CUESTIONES Y EJERCICIOS - 4ª VUELTA A LA ESPIRAL

1. Teclea la siguiente clase y añade los comentarios necesarios para que se genere una documentación completa con la utilidad Javadoc:

Menu
+ primerPlato: texto + segundoPlato: texto + calorías: entero + precio: real
+ getPrimerPlato() devuelve texto + getSegundoPlato() devuelve texto + getCalorias() devuelve entero + getPrecio() devuelve real + setPrimerPlato (plato1: texto) + setSegundoPlato(plato2: texto) + setCalorias(cal: entero) + setPrecio(nuevoPrecio: real)

2. Completa el ejercicio anterior de modo que se incluya un nuevo método llamado imprimeMenu() que imprima el siguiente contenido formateado.

```
PRIMER PLATO                    SEGUNDO PLATO
Ensalada de la casa          Solomillo al whisky
--------------------------------------------
CALORÍAS                                PRECIO
798                                     11,95€
```

Debes saber que el ancho total de la línea es de 44 caracteres y que se están usando dos zonas de formateo de 22 caracteres, una alineada a la izquierda y otra a la derecha.

3. Conociendo la declaración de las siguientes variables:

int edad;	double precio;	long entero;
short corto;	float rango;	byte miByte

Indica el tipo de cada una de las siguientes expresiones:

a) corto*miByte + edad*2/3
b) corto*rango + edad*entero
c) edad*50L + corto + miByte
d) (int) (precio*rango+entero*edad)
e) edad*23.33 + entero*rango
f) corto + miByte + rango + edad + precio
g) (float) (edad + entero)
h) (float) edad + precio
i) miByte*23.33F + edad

CUESTIONES Y EJERCICIOS - 5ª VUELTA A LA ESPIRAL

4. Escribir las clases:

```
public class PersonaCondicional {
    public int edad, altura, peso;
    boolean estaCasado;
    //Crear los setters y getters con ayuda del IDE presionando Alt+Ins
}
```

```
public class PruebaPersonaCondicional {
    public static void main (String args[]) {
        PersonaCondicional p = new PersonaCondicional();
        p.setEdad(20);
        p.setAltura(165);
        p.setEstaCasado(false);

        // Pon tus sentencias condicionales aquí
    }
}
```

Escribe el código que falta para que se realicen las siguientes comprobaciones:

- Si la edad >= 18 se debería imprimir "Ya eres un señor/a "
- Si la altura >= 185 se debe imprimir "Eres alto/a"
- Si estaCasado == false entonces se debe imprimir: "No está casado".

A continuación, modifica el código para que la edad y la altura se lean por teclado. Ejecútalo 4 veces introduciendo las distintas posibilidades para que se impriman o no los mensajes

5. Retoma el programa anterior y realiza las siguientes acciones:
   a) Depurar el programa paso a paso, añadiendo “p” como un elemento a observar (watch) y comprobar que el resultado impreso en pantalla es el correcto.
   b) Modificar los valores de la persona para que solo se imprima “Eres alto/a”. Depurar para comprobar que los cambios han hecho lo que deben.

6. Las propiedades de un objeto *Persona* se han establecido del siguiente modo:

   edad = 39; altura = 200; casado = true;

   trabajador = false; salario = 1800.50; cochePropio = true;

   A continuación, indica si el resultado de las siguientes condicionales es true o false:

CONDICIÓN	**RESULTADO**
if (edad >= 18 && altura < 180 && casado == true && cochePropio == true)	
if (edad >= 18 || altura < 180 || casado == true || cochePropio == true)	
if (salario >= 1800 && !(trabajador == false) && !(edad < 18))	
if (salario >= 1800 || !(trabajador == false) || !(edad < 18))	
if (!trabajador && !casado)	
if (edad < 39 && altura >= 200 && cochePropio && casado)	
if (edad > 39 || altura > 200 || cochePropio)	
if (edad < 39 && altura >= 200 && cochePropio || casado)	
if (edad < 39 && altura >= 200 && (cochePropio || casado))	

   Por último:

   a) Observa las condiciones que solo tienen && y describe con tus palabras cómo funcionan estos operadores encadenados.
   b) Observa las condiciones que solo tienen || y describe con tus palabras cómo funcionan estos operadores encadenados.

7. Escribir una clase llamada *PruebaCasting.java* que contenga el siguiente código y que utilice la clase *PersonaCondicional* del ejercicio 4:

```
public class PruebaCasting {
    public static void main (String args[]) {

        // Escribe tu código aquí

    }
}
```

Escribe el código que falta para que se realicen las siguientes acciones:

- Crea un objeto de la clase *PersonaCondicional*.
- Añade código para que la *edad* y la *altura* se lean por teclado.
- Establece la propiedad *casado* a *false*.
- Si la edad está entre 18 y 30 años y no está casado y la altura es mayor de 175 cm entonces se debe escribir *"Has conseguido el papel"*
- Ejecutar el programa y comprobar que el resultado impreso en pantalla es el correcto.
- Por último, cambia los valores de la edad para que no se imprima ningún mensaje

8. Vamos a reescribir la clase llamada *PruebaCasting.java* para que se imprima "Has conseguido el papel" se cumpla alguna de las siguientes condiciones:
   - La edad está entre 18 y 30 y no está casado.
   - La edad sea mayor que 30 y menor que 40, no está casado pero la altura debe ser mayor de 190 cm.
   - La edad sea mayor o igual que 40 años y esté casado.

   Piensa que < no es lo mismo que <= y que > no es lo mismo que >=

9. ¿Cómo reescribirías el if del ejercicio anterior si la última condición fuera la siguiente?
   - La edad sea mayor o igual que 40 años **O** esté casado.

10. Imagina que el profesor se equivoca al escribir el enunciado del ejercicio anterior y que ahora dice *"…para que se imprima "Has conseguido el papel" se deben cumplir TODAS las condiciones:*
    - *… Idem"*

    ¿Existiría alguna persona que pudiera pasar el casting? Justifica tu respuesta.

11. Escribir las clases:

```
public class PersonaCondicionalDoble {
    public int edad, altura, peso;
    boolean estaCasado, tieneTrabajo;
    //Crear los setters y getters con ayuda del IDE presionando Alt+Ins
}
```

```
public class PruebaPersonaCondicionalDoble {
    public static void main (String args[]) {
        PersonaCondicionalDoble p = new PersonaCondicionalDoble ();
        p.setEdad(20);
        p.setAltura(165);
        p.setEstaCasado(false);
        p.setTieneTrabajo(true);

        // Pon tus instrucciones aquí
    }
}
```

Escribe el código que falta para que se realicen las siguientes acciones:

- Si la edad < 18 se debería imprimir "Eres menor de edad", en caso contrario "Eres mayor de edad"
- Si la altura >= 185 se debe imprimir "Eres alto/a", en caso contrario "No eres alto/a"
- Si estaCasado == true entonces se debe preguntar, además, si tieneTrabajo == true. Dependiendo de las posibles respuestas a ambas preguntas tendremos que imprimir los siguientes mensajes según corresponda: *"Casado/a y con trabajo", "Casado/a y sin trabajo"* y "No está casado".

Ejecutar el programa y comprobar que el resultado impreso en pantalla es el correcto.

12. Partir de la clase *BombillaInteligente* y modificarla para crear una nueva clase llamada *BombillaCondicional* con el siguiente diagrama de clase:

BombillaCondicional
+ marca: texto + potencia: entero + encendida: lógico + numVecesEncendida: entero
+ setMarca (nuevaMarca: texto) + imprimeMarca () + setPotencia (nuevaPotencia: entero) + imprimePotencia () + encender () + apagar () + imprimeEstado () + getNumVecesEncendida() devuelve entero

Se debe modificar el comportamiento de los siguientes métodos:

- encender(): este método debe comprobar si la bombilla está previamente encendida y mostrar el mensaje "La bombilla ya estaba encendida" si intentas encender la bombilla cuando esta ya estaba encendida.
- apagar(): este método debe comprobar si la bombilla está previamente apagada y mostrar el mensaje "La bombilla ya estaba apagada" si intentas apagar la bombilla cuando ya estaba apagada.

Por último, se debe crear un nuevo fichero llamado *PruebaBombillaCondicional.java* que contenga una clase *PruebaBombillaCondicional* que tendrá un método *main* y que realizará las siguientes operaciones:

- Definir y crear un objeto de la clase BombillaCondicional
- Cambiar la marca a "Phillips"
- Cambiar la potencia a 100
- Encender y apagar
- Apagar, encender y encender.
- Apagar e imprimir su estado.
- Imprimir el texto: "La bombilla se ha encendido XX veces" utilizando el número que devuelve el método *obtieneNumVecesEncendida()*

Utiliza el depurador si te atascas o el programa empieza a tener "vida propia"

13. Se desea mejorar la clase *BombillaCondicional* del siguiente modo:
    - Se va a añadir una propiedad de tipo lógico llamada *fundida* que indicará si la bombilla está fundida o no.
    - Se debe modificar el funcionamiento de los siguientes métodos:
        - *encender():* Si la bombilla está fundida entonces se debe imprimir el texto "La bombilla está fundida y no se puede encender". En caso de que no esté fundida se debe incrementar en 1 la propiedad *numVecesEncendida* y, en el caso de que el valor de esta propiedad sea 10, se debe fundir la bombilla poniendo la propiedad fundida a cierto. Si el valor de *numVecesEncendida* es inferior a 10 entonces se debe encender la bombilla o bien, ofrecer el mensaje "La bombilla ya estaba encendida" si se está intentando encender cuando ya está encendida.
        - *apagar():* Si la bombilla está fundida entonces se debe imprimir el texto "La bombilla está fundida y no se puede apagar". En caso de que no esté fundida se realizarán las acciones que se venían haciendo anteriormente en este método.
        - *imprimeEstado()*: Si la bombilla está fundida entonces se debe imprimir el texto "La bombilla está fundida", en caso contrario se realizarán las acciones que se venían haciendo anteriormente en este método.
    - Por último, se debe abrir un fichero nuevo llamado *PruebaBombillaCondicional2* que defina la clase *PruebaBombillaCondicional2*. Esta clase contendrá el método *main* que hará las siguientes acciones:
        - Definir y crear un objeto *bc* de la clase *BombillaCondicional*
        - Encender y apagar el objeto *bc* 10 veces hasta que fundamos la bombilla.
        - Una vez fundida se debe apagar y encender una vez más y debe mostrarlos los mensajes que nos advierten la que la bombilla está encendida.
        - A continuación se debe imprimir el texto: "La bombilla se ha encendido XX veces" utilizando el número que devuelve el método *obtieneNumVecesEncendida()*
        - Por último se debe llamar al método *imprimeEstado()*

14. Escribe un programa que cree una clase Persona que tenga dos propiedades: nombre y edad. Además, tendrá los getters y setters correspondientes (getNombre(), setNombre(String nuevoNombre), …). A continuación, crea una nueva clase llamada PruebaEdades que tenga un método *main* que cree un objeto de la clase Persona, le asigne el nombre Pedro y la edad 30 y que muestre los siguientes mensajes según corresponda.

    - “Es un bebé” si su edad está comprendida entre 0 y 3 años.
    - “Es un niño” si su edad está comprendida entre 4 y 12 años.
    - “Es un adolescente” si su edad está comprendida entre 12 y 20 años.
    - “Es un adulto” si su edad es mayor o igual a 21 años.

15. Escribe un programa que le pida al usuario un número del 1 al 5 y le responda con una frase que rime con el número elegido. Por ejemplo: para el 1 “Que no pare ninguno”. Debes usar una condicional múltiple y si el usuario escoge un número fuera del rango debe salir un texto que diga “No me sé ninguna rima para el número XX”.

16. Escribe un programa que pida un número real al usuario y, a continuación, presente un menú de opciones del siguiente modo:

    MENU DE POTENCIAS
    1 – Calcular el cuadrado
    2 – Calcular el cubo
    3 – Calcular la raíz cuadrada
    Escoja una opción:

    Se deben mostrar los resultados correctos, utilizando el método Math.pow(…) para los cálculos. La raíz cuadrada de un número se puede calcular elevando dicho número a ½.

    Debes usar un switch. En el caso de que el usuario escoja una opción fuera del rango permitido debe mostrarse un mensaje que diga “Opción incorrecta”.

17. Escribe un programa que pida dos números enteros por teclado y después le pida al usuario un carácter que represente el operador a aplicar, pudiendo ser: suma ‘+’, resta ‘-‘, multiplicación ‘*’ y división ‘/’. Finalmente se ofrece un mensaje con el resultado.

    Debes usar un switch.

    Ejemplo de posible ejecución:
    Dime el primer número:
    4
    Dime el segundo número:
    10
    Dime el operador a aplicar:
    +
    El resultado de 4+10 es 14

18. Escribe un programa que escriba en pantalla las fases por las que pasa una lavadora. Para ello debes saber que:
    - La lavadora dispone de 2 ciclos de lavado: lavado corto y lavado largo.
    - El ciclo de lavado corto abarca los números del 1 al 4 y los mensajes que deberían mostrarse en la pantalla si empezamos el ciclo en el número 1 serían:
        - 1 – Prelavado
        - 2 – Lavado
        - 3 – Centrifugado
        - 4 – Fin

- El ciclo largo abarca los números del 5 al 12 y los mensajes que deberían mostrarse en la pantalla si empezamos el ciclo en el número 5 serían:
    - 5 – Prelavado
    - 6 – Lavado en caliente
    - 7 – Centrifugado 1
    - 8 – Lavado en frío
    - 9 – Suavizante
    - 10 - Centrifugado 2
    - 11 – Secado
    - 12 – Fin
- Debes preguntar al usuario en qué número va a empezar la lavadora y el programa mostrará los mensajes correspondientes. Si el usuario escoge un número fuera del rango le saldrá el mensaje "Opción incorrecta".

- Ejemplo de ejecución:

    ¿En qué número quieres poner la lavadora?
    2
    2 – Lavado
    3 – Centrifugado
    4 – Fin

CUESTIONES Y EJERCICIOS - 6ª VUELTA A LA ESPIRAL

19. Vamos a crear una clase *PruebaValidaEntrada* que tenga un método main que realice las siguientes acciones una tras otra:

    - Se le pide un número entero al usuario que tiene que ser mayor estricto que cero. Se debe almacenar en una variable llamada *mayorCero*. En el caso de que el usuario introduzca un número fuera del rango permitido, se le debe mostrar el mensaje "El número debe ser mayor que cero" y se le vuelve a pedir que lo introduzca de nuevo.
    - Se le pide un número entero al usuario que tiene que ser menor o igual que cero. Se debe almacenar en una variable llamada *menorIgualCero*. En el caso de que el usuario introduzca un número fuera del rango permitido, se le debe mostrar el mensaje "El número debe ser menor o igual que cero" y se le vuelve a pedir que lo introduzca de nuevo.
    - Se le pide un número real al usuario que tiene que estar comprendido entre 1.3 y 19.8 ambos incluidos. Se debe almacenar en una variable llamada *realRango*. En el caso de que el usuario introduzca un número fuera del rango permitido, se le debe mostrar el mensaje "El número debe ser estar comprendido entre 1.3 y 19.8" y se le vuelve a pedir que lo introduzca de nuevo.
    - Se le pide un carácter al usuario que tiene que sólo se considerará válido si es "S" o "N". Se debe almacenar en una variable llamada *siNo*. En el caso de que el usuario introduzca un carácter distinto, se le debe mostrar el mensaje "El carácter solo puede ser S o N" y se le vuelve a pedir que lo introduzca de nuevo.
    - Finalmente se deben mostrar en pantalla los valores de las variables leídas.

20. Descarga del repositorio el proyecto U2.P3.PruebaMenu, ábrelo y modifícalo del siguiente modo:
    - Ahora queremos que el usuario se le muestre el menú de opciones, elija una opción, le salga el resultado correspondiente y, sin cambiar los valores de *a* y *b,* se le vuelva a mostrar el menú de opciones.
    - El menú de opciones que se le muestra al usuario quedará como este:

      MENU DE OPCIONES:
      0 - SALIR
      1 - Suma (a+b)
      2 - Multiplicación (a*b)
      3 - División entera (a/b)
      Escoge una opción:

    - Este ciclo se repetirá hasta que elija la opción de SALIR del programa. Cuando esto ocurra se debe escribir un mensaje de despedida que diga: "Aaaadios".
    - Piensa bien dónde empieza y acaba el do… while y qué condición de permanencia hay que escribir.

21. Retoma el código de la *BombillaCondicional* del ejercicio 13, retócalo para que la bombilla se funda la quinta vez que se encienda. A continuación, crea una clase *PruebaBombillaMenu* que tenga un método main que realice las siguientes acciones:
    - Crea un objeto Bombilla
    - Ofrece un menú al usuario que le permita elegir entre las siguientes opciones:

      MENU DE OPCIONES:
      0 - SALIR
      1 - Enciende
      2 – Apaga
      3 – Imprime estado
      4 – Crea una nueva bombilla
      Escoge una opción:

    - Hasta que el usuario no seleccione la opción 0 – SALIR, se seguirá repitiendo el ciclo: MENÚ, ESCOGER, REALIZAR ACCIÓN, MENÚ, ESCOGER…
    - La opción 4 permite crear una nueva bombilla que será la que se utilice en los siguientes ciclos. Digamos que la bombilla anterior "la reemplazamos" por una nueva.

    Cuando tengas listo el programa, haz una ejecución de prueba para ver si todo funciona correctamente.

22. Escribir una clase llamada PruebaCuentaAsc.java que contenga un método main y que imprima los 100 primeros números naturales (1, 2, 3... 100) usando la instrucción while.

23. Escribir una clase llamada PruebaCuentaDesc.java que contenga un método main y que imprima del 100 hasta el 0 contando hacia atrás (100, 99, 98, ... 0) usando la instrucción while.

24. Escribir una clase llamada PruebaCuentaPar.java que contenga un método main y que imprima los números pares desde el 0 hasta el 100 (0, 2, 4, ..., 100) usando la instrucción while.

25. Escribir una clase llamada PruebaCuentaImpar.java que contenga un método main y que imprima los números impares desde el 1 hasta el 99 (1, 3, 5, ..., 99) usando la instrucción while.

26. Escribir una clase llamada PruebaTablaMulti9.java que contenga un método main y que imprima la tabla de multiplicar del 9 en el siguiente formato:
    0 x 9 = 0
    1 x 9 = 9
    ...
    10 x 9 = 90

27. Modifica el programa anterior para que ahora sea el usuario quien introduzca por teclado el número del que se va a mostrar su tabla de multiplicar.

28. Crea una clase llamada *ParesDescendientes* que tiene un método *main* que realiza las siguientes acciones:
    - Le pide al usuario un número entero comprendido entre 0 y 100. En el caso de que esté fuera de rango, el programa se "quejará" y le pedirá al usuario que vuelva a intentarlo.
    - A continuación, el programa debe imprimir los números pares comprendidos entre el número elegido y cero, de forma descendiente.
    - Para saber si un número es par se usa el operador % que me devuelve el resto de una división entera. Por ejemplo:

5 % 2 devuelve 1	5 | 2 1 2

6 % 2 devuelve 0	6 | 2 0 3

29. Escribe una clase llamada *Dado* que me permite simular la tirada de un dado de 6 caras. El diagrama UML sería el siguiente:

Dado
+ tirada() devuelve entero

El método *tirada* me devolverá un número aleatorio entre comprendido entre 1 y 6. Para calcular la tirada usaremos la siguiente expresión:

tirada = (int) (Math.random()*6 + 1);

Math.random devuelve un número aleatorio de tipo double cuyo valor está en el conjunto [0.0, 1.0). Es decir, puede devolver cero y 0.9999999… pero nunca llega a devolver 1.0.

Antes de seguir intenta "digerir" la fórmula anterior.

A continuación, vamos a crear una clase *PruebaDado* que tiene un método *main.* En el *main* se le pedirá al usuario qué número le gustaría que saliera del dado (el valor leído debe restringirse a los números del 1 al 6). A continuación, el programa debe repetir e imprimir las tiradas del dado hasta obtener el número que deseaba el usuario.

PISTA: puede que te ayude hacer una variable como *boolean coinciden* que se ponga a true cuando hayamos conseguido una tirada que coincida con la que quería el usuario.

30. Modifica el programa anterior para que ahora repita las tiradas del dado hasta obtener un 6 triple, es decir, que aparezca un 6 en 3 ocasiones.

31. Escribe la clase *CandadoNumerico* que simula el funcionamiento de un candado que se abre mediante una combinación numérica.

CandadoNumerico
+ numSecreto: entero
+ setNumSecreto (nuevoNumSecreto: entero) + intentaAbrir (numero: entero) devuelve lógico

El método *intentaAbrir* devuelve *true* si el número que recibe como parámetro coincide con el número secreto y *false* en caso contrario.

A continuación, crea una clase *ReventaCandado* que tenga un método *main* que realice las siguientes acciones:

- Crea un objeto *CandadoNumerico* y le pide al usuario que le ponga le establezca el número secreto. Este número debe estar comprendido entre 0 y 9999, si el usuario introduce un valor fuera del rango debe imprimir un mensaje de error y volver a leer

otro número hasta que lo haga correctamente.

- A continuación, el *main* debe hacer un bucle que utilice el método *intentaAbrir* con todas las combinaciones posibles hasta dar con el número secreto. En cuanto se encuentre el número secreto el bucle debe terminar y no seguir probando combinaciones.
- Cada vez que se falle en la combinación se debe mostrar el mensaje: "Probando con el número XXXX: no se abre"
- Cuando se acierte con la combinación secreta se debe mostrar el mensaje: "Probando con el número XXXX: SE ABRIÓ"

¿Qué cambios tendrías que hacer para que ahora la combinación secreta no la establezca el usuario, sino que se realice mediante la generación de un número aleatorio entre 0 y 9999?

32. Escribe una clase llamada *BuscaParejaMagica* que debe buscar una pareja de "dígitos mágicos". Cada dígito podrá valer 0…9. Entendemos como "pareja mágica" aquella formada por dos dígitos XY de forma que se cumplen simultáneamente las siguientes restricciones:
    - X es par
    - Y es impar
    - X+Y > 6
    - X * Y < 60

    Usa dos bucles anidados, uno para la X y otro para la Y de forma que puedas probar todas las combinaciones. Cuando encuentres un "número mágico" imprímelo en la pantalla y sigue buscando, tienes que mostrarlos todos.

33. Escribe una clase llamada *BuscaTrioMagico* que debe buscar un trio de "dígitos mágicos". Entendemos como "trío mágico" aquel formado por tres dígitos XYZ de forma que se cumplen simultáneamente las siguientes restricciones:
    - X es par
    - Y es impar
    - Z es par
    - X+Y+Z > 10
    - X * Y * Z < 90

34. Escribir una clase llamada *PruebaDibujoTriangulo* que contenga un método *main* que imprima en pantalla "Dime el tamaño del triángulo (5-30):" y que lea un número entero en el rango indicado. Acto seguido se debe dibujar un triángulo en la pantalla usando el carácter * del siguiente modo (ejemplo para tamaño = 5).

    *
    **
    ***
    ****
    *****

    Ayuda:
    - System.out.print("*"); // Añade un * sin pasar a la fila siguiente
    - System.out.println(""); // No imprime nada, solo pasa a la fila siguiente

CUESTIONES Y EJERCICIOS - 7ª VUELTA A LA ESPIRAL

35. Escribir una clase llamada *PruebaArrays* que realice las siguientes acciones:
    - Declara y crea un array de números reales llamado *notas* de 25 posiciones.
    - Le asigne las notas 4.3, 5.3, 5.6 y 7.8 a las posiciones 2, 3, 5 y 7 del array.
    - A continuación, imprima en pantalla las notas de las posiciones 0, 2, 4 y 5.
    - Además, declara y crea otro array de enteros llamado *metrosRecorridos* que se inicie con los siguientes valores: 200, 400, 800, 1500, 42000.
    - Ahora imprime los valores de las posiciones 0, 1 y 2.
    - Prueba a ver qué ocurre si quieres imprimir el valor de una posición fuera del rango permitido, por ejemplo, la posición 20.

36. Creamos una clase *Persona* que contenga solo las propiedades *rol* y *edad* y con sus respectivos métodos *getters* y *setters*. A continuación, vamos a escribir una clase llamada *PruebaArrayPersonas* que tenga un método *main* y que cree un array de 4 objetos de la clase Persona que va a representar las 4 personas que conviven en una vivienda. Prosigue realizando las siguientes acciones:
    - Crea un objeto Persona con el rol “padre” y la edad 43 y lo asigna a la primera posición del array.
    - Crea un objeto Persona con el rol “madre” y la edad 40 y lo asigna a la segunda posición del array.
    - Crea un objeto Persona con el rol “hijo” y la edad 24 y lo asigna a la tercera posición del array.
    - Crea un objeto Persona con el rol “hija” y la edad 14 y lo asigna a la cuarta posición del array.
    - En este momento, añade las instrucciones para que se imprima el siguiente texto para cada elemento del array: “La posición *POSICION* la ocupa el/la *ROL* con EDAD años”
    - A continuación, el “hijo” se va de casa ¿cómo podrías reflejar esa situación en el array?
    - Por último, el “abuelo” de 80 años viene a casa a ocupar el “hueco” del “hijo” ¿cómo lo reflejarías en el array?

37. Realiza un programa que cree una constante entera llamada NUM_TIRADAS que valdrá inicialmente 50.

    A continuación, debes crear un array de números enteros de tamaño NUM_TIRADAS y cargarlo con los resultados las tiradas de un dado de 6 caras utilizando la generación de números aleatorios del lenguaje Java. Después de esto, recorre el array para realizar estadísticas, de modo que el programa termine imprimiendo los siguientes mensajes:

```
Se ha analizado NUM_TIRADAS tiradas de un dado de 6 caras y se obtienen los
siguientes resultados:
El número 6 ha aparecido XX veces.
El número 5 ha aparecido XX veces.
El número 4 ha aparecido XX veces.
El número 3 ha aparecido XX veces.
El número 2 ha aparecido XX veces.
El número 1 ha aparecido XX veces.
```

    IMPORTANTE: si has utilizado correctamente la constante NUM_TIRADAS deberías poder cambiar el tamaño del array y el funcionamiento de todo el código solamente modificando el valor de la constante, así podríamos probar el comportamiento del código con 1000 tiradas, por ejemplo.

38. Realiza un programa que cree un array de objetos Traduccion. La clase Traduccion tendrá dos propiedades de tipo cadena llamadas: *english* y *spanish*. Además, tendrá los getters y setters correspondientes. Debes inicializar el array con 5 objetos Traduccion con los siguientes pares de valores:
    “To break down”, “Averiar”
    “To fix”, “Arreglar”
    “To reboot”, “Reiniciar”
    “To work”, “Funcionar”
    “To type”, “Teclear”

    A continuación, se debe recorrer el bucle utilizando la notación **for** (tipo elem: colección) para imprimir el siguiente mensaje para cada elemento:

    XXXXX en español se traduce por YYYYY

39. Realiza un programa que cree un array de 2000 posiciones con números enteros generados aleatoriamente en un rango entre 0 y 1000, ambos inclusive.
    A continuación, crea un código que le pida al usuario qué número quiere buscar, lo busque en el array e informe de la posición en la que está ese número, además, se debe imprimir en pantalla el contenido de esa posición del array para confirmar que ahí se encuentra el número buscado.
    En el caso de que no se encuentre el número se debe imprimir el texto “Número no encontrado”.

40. Realiza un programa que cree un array de 5 objetos Articulo en el que cada objeto se modela del siguiente modo:

Articulo
+ idArticulo: entero + nombre: texto + precio: real + static NOT_FOUND: Articulo
+ getters y setters

Teniendo en cuenta que:

- NOT_FOUND: es una constante estática de la clase Articulo que nos servirá como marca de "no encontrado. La definiremos así:

```
public static final Articulo NOT_FOUND = new Articulo();
```

  Esta marca la tendremos que usar en el ejercicio siguiente.

A continuación, crea 5 objetos con los valores que consideres oportunos y almacénalos en el array. Por último, pregunta al usuario qué identificador de artículo (idArticulo) desea buscar y realiza la búsqueda de ese elemento.

Si se encuentra el elemento se deben mostrar los valores de las propiedades del artículo encontrado. En caso contrario se debe mostrar "artículo no encontrado".

41. Aprovechando la clase Articulo del ejercicio anterior, vamos a crear lo siguiente:

Carrito
+ carrito: array de Articulo + numArticulos: entero + inicializado: lógico
+ inicializaCarrito (numPosiciones: entero) + getNumArticulos () devuelve entero + guardaArticulo (nuevoArtic: Articulo) + muestraArticulos () + buscaArticuloPorId (idABuscar: entero) devuelve Articulo + getPrecioTotalCarrito () devuelve real

Teniendo en cuenta las siguientes restricciones:

- La propiedad *numArticulos* llevará la cuenta del número de artículos introducidos en el carro.
- El método *inicializaCarrito* crea un array de referencias a Articulo con *numPosiciones* posiciones y lo guarda en la propiedad *carrito.* Este método debe ser llamado en primer lugar, antes que cualquier otro del objeto *Carrito*, para garantizarlo usaremos la propiedad lógica *inicializado*, que se pondrá a true cuando ejecutemos este método.
- El método *guardaArticulo* almacena el artículo que se recibe como parámetro en la siguiente posición libre del array. Pero además realizará las siguientes comprobaciones:

    - Si *inicializado* es falso entonces imprimirá el mensaje "Error: carrito aún no inicializado" y terminará el método.
    - Si el número de artículos del carro es igual que el tamaño del array se imprimirá el mensaje "Error: carrito lleno".
    - Si el objeto que se intenta guardar en el carro es igual (==) a la marca NOT_FOUND entonces se imprimirá el mensaje "Error: no se puede guardar la marca de NO ENCONTRADO"

- El método *muestraArticulos* recorre las posiciones rellenas del array mostrando los artículos que contiene. De cada artículo se mostrará un mensaje como el siguiente: *"Id=XX, nombre=NNNNN, precio=PPP.PP€".* En el caso de que no tenga ningún artículo se mostrará el mensaje: "No hay artículos en el carrito". En el caso de que el carrito aún no haya sido inicializado se mostrará "Error: carrito aún no inicializado" y terminará el método.
- El método *buscaArticuloPorId* devuelve el artículo cuya propiedad *id* coincida con la que se recibe como parámetro. En el caso de que no lo encuentre, devuelve una "marca de no encontrado" (NOT_FOUND). En el caso de que el carrito aún no haya sido inicializado se mostrará "Error: carrito aún no inicializado", terminando el método y devolviéndose un NOT_FOUND.
- El método *getPrecioTotalCarrito* devuelve la suma de los precios de todos los productos del carrito. En el caso de que no haya ningún producto, se devuelve 0.0. En el caso de que el carrito aún no haya sido inicializado se mostrará "Error: carrito aún no inicializado", terminando el método y devolviéndose un -1.0.

Por último, crea una clase *PruebaCarrito* que cree un objeto Carrito, lo inicialice y pruebe todos los métodos de la clase Carrito.

42. Escribir una clase llamada *PruebaDiametro.java* que contenga el siguiente código:

```
public class PruebaDiametro {
    public static void main (String args[]) {
            double diametro;
            double limite;
        Teclado t = new Teclado();

            limite = 1.4;

        // Escribe las instrucciones para pedir por teclado el valor del diámetro
            // Pon tus sentencias condicionales aquí
    }
}
```

Escribe el código que falta para que se realicen las siguientes acciones:

- Escribir en pantalla el mensaje "Dime el diametro de la rueda: " y acto seguido leer por teclado un valor double y guárdalo en la variable *diametro*
- Si el diámetro es superior a 1.4 mostrará un mensaje en la pantalla que diga “La rueda es para un vehículo grande”.
- Si el diámetro es igual a 1.4 mostrará un mensaje en la pantalla que diga “La rueda es para

un vehículo mediano".

- Si no se cumplen ninguna de las dos condiciones anteriores mostrará en pantalla el mensaje "La rueda es para un vehículo pequeño".

43. Escribir una clase llamada *PruebaDiametro2.java* partiendo de la clase del ejercicio anterior que lo modifique para que ahora se comporte del siguiente modo:

- Si el diámetro es superior a 1.4 mostrará un mensaje en la pantalla que diga "La rueda es para un vehículo grande".
- Si el diámetro es menor o igual a 1.4 y mayor que 0.8 mostrará un mensaje en la pantalla que diga "La rueda es para un vehículo mediano".
- Si no se cumplen ninguna de las dos condiciones anteriores mostrará en pantalla el mensaje "La rueda es para un vehículo pequeño".

44. Escribir una clase llamada *PruebaDiametro3.java* partiendo de la clase del ejercicio anterior que lo modifique para que ahora se comporte del siguiente modo:

- Le añadimos una variable nueva *double* grosor.
- Mostramos al usuario el texto "Dime el grosor de la rueda: " y acto seguido leemos por teclado un valor double y guárdalo en la variable *grosor*
- Si el diámetro es superior a 1.4 y el grosor es inferior a 0.4 mostrará un mensaje en la pantalla que diga "El grosor para esta rueda es inferior al recomendado".
- Si el diámetro es menor o igual a 1.4 pero mayor que 0.8 y el grosor es inferior a 0.25 mostrará por pantalla "El grosor para esta rueda es inferior al recomendado".

45. Escribir una clase llamada *PruebaCuantoCompro.java* que contenga el siguiente código:

```
public class PruebaCuantoCompro {
    public static void main (String args[]) {
        double precioArticulo;
        double dineroTotal;
            double numArticulos;
        // Pon tus instrucciones aquí
    }
}
```

- Se debe mostrar el texto "Cual es el precio de un articulo?: "
- Después se leerá del teclado el valor que introduzca el usuario y se guardará en la variable *precioArticulo.*
- Acto seguido se debe mostrar el texto "De cuanto dinero dispones?: "
- Después se leerá del teclado el valor que introduzca el usuario y se guardará en la variable dineroTotal.

- Ahora se debe comprobar que el *precioArticulo* sea un número entero mayor que cero.
  - Si es así se divide el *dineroTotal* entre el *precioArticulo* y el resultado se guarda en *numArticulos.* Por último, se mostrará un texto en el que se pueda leer: "Con *dineroTotal* puedes comprar *numArticulos* artículos a un precio de *precioArticulo* euros" *(reemplazando cada palabra en cursiva por su valor...)*
- En el caso de que *precioArticulo* sea un número menor o igual que cero entonces se debe mostrar un texto que sea: "El precio de un artículo debe ser un entero positivo"

46. Escribir una clase llamada PruebaMuchosTriangulos.java que se reutilice el código de la clase *PruebaDibujoTriangulo.java* del ejercicio 14 del siguiente modo:

Se imprime "Cuantos triangulos quieres imprimir?: ".

Después se le pide al usuario un número entero utilizando la clase Teclado.java

Se imprime "Cuál es el tamaño del triángulo?: ".

Se lee un entero por teclado para el tamaño del triángulo.

Acto seguido se imprimen tantos triángulos como haya indicado el usuario del tamaño elegido.

Ejemplo para un número de triángulos igual a tres y 5 de tamaño:

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

47. Juego del ahorcado.

48. Escribir una clase llamada *PruebaAdivinaNumero.java* que comience así:

```
import java.util.Random;
public class PruebaAdivinaNumero {
        public static void main(String arg[ ]) {
                Random rnd = new Random();
                int intento;
                boolean acierto = false;
                int adivina =  (int) (rnd.nextDouble() * 100+1);
                // Pon tu código aquí
        }
}
```

- La variable *adivina* tendrá un número aleatorio comprendido entre 1 y 100.
- La variable *intento* guardará los sucesivos números que introduzca el usuario intentando adivinar el número guardado en *adivina.*
- El programa debe imprimir el texto "Dime un número: " y pedir al usuario que introduzca por teclado un número entero que se guardará en la variable *intento.* A continuación, se compararán ambos números pudiendo resultar lo siguiente:
  - Si *adivina* == intento se debe imprimir "Acertaste!!" y el programa debe acabar.
  - Si *adivina* > intento entonces se debe imprimir "El número a adivinar es mayor que *intento*".
  - Si *adivina* < intento entonces se debe imprimir "El número a adivinar es menor que *intento*".
- Mientras el usuario no acierte el número se le seguirán haciendo las preguntas.