

PROGRAMACIÓN JAVA EJERCICIOS-2 TEMA 6. Clases compuestas y arrays de objetos.

1. Crea una clase Persona con los siguientes atributos: nombre, fechaNacimiento(objeto de tipo Fecha, clase creada en el Ejercicio 7 de ejercicios básicos POO), dirección, códigoPostal y ciudad.

Esta clase se va a utilizar en un programa que pida por teclado los datos de varias personas y las guarde en un ArrayList de objetos de tipo Persona y a continuación muestre información sobre ellas.

Clase principal del proyecto y el método main:


```
public class Ejercicios2_1 {  
  
    static ArrayList<Persona> personas = new ArrayList<>();  
    static Scanner sc = new Scanner(System.in);  
  
    public static void main(String[] args) {  
  
        leerPersonas();  
        if (numeroDePersonas() > 0) {  
            System.out.println("\nPersonas introducidas: ");  
            mostrar();  
            System.out.println("\nPersona de mayor edad: ");  
            System.out.println(personaMayorEdad());  
            System.out.println("\nNúmero de personas que viven en Elche: "  
                               + cuantasPersonasVivenEn("Elche"));  
            System.out.println("\nNúmero de personas mayores de edad : "  
                               + personasMayoresDeEdad());  
        }  
    }  
    //Resto de métodos  
}
```

Escribe a continuación del método main el código del resto de métodos:

- Método leerPersonas: pide por teclado los datos de las personas y las añade al array.
- Método mostrar: muestra los datos de todas las personas introducidas.
- Método númeroDePersonas: devuelve el número de personas que contiene el array.
- Método personaDeMayorEdad: devuelve la persona (objeto Persona) de mayor edad. Si hay varias devuelve la primera encontrada.
- Método cuantasPersonasVivenEn: método que recibe el nombre de una población y devuelve cuántas personas viven en ella.
- Método personasMayoresDeEdad: método que devuelve cuántas personas son mayores de edad.

Sobrescribe el método toString() en la clase Persona para mostrar los datos de la siguiente forma:

```
Nombre: Adolfo Pérez Clavarana  
Fecha Nacimiento: 12-07-1987  
Dirección: C/ La Isla 19 2-B  
03030 Elche
```

 Recuerda que desde NetBeans puedes generar el código del método *toString()* pulsando: *botón derecho -> insertar código -> toString()*.

2. Crea una clase Empleado que tenga como atributos:

- DNI.
- Nombre.
- Sueldo base.
- Horas extra realizadas en el mes.
- Tipo de IRPF (%).
- Casado o no.
- Número de hijos.
- **Atributo static**: importe de la hora extra.

Los objetos Empleado se podrán crear con un constructor por defecto o con un constructor con un solo parámetro correspondiente al DNI.

Además de los métodos getter/setter y el método toString, la clase tendrá estos métodos:

- Método para el cálculo del complemento correspondiente a las horas extra realizadas.
- Método para calcular el sueldo bruto (sueldo base + complemento por horas extras)
- Método para calcular las retenciones (IRPF). El porcentaje de IRPF se aplica sobre el sueldo bruto, teniendo en cuenta que el porcentaje que hay que aplicar es el tipo menos 2 puntos si el empleado está casado y menos 1 punto adicional por cada hijo que tenga.

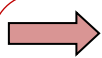
Una vez creada la clase, escribe un programa que lea empleados y los guarde en un **array estático**.

El tamaño del array de Empleados será de 20 elementos. Este array se creará de ese tamaño. Esto quiere decir que en la empresa puede haber 20 empleados como máximo pero puede haber también menos.

Lo primero que hará el programa será introducir los datos de los empleados.

Debemos tener en cuenta que **no podemos introducir empleados repetidos**.

Si un empleado ya existe en el array se vuelve a pedir. Consideramos que dos empleados son iguales si tienen el mismo dni. Para comprobar si dos empleados son iguales, la clase Empleado debe contener el **método equals** donde debes indicar que dos empleados son iguales si tienen el mismo dni. Este método será el que invoques cuando quieras comprobar si un empleado ya existe en el array.



Recuerda que desde NetBeans puedes generar el código del método *equals* pulsando: *botón derecho -> insertar código -> equals() and hashCode()*.

Los métodos equals y hashCode están relacionados y por eso NetBeans genera los dos al mismo tiempo. Veremos más adelante en el curso para qué sirve el método hashCode, de momento no lo vamos a utilizar así que dejaremos en la clase el método hashCode que genera NetBeans sin preocuparnos de para qué sirve.

A continuación mostrar por pantalla todos los empleados introducidos ordenados alfabéticamente por nombre.

A continuación se pedirá que se introduzca el importe correspondiente al pago por hora extra asignándose al atributo correspondiente.

A continuación se mostrará:

- El empleado que más cobra y el que menos. Para realizar esto la clase principal contendrá un método que devuelva el empleado que más cobra y otro método que devuelva el empleado que menos cobra. Estos métodos no muestran nada por pantalla sino que devuelve un empleado (objeto), el que más cobra en un caso y el que menos cobra en el otro.

- El empleado que cobra más por horas extra y el que menos. Igual que en el punto anterior, la clase principal contendrá un método que devuelva el empleado que más cobra por horas extras y otro método que devuelva el empleado que cobra menos por horas extras.
 - Todos los empleados ordenados por salario de menor a mayor. Para realizar esto la clase principal contendrá un método que ordene los empleados por sueldo y otro que muestre el array después de la ordenación.
3. Escribe una clase Alumno con los atributos: nia (número de identificación del alumnos de tipo int), Curso (String), Nombre (String), NIF (de tipo NIF. Puedes tomar como base la clase NIF del Ejercicio 8 Ejercicios básicos POO), fecha de nacimiento (de tipo Fecha. Puedes tomar como base la clase Fecha del Ejercicio 7 Ejercicios básicos POO), dirección (String), código postal (int), ciudad (String), nota media (double), número de faltas graves (int), límite de faltas graves: atributo static que contendrá el máximo de faltas graves para que un alumno sea expulsado. Valor por defecto = 5.

Además de los constructores, métodos getter/setter, método toString, método equals, etc. la clase contendrá un **método leer()** para introducir por teclado los datos de los alumnos.

Realiza un programa para utilizar la clase. El programa creará alumnos y los guardará en un **ArrayList** de objetos Alumno.

Mediante un **Menú** se realizarán las diferentes operaciones a realizar con los alumnos.

Opciones del menú:

1. Nuevo alumno
2. Mostrar alumnos
3. Buscar alumnos
4. Bajas de alumnos
5. Modificar Alumnos
6. Calcular la media de todos los alumnos
7. Modificar el límite de faltas graves
8. Mostrar alumnos con faltas graves
9. Mostrar los alumnos expulsados
0. FIN

Introduzca opción:

El programa principal contendrá los métodos necesarios para llevar a cabo las operaciones indicadas en el menú:

- Añadir un nuevo alumno: se introducen los datos de un alumno por teclado y se añade al ArrayList. En el ArrayList no puede haber dos alumnos iguales. Consideramos que dos alumnos son iguales si tienen el mismo NIF.
- Mostrar todos los alumnos. Ordenados alfabéticamente por nombre.
- Buscar alumnos: Se introduce un NIF y se muestra el alumno correspondiente o un mensaje si no existe.
- Eliminar alumnos: Se introduce un NIF y se elimina el alumno correspondiente.
- Modificar alumnos: Se introduce un NIF y si el alumno existe se muestran sus datos actuales, se leen los nuevos y se modifica. El campo NIF no se puede modificar.
- Calcular la media de todos los alumnos: la media total de todos los alumnos de la clase.
- Modificar el límite de faltas graves: modificar el valor del atributo static.
- Mostrar alumnos con faltas graves. Ordenados por número de faltas graves de menor a mayor.
- Mostrar todos los alumnos expulsados: muestra los alumnos cuyo número de faltas graves sea mayor o igual que el límite de faltas graves. Ordenados alfabéticamente.

4. Crea una clase Biblioteca que modele el funcionamiento básico de una pequeña biblioteca.

Los atributos de la clase serán:

- un **array** de libros (array de objetos de tipo Libro. Utiliza como base la clase Libro del Ejercicio 5 Ejercicios básicos POO)
- un entero *indice* que servirá para indicar la posición del siguiente elemento libre dentro del array.

A la clase Libro se le debe añadir un campo *referencia* de tipo String, un método *estaDisponible()* que devuelva true o false si hay ejemplares disponibles del libro para prestar y el método *toString()* que muestre los datos del libro.

Consideramos que en la biblioteca caben 100 libros como máximo.

La clase Biblioteca contendrá al menos los siguientes métodos:

- Un constructor por defecto que crea el array de libros de 100 elementos.
- Método para añadir un nuevo libro en la biblioteca. El método recibe un Libro (objeto) para añadir a la biblioteca. El nuevo libro se añade a continuación del ultimo libro añadido al array, siempre que haya espacio. El método devuelve true o false para indicar si el libro se ha añadido o no. Si no hay espacio el método devuelve false. Devuelve true si se ha podido añadir.
- Método que recibe la referencia de un libro y si existe lo elimina de la biblioteca. Eliminar un libro del array significa desplazar un lugar a la izquierda el resto de libros que se encuentran a continuación de éste. El método devuelve true si se ha eliminado o false en caso contrario.
- Método que recibe la posición que ocupa un libro dentro del array y lo elimina del array de la misma forma que en el punto anterior. Se debe comprobar que la posición recibida sea válida. El método devuelve true si se ha eliminado o false en caso contrario.
- Método para saber cuantos libros hay disponibles para prestar en la biblioteca.
- Método *contiene* para saber si un libro se encuentra en la biblioteca. El método recibe la referencia del libro a buscar. Devuelve true si el libro se encuentra en la biblioteca y false si no se encuentra.
- Método *get* que devuelva el libro cuya referencia recibe como parámetro. Si el libro no está devuelve null.
- Método *toString* para mostrar todos los libros de la biblioteca.

Escribe un programa que mediante un menú realice las siguientes operaciones:

- Añadir un libro Nuevo. Tendremos en cuenta que no se pueden añadir libros con la misma referencia.
- Eliminar libros por referencia.
- Eliminar libros por posición.
- Realizar un préstamo
- Realizar una devolución.
- Listado alfabético por título de los libros disponibles.
- Listado alfabético por título de todos los libros.

5. Escribe una clase `miArrayList` que **simule el funcionamiento de la clase `ArrayList`** de java para objetos de tipo `Cuenta` (toma como base la clase `Cuenta` del ejercicio Ejercicio 1 Ejercicios básicos POO).

Los atributos privados de la clase `miArrayList` serán un array de tipo `Cuenta` y un entero *índice* que servirá para indicar la posición del siguiente elemento libre dentro del array.

La clase contendrá los siguientes constructores:

- Constructor por defecto que crea el array con 20 elementos.
- Constructor que recibe un entero `n` y crea el array de `n` elementos.
- Constructor copia que crea el array del tamaño del objeto `miArrayList` que recibe + 20 y asigna a los valores del objeto creado los del objeto recibido. Las posiciones restantes (las 20 finales) contendrán `null`.

Además contendrá los métodos:

- **`add(Cuenta a)`** que añade la cuenta al final del array. Si no hay espacio se debe ampliar el tamaño del array. Ampliar el tamaño significa crear un array nuevo de tamaño 20 elementos superior al actual, copiar los valores del array actual en el nuevo array y finalmente asignar la referencia del nuevo array creado al array de la clase.
- **`add(int posición, Cuenta a)`** que inserta en la posición indicada la `Cuenta a`. Debe ser una posición válida. Si no lo es se añade al final del array. El resto de elementos desde la posición hasta el final se desplazan un lugar a la derecha. Si no hay más espacio para insertar se procede como en el punto anterior.
- **`boolean contains(Cuenta a)`** si la cuenta se encuentra dentro del array devolverá `true`. Devolverá `false` en caso contrario.
- **`remove(Cuenta a)`** si la cuenta se encuentra en el array la elimina. Eliminar significa desplazar todos los elementos a partir del que vamos a eliminar un lugar a la izquierda.
- **`set(int pos, Cuenta a)`** cambia el valor del elemento que se encuentra en la posición *pos* por el nuevo valor *a*. *pos* debe ser una posición válida. Si no lo es no se hace nada. Devuelve `true` si se ha realizado la modificación o `false` en caso contrario.
- **`get(int posición)`** devuelve la `Cuenta` que está en la posición indicada. Si la posición es superior a la que ocupa el último elemento del array se devuelve el último elemento.
- **`size()`** devuelve el número de elementos del array.
- **`int indexOf(Cuenta a)`** si la cuenta se encuentra dentro del array devolverá su posición. Devolverá -1 en caso contrario.
- **`clear()`** elimina todos los elementos del array.
- **`sort()`** que ordena todas las cuentas por saldo de mayor a menor.

Escribe un programa para probar la clase.

Utiliza un menú para pedir qué opción se quiere realizar (añadir, eliminar, modificar, mostrar cuentas, etc). Al añadir una nueva cuenta se debe tener en cuenta que no puede haber cuentas con el mismo número de cuenta. En la opción de modificar no se podrá modificar el número de cuenta. Además se podrá calcular y mostrar la cuenta con el mayor saldo, la cuenta con el menor saldo y el saldo medio de todas las cuentas.

6. Una **Pila** es una estructura de datos en la que las inserciones y las eliminaciones de elementos se producen siempre por el mismo lugar. A las estructuras de tipo *pila* se les llama estructuras LIFO (Last IN First Out, el último que entra es el primero que sale).

Un ejemplo típico de la vida real es una pila de platos colocados uno encima de otro. Un plato nuevo se coloca sobre el último que se ha colocado en la pila y si necesitamos un plato tomaremos el que se encuentra arriba del todo que es el último que se ha introducido.

Escribe una clase Pila que simule el funcionamiento de una estructura de datos de este tipo.

Los atributos privados de la clase Pila serán un **ArrayList** de tipo Libro (Utiliza la clase Libro del ejercicio 4 de esta hoja).

La clase contendrá un constructor por defecto que crea el ArrayList.

Además contendrá los métodos:

- **void push(Libro a)** que añade el libro a la pila.
- **Libro pop()** extrae y devuelve un libro de la pila. Si la pila está vacía devuelve null.
- **Libro cima()** devuelve el libro que se encuentra en la cima de la pila sin extraerlo de la misma. Si la pila está vacía devuelve null.
- **int size()** devuelve el número de elementos que contiene la pila.
- **void clear()** elimina todos los elementos de la pila.
- **toString()** muestra todos los elementos que contiene la pila en el orden en que se extraerían de la misma (desde la cima hasta el fondo).

Escribe un programa para probar la clase.

Utiliza un menú para pedir qué opción se quiere realizar (push, pop, mostrar, etc).