PROGRAMACIÓN
ACTIVIDADES Unidad 05

CLASES

1.-Diseñar la clase CuentaCorriente, que almacena los datos: DNI y nombre del titular, así como el saldo. Las operaciones típicas con una cuenta corriente son:

- Crear una cuenta: se necesita el DNI y nombre del titular. El saldo inicial será 0.
- Sacar dinero: el método debe indicar si ha sido posible llevar a cabo la operación, si existe saldo suficiente.
- Ingresar dinero: se incrementa el saldo.
- Mostrar información: muestra la información disponible de la cuenta corriente.

2.-En la clase CuentaCorriente sobrecargar los constructores para poder crear objetos.

- Con el DNI del titular de la cuenta y un saldo inicial.
- Con el DNI, nombre y el saldo inicial.

Escribir un programa que compruebe el funcionamiento de los métodos.

3.- Modificar la visibilidad de la clase CuentaCorriente para que sea visible desde clases externas y la visibilidad de sus atributos para que:

- saldo no sea visible para otras clases.
- nombre sea público para cualquier clase.
- dni solo sea visible por clases vecinas.

Realizar un programa para comprobar la visibilidad de los atributos.

4.- Todas las cuentas corrientes con las que se va a trabajar pertenecen al mismo banco. Añadir un atributo que almacene el nombre del banco (que es único) en la clase CuentaCorriente. Diseñar un método que permita recuperar y modificar el nombre del banco (al que pertenecen todas las cuentas corrientes).

5.- Existen gestores que administran las cuentas bancarias y atienden a sus propietarios.

Cada cuenta, en caso de tenerlo, cuenta con un único gestor. Diseñar la clase Gestor de la que interesa guardar su nombre, teléfono y el importe máximo autorizado con el que pue den operar. Con respecto a los gestores, existen las siguientes restricciones:

Un gestor tendrá siempre un nombre y un teléfono.

Si no se asigna, el importe máximo autorizado por operación será de 10000 euros.

Un gestor, una vez asignado, no podrá cambiar su número de teléfono. Y todo el mundo podrá consultarlo.

El nombre será público y el importe máximo solo será visible por clases vecinas.

Modificar la clase CuentaCorriente para que pueda disponer de un objeto Gestor. Escribir los métodos necesarios.

Cada cuenta corriente tendrá una referencia a un objeto de tipo Gestor

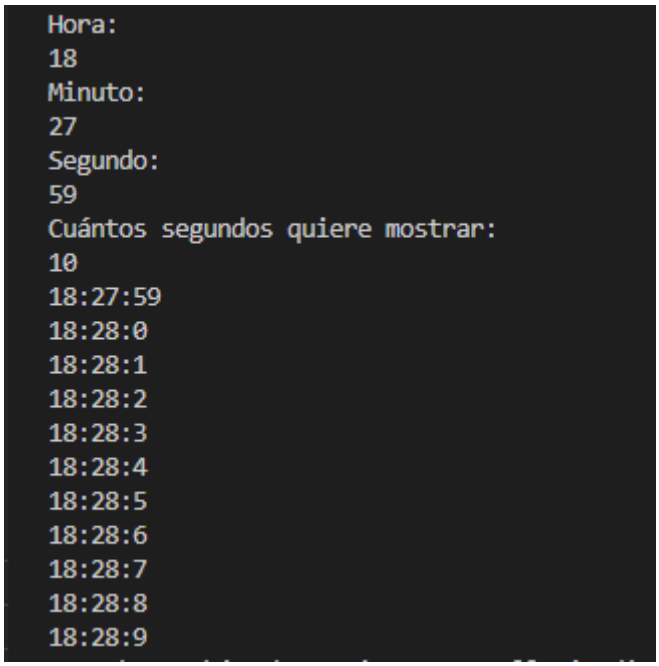
6.- Escribir un programa que lea por teclado una hora cualquiera y un número n que representa una cantidad en segundos. El programa mostrará la hora introducida y las n siguientes, que se diferencian en un segundo. Para ello hemos de diseñar previamente la clase Hora que dispone de los atributos hora, minuto y segundo. Los valores de los atributos se controlarán mediante métodos set/get.

--La clase Hora es muy simple y dispone de los atributos: hora, minutos y segundos. Estos serán privados y para acceder a ellos usaremos métodos get/set.

Internamente vamos a utilizar byte para almacenar los atributos, pero desde fuera de la clase nos interesa dar la sensación que los atributos son int: estamos ocultando la verdadera implementación.

No escribiremos ningún constructor.

Ejemplo;



7.- Diseñar la clase Texto que gestiona una cadena de caracteres con algunas características:

- La cadena de caracteres tendrá una longitud máxima que se especifica en el constructor.
- Permite añadir un carácter al principio o al final, siempre y cuando no se exceda la longitud máxima, es decir, exista espacio disponible.
- Igualmente, permite añadir una cadena, al principio o al final del texto, siempre y cuando no se rebase el tamaño máximo establecido.
- Es necesario saber cuántas vocales (mayúsculas y minúsculas) hay en el texto.

- Cada objeto de tipo Texto tiene que conocer la fecha en la que se creó, así como la fecha y hora de la última modificación efectuada.
- Deberá existir un método que muestre la información que gestiona cada texto.
- La clase Texto contendrá:
    - un String donde guardaremos la cadena de caracteres
    - un numero entero que indicará la longitud maxima del texto
    - fecha de creacion del texto
    - fecha y hora de la ultima modificacion;

Contructor publicTexto (int logitudMax)

public void addFinal(char c)

public void addFinal(String c)

public void addInicio(char c)

public void addInicio(String c)

public void mostrar()

public int numVocales()

private boolean esVocal(char c)

Debes de importar las clases

import java.time.LocalDate;

import java.time.LocalDateTime;

8.- Definir una clase que permita controlar un sintonizador digital de emisoras FM; concreta mente, se desea dotar al controlador de una interfaz que permita subir (up) o bajar (down la frecuencia (en saltos de 0,5 MHz) y mostrar la frecuencia sintonizada en un momento dado (display). Supondremos que el rango de frecuencias para manejar oscila entre los 80 MHz y los 108 MHz y que, al inicio, el controlador sintonice la frecuencia indicada en el constructor o 80 MHz por defecto. Si durante una operación de subida o bajada se sobrepasa

uno de los dos límites, la frecuencia sintonizada debe pasar a ser la del extremo contrario. Escribir un pequeño programa principal para probar su funcionamiento.

/* La clase tiene un atributo real que almacena la frecuencia a la que estamos

* sintonizando, junto a los métodos necesarios para utilizar el sintonizador. */

9.- Modelar una casa con muchas bombillas, de forma que cada bombilla se pueda encender o apagar individualmente. Para ello, hacer una clase Bombilla con una variable privada que indique si está encendida o apagada, así como un método que nos diga el estado de una bombilla concreta. Además, queremos poner un interruptor general, de forma que, si este se apaga, todas las bombillas quedan apagadas. Cuando el interruptor general se activa, las bombillas vuelven a estar encendidas o apagadas, según estuvieran antes. Cada bombilla se enciende y se apaga individualmente, pero solo responde que está encendida si su interruptor particular está activado y además hay luz general.

/* La clase Bombilla se implementa con un indicador de estado (apagada/encendida), que será individual para cada bombilla (para cada objeto bombilla). Además, el interruptor general (que afecta a todas las bombillas) se implementa con un atributo estático, cuyo valor será el mismo para todos los objetos de la clase. */

10.- Hemos recibido el encargo de un cliente para definir los paquetes y las clases necesarias (solo implementar los atributos y los constructores) para gestionar una empresa ferroviaria, en la que se distinguen dos grandes grupos: el personal y la maquinaria. En el primero se ubican todos los empleados de la empresa, que se clasifican en tres grupos: los maquinistas, los mecánicos y los jefes de estación. De cada uno de ellos es necesario guardar:

Maquinistas: su nombre, DNI, sueldo y el rango que tienen adquirido.

Mecánicos: su nombre, teléfono (para contactar en caso de urgencia) y en qué especialidad desarrollan su trabajo (esta puede ser: frenos, hidráulica, electricidad o motor).

Jefes de estación: su nombre, DNI y la fecha en la que fue nombrado jefe de estación.

En la parte de maquinaria podemos encontrar trenes, locomotoras y vagones. De cada uno de ellos hay que considerar:

Vagones: tienen un número que los identifica, una carga máxima (en kilos), la carga actual y el tipo de mercancía con el que están cargados.

Locomotoras: disponen de una matrícula (que las identifica), la potencia de sus motores y una antigüedad (año de fabricación). Además, cada locomotora tiene asignado un mecánico que se encarga de su mantenimiento.

Trenes: están formados por una locomotora y un máximo de 5 vagones. Cada tren tiene asignado un maquinista que es responsable de él.

Todas las clases correspondientes al personal (Maquinista, Mecánico y Jefe Estación) serán de uso público. Entre las clases relativas a la maquinaria solo será posible construir, desde clases externas, objetos de tipo Tren y de tipo Locomotora. La clase Vagón será solo visible por sus clases vecinas.

11.- Las listas son estructuras dinámicas de datos donde se pueden insertar o eliminar elementos de un determinado tipo sin limitación de espacio.

Implementar la clase Lista correspondiente a una lista de números de la clase Integer. Los números se guardarán en una tabla que se redimensionará con las inserciones y eliminaciones, aumentando o disminuyendo la capacidad de la lista según el caso.

Entre los métodos de la clase, se incluirán las siguientes tareas:

- Un constructor que inicialice la tabla con un tamaño 0.
- Obtener el número de elementos insertados en la lista.

- Insertar un número al final de la lista.
- Insertar un número al principio de la lista.
- Insertar un número en un lugar de la lista cuyo índice, que es el de la tabla, se pasa como parámetro.
- Añadir al final de la lista los elementos de otra lista que se pasa como parámetro.
- Eliminar un elemento cuyo índice en la lista se pasa como parámetro.
- Obtener el elemento cuyo índice se pasa como parámetro.
- Buscar un número en la lista, devolviendo el índice del primer lugar donde se encuentre. Si no está, devolverá-1.
- Mostrar los elementos de la lista por consola.

Recuerda el uso de Arrays.CopyOf() y System.arraycopy();

```java
package AR_Clases711;

public class MainLista {
    Run | Debug
    public static void main(String[] args) {
        Lista l1 = new Lista();
        Lista l2 = new Lista();

        l1.insertarFinal(nuevo: 4);
        l1.insertarFinal(nuevo: 5);
        l1.insertarFinal(nuevo: 6);
        l1.mostrar();

        l1.insertarPrincipio(nuevo: 3);
        l1.insertarPrincipio(nuevo: 2);
        l1.insertarPrincipio(nuevo: 1);
        l1.mostrar();

        l1.insertar(posicion: 2, nuevo: 99);
        l1.mostrar();

        l1.eliminar(indice: 2);
        l1.mostrar();

        System.out.println(l1.buscar(claveBusqueda: 4));

        l2.insertarFinal(nuevo: 10);
        l2.insertarFinal(nuevo: 20);
        l2.insertarFinal(nuevo: 30);
        l2.insertarFinal(nuevo: 40);
        l2.insertarFinal(nuevo: 50);
        l2.mostrar();

        l1.insertarFinal(l2);
        l1.mostrar();

    }
}
```

```
Lista: [4, 5, 6]
Lista: [1, 2, 3, 4, 5, 6]
Lista: [1, 2, 99, 3, 4, 5, 6]
Lista: [1, 2, 3, 4, 5, 6]
3
Lista: [10, 20, 30, 40, 50]
Lista: [1, 2, 3, 4, 5, 6, 10, 20, 30, 40, 50]
PS C:\Users\joan\OneDrive - Conselleria d'Educació\Programacion2223\JavaFolderBook\es.simarro.inicios\src>
```

12.- Añadir a la clase Lista el método estático:

Lista concatena (Lista 11, Lista 12)

que construye y devuelve una lista que contiene, en el mismo orden, una copia de todos los elementos de 11 y 12.