how many positions with zero are at the end of the array, using the lowest possible number of operations

19. Write a class method that receives an array of integers `v` and returns the position of the last odd element in `v` (or -1 if there are no odd elements)

20. Write a class method that receives an array of integers `v` and returns the sum of all the elements that appear after the first odd element

21. Write a class method that receives two arrays of real numbers of the same length and returns the scalar product of the vectors represented by the arrays; remember that the scalar product of two vectors v and w, $v, w \in \mathbb{R}^n$, is defined by $v \cdot w = \sum_{i=1}^{n} v_i \cdot w_i$

22. Write a class method that receives two arrays of integers and returns another array which is the sum of the two passed arrays

23. Write a class method that receives an array of words (`String`) and returns whether it is palindrome or not, i.e., the first and last word are the same, and the second and before-last, and the third and the two-before-last, etc.; it must return a `boolean` value

24. Implement a class method that receives two `double` arrays and returns whether the first one is a prefix of the second, i.e., all elements of the first are in the same order at the beginning of the second; sizes of the array could be anyone, i.e., it is possible that the first array is shorter, equal length, or longer than the second one

25. Write a class method that receives an array of `char` and substitutes each occurrence of the sequence "not" with "yes"

26. Implement a class method with header:

```
public static void sumInB(int [] a, int [] b)
```

which modifies in `b` as many components as possible, by summing to them the value in the equivalent position in `a`; i.e., if `n` is the minimum between `a.length` and `b.length`, the first `n` positions of `b` must be modified by adding the corresponding value in `a`

27. Given an array of integers `a`, with `a.length`> 0, where its elements are all between 0 and 9 (included), write a Java method with the following header:

```
public static void digits(int [] a)
```

that **writes on the standard output** the first elements of the array where there are no repeated consecutive elements, e.g.:

- When `a` is `{8,8,4,3}`, the result is `8`
- When `a` is `{4,0,5,9,9}`, the result is `4059`
- When `a` is `{0,9,4,5,9}` the result is `09459`
- When `a` is `{1,7,1,0,0,8,7}` the result is `1710`

28. Write a different version of the previous method with header:

```
public static int digits(int [] a)
```

where `a` is an array with the same features, and the method **calculates and returns** an integer number whose digits are the first elements of the array where there are no repeated consecutive elements, e.g.:

- When `a` is `{1,3,4,4,0,5,1}`, returns `134`
- When `a` is `{0,4,2,8,8,7}`, returns `428`
- When `a` is `{8,7,8,5,5}`, returns `8785`
- When `a` is `{8,8,4,3}`, returns `8`

29. Given an array of integer numbers, design a Java method with the following header:

```
public static int sum(int [] a)
```

that returns the sum of the elements that are symmetric and equals in the array `a`. In the case there is an odd number of elements the central element is considered as symmetric and equal to itself. For example:

- When `a` is `{1,2,3,2,1}` returns `9`
- When `a` is `{1,2,3,2,5}` returns `7`
- When `a` is `{1,2,3,5,1}` returns `5`
- When `a` is `{1,2,3,2}` returns `0`
- When `a` is `{1,2,3,1}` returns `2`

30. Generalise the method of the previous exercise with the following header:

```
public static int sum(int [] a, int i, int j)
```

that returns the sum of the elements that are symmetric and equals in the array `a` and are between positions `i` and `j` ($0 \leq$`i`$\leq$`j`$<$`a.length`). In the case there is an odd number of elements in the range [`i`,`j`], the central element is considered as symmetric and equal to itself. For example, given that `i=1` and `j=3`:

- When `a` is `{1,2,3,2,1}` returns `7`

- When `a` is `{1,2,3,2,5}` returns `7`
- When `a` is `{1,2,3,5,1}` returns `3`
- When `a` is `{1,2,3,2}` returns `7`
- When `a` is `{1,2,3,1}` returns `3`

31. Write a Java method with the following header:

```
public static boolean detect(char [] s1, char [] s2)
```

whose parameters are two arrays of char and returns a `boolean` value. It returns `true` when the sequence of char stored in `s1` is in `s2`, although is not present as an unique block, i.e., in can be fragmented but in the same order. For example, "Castor" is present in "Yesterday in **Cas**ablanca was s**tor**my" and in "**C**up ch**a**mpion**s** will no**t** give up s**o** soon, ne**r**d!", but not in "Yesterday was stormy in Casablanca"

32. The following class defines a track for a musical CD:

```
public class Track
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`Track(java.lang.String n, int m, int s)` Constructor: creates a `Track` object with a name and a duration in minutes and seconds

Method Summary

Methods

Modifier and Type	Method and Description
`int`	`getMinutes()` Consultor: retrieves minutes of the track
`int`	`getSeconds()` Consultor: retrieves seconds of the track
`java.lang.String`	`getTname()` Consultor: retrieves name of the track
`void`	`setMinutes(int m)` Modifier: changes minutes of the track (if they are non-negative)
`void`	`setSeconds(int s)` Modifier: changes seconds of the track (if they are non-negative)
`void`	`setTname(java.lang.String n)` Modifier: changes name of the track

Implement the class `MusicCD` that:

- Stores as attributes the name of the CD record (`String`) and an array of `Track` objects
- Implement a constructor that receives the name of the record and the number of tracks
- Implement a method that adds a track, receiving as parameters the track position (must be validated), the name of the track, and the minutes and seconds of the track (supposed to be valid values)

- Implement a method that returns the longest track of the CD
- Implement a method that returns the total duraction of the CD in a `String` with format "hh:ss"

33. A polygon can be defined as set of `Point` objects that can be stored in an array. Implement a Java class `Polygon` that defines a polygon by using the following features:
    - An attribute array of `Point` (last point is suposed to be connected to first point) and an integer attribute that tells how many `Point` have been actually defined in the polygon
    - A constructor that receives the number of vertexes of the polygon; the initial number of defined `Point` must be initialised to zero
    - A method that receives and adds a new vertex (in case not all the vertexes were already defined); it must be added to the position given by the current number of vertexes, that must be properly incremented
    - Implement a method that returns the perimeter of the polygon (in case it is fully defined, otherwise if must return 0)
    - Implement a method that returns if the polygon has any concave vertex (i.e., the angle formed by the sides that join in that vertex is higher than 180º); in case it is not fully defined, return `false`
34. Given the following `News` class:

```
public class News
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`News(java.lang.String t)` Constructor: creates a `News` object with the given text and an initial number of 0 visits and comments

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`comment()` Increment number of comments in one
`int`	`getComments()` Consultor: retrieves number of comments
`java.lang.String`	`getText()` Consultor: retrieves text of the news
`int`	`getVisits()` Consultor: retrieves number of visits (times that the news was read)
`void`	`read()` Increment number of visits in one
`void`	`setComments(int c)` Modifier: changes number of comments of the news
`void`	`setText(java.lang.String t)` Modifier: changes text of the news
`void`	`setVisits(int v)` Modifier: changes number of visits of the news

implement the `Newspaper` class that:

- Stores the date of a newspaper as three integers attributes (that represent day, month, and year), an attribute for the news of that day (array of `News`), and an attribute with the total number of news stored (integer)
- Implement a constructor that receives the date (suppose it is valid) and makes the *empty* newspaper (with zero `News`)
- Implement a method that adds a news if possible (maximum number of news is 100), by receiving its text
- Implement a method to remove a news given its position in the array (if position is not valid, a warning message must be shown and no news will be deleted); the rest of `News` must be moved to the initial positions (i.e., if current number of `News` is n, they must be placed in indexes from 0 to $n-1$)
- Implement a method that retrieves the most popular news (i.e., that with highest number of visits).

35. A cash machine can be represented by an array that stores the amount of coins and notes of each value. Apart from that, the value of each coin and note is stored in another array. For example, a cash machine with 2 coins of 1 euro cent, 5 coins of 20 euro cent, 2 coins of 1 euro, 7 notes of 5 euros, and 3 notes of 50 euros will be represented by:

```
values = {0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500};
amount = {   2,    0,    0,   0,   5,   0, 2, 0, 7,  0,  0,  3,   0,   0,   0};
```

Implement a class `CashMachine` that represents a cash machine by using the following structure:

- Two array attributes that represent the `values` and `amount` arrays
- Implement a constructor that initialises the values and makes available 5 units of each coin and note with a value of 50 euros or less
- Implement a consultor method to get the `values` array
- Implement a method that represents the change operation: it must receive an array with the amounts of coins and notes given by the costumer (with the same structure that the `amount` attribute) and another value with the real amount to be paid, and it must return another array with the corresponding change (with the same format); in case no change is possible with the current amounts, the original array is returned; internal values for `amount` must be properly updated for each change operation.

36. Define a class `Parking` that represents a parking lot. The class is composed of the following attributes:

- An array of `Car` objects (a position is `null` when no car is occupying the space)
- A fare per minute (positive real)

Define the car class with:

- Attributes for the plate number (string) and the hour and minute they entered into the parking lot (two integers)
- A constructor with all the parameters needed for the attributes; you can suppose that all parameters are correct
- The consultors and modifiers for all attributes (you can suppose that parameters in modifiers are always valid)

The parking class must include:

- A constructor that receives the number of spaces, the fare per minute, and makes the parking lot empty; you can suppose that all parameters are valid values
- A method that receives a car and situates it in the first free space; if no free spaces are available, it must return `false`, otherwise return `true`
- A method that receives a car and a space number and situates the car in that space if empty; otherwise, it must inform the user and situate it in the first free space (if no free spaces available, inform the user); the method must return `true` if the car was inserted in the demanded space, and `false` otherwise
- A method that extracts a car from a given space in a given hour and minute and returns the total fare for that car; you must check that the space is not empty (in that case, it must return 0); you can suppose that given hour and minute are posterior to that when the car entered the parking
- A method to return the place number of a car given its plate number (-1 if the car is not present)

37. What calculates the following method, where `m` is supposed to store a square matrix?

```
public static int calculate(int [][] m) {
     boolean found = false;
     int n = m.length, i = 0;
     while (i < n && !found) {
          int s = 0;
          for (int j = 0; j < n - 1; j++) s = s + m[i][j];
          if (m[i][n - 1] <= s) found = true;
          else i++;
     }
     return i;
}
```

38. Implement a class method that receives two `double` matrixes `a` and `b`, of sizes `m`×`n` and `n`×`p`, respectively, and returns the product matrix

39. Implement a class method that receives a `double` square matrix and returns the product of all the elements of its main diagonal.

40. Implement a class method that receives a `double` matrix and returns an array which contains the sum of each column of the matrix.

41. Implement a class method that receives a `double` matrix and returns an array which contains the maximum elements for each row of the matrix.

42. Implement a class method that receives a `double` matrix `a` of size `n`×`m` and returns the transpose matrix `b` with size `m`×`n`; the transpose matrix is that resulting from interchanging rows and columns

43. Implement a class method that receives a `double` square matrix `a` of size `n`×`n`, and modifies and returns `a` to be its transpose matrix (no auxiliar matrixes must be employed)

44. Implement a class method that receives a `double` square matrix `a` of size `n`×`n` and returns whether it is symmetric or not (with respect its main diagonal)

45. Implement a class method that receives a `double` matrix `a` and returns whether it is any element in position `[i][j]` such that is equal to the sum of all the elements of the submatrix from `[0][0]` to `[i-1][j-1]`

46. Implement a class method that receives a `double` square matrix `a` of size `n`×`n` and returns whether it is strictly dominant by rows, i.e., for each row, the element in the diagonal is strictly greater than the sum of the absolute value of the rest of the elements of the row

47. Implement a class method that receives a `double` square matrix `a` of size `n`×`n` and returns its 1-norm, i.e., the maximum for the sum of the absolute values of all the elements of the columns of the matrix

48. Implement a class method that receives a `double` square diagonal matrix `a` (it only has non-zero elements in the diagonal) and an integer `x`, and returns the result of multiplying `x` by `a`; you must use the lowest possible number of operations

49. Implement a class method that receives a `double` matrix and a pair of integers (position), and returns the sum of the 3×3 submatrix centered in the position given. If the position is on any border of the matrix, the submatrix must be restricted to take only positions that really exists in the matrix.

50. Implement a class method that receives a `double` square matrix and returns `true` if it is lower triangular (i.e., all elements over the main diagonal are 0) or `false` otherwise.

51. Implement a class method that receives a `double` bidimensional array `a` and returns whether it is a matrix (all rows have the same number of columns) or not

52. Implement a class method that receives a `double` bidimensional array `a` and returns the array of the row with maximum sum of its elements

53. Implement a class method that receives a `double` bidimensional array `a` and returns an array with the column with minimum sum of its elements; elements not present sum as 0

54. Implement an utility class `BidimensionalArrayUtil` that implements the methods of the exercises from 38 to 53

55. Implement a method that, given an array of char `word` and a matrix of char `m` where the size of all the rows is equal to the length of `word`, iteratively searchs if `word` is equal, char by char, to any row of `m`. The method must return the index of the first row where there is the coincidence (lowest index). Otherwise, it must return -1. Unnecessary comparisons must be avoided

56. Given the following implementation of the `Image` class, that allows to manage images in grayscale:

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.awt.*;
import javax.swing.*;

public class Image {

    // Matrix that stores the image in gray level format (0-255)
    // 0 = black, 255 = white
    private int [][] img;

    // Constructor: create grayscale image from JPG file (parameter)
    public Image(String fileName) {
        try {
            BufferedImage imageRGB = ImageIO.read(new File(fileName));
            BufferedImage image = getGrayScale(imageRGB);
            int height = image.getHeight();
            int width = image.getWidth();
            img = new int[height][width];

            for(int i = 0; i < height; i++) {
                for(int j = 0; j < width; j++) {
                    int pix = image.getRGB(j,i);
                    Color c = new Color(pix);
                    int R = c.getRed(); int G = c.getGreen(); int B = c.getBlue();
                    img[i][j] = (int) ((R + G + B) / 3);
                }
            }
        }
        catch (Exception e) {
            img = null;
        }
    }

    // Show image
    public void show() {

        // Create image from integer matrix
        BufferedImage image = convert();

        // Show image
        JFrame myWindow = new JFrame();
        myWindow.setTitle("Image");
        JLabel label = new JLabel(new ImageIcon(image));
        myWindow.add(label);
        myWindow.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        myWindow.pack();
        myWindow.setVisible(true);
```

```
    }

    // Save in JPG format (filename given) the grayscale image
    // Return true when saving was successful, false otherwise
    public boolean save(String fileName) {

        // Create image from integer matrix
        BufferedImage image = convert();

        // Save image
        try {
            File f = new File(fileName);
            return ImageIO.write(image, "jpg", f);
        }
        catch (IOException e) {
            System.out.println("Error saving image " + fileName);
            System.out.println(e.getMessage());
        }

        return false;
    }

    // Converts color image in grayscale image (private class method)
    private static BufferedImage getGrayScale(BufferedImage inputImage){
        BufferedImage img = new BufferedImage(inputImage.getWidth(), inputImage.getHeight(),
                                              BufferedImage.TYPE_BYTE_GRAY);
        Graphics g = img.getGraphics();
        g.drawImage(inputImage, 0, 0, null);
        g.dispose();
        return img;
    }

    // Converts matrix of int into image (private method)
    private BufferedImage convert() {
        // Create image from integer matrix
        int width = img[0].length;
        int height = img.length;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);

        for(int i=0; i<height; i++) {
            for(int j=0; j<width; j++) {
                int value = img[i][j] << 16 | img[i][j] << 8 | img[i][j];
                image.setRGB(j, i, value);
            }
        }

        return image;
    }

}
```

Implement a new constructor that receives the width and height of the image and creates a random grayscale image (i.e., each pixel in the matrix is a random value between 0 and 255, both included)

57. Implement a method in the `Image` class that converts the grayscale image into its negative

58. Implement a method in the `Image` class that, given an integer value t (you can suppose $0 \leq t \leq 255$), binarises the image according to this threshold value (i.e., pixels with values lower than t will take value 0 and rest of pixels will take value 255)

59. Implement a method in the `Image` class that obtains the symmetric image (i.e., mirror with respect to vertical axis)

60. Implement a method in the `Image` class that obtains the upside down image (i.e., mirror with respect to the horizontal axis)

61. Implement a method in the `Image` class that, given a positive integer number f, adds a black frame of width f to the image (i.e., all positions that are f pixels or less away from the border take a black value, 0)

62. Implement a method in the `Image` class that, given a position in the image (two integers) and a positive integer c, puts a black square whose upper leftmost position is the given position and its side length is c; be careful to not exceed the borders of the matrix

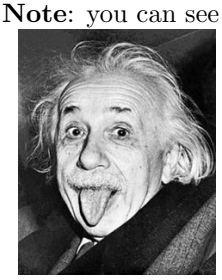
63. Implement a method in the `Image` class that reduces its side to the half for both width and height; you must create a new matrix that will substitute the previous one; the value for each pixel must be calculated as the average of the four pixels that correspond in the original image

64. Implement a method in the `Image` class that increases its side to the double for both width and height; you must create a new matrix that will substitute the previous one; the value for each pixel must be calculated as the value of the original pixel that corresponds in the original image

65. Implement a method in the `Image` class that rotates 90º clockwise the image; you must create a new matrix that will substitute the previous one

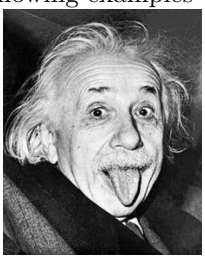
66. Implement a method in the `Image` class that rotates 90º counterclockwise the image; you must create a new matrix that will substitute the previous one

Note: you can see the effect of the different operations in the following examples

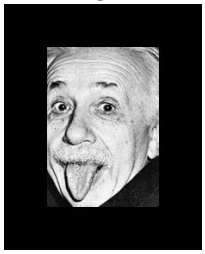

Original

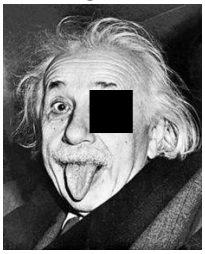

Negative

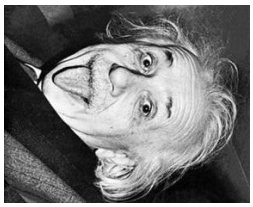
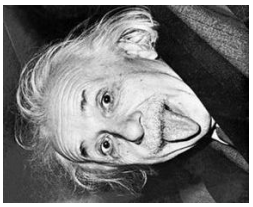

Binary ($t = 100$)


Symmetric


Upside-down


Frame ($f = 50$)


Square ((100,100),50)



Clockwise/Counterclockwise rotation

67. Given the following class `Set`:

```
public class Set {
    private boolean [] set;
    private int last;
    /** Constructor of empty set, which will
      * contain natural numbers in the interval [0..l].  */
    public Set(int l) {
        set = new boolean[l + 1];
        last = l;
    }
    /** Check pertaining to the set of
      * a given x , 0<=x<=last. */
    public boolean pertain(int x) { return set[x]; }
```

```
    /** Gives cardinal of the set. */
    public int cardinal() {
        int card = 0;
        for (int i = 0; i < set.length; i++)
            if (set[i]) card++;
        return card;
    }
    /** Add to the set a given x, 0<=x<=last. */
    public void add(int x) { set[x] = true; }
    /** Generate random elements in the current set. */
    public void randomSet() {
        for (int i = 0; i < set.length; i++)
            set[i] = (Math.random() >= 0.5);
    }
    ...
}
```

Add the methods that allow to:

(a) Return the set resulting of the union of the current set (`this`) and another given set

(b) Return the set resulting of the intersection of the current set (`this`) and another given set

(c) Return the set resulting of the difference of the current set (`this`) and another given set

(d) Return the complementary set of the current set (`this`)

68. Supose that the mean daily temperatures of a city during a year have been measured and stored in a bidimensional array `meanTemp` defined as follows:

```
final int[] NUM_DAYS = {0,31,28,31,30,31,30,31,31,30,31,30,31}
// NUM_DAYS[i] = number of days for each month, 1<=i<=12
// NUM_DAYS[0] = 0 (no month associated)

double [][] meanTemp = new double[13][];
// meanTemp[i] represents month, 1<=i<=12
// meanTemp[0] must be null (no month associated)

for (int i = 1; i < meanTemp.length; i++)
        meanTemp[i] = new double[NUM_DAYS[i] + 1];
// number of elements of meanTemp[i] is NUM_DAYS[i]+1, 1<=i<=12
```

Modify the following method:

```
public static int mode0ToN(int[] a, int n) {
    // Build array between 0 and n
    int [] frequency = new int[n + 1];
    // Listing a and obtaining frequencies
    for (int i = 0; i < a.length; i++) frequency[a[i]]++;
    // The mode is the index of the maximum of the frequency array
    int mode = 0;
    for (int i = 1; i < frequency.length; i++)
        if (frequency[i] > frequency[mode]) mode = i;
    return mode;
}
```

to show on the screen the histogram of frequence of temperatures:

(a) Knowing that minimum temperature was 0 degress and maximum was 38. The histogram must contain 39 lines with the temperatures whose integer part are from 0 to 38, and the number of days in which that temperature was reached

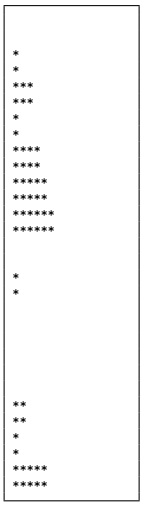
(b) When no maximum nor minimum temperatures are known in advance and they can be negative

69. Given a sequence of words stored in an array of `String`, write class methods that:

(a) Calculate the histogram of the lengths of the words in the array; use an array `hist` with 15 counters, such that `hist[i]` contains the number of words with length `i+1` that were in the text, for $0 \leq$`i`$<$ 14, and `hist[14]` contains the number of words with length greater than or equal to 15

(b) Show the histogram on the screen by using asterisks (`*`): for each length `i`, two lines of asterisks with as many asterisks as `hist[i-1]` stores; for example, for `hist={0,1,3,1,4,5,6,0,1,0,0,0,2,1,5}` it must show the figure in the right side

(c) Calculate the mode of the lengths of the words of the sequence (i.e., the most frequent length)



70. Given the following class:

```
/** Class CircleSeq
 * Atributtes:
 * - size, number of circles
 * - theArray, where circles get stored
 * - MAX_ARRAY, maximum number of circles
 */
public class CircleSeq {
    private int size;
    private Circle [] theArray;
    private static final int MAX_ARRAY = 10;
    /** Constructor: empty sequence. */
    public CircleSeq() {
        theArray = new Circle[MAX_ARRAY];
        size = 0;
    }
    /** Add a cicle at the end.
     * @param c Circle to add.
     * @return boolean true if successful
     *                 false if array is full.
     */
    public boolean insert(Circle c) {
        boolean er = true;
        if (size < theArray.length) theArray[size++] = c;
        else er = false;
        return er;
    }
    /** Retrieves circle in the given position.
     * @param pos position (first position is 0).
     * @return Circle circle in that position or null
     *                 if pos<0 || pos>=size.
     */
    public Circle retrieve(int pos) {
        if (pos >= 0 && pos < size) return theArray[pos];
        return null;
    }
    /** Consult number of circles.
     * @return int number of circles.
     */
    public int size() { return size; }
```

```
  /** Erases first circle of color col.
   * @param col String color of circle to erase.
   * @return boolean true if successful
   *          false if not circle of that color.
   */
  public boolean erase(String col) {
      boolean ok = false;
      for (int i = 0; i < size && !ok; i++)
        if (theArray[i].getColor().equals(col)) {
          ok = true;
          for(int j = i; j < size - 1; j++)
            theArray[j] = theArray[j + 1];
          size--;
        }
      return ok;
  }
  /** Returns position of first circle of color col
   * and radius r.
   * @param col String color of circle.
   * @param r double radius of circle.
   * @return int position of circle (first circle in pos 0)
   *              or -1 if not exists.
   */
  public int indexIf(String col, double r) {
      int pos = -1;
      for (int i = 0; i < size && pos == -1; i++)
        if (theArray[i].getColor().equals(col) &&
            theArray[i].getRadio() == r) pos = i;
      return pos;
  }
  /** Returns sums of areas of all the circles.
   * @return double sum of the areas of all circles.
   */
  public double area() {
      double res = 0.0;
      for (int i = 0; i < size; i++) res += theArray[i].area();
      return res;
  }
  /** Retuns a description of the circles.
   * @return String string with the description.
   */
  public String toString(){
      String res = "Sequence of " + size + " circles:\n";
      if (size > 0)
          for(int i = 0; i < size; i++)
              res += theArray[i].toString() + "\n";
      else res = "Empty sequence\n";
      return res;
  }
}
```

Add the following methods:

(a) Modifier that erases (if exists) a `Circle` object given as parameter. It will return `true` if it exists and `false` otherwise

(b) Consultor that returns the position of a `Circle` object given as parameter. It will return -1 if the object is not present.

(c) Method that draws all the `Circle` objects in `theArray` in a `Blackboard` object; supose the two classes are in the same package