EJERCICIOS T. 5. Arrays y cadenas

1. Cree dos arrays de enteros. El primero de ellos servirá para albergar un número de mes, y el segundo para albergar el número de días de cada uno de los meses, para un año no bisiesto. Rellene y recorra dichos arrays, mostrando su contenido por pantalla.

2. Realice el mismo ejercicio que en el caso anterior, pero ahora con un único array bidimensional para albergar los datos.

3. Cree una clase para guardar los datos de un mes: nombre, número de mes, días que tiene (año no bisiesto) y temperatura media.
Cree un array de objetos de esas clases, uno para cada mes del año.
Cree un método para rellenar los datos de un mes, e invóquelo 12 veces para completar todos los datos del año.
Cree un método para mostrar los datos de un mes, e invóquelo 12 veces para visualizar todos los datos del año.

4. Cargue un array (vector) de N números enteros. Rellénelo, y diga cuál es el mayor valor que contiene, y qué posición ocupa dicho valor.

5. Cargue un array (vector) de N números enteros. Rellénelo y calcule la media de los que ocupan posiciones pares.

6. Cargue una tabla de NUM_FILAS filas y NUM_COLS columnas de enteros. Rellene el contenido de dicha matriz, pidiendo por teclado sus valores. Al final, muestre la suma de las diferentes filas y columnas que componen la matriz obtenida.

7. Desarrolle un programa que pida por teclado dos matrices de enteros, la primera N x M, y la segunda M x P; el programa habrá de averiguar la matriz N x P resultante de multiplicar la primera y la segunda matriz introducidas.

8. Diseñe un array de almacenamiento apto para albergar el siguiente triángulo:

1
121
12321
1234321
123454321
12345654321
1234567654321
123456787654321

Observe que el número de columnas de cada fila no es el mismo. Rellene el array diseñado con los datos del triángulo, y muéstrelo en pantalla.

9. Pida por teclado una cadena de caracteres, y muéstrela carácter por carácter por pantalla.

10. Una clase recibe una frase. Se pide invertir la frase en cuestión carácter a carácter, mostrando el resultado en pantalla.

11. Una clase recibe una frase. Se pide construir un programa que muestre esa frase en mayúsculas y minúsculas.

12. Cree un método llamado *invierte*, que reciba como parámetro una cadena de caracteres, y la modifique cambiando las minúsculas por mayúsculas, las mayúsculas por minúsculas y los dígitos por puntos. Pruebe el funcionamiento de dicho método usándolo en el main, con cadenas de caracteres introducidas por teclado.

13. Un palíndromo es una frase que se lee igual de izquierda a derecha que de derecha a izquierda. Cree un método llamado *palindromo*, que reciba como argumento una frase, e indique si dicha frase es o no un palíndromo.

14. Realice un programa que extraiga de un número de tipo float los n primeros decimales.

15. Escriba un programa que lea una línea por teclado, y separe dicha línea en palabras. El programa mostrará cada palabra que compone la línea en una línea diferente. Suponga que la separación de palabras en la línea se consigue con un espacio en blanco.

16. Una clase construye internamente las tablas de multiplicar (hasta diez). Se pide hacer que el programa construya internamente una cadena para cada tabla, separando los elementos con asteriscos, para mostrar después el resultado en pantalla. Se descompondrá la tarea en métodos.

17. Una clase recibe una frase. Se pide hacer una estadística de letras de esa frase (que puede contener dígitos), considerando como equivalentes las mayúsculas y minúsculas.

18. Una clase recibe una frase. Se pide hacer la estadística de composición de esa frase, mostrando en pantalla únicamente las palabras que aparezcan una o más veces. Las palabras en mayúsculas y minúsculas se consideran equivalentes.

19. Un programa recibe varias frases como argumentos de la línea de órdenes. Construir un método que produzca como resultado un lista de String que contenga el léxico con indicación de frecuencias.

20. Una clase recibe un nombre de fichero. El programa debe comprobar que la extensión de fichero sea viable (entre uno y tres caracteres separados por un punto, con un único punto en posición correcta).

21. Una clase solicita al usuario una nueva pareja de palabras usuario/contraseña. El programa debe verificar la corrección de esa contraseña, que deberá satisfacer las condiciones siguientes:

· La contraseña no puede tener menos de seis caracteres
· Al menos dos de esos caracteres deben ser dígitos
· El usuario no puede formar parte de la contraseña

22. Una clase recibe un número IP. Se pide construir un método que devuelva un valor boolean indicativo de si el número en cuestión está o no bien formado.

23. Cree un array con datos sobre los meses del año: el número de días del mes y el orden en el que se suceden a lo largo del año. Cree también una clase que le permita manejar fechas (número de día, día de la semana, mes y año). Use ambas estructuras en un programa que solicite al usuario un día, mes y año, y devuelva el número total de días transcurridos en ese año hasta la fecha indicada, suponiendo que no hubiese años bisiestos. Amplíe el ejercicio para calcular el número de días que el usuario lleva viviendo hasta hoy.

24. Cree una aplicación que lea fechas por teclado con el formato dd/mm/aaaa. Dicha aplicación mostrará por pantalla la fecha siguiente a la introducida por el usuario. No considere de momento años bisiestos, y suponga que no se introducirán fechas incorrectas. En una versión mejorada podría considerar bisiestos, y comprobar que la fecha introducida es realmente válida.

25. Implemente un método llamado *aBinario*, que reciba un entero y devuelva una cadena con su valor convertido a binario (análogamente a lo que hace `Integer.toBinaryString`, sin hacer uso de dicho método). De igual modo, implemente el método *aOctal* y *aHexa*, que proporcionen la cadena de resultante de convertir un entero a octal y hexadecimal, respectivamente.

26. Use las clases y / o arrays que estime oportunos para representar una baraja española de cartas. Represente también la información relevante de una baza del juego de la escoba por medio de una clase (número de cartas, número de sietes, número de oros y posibilidad de hacer escoba, sin dejar ninguna otra carta en la mesa).

27. Usando las clases creadas en el ejercicio anterior, escriba un programa que asigne valor a los campos de la clase de la jugada de la escoba (baza), a partir de un vector de clases de tipo carta (correspondientes a las cartas de la baza en cuestión).

28. Realice un programa que implemente un juego de adivinanza de palabras. Al jugador se le mostrará por pantalla un guión por cada letra que compone la palabra, y se le dará la oportunidad de adivinar una letra. En caso de que la letra forme parte de la palabra, en la siguiente oportunidad se le mostrará dicha letra en vez del guión en la posición correspondiente (a modo de *La ruleta de la fortuna*). El jugador contará con un máximo de 10 intentos para adivinar la palabra en cuestión. Si se completa la palabra con las letras solicitadas antes de terminar con los 10 intentos, el juego felicitará al usuario por su triunfo. Si pasados los 10 intentos de letras no se ha completado la palabra, el juego dará la posibilidad de intentar adivinar introduciendo una palabra. Si se adivina, el juego felicitará al jugador, y si no, se lamentará de su derrota y le informará de cuál era la palabra que perseguía.

NOTA 1 : Rellene, al comienzo del programa, un array con las palabras que serán candidatas de ser adivinadas en el juego.

NOTA 2 : Realice, al comienzo del programa, una selección aleatoria de una de dichas palabras, que se convertirá en la palabra a adivinar.

29. Cree una versión reducida de un guerra barcos, en el que un jugador tratará de hundir los barcos previamente colocados por otro jugador. Se dispondrá de un tablero clásico de *guerra barcos*, con 8 filas (identificadas por números) y 8 columnas (identificadas por letras). El primer jugador colocará en él un barco de cuatro casillas, otro de tres, otro de dos, y otro de una casilla, siguiendo las reglas conocidas del juego. Una vez colocados, el segundo jugador dispondrá de un número N de intentos (por ejemplo 20), para hundir los barcos. Ante una propuesta (por ejemplo C - 6), el juego responderá diciendo si *Agua*, *Tocado* o *Hundido*, e irá llevando la cuenta de los barcos y piezas hundidas. Si al final el segundo jugador consigue hundir toda la flota del primero, habrá ganado la partida; en otro caso, la habrá perdido.

NOTA: Amplíe dicho programa tanto como le permita su imaginación y conocimientos (permita un "contraataque" del primer jugador, haciendo que tras disparar el segundo, pueda disparar el primero, amplíe el número de barcos, ...).

30. Escriba diferentes programas de prueba que le permitan practicar con los diversos aspectos de arrays y cadenas en Java.