Ejercicios de iniciación.

1. Diseña un programa en Java que trabaje con una clase llamada Alumno. Serán atributos del alumno su número de expediente, nombre, apellidos, fecha de nacimiento y curso en el que se matricula. La clase debe incluir un constructor y los métodos get y set correspondientes y toString. Instancia varios objetos de esta clase y prueba los métodos creados.
2. Diseña una clase llamada Empleado con las características que se citan a continuación. Prueba sus métodos.
    a. Atributos. id, nombre, apellidos, salario mensual.
    b. Métodos: constructor, getters y setters.
    c. Añade los métodos:
        i. getNombreCompleto. Devuelve el nombre del empleado con la forma: Apellidos, Nombre.
        ii. getSalarioAnual. Este se obtiene multiplicando por doce el salario mensual.
        iii. incrementaSalario (int porcentaje). Incrementa el salario en el porcentaje indicado.
        iv. toString.
    d. Crea dos empleados y prueba los métodos programados, incrementa su salario y obtén el salario anual.

3. Diseña un programa Java que guarde información sobre los taxis de una compañía. De cada taxi debe guardarse la matrícula, distrito en el que opera (sur, norte, oeste…), tipo motor (diesel o gasolina) y coordenadas (latitud y longitud por separado) en las que se ubica. Crea la clase necesaria y añade métodos get/set, dos constructores (uno con todos los atributos y otro sin coordenadas, poner a 0). Diseña otro método que servirá para comprobar si dos taxis operan en el mismo distrito devolviendo true o false. Este método recibirá un taxi como parámetro. Añade un nuevo método que devuelva un String con las coordenadas en conjunto. Crea dos objetos Taxi y prueba sus métodos.

4. Diseña un programa Java que trabaje con una clase llamada DvdCine.
    a. Su atributos serán: título, director, productora, actores principales, género, resumen y duración.
    b. Debe incluir un constructor que reciba como parámetros los atributos.
    c. Crea un método llamado esThriller que devuelva true o false según la película sea o no de este género.
    d. Crea todos los métodos gets/sets.
    e. Crea un método llamada mismaProductora que reciba un objeto de la clase DvdCine como parámetro. Este método devuelve true o false si el objeto utilizado con el método es de la misma productora que el pasado como parámetro.
    f. Crea objetos y prueba estos métodos.

5. En un nuevo proyecto, copia la clase Alumno del ejercicio 1, y añade:

a. Dos atributos que guarden sendas notas (tipo double).
b. Crea los métodos get/set correspondientes, modifica también el constructor para que incluya estas notas.
c. Crea un método que devuelva la media de las notas (double).
d. Crea un método que devuelva número de expediente, nombre y nota media.

Ejercicios de profundización.

6. Diseña un programa Java que trabaje con la clase Alumno del ejercicio 5, copiándola y pegándola en este. Crea otra clase llamada GestionAlumnos con las siguientes características:
    - Atributos: array de 10 Alumnos
    - Métodos:
        - llenarArray, rellena el array de alumnos con datos.
        - mostrarAlumnos, muestra en pantalla cada alumno y su nota media.
        - mostrarNotas, muestra nº de expediente, nombre y nota media de cada alumno.
        - buscarAlumnoPorNumeroExpediente. Recibe como parámetro un nº de expediente y devuelve/retorna al alumno con ese nº expediente, o null si no lo encuentra.
    - Prueba la clase GestionAlumnos.

7. Diseña un programa Java que trabaje con la clase DvdCine creada en el ejercicio anterior. Crea una clase (GestionDvd) que incluya como atributo un array de objetos pertenecientes a la clase DvdCine y como métodos las opciones del menú.

    El menú de la aplicación sería:
    Menú de opciones:
        1. Mostrar la lista de películas (Título y productora)
        2. Mostrar la película de más duración (Título y minutos)
        3. Pedir un género y mostrar el título de las pelis de ese género.
        4. Duración de la peli. Pedir título y mostrar su duración.

    Aquí tienes una serie de datos de partida.

    ```
    private DvdCine [] pelis = {
        new DvdCine ("El hobbit. La desolación de Smaug", "Peter Jackson", "New Line Cinema." + "Metro-Goldwyn-Mayer", "Ian McKellen, Martin Freeman", "Ciencia ficción", "Bla...", 123),
        new DvdCine ("El Padrino", "Francis Ford Copola", "Paramount Pictures" , "Al Pacino, Marlon Brando", "Drama", "Bla...", 175),
        new DvdCine ("Titanic", "Francis Ford Copola", "Paramount Pictures. 20th Century Fox" , "Di Caprio", "Drama", "Bla...", 123),
        new DvdCine ("El Rey León", "WD", "Walt Dysney", "------", "Animación", "Bla...", 86),
        new DvdCine ("Los Vengadores", "--", "Marvel Studios. Walt Dysney", "--", "Ciencia ficción", "Bla...", 114),
    ```

new DvdCine ("Avatar", "Francis Ford Copola", "20th Century Fox", "--", "Ciencia ficción", "Bla...", 126),
new DvdCine ("Harry Potter. Las reliquias de la muerte", "--", "Warner Bros.", "--", "Ciencia ficción", "Bla...", 131),
new DvdCine ("El señor de los anillos. El retorno del rey", "Francis Ford Copola", "New Line Cinema", "--", "Ciencia ficción", "Bla...", 175),
new DvdCine ("Toy Story 3", "----", "Dysney/Pixar", "--", "Animación", "Bla...", 78),
new DvdCine ("The Dark Knight Rises", "--", "Warner Bros.", "--", "Ciencia ficción", "Bla...", 99)};

8. Clase Mates. Crea una clase con este nombre y con las siguientes características.
   a. Atributo constante con el valor de PI.
   b. Método estático que retorna la longitud de una circunferencia a partir de un radio pasado como argumento.
   c. Método estático que devuelva si un número es par o no (true o false).
   d. Método estático que devuelva si un número es primo o no (true o false).
   e. Método estático que devuelva la suma de los divisores de un número.
   f. Método estático que devuelva verdadero o falso si dos números son amigos o no (true o false).

9. Diseña una aplicación Java que trabaje con una clase llamada **Programa** (Software). Cada programa tendrá como atributos su nombre, versión, función (sistema operativo, servidor de bases de datos, ofimática, …), año de lanzamiento, empresa que lo desarrolla, tipo de licencia y precio.
   Diseña también una clase llamada **GestionProgramas** que contendrá:
   - un array de 25 elementos de la clase Programa
   - métodos que sirvan para programar las funciones del siguiente menú:

   Menú
   1. Insertar nuevo programa (controlar elementos libres)
   2. Mostrar lista de software
   3. Filtrar por licencia
   4. Filtrar por función
   5. Eliminar el último elemento
   6. Buscar programa por nombre y retornar programa
   7. Salir

10. Diseña un programa Java que gestione un Parking con capacidad para 200 vehículos.
    a. Cada vez que entra un vehículo en el parking guardamos su matrícula, el instante en que entró en milisegundos y decrementamos el contador de plazas disponibles.
    b. Cuando un vehículo deja el parking libera una plaza.
    c. El precio del segundo es de 0,015 euros. Antes de salir, el propietario debe pagar el importe en función del tiempo que el vehículo estuvo en el parking.

Para realizar este cálculo utilizaremos la clase Date (java.util.Date) como sigue:

- De esta forma guardamos el instante actual en el objeto miFecha:
  Date miFecha= new Date();
-
- En nuestro caso debemos calcular la diferencia entre el instante de entrada y el de salida
- Así calculamos la diferencia entre 2 instantes, el actual y el pasado como parámetro. Los instantes están expresados en milisegundos, por tanto al dividir entre 1000 se obtienen segundos.

```
public static long secondsDiff(Date date) {
    return (new Date().getTime() - date.getTime()) / 1000;
}
```

En definitiva, debemos crear:

- clase Vehiculo con los atributos matrícula e instante de entrada (Date).
- clase Parking.
  - Atributos:
    - array de N vehículos.
    - plazas libres
  - Métodos.
    - entradaVehículo
    - salidaVehículo
    - mostrarVehiculosParking
    - mostrarPlazasDisponibles

11. Diseña un programa Java que sirva para gestionar los vehículos de una empresa de alquiler.
    - De cada vehículo se guarda matrícula, marca, modelo, potencia, consumo medio y precio de alquiler diario. Diseña la clase **Vehiculo** con los atributos que se han indicado, getters/setters y constructor. Diseña también un segundo constructor que reciba como parámetros la matrícula y el precio diario de alquiler.
    - Diseña una segunda clase llamada **GestionVehiculos**, que tendrá un array de 100 vehículos como atributo. El constructor de esta clase llenará el array con datos de 100 vehículos. Esta clase tendrá los métodos **filtrarPorConsumo**, que recibe un parámetro double relativo al consumo y devolverá una lista de los vehículos con un consumo igual o inferior al pasado. Diseña también dentro de esta clase el método **mostrarTodos**, que imprimirá en pantalla datos de todos los vehículos. Por último, diseña el método **precioMedioAlquiler** que devolverá un double con este dato.
    - Desde el main crea un objeto de la clase **GestionVehiculos** y llama a sus métodos. **No es necesario crear un menú.**

12. Diseñar una clase llamada Password con las características:

- Que tenga los atributos longitud y contraseña . Por defecto, la longitud será de 8.
- Los constructores serán los siguientes:
  - Un constructor por defecto. Generará una contraseña de 8 caracteres.
  - Un constructor con la longitud que nosotros le pasemos. Generará una contraseña aleatoria con esa longitud.
- Métodos:
  - esFuerte(): devuelve un booleano indicando si la contraseña es fuerte o no. Para que sea fuerte debe tener más de 2 mayúsculas, más de 1 minúscula y más de 3 dígitos.
  - generarPassword(): genera la contraseña del objeto con la longitud que tenga. Es un método privado.
  - Métodos get para contraseña y longitud.

13. Diseña un programa Java que sirva para gestionar productos que se venden en una ferretería. De cada producto se guarda un código, nombre, tipo (pintura, herramienta, recambio, cinta,...), precio de venta, precio de compra y unidades disponibles.

    Crea una clase **Producto** con los atributos nombrados, los métodos getters, setters y el constructor. Diseña también una clase llamada **GestionProductos** que tendrá un array de 100 productos como atributo. Esta clase llevará los siguientes métodos:

    a. Constructor. Llenará el array de productos con valores al azar.
    b. filtrarPorPrecioVenta. Pasar un precio y mostrar todos productos con precio superior al pasado.
    c. filtrarPorTipo. Pasar un tipo de producto y mostrar los que coinciden con él.
    d. mostrarDiferencia. Mostrar nombre de producto, precio de compra, precio de venta y la diferencia entre ambos.
    e. mostrarProductoPosicion. Recibe como parámetro una posición y muestra el producto que está en esa posición del array.
    f. cambiarPor. Este método recibe un producto y una posición y sustituirá en el array el producto situado en la posición indicada por el producto pasado como parámetro. Devuelve verdadero o falso, según se haya podido hacer el cambio o no.

    En el main crea un objeto de la clase GestionProductos y prueba sus métodos.

14. Descarga y estudia este proyecto y completa la clase GestionEmpleados con los siguientes métodos. Prueba los métodos creados en la clase principal (App) de la aplicación.
    a. buscarEmpleadoPorId. Recibe un id de empleado y retorna el empleado con esa id o null si no lo encuentra.

b. empleadosCorreo. Muestra todos los empleados cuyo correo electrónico contenga una cadena pasada como parámetro. Utiliza el método “contains” de la clase String. Busca información sobre el mismo.
c. filtrarPorSalario. Muestra id, nombre, apellidos y salario de los empleados cuyo salario está comprendido entre un valor mínimo y un máximo pasados como parámetros.
d. actualizaSalario. Actualizar el salario de un empleado. El método recibe la id y el nuevo salario. Retorna true si se llevó a cabo el cambio, false en caso contrario.
e. nacidosEn. Recibe un año y muestra los empleados nacidos en ese año (toString).
f. nacidosEn. Recibe un año y un mes y muestra los empleados nacidos ese año y mes (toString).

15. Descarga y estudia este proyecto y completa la clase GestionCiudades con los siguientes métodos. Prueba los métodos creados en la clase principal (App) de la aplicación.
    a. buscarCiudadPorId. Recibe una id y retorna la ciudad con esa id o null si no existe.
    b. buscarCiudadMasPoblada. Retorna la ciudad más poblada.
    c. mostrarCiudadesDelPais. Recibe un país como argumento e imprime las ciudades de ese país.
    d. comparaCiudades. Método estático que recibe dos objetos ciudad (Ciudad) como parámetro y retorna true o false si son iguales o no. Dos ciudades son iguales si tienen el mismo id y nombre.
    e. mostrarVariasCiudades. Recibe varios ids como argumentos y muestra las ciudades (toString) asociadas a esas ids. (varagrs)
    f. mostrarCiudadesPaises. Recibe varios códigos de país como argumentos y muestra las ciudades (toString) pertenecientes a esos países.