UD02.Estruturas de control
Ejercicios

DAM1-Programación 2023-24

Sobre la entrega de los ejercicios.

1. Agrupa los ejercicios en los paquetes cuyo nombre se indica al inicio de cada apartado.
2. Nombra los archivos .java con el nombre que se indica en negrita en el enunciado.
3. Utiliza un *comentario de documentación* para indicar tu nombre y apellidos como autor del código.

Condicionales Paraninfo

Paquete: `paraninfo02`

EP0211. Escribe una aplicación que solicite al usuario un número comprendido entre 0 y 9999. La aplicación tendrá que indicar si el número introducido es capicúa.

EP0212. El DNI consta de un entero de 8 digitos seguido de una letra que se obtiene a partir del número de la siguiente forma:

letra = número DNI módulo 23

Basándote en esta información, elige la letra a partir de la numeración de la siguiente tabla:

CÓDIGO PARA LA LETRA DEL D.N.I. O DEL N.I.F.																					
RESTO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LETRA	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

y diseña una aplicación en la que, dado un número de DNI, calcule la letra que le corresponde. Observa que un número de 8 digitos está dentro del rango del tipo int.

EP0213. En una granja se compra diariamente una cantidad (`comidaDiaria`) de comida para los animales. El número de animales que alimentar (todos de la misma especie) es `numAnimales`, y sabemos que cada animal come una media de `kilosPorAnimal`.

Diseña un programa que solicite al usuario los valores anteriores y determine si disponemos de alimento suficiente para cada animal. En caso negativo, ha de calcular cuál es la ración que corresponde a cada uno de los animales.
Nota: Evitar que la aplicación realice divisiones por cero.

EP0214. Escribe un programa que solicite al usuario un número comprendido entre 1 y 99. El programa debe mostrario con letras, por ejemplo, para 56, se verá: “cincuenta y seis”.

EP0215. Escribe una aplicación que solicite por consola dos números reales que corresponden a la base y la altura de un triángulo. Deberá mostrarse su área, comprobando que los números introducidos por el usuario no son negativos, algo que no tendría sentido.

EP0216. Utiliza el operador ternario para calcular el valor absoluto de un número que se solicita al usuario por teclado.

EP0217. Realiza el “juego de la suma”, que consiste en que aparezcan dos números aleatorios (comprendidos entre 1 y 99) que el usuario tiene que sumar. La aplicación debe indicar si el resultado de la operación es correcto o incorrecto.

EP0218. Modifica la actividad anterior (EP0217) para que, además de los dos números aleatorios, también aparezca aleatoriamente la operación que debe realizar el jugador: suma, resta o multiplicación.

Amplía el programa anterior para que muestre el número de segundos que el usuario ha tardado en responder. Utiliza la clase LocalTime.

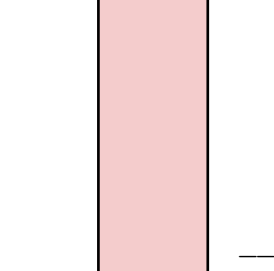
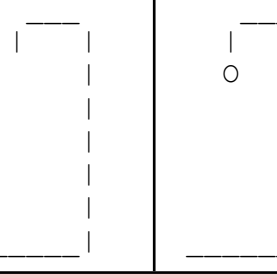
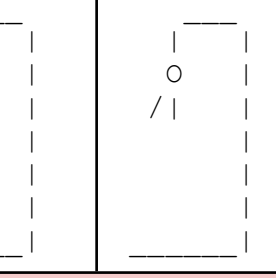
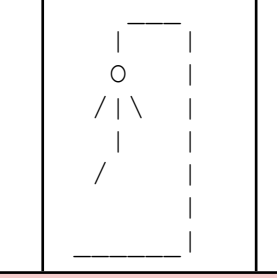
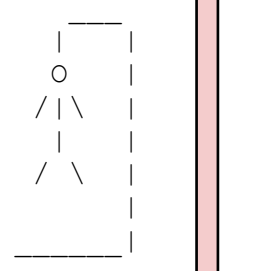
EP0219. Crea una aplicación que solicite al usuario cuántos grados tiene un ángulo y muestre el equivalente en radianes. Si el ángulo introducido por el usuario no se encuentra en el rango de 0° a 360°, hay que transformarlo a dicho rango. Nota: El operador módulo (%) puede ayudarnos a convertir un ángulo a su equivalente en el rango comprendido de 0 a 360°.

Ejercicios Condicionales

Paquete: `condicionales`

JuegoCuantoTiempo. Realiza un juego que solicite al usuario pulsar Enter después de trasncurrir un número aleatorio de segundos, que se mostrará al usuario, elegido al azar entre un número mínimo, por ejemplo 5, y un número máximo, por ejemplo 30. Los límites mínimo y máximo de segundos se definirán como constantes.
El programa contará el tiempo que el usuario ha tardado en pulsar Enter y responderá con un mensaje diciendo si acertó en el momento correcto o, en caso contrario, cuántos segundos, de adelanto o de retraso, se desvió.

Ahorcado. Haz un programa que lea por teclado un número de fallos en el juego del ahorcado, entre 0 y 7, y que imprima un dibujo de un ahorcado más o menos completo en función del número de fallos, como los de los siguientes ejemplos:

Fallos = 0	Fallos = 1	Fallos = 3	Fallos = 6	Fallos = 7
				

...

```
// Imprime partes comunes iniciales del ahorcado
System.out.println("    ___");
System.out.println("   |   |");

switch (numFallos) {
    case 0:
        System.out.println("       |");
        System.out.println("       |");
        System.out.println("       |");
        System.out.println("       |");
        break;
    case 1:
        System.out.println("   O   |");
        System.out.println("       |");
        System.out.println("       |");
        System.out.println("       |");
        break;
    case 2:
        System.out.println("   O   |");
        System.out.println("  /    |");
        System.out.println("       |");
        System.out.println("       |");
        break;
    case 3:
        System.out.println("   O   |");
        System.out.println("  /|   |");
        System.out.println("       |");
        System.out.println("       |");
        break;
    case 4:
        // Hay que "escapar" la barra '\' para imprimirla
        System.out.println("   O   |");
        System.out.println("  /|\\  |");
        System.out.println("       |");
        System.out.println("       |");
        break;
    case 5:
        System.out.println("   O   |");
        System.out.println("  /|\\  |");
        System.out.println("   |   |");
        System.out.println("       |");
        break;
    case 6:
        System.out.println("   O   |");
        System.out.println("  /|\\  |");
        System.out.println("   |   |");
        System.out.println("  /    |");
        break;
    case 7:
        System.out.println("   O   |");
        System.out.println("  /|\\  |");
        System.out.println("   |   |");
```

```
        System.out.println("  / \\  |");
        break;
}

// Imprime partes comunes finales
System.out.println("       |");
System.out.println(" ______|");
...
```

Bucles Paraninfo

Paquete: `paraninfo03`

EP0311. Realiza un programa que convierta un número decimal en su representación binaria. Hay que tener en cuenta que desconocemos cuántas cifras tiene el número que introduce el
usuario.
Por simplicidad, iremos mostrando el número binario con un digito por línea.

EP0312. Modifica el programa anterior (3.11) para que el usuario pueda introducir un número en binario y el programa muestre su conversión a decimal.

EP0313. Escribe un programa que incremente la hora de un reloj. Se pedirán por teclado la hora, minutos y segundos, así como cuántos segundos se desea incrementar la hora introducida. La aplicación mostrará la nueva hora. Por ejemplo, si las 13:59:51 se incrementan en 10 segundos, resultan las 14:00:01.

EP0314. Realiza un programa que nos pida un número n, y nos diga cuántos números hay entre 1 y n que sean primos. Un número primo es aquel que solo es divisible por 1 y por él mismo. Veamos un ejemplo para n = 8.

Comprobamos todos los números del 1 al 8:

1 - primo
2 - primo
3 - primo
4 - no primo
5 - primo
6 - no primo
7 - primo
8 - no primo

Resultan un total de 5 números primos.

EP0315. Diseña una aplicación que dibuje el triángulo de Pascal, para n filas. Numerando las filas y elementos desde 0, la fórmula para obtener el m-ésimo elemento de la n-ésima fila es

$E(n, m) = n! \ / \ m!(n - m)!$

Donde n! es el factorial de n.

Un ejemplo de triángulo de Pascal con 5 filas (n = 4) es :

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

EP0316. Solicita al usuario un número n y dibuja un triángulo de base y altura n, de la forma (para n igual a 4):

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

EP0317. Para dos números dados, a y b, es posible buscar el máximo común divisor (el número más grande que divide a ambos) mediante un algoritmo ineficiente pero sencillo: desde el menor de a y b, ir buscando, de forma decreciente, el primer número que divide a ambos simultáneamente. Realiza un programa que calcule el máximo común divisor de dos números.

EP0318. De forma similar a la actividad anterior (3.17), implementa un algoritmo que calcule el minimo común múltiplo de dos números dados.

EP0319. Calcula la raíz cuadrada de un número natural mediante aproximaciones. En el caso de que no sea exacta, muestra el resto. Por ejemplo, para calcular la raíz cuadrada de 23, probamos $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 16$, $5^2 = 25$ (nos pasamos), resultando 4 la raiz cuadrada de 23 con un resto (23 - 16) de 7.

EP0320. Escribe un programa que solicite al usuario las distintas cantidades de dinero de las que dispone. Por ejemplo: la cantidad de dinero que tiene en el banco, en una hucha, en un cajón y en los bolsillos. La aplicación mostrará la suma total de dinero de la que dispone el usuario.

La manera de especificar que no se desea introducir más cantidades es mediante el cero.

Ejercicios Bucles

Paquete: `bucles`

MayorMenorNNumeros. Diseña una aplicación que muestre el mayor y menor número entero de un conjunto indeterminado de números positivos introducidos por teclado. El programa terminará al introducir un 0 o un número negativo.

Funciones Paraninfo

Paquete: `paraninfo04`

EP0411. Diseña una función que calcule y muestre la superficie y el volumen de una esfera.

Superficie = 4pi * radio^2
Volumen = (4pi / 3) * radio^3

EP0412. Implementa la función

```
static double distancia (double x1, double y1, double x2, double y2)
```

que calcula y devuelve la distancia euclídea que separa los puntos (x1, y1) y (x2, y2). La fórmula para calcular esta distancia es:

distancia = $\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$

EP0413. Crea la función `muestraPares(int n)` que muestre por consola los primeros `n` números pares.

EP0414. Escribe una función a la que se pase como parámetros de entrada una cantidad de días, horas y minutos. La función calculará y devolverá el número de segundos que existen en los datos de entrada.

EP0415. Diseña una función a la que se le pasan las horas y minutos de dos instantes de tiempo, con el siguiente prototipo:

```
static int diferenciaMin (int hora1, int minuto1, int hora2, int minuto2)
```

La función devolverá la cantidad de minutos que existen de diferencia entre los dos instantes utilizados.

EP0416. Implementa la función `divisoresPrimos()` que muestra, por consola, todos los divisores primos del número que se le pasa como parámetro.

EP0417. Escribe una función que decida si dos números enteros positivos son amigos. Dos números a y b son amigos si la suma de los divisores propios (distintos de él mismo) de a es igual a b. Y viceversa.

Para probar se pueden usar los números 220 y 284, que son amigos.

EP0418. Crea una función que muestre por consola una serie de números aleatorios enteros. Los parámetros de la función serán: la cantidad de números aleatorios que se mostrarán y los valores mínimos y máximos que estos pueden tomar.

EP0419. Sobrecarga la función realizada en la Actividad 4.18 para que el único parámetro sea la cantidad de números aleatorios que se muestra por consola. Los números aleatorios serán reales y estarán comprendidos entre 0 y 1.