Ejercicios de estructuras de control

1. Dado un número escribir “Buena edad”, “Demasiado joven” o “Viejales” en función de que dicho número sea igual, menor o mayor que nuestra edad.
    - Utilizar if..else if...
2. Dado un número, indicar por pantalla si es par o impar.
    - Utilizar if...else
3. Dado un número, mostrar por pantalla su nombre (es decir, si es 1, “UNO”, si es 2, “DOS”, si es 3, “TRES”, y si es cualquier otro número que muestre “NO SÉ). Hacer una versión alternativa utilizando “switch”.
    - Utilizar if..else if…
    - Utilizar switch
4. Dada una letra indicar si es una vocal o una consonante.
    - Utilizar switch, capturando la entrada del teclado con “scanNext()”. Utilizar una variable tipo String y no char, porque el Scanner no está preparado para leer char. Recordad que los literales String van entre dobles comillas, y no simples como char
5. Mostrar los 10 primeros números (utilizando una sola sentencia “print”)
6. Mostrar los números desde el 5 hasta el 10 (utilizando una sola sentencia “print”)
7. Mostrar los números desde el 10 hasta el 1 (utilizando una sola sentencia “print”)
8. Mostrar los números desde el 10 hasta el 4 (utilizando una sola sentencia “print”)
9. Mostrar los números IMPARES que haya (en orden) desde el 1 hasta el 10 (utilizando una sola sentencia “print”)
10. Mostrar la tabla de multiplicar del 4 (utilizando una sola sentencia “print”);
11. Pedir un número por teclado y mostrar la tabla de multiplicar del ese número (utilizando una sola sentencia “print”)
12. Mostrar la tabla de multiplicar de los 10 primeros números (utilizando una sola sentencia “print”)
13. Pedir un número por teclado y mostrar las tablas de multiplicar desde el 1 hasta ese número introducido (utilizando una sola sentencia “print”)
14. Leer números por teclado de manera que al escribir el número 0, pare el proceso y se dejen de leer números.
    - Utilizar un bucle do..while, que lea un número “n” y permanezca en el bucle si n es distinto a cero
15. Leer números por teclado de manera que sólo permita que el recién leído sea mayor que el anterior. En el momento que esto no se cumpla el programa se detiene. La lectura del primer número siempre será exitosa, ya que no hay ningún número con el que compararlo.
    - Hacer una primera lectura de “n” fuera del bucle. Utilizar un bucle do..while en el que, se guarde el valor de “n” en una variable “ant”, se lea el nuevo “n” y se permanezca en el bucle si n>ant
16. Leer números, permitiendo únicamente que el recién leído sea distinto de cero y mayor que el anterior. En caso contrario se debe detener el programa.

- Igual que 6, pero la condición de permanencia en el bucle, debe incluir que “n” sea distinto de cero. Además, antes de entrar en el bucle se deberá comprobar que “n” sea distinto de cero con un “if”.

17. Leer números, permitiendo únicamente que el recién leído sea mayor que el anterior. En caso contrario se debe leer otro número. Detenerse cuando se introduzca el cero. Informar en cada número que se pida, a qué otro número debe superar.
    - Igual que 8, excepto que la condición de permanencia en el bucle es únicamente que sea distinto de cero. Dentro del bucle, y después de leer el nuevo “n” se deberá comprobar con un if que n>ant, escribiendo por pantalla “n” si fuera cierto, y no haciendo nada en caso contrario.
18. Dado un número entero mayor que cero, escribirlo en orden inverso. (ej. Dado 1234, mostraría 4321)
    - Obtener las unidades de “n” observando el resto de dividir “n” por 10. Imprimir esas unidades por pantalla y repetir el proceso anterior pero trabajando con la décima parte de “n”
19. Dado un número entero mayor que cero, indicar cuántos dígitos tiene.
    - Introducir el número en un bucle e irlo dividiendo por 10 hasta que sea menor que cero. Llevar un contador para ver cuántas vueltas se dan al bucle. Ese número será el número de cifras.
20. Indicar si un número entero mayor que cero, introducido, es capicúa.
    - Aprovechando la experiencia de los dos ejercicios anteriores crear un método “int reves(int n)” que devuelva el inverso de un número (no que lo muestre por pantalla, sino que lo devuelva).
    - Para hacer el método “int reves (int n)”, la estrategia es un poco distinta del ejercicio que tan sólo lo mostraba. Lo que hay que hacer es ir dividiendo el número por 10, y sumarle el resto a 10*inverso que hayamos calculado hasta el momento.
    - Para ver que “n” sea capicúa basta con que n sea igual a reves(n)
21. Indicar la suma de los dígitos de un número (ej. 178 = 1+7+8=16)
    - Utilizar una variable “acc” (acumuladora) en la que vayamos acumulando la suma parcial de los distintos dígitos. “acc” comenzará valiendo 0, y le iremos sumando, en un bucle, la cifra de la posición de las unidades de “n”. Antes de salir del bucle, dividiremos “n” por 10 para que la posición que estaba en las decenas pase ahora a ser la de las unidades. Permaneceremos en el bucle hasta que n<0 (es decir que no queden más unidades que procesar).
22. Indicar la suma de los dígitos de posición impar de un número dado (ej. Dado 24567, la suma sería 2+5+7=**14**)
    - Parecido al anterior, pero sólo deberemos actualizar la variable acumuladora, en una de cada dos vueltas, para lo cual podemos utilizar una variable booleana, a la que podemos llamar “impar” (para indicarnos si estamos en un dígito de posición impar o no). Inicialmente valdrá true, ya que las unidades son impares, y en cada vuelta comprabaremos si es impar o no, para acumular o no, y en cualquier caso cambiar el valor de la variable par, ya que irá alternando su valor de true a false en cada vuelta del bucle.
23. Dado un número mayor que cero, indicar si es primo o no.

- Hacer un bucle que vaya desde 2 hasta n-1 (ya que cualquier número es divisible por la unidad y por sí mismo) , comprobando que el resto de dividir n por i (siendo i el contador del bucle), sea distinto de cero, ya que en caso contrario sería divisible y por tanto no primo. Evaluar con una variable booleana si el número ha permanecido siendo primo durante todas las vueltas del bucle.

24. Mostrar los números primos existentes entre el número 1 y el 100.
    - Hacer un método “boolean primo(int n)” que devuelva true si n es primo y false en caso contrario. Apoyarse en el código anteriorsd
    - Hacer un bucle de 1 a 100 que imprima el número en función de si es primo o no..
25. Dados dos números a y b, mostrar los números primos que existen entre ellos.
    - Hacer un bucle que recorra desde a hasta b y en cada vuelta consulte al método “primo” para decidir si lo debe mostrar o no.
26. Dados dos números, indicar el máximo común divisor de ambos.
    - Existen muchos algoritmos para calcular el mcd(x,y). Uno de los más populares es el algoritmo de Euclides que consiste en lo siguiente: Siendo x >= y (y si no los intercambiaremos), permaneceremos en un bucle mientras “y” sea distinto de cero. En cada vuelta del bucle “x” pasará a valer “y”, e “y” pasará a valer el resto de dividir “x” por “y”. Al terminar el bucle, el mcd será el último valor de “x”.
27. Dado un número n, mostrar los n primeros términos de la serie de Fibonacci. La serie de Fibonacci, comienza por el cero, sigue por el uno, y los siguientes números se van calculando como la suma de los dos anteriores, es decir: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 …
    - Arreglar un bucle en el que se vayan manteniendo los dos valores de la serie anteriores
28. Dado un número mostrar n! (n! = n (n-1) (n-2) ….. 1). Téngase en cuenta que 0! = 1
    - Utilizar una variable “acc” acumuladora de las multiplicaciones parciales, que comience valiendo 1 y se actualice dentro de un bucle que vaya desde 1 hasta n. Mirar las formas de optimizar dicho bucle.
29. Dado un par de números “n” y “k”, con n>k, mostrar el valor del número combinatorio,

    $$\binom{n}{k}$$

    que se calcula de la siguiente manera:

    $$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

    - hacer un método “int fact(int n)” que calcule el factorial de un número n, basándose en el algoritmo del ejercicio anterior.
    - Optimizar el cálculo observando el comportamiento de la fórmula

30. Mostrar los 5 primeros términos de la serie:
    $x_i = x_{i-1} * 3$ sabiendo que $x_1 = 1$;
31. Calcular el sig. sumatorio:

32. Dado el siguiente programa, prever el resultado de la ejecución, sin compilarlo ni ejecutarlo en el ordenador.

```
public class Ejercicio {

  public static int f(int x)    {
    int resultado;
    if (x%2==0)   {
      resultado=x+1;
    }
    else       {
      resultado=x+3;
    }
    return resultado;
  }

  public static void main(String[] args) {
    System.out.println ( f(2) );
    System.out.println ( f(3) );
    for (int i=0;i<4;i=i+2) {
      System.out.println ( f(i) );
    }
    System.out.println ( f(2+1) );
    System.out.println ( f(2)+1 );
    System.out.println ( f(1) + " " +  f(f(1))
);
    for (int i=0;i<4;i=i+2) {
      System.out.println ( f(f(i)) );
    }
  }

}
```

```
3
6
1
3
6
```

4
4 5
4
6

33. Escribir los siguientes bucles “while” con la sintaxis de un bucle “for”
    a.
```
int i=10;
while (i>0) {
        System.out.println(i);
        i=i-2;
}
```
    b.
```
int i=10;
int j=20;
while (j>i) {
        int k=i+1;
        i=i*2;
        System.out.println(i+”,”+j+”,”+k);
        j=j-1;
}
```
    c.
```
boolean amor=true;
int i=1;
while (amor || i<=4) {
        if (amor) {
                System.out.println(“Me quiere”);
        }
        else {
                System.out.println(“NO Me quiere”);
        }
        amor = ! amor;
        i++;
}
```
34. Escribir los siguientes bucles “for” con la sintaxis de un bucle “while” o “do...while”
    a.
```
for (int j=10;(j<20 || j%3==0);j*=2) {
        int k=j/4;
        System.out.println(k);
}
```
    b. ----

35. Dado un número n, dibujar una figura como ésta, en la que la base del triángulo más grande es de n asteriscos (tal que 1<= n <= 20)

Por ejemplo:

```
Introduce n: 7
*
**
***
****
*****
******
*******
```

36. Modificar el programa anterior, para que haga la “bajada de la montaña” también
    Por ejemplo:

```
Introduce n: 7
*
**
***
****
*****
******
*******
******
*****
****
***
**
*
```

37. Modificar el programa anterior, para que haga la dibuje m montañas de altura a.
    Por ejemplo:

```
Introduce a: 3
Introduce m: 4
*
**
***
**
*
*
**
***
**
*
*
```

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

38. Similar al 35, pero haciendo el triángulo a la inversa.

```
Introduce n: 7
        *
       **
      ***
     ****
    *****
   ******
  *******
```

39. ---

PRÁCTICAS

1. Realizar un programa que muestre el resultado del cálculo combinatorio adecuado en función de los siguientes datos:
    - Un número **n** de elementos.
    - Un número **k** de elementos a combinar en CADA combinación válida.
    - La tecla “s” o “n” en función de si importa el orden de combinación.
    - La tecla “s” o “n” en función de si se pueden repetir los elementos.
2. Realizar un programa que permita la visualización de ciertas magnitudes físicas en varios sistemas, de la siguiente manera:
    1. Al arrancar el programa mostrará un menú, en el que se proporcionará la siguiente información y opciones:
        a. Un título: “CONVERSOR de MAGNITUDES FÍSICAS”
        b. Información acerca del sistema métrico utilizado en este momento, p.ej “Sistema Internacional (S.I. ó MKS) o sistema cegesimal (CGS)”
        c. Tres números, que denoten la cantidad de espacio, masa y tiempo con la que estamos trabajando (p.ej. si estamos trabajando en el S.I., y los números son 1, 2 y 3, querría decir que estamos trabajando con 1 metro, 2 kilo y 3 segundos), es importante indicar al usuario, detrás de cada uno de estos números la unidad de medida con la que se está trabajando. (es decir: 1 m, 2 kg, 3 s)
        d. Una lista de magnitudes físicas derivadas (con sus unidades) calculadas a partir de los datos de masa, tiempo y espacio. Éstas serán:

i. Velocidad. (v=e/t)
ii. Aceleración (a=e/t2)
iii. Fuerza (F=m.a)
iv. Trabajo (T=F.e)
v. Potencia (W=T/t)
vi. Presión (P=F/e2)

NOTA: Buscar los nombres de las unidades en Wikipedia (entradas: “Sistema Internacional de unidades” o “Sistema cegesimal de unidades”)

e. Opciones de usuario (de significado obvio)
    1. Cambiar la cantidad de masa
    2. Cambiar la cantidad de tiempo
    3. Cambiar la cantidad de espacio.
    4. Cambiar el sistema de unidades (conmuta entre S.I. y CGS)

1. Cuando un usuario cambia la cantidad de masa, tiempo o espacio, todas las magnitudes físicas se recalculan respecto a estos nuevos valores y se muestran al usuario.
2. Cuando un usuario cambia el sistema de medida, se recalculan, no sólo las magnitudes físicas derivadas, sino la cantidad de masa, tiempo y espacio, que corresponden a los valores ya fijados (p.ej. si estábamos trabajando con 1 m en el sistema internacional y conmutamos a CGS, automáticamente aparecerá que estamos trabajando con 100 cm). Por supuesto, se tendrán que recalcular todas las fórmulas.
3. El menú permitirá al usuario permanecer ejecutando el programa hasta que pulse una tecla (p.ej. la tecla “T”), para terminar.