Backend 1

Arreglos

Ejercicios

extras

# Ejercicios extras

Estos ejercicios son para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado con los ejercicios prácticos del día y aún no haya finalizado el encuentro. Recuerda que no es necesario que termines estos ejercicios extra para continuar con lo siguiente.

1. Realizar un algoritmo que calcule la suma de todos los elementos de un vector de tamaño N, con los valores ingresados por el usuario.
2. /\*Ejercicios Extras-
3. \*1./Realizar un algoritmo que calcule la suma de todos los elementos de un vector de tamaño N, con los valores ingresados por el usuario.
4. \*/
5. package sumavector;
6. // @author Horacio
7. import java.util.Scanner; // Importamos la clase Scanner para leer la entrada del usuario
8. public class SumaVector {
9. public static void main(String[] args) {
10. try (Scanner input = new Scanner(System.in) // Creamos una instancia de Scanner para leer la entrada del usuario
11. ) {
12. System.out.print("Ingrese el tamaño del vector: ");
13. int N = input.nextInt(); // Leemos el tamaño del vector ingresado por el usuario
14. int[] vector = new int[N]; // Creamos un arreglo de enteros con el tamaño N especificado
15. System.out.println("Ingrese los elementos del vector: ");
16. for (int i = 0; i < N; i++) { // Usamos un ciclo for para iterar sobre el arreglo y leer los elementos ingresados por el usuario
17. vector[i] = input.nextInt(); // Leemos cada elemento y lo almacenamos en el arreglo
18. }   int suma = 0; // Inicializamos una variable para almacenar la suma de los elementos del vector
19. for (int i = 0; i < N; i++) { // Usamos otro ciclo for para iterar sobre el arreglo y sumar sus elementos
20. suma += vector[i]; // Sumamos cada elemento al total acumulado en la variable suma
21. }   System.out.println("La suma de los elementos del vector es: " + suma); // Mostramos el resultado de la suma
22. // Cerramos el Scanner para liberar recursos
23. }
24. }
25. }
26. /\*
27. El algoritmo utiliza la clase Scanner de Java para leer la entrada del usuario desde la consola.
28. El usuario debe ingresar primero el tamaño del vector, y luego los elementos del vector uno por uno.
29. Luego, se utiliza un ciclo for para iterar sobre el arreglo y sumar sus elementos,
30. y finalmente se muestra el resultado de la suma en la consola.
31. \*/
32. Escriba un programa que averigüe si dos vectores de N enteros son iguales (la comparación deberá detenerse en cuanto se detecte alguna diferencia entre los elementos).
33. /\* Extras Guia5
34. \*2.-Escriba un programa que averigüe si dos vectores de N enteros son iguales
35. (la comparación deberá detenerse en cuanto se detecte alguna diferencia entre los elementos).
36. \*/
37. package compararvectores;
38. // @author Horacio
39. import java.util.Scanner; // Importamos la clase Scanner para leer la entrada del usuario
40. public class CompararVectores {
41. public static void main(String[] args) {
42. Scanner input = new Scanner(System.in); // Creamos una instancia de Scanner para leer la entrada del usuario
43. System.out.print("Ingrese el tamaño de los vectores: ");
44. int N = input.nextInt(); // Leemos el tamaño de los vectores ingresado por el usuario
45. int[] vector1 = new int[N]; // Creamos un arreglo de enteros para el primer vector con el tamaño N especificado
46. int[] vector2 = new int[N]; // Creamos un arreglo de enteros para el segundo vector con el tamaño N especificado
47. System.out.println("Ingrese los elementos del primer vector: ");
48. for (int i = 0; i < N; i++) { // Usamos un ciclo for para iterar sobre el primer vector y leer los elementos ingresados por el usuario
49. vector1[i] = input.nextInt(); // Leemos cada elemento del primer vector y lo almacenamos en el arreglo
50. }
51. System.out.println("Ingrese los elementos del segundo vector: ");
52. for (int i = 0; i < N; i++) { // Usamos otro ciclo for para iterar sobre el segundo vector y leer los elementos ingresados por el usuario
53. vector2[i] = input.nextInt(); // Leemos cada elemento del segundo vector y lo almacenamos en el arreglo
54. }
55. boolean sonIguales = true; // Inicializamos una variable booleana para indicar si los vectores son iguales o no
56. for (int i = 0; i < N; i++) { // Usamos otro ciclo for para comparar los elementos de los dos vectores
57. if (vector1[i] != vector2[i]) { // Comparamos los elementos en la misma posición de los dos vectores
58. sonIguales = false; // Si encontramos una diferencia, actualizamos la variable sonIguales a false
59. break; // Detenemos la comparación en cuanto se detecte alguna diferencia entre los elementos
60. }
61. }
62. if (sonIguales) {
63. System.out.println("Los vectores son iguales."); // Si la variable sonIguales es true, mostramos que los vectores son iguales
64. } else {
65. System.out.println("Los vectores son diferentes."); // Si la variable sonIguales es false, mostramos que los vectores son diferentes
66. }
67. input.close(); // Cerramos el Scanner para liberar recursos
68. }
69. }
70. /\*
71. En resumen, este programa solicita al usuario el tamaño de dos vectores, lee los elementos de ambos vectores ingresados por el usuario,
72. los almacena en arreglos de enteros, los compara elemento por elemento utilizando un ciclo for, y muestra si los vectores son iguales o diferentes en la consola.
73. La clase Scanner se utiliza para leer la entrada del usuario, y se cierra al final del programa para liberar recursos.
74. \*/
75. Crear una función rellene un vector con números aleatorios, pasándole un arreglo por parámetro. Después haremos otra función o procedimiento que imprima el vector.
76. /\* Extras Guia 5
77. \*3.-Crear una función rellene un vector con números aleatorios, pasándole un arreglo por parámetro.
78. Después haremos otra función o procedimiento que imprima el vector.
79. \*/
80. package vectoraleatorio;
81. // @author Horacio
82. import java.util.Random; // Importamos la clase Random para generar números aleatorios
83. public class VectorAleatorio {
84. public static void main(String[] args) {
85. int N = 10; // Tamaño del vector
86. int[] vector = new int[N]; // Creamos un arreglo de enteros con el tamaño N especificado
87. llenarVectorAleatorio(vector); // Llamamos a la función llenarVectorAleatorio para llenar el vector con números aleatorios
88. imprimirVector(vector); // Llamamos a la función imprimirVector para imprimir el vector lleno
89. }
90. // Función para llenar un vector con números aleatorios
91. public static void llenarVectorAleatorio(int[] vector) {
92. Random rand = new Random(); // Creamos una instancia de la clase Random para generar números aleatorios
93. for (int i = 0; i < vector.length; i++) { // Usamos un ciclo for para iterar sobre el vector
94. vector[i] = rand.nextInt(101); // Generamos un número aleatorio entre 0 y 100, y lo almacenamos en la posición i del vector
95. }
96. }
97. // Función para imprimir un vector
98. public static void imprimirVector(int[] vector) {
99. System.out.println("Elementos del vector:");
100. for (int i = 0; i < vector.length; i++) { // Usamos un ciclo for para iterar sobre el vector
101. System.out.println(vector[i]); // Imprimimos el elemento en la posición i del vector
102. }
103. }
104. }
105. /\*
106. En resumen, este programa crea una función llamada llenarVectorAleatorio que recibe un arreglo de enteros como parámetro
107. y lo llena con números aleatorios entre 0 y 100 utilizando la clase Random.
108. Luego, tiene otra función llamada imprimirVector que recibe un arreglo de enteros como parámetro y lo imprime en la consola.
109. En la función main, se crea un vector de tamaño N, se llama a la función llenarVectorAleatorio para llenar el vector con números aleatorios
110. y se llama a la función imprimirVector para imprimir el vector lleno.
111. \*/
112. Los profesores del curso de programación de Egg necesitan llevar un registro de las notas adquiridas por sus 10 alumnos para luego obtener una cantidad de aprobados y desaprobados. Durante el período de cursado cada alumno obtiene 4 notas, 2 por trabajos prácticos evaluativos y 2 por parciales. Las ponderaciones de cada nota son:

**Primer trabajo práctico evaluativo 10%**

**Segundo trabajo práctico evaluativo 15%**

**Primer Integrador 25%**

**Segundo integrador 50%**

Una vez cargadas las notas, se calcula el promedio y se guarda en el arreglo. Al final del programa los profesores necesitan obtener por pantalla la cantidad de aprobados y desaprobados, teniendo en cuenta que solo aprueban los alumnos con promedio mayor o igual al 7 de sus notas del curso.

\*Extras guia5

4.-Los profesores del curso de programación de Egg necesitan llevar un registro de las notas adquiridas

por sus 10 alumnos para luego obtener una cantidad de aprobados y desaprobados.

Durante el período de cursado cada alumno obtiene 4 notas, 2 por trabajos prácticos evaluativos y 2 por parciales.

Las ponderaciones de cada nota son:

Primer trabajo práctico evaluativo 10%

Segundo trabajo práctico evaluativo 15%

Primer Integrador 25%

Segundo integrador 50%

Una vez cargadas las notas, se calcula el promedio y se guarda en el arreglo.

Al final del programa los profesores necesitan obtener por pantalla la cantidad de aprobados y desaprobados,

teniendo en cuenta que solo aprueban los alumnos con promedio mayor o igual al 7 de sus notas del curso.

 \*/

package registronotasalumnos;

// @author Horacio

 import java.util.Scanner; // Importar la clase Scanner para leer entrada del usuario

public class RegistroNotasAlumnos {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in); // Crear una instancia de la clase Scanner para leer entrada del usuario

        int cantidadAlumnos = 10; // Cantidad de alumnos

        double[][] notas = new double[cantidadAlumnos][4]; // Crear un arreglo bidimensional para almacenar las notas de los alumnos

        double[] promedios = new double[cantidadAlumnos]; // Crear un arreglo para almacenar los promedios de los alumnos

        int aprobados = 0; // Inicializar contador de aprobados en 0

        // Ciclo para ingresar las notas de los alumnos

        for (int i = 0; i < cantidadAlumnos; i++) {

            System.out.println("Ingrese las notas del alumno " + (i + 1) + ":");

            for (int j = 0; j < 4; j++) {

                System.out.print("Nota " + (j + 1) + ": ");

                notas[i][j] = sc.nextDouble(); // Leer la nota del usuario y guardarla en el arreglo de notas

            }

            promedios[i] = calcularPromedio(notas[i]); // Calcular el promedio de las notas del alumno y guardarlo en el arreglo de promedios

            if (promedios[i] >= 7) { // Verificar si el alumno aprobó

                aprobados++; // Incrementar el contador de aprobados

            }

        }

        int desaprobados = cantidadAlumnos - aprobados; // Calcular la cantidad de desaprobados

        // Mostrar resultados

        System.out.println("Cantidad de aprobados: " + aprobados);

        System.out.println("Cantidad de desaprobados: " + desaprobados);

    }

    // Método para calcular el promedio de un arreglo de notas

    public static double calcularPromedio(double[] notas) {

        double sum = 0;

        for (int i = 0; i < notas.length; i++) {

            sum += notas[i]; // Sumar todas las notas del arreglo

        }

        return sum / notas.length; // Calcular el promedio dividiendo la suma de las notas por la cantidad de notas

    }

}

/\*

 Se utiliza un arreglo bidimensional notas para almacenar las notas de los 10 alumnos,

donde cada fila representa las notas de un alumno y cada columna representa una nota específica

(2 por trabajos prácticos evaluativos y 2 por parciales).

Luego, utiliza un arreglo promedios para almacenar los promedios de notas de cada alumno.

Además utiliza un ciclo for anidado para ingresar las notas de los alumnos a través de la entrada del usuario,

calcular los promedios de notas y contar la cantidad de aprobados.

Luego, calcula la cantidad de desaprobados restando la cantidad de aprobados del total de alumnos.

Al final, muestra por pantalla la cantidad de aprobados y desaprobados obtenidos.

\*/

1. Realizar un programa que llene una matriz de tamaño NxM con valores aleatorios y muestre la suma de sus elementos.
2. /\*
3. 5.-Realizar un programa que llene una matriz de tamaño NxM con valores aleatorios y muestre la suma de sus elementos.
4. \*/
5. package matrizsuma;
6. // @author Horacio
7. import java.util.Random; // Importar la clase Random para generar números aleatorios
8. public class MatrizSuma {
9. public static void main(String[] args) {
10. int N = 3; // Número de filas de la matriz
11. int M = 4; // Número de columnas de la matriz
12. int[][] matriz = new int[N][M]; // Crear una matriz de NxM para almacenar los valores aleatorios
13. Random rand = new Random(); // Crear un objeto de la clase Random para generar números aleatorios
14. // Llenar la matriz con valores aleatorios entre 1 y 10
15. for (int i = 0; i < N; i++) {
16. for (int j = 0; j < M; j++) {
17. matriz[i][j] = rand.nextInt(10) + 1; // Generar un número aleatorio entre 1 y 10 y asignarlo a la posición (i,j) de la matriz
18. }
19. }
20. int suma = 0; // Variable para almacenar la suma de los elementos de la matriz
21. // Calcular la suma de los elementos de la matriz
22. for (int i = 0; i < N; i++) {
23. for (int j = 0; j < M; j++) {
24. suma += matriz[i][j]; // Sumar el valor de cada elemento de la matriz a la variable "suma"
25. }
26. }
27. // Imprimir la matriz generada
28. System.out.println("Matriz generada:");
29. for (int i = 0; i < N; i++) {
30. for (int j = 0; j < M; j++) {
31. System.out.print(matriz[i][j] + " "); // Imprimir el valor de cada elemento de la matriz separado por espacio
32. }
33. System.out.println(); // Imprimir una nueva línea después de cada fila de la matriz
34. }
35. // Imprimir la suma de los elementos de la matriz
36. System.out.println("Suma de los elementos de la matriz: " + suma);
37. }
38. }
39. /\*
40. Se declara una clase llamada "MatrizSuma" que contiene el método "main" como punto de entrada del programa.
41. Se definen las variables "N" y "M" para representar el número de filas y columnas de la matriz, respectivamente.
42. Se crea una matriz bidimensional llamada "matriz" de tamaño NxM para almacenar los valores aleatorios.
43. Se crea un objeto de la clase "Random" para generar números aleatorios.
44. Se utiliza un bucle "for" anidado para llenar la matriz con valores aleatorios entre 1 y 10.
45. Se utiliza otro bucle "for" anidado para calcular la suma de todos los elementos de la matriz y se almacena en la variable "suma".
46. Se utiliza otro bucle "for" anidado para imprimir por pantalla la matriz generada.
47. Se imprime por pantalla la suma de los elementos de la matriz.
48. \*/
49. Construya un programa que lea 5 palabras de mínimo 3 y hasta 5 caracteres y, a medida que el usuario las va ingresando, construya una “sopa de letras para niños” de tamaño de 20 x 20 caracteres. Las palabras se ubicarán todas en orden horizontal en una fila que será seleccionada de manera aleatoria. Una vez concluida la ubicación de las palabras, rellene los espacios no utilizados con un número aleatorio del 0 al 9. Finalmente imprima por pantalla la sopa de letras creada.

**Nota:** Para resolver el ejercicio deberá investigar cómo se utilizan las siguientes funciones de Java substring(), Length() y Math.random().

/\*

 \*6.-Construya un programa que lea 5 palabras de mínimo 3 y hasta 5 caracteres y, a medida que el usuario las va ingresando, construya una “sopa de letras para niños” de tamaño de 20 x 20 caracteres. Las palabras se ubicarán todas en orden horizontal en una fila que será seleccionada de manera aleatoria. Una vez concluida la ubicación de las palabras, rellene los espacios no utilizados con un número aleatorio del 0 al 9. Finalmente imprima por pantalla la sopa de letras creada.

Nota: Para resolver el ejercicio deberá investigar cómo se utilizan las siguientes funciones de Java substring(), Length() y Math.random().

 \*/

package sopadeletras;

import java.util.Scanner;

public class SopaDeLetras {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String[][] matriz = new String[20][20];

        int cont = 0;

        for (int i = 0; i < 5; i++) {

            cont = 0;

            System.out.println("Ingrese 5 palabras para la sopa de letras:");

            String palabra = sc.nextLine();

            // Validar que la palabra tenga entre 3 y 5 caracteres

            while (palabra.length() < 3 || palabra.length() > 5) {

                System.out.println("Palabra incorrecta. Ingrese una palabra de 3 a 5 caracteres:");

                palabra = sc.nextLine();

            }

            // Generar una fila aleatoria para colocar la palabra en la matriz

            int fila = (int) (Math.random() \* matriz.length);

            while (matriz[fila][0] != null) {

                fila = (int) (Math.random() \* 20);

            }

            // Generar una columna aleatoria para colocar la palabra en la matriz

            int columna = (int) (Math.random() \* 14);

            // Colocar cada letra de la palabra en la matriz consecutivamente

            for (int j = columna; j < (columna + palabra.length()); j++) {

                matriz[fila][j] = palabra.toUpperCase().substring(cont, cont + 1);

                cont++;

            }

            cont = 0;

            // Llenar los espacios vacíos en la fila con números aleatorios del 1 al 9

            for (int j = 0; j < matriz.length; j++) {

                if (matriz[fila][j] == null) {

                    matriz[fila][j] = String.valueOf((int) (Math.random() \* 9 + 1));

                }

            }

        }

        // Llenar cualquier espacio vacío restante en la matriz con números aleatorios del 1 al 9

        for (int i = 0; i < matriz.length; i++) {

            for (int j = 0; j < matriz.length; j++) {

                if (matriz[i][j] == null) {

                    matriz[i][j] = String.valueOf((int) (Math.random() \* 9 + 1));

                }

            }

        }

        // Mostrar la matriz resultante en la consola

        for (int i = 0; i < matriz.length; i++) {

            for (int j = 0; j < matriz.length; j++) {

                System.out.print(matriz[i][j] + "  ");

            }

            System.out.println("");

        }

    }

}

/\*

Se importa la clase Scanner para leer la entrada del usuario.

Se crea un arreglo de Strings llamado "palabras" para almacenar las palabras ingresadas por el usuario.

Se crea un arreglo bidimensional de caracteres llamado "sopaDeLetras" para representar la sopa de letras.

Se solicita al usuario que ingrese 5 palabras de mínimo 3 y máximo 5 caracteres.

Se lee cada palabra ingresada por el usuario y se guarda en el arreglo de palabras.

Se selecciona aleatoriamente una fila donde se ubicarán las palabras en la sopa de letras.

Se itera a través de las palabras ingresadas y se coloca cada letra de la palabra en la fila seleccionada de la sopa de letras.

Se verifica si alguna celda de la sopa de letras está vacía y se llena con un número aleatorio del 0 al 9.

Se imprime la sopa de letras por pantalla.

El programa finaliza.

\*/

1. Realizar un programa que complete un vector con los N primeros números de la sucesión de Fibonacci. Recordar que la sucesión de Fibonacci es la sucesión de los siguientes números:

1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Donde cada uno de los números se calcula sumando los dos anteriores a él. Por ejemplo:

La sucesión del número 2 se calcula sumando (1+1)

Análogamente, la sucesión del número 3 es (1+2),

Y la del 5 es (2+3),

Y así sucesivamente…

La sucesión de Fibonacci se puede formalizar de acuerdo a la siguiente fórmula:

Fibonaccin = Fibonaccin-1 + Fibonaccin-2 para todo n>1

Fibonaccin = 1 para todo n<=1

Por lo tanto, si queremos calcular el término “n” debemos escribir una función que reciba como parámetro el valor de “n” y que calcule la serie hasta llegar a ese valor.

Para conocer más acerca de la serie de Fibonacci consultar el siguiente link: <https://quantdare.com/numeros-de-fibonacci/>

**/\***

**\*7.-Realizar un programa que complete un vector con los N primeros números de la sucesión de Fibonacci.**

**Recordar que la sucesión de Fibonacci es la sucesión de los siguientes números:**

**1, 1, 2, 3, 5, 8, 13, 21, 34, ...**

**Donde cada uno de los números se calcula sumando los dos anteriores a él. Por ejemplo:**

**La sucesión del** número 2 se calcula sumando (1+1)

Análogamente, la sucesión del número 3 es (1+2),

Y la del 5 es (2+3),

Y así sucesivamente…

La sucesión de Fibonacci se puede formalizar de acuerdo a la siguiente fórmula:

Fibonaccin = Fibonaccin-1 + Fibonaccin-2 para todo n>1

Fibonaccin = 1 para todo n<=1

Por lo tanto, si queremos calcular el término “n” debemos escribir una función

que reciba como parámetro el valor de “n” y que calcule la serie hasta llegar a ese valor.

Para conocer más acerca de la serie de Fibonacci consultar el siguiente link: https://quantdare.com/numeros-de-fibonacci/

 \*/

package fibonacci;

// @author Horacio

public class Fibonacci {

    public static void main(String[] args) {

        int n = 10; // Número de términos de Fibonacci a generar

        int[] fibonacci = new int[n]; // Crear un arreglo para almacenar los términos de Fibonacci

        // Calcular la sucesión de Fibonacci

        for (int i = 0; i < n; i++) {

            if (i <= 1) {

                fibonacci[i] = 1; // El primer y segundo término de Fibonacci es 1

            } else {

        fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2]; // Los términos siguientes se calculan sumando los dos anteriores

            }

        }

        // Imprimir los términos de Fibonacci generados

        System.out.println("Los primeros " + n + " números de la sucesión de Fibonacci son:");

        for (int i = 0; i < n; i++) {

            System.out.print(fibonacci[i] + " ");

        }

    }

}

/\*

Se declara una clase llamada "Fibonacci" que contiene el método "main" como punto de entrada del programa.

Se define una variable "n" que representa el número de términos de la sucesión de Fibonacci que se desea generar.

Se crea un arreglo llamado "fibonacci" con un tamaño de "n" para almacenar los términos de Fibonacci generados.

Se utiliza un bucle "for" para calcular los términos de Fibonacci. Si "i" es menor o igual a 1, se asigna el valor 1 al arreglo "fibonacci[i]" ya que los dos primeros términos de Fibonacci son 1. De lo contrario, se suma el valor de los dos términos anteriores (fibonacci[i - 1] y fibonacci[i - 2]) para obtener el siguiente término de Fibonacci.

Se utiliza otro bucle "for" para imprimir los términos de Fibonacci generados por pantalla.

Se imprime por pantalla los términos de Fibonacci generados.

\*/