SESIÓN 8: Arrays de objetos. Búsqueda y Ordenación

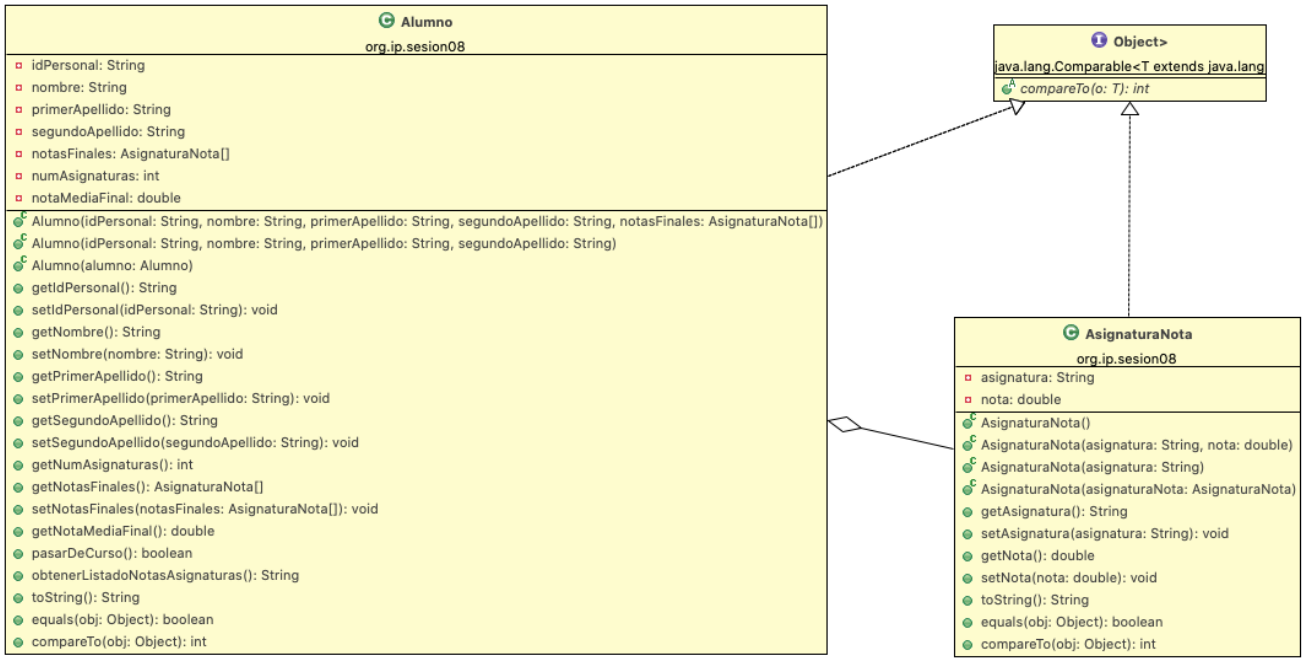
Objetivos

- Saber declarar, crear, inicializar y manejar arrays de objetos.
- Saber realizar operaciones comunes con arrays de objetos.
- Saber ordenar un array utilizando los métodos de *selección* y *burbuja*.
- Saber ordenar un array de objetos utilizando métodos de ordenación genéricos.
- Introducir el uso de los tests unitarios con JUnit con Java en Eclipse. (JUnit 5). **Tutorial JUnit**.

Nota importante: Siga el esquema de nombrado de paquetes que se indicó en la sesión 01 es decir: **org.ip.sesion08**. En ese paquete se crearán todos los programas que se proponen en la sesión dándoles un nombre al programa y que se indica en cada ejercicio entre **paréntesis y en negrita**.
Al final de la sesión, el alumno deberá cargar el trabajo realizado a su repositorio personal indicando la clave correspondiente a la sesión.

Ejercicios propuestos

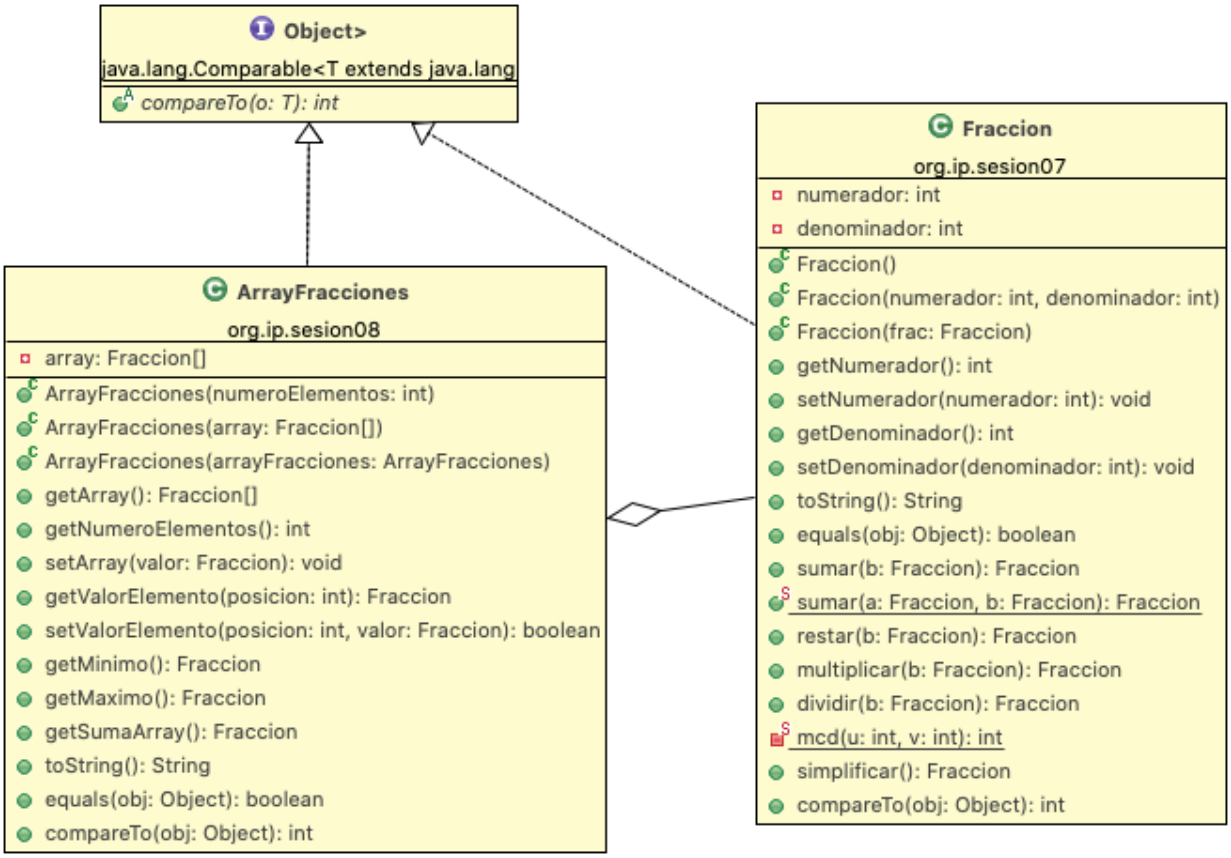
1. Implementa la clase `Alumno` **(Alumno)** y la clase `AsignaturaNota` **(AsignaturaNota)** tal y como se muestra en el siguiente diagrama de clases UML. Como podemos comprobar, dentro de la clase `Alumno` tenemos un array de objetos (dato miembro) de la clase `AsignaturaNota` (composición). Todo ello para trabajar también con arrays de objetos de tipo `Alumno`.



Conforme se vaya implementado dicha clase se deberá ir pasando el test (**ArrayAlumnosTestJUnit5**) que permita comprobar su correcto funcionamiento. Dicho test se proporcionará con el material de esta sesión y se deberá almacenar en el paquete **test.org.ip.sesion08**. Recordemos que hay que crear un nuevo paquete en la carpeta de fuentes **test**, éste será **org.ip.sesion08**. A continuación, debemos copiar el archivo de test **ArrayAlumnosTestJUnit5.java** y a partir de este momento hay que implementar todas las operaciones en el archivo **Alumno.java** y en otro archivo **AsignaturaNota.java** que deberán estar ubicados en la carpeta de fuentes **src** en el paquete **org.ip.sesion08**.

Trabajo autónomo

2. Implementa la clase, **ArrayFracciones**, cuyo diagrama de clases se proporciona a continuación, y destacando que se hará uso de (`import`) la clase **`Fraccion`** implementada en la **sesión 07** (`org.ip.sesion07`) y que habrá que importar en nuestra clase ArrayFracciones.

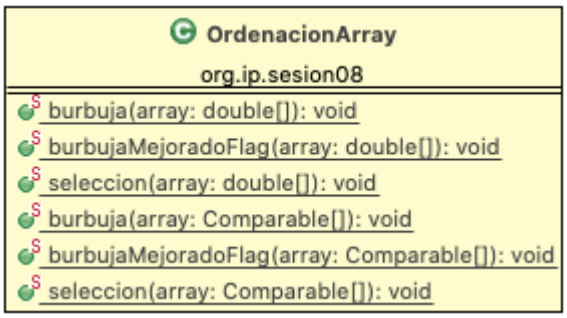


1. Implementar 3 constructores diferentes según los parámetros que se le pasen. Destacar que para el constructor `public ArrayFracciones(int numeroElementos)` se creará un array con tantas fracciones (constructor por defecto de la clase `Fraccion`) como indique el valor de `numeroElementos`.
2. El método `public Fraccion [] getArray()` devolverá el array de fracciones.
3. El método `public int getNumeroElementos()` devuelve el número de elementos que hay en el array de fracciones, es decir, el número de fracciones.
4. El método `public void setArray(Fraccion valor)` le asignará a **todos** los elementos del array el mismo valor de fracción, y éste será el que se le pasará como parámetro a través de la variable *valor*, que es de tipo `Fraccion`.
5. El método `public Fraccion getValorElemento(int posicion)` devolverá el valor de la `Fraccion` que tiene el array en la posición *posicion*, sabiendo que $0 \leq posicion \leq array.length - 1$.
6. El método `public boolean setValorElemento(int posicion, Fraccion valor)` devolverá `true` si ha podido realizar correctamente la asignación de la Fraccion *valor* en la posición *poiscion* del array, `false` en caso contrario. Recordad el rango, $0 \leq posicion \leq array.length - 1$.
7. El método `public Fraccion getMinimo()` devolverá la `Fraccion` de menor valor almacenada en el array de Fracciones.

8. El método `public Fraccion` **`getMaximo`**`()` devolverá la `Fraccion` de mayor valor almacenada en el array de Fracciones.
9. El método `public Fraccion` **`getSumaArray`**`()` devolverá la `Fraccion` suma de todas las Fracciones almacenada en el array. Cabe destacar también que el resultado de la suma se deberá de `simplificar()` y que si tanto el numerador como el denominador son negativos, la `Fraccion` suma resultante será positiva (numerador y denominador serán ambos positivos).
10. La salida del método `public String` **`toString`**`()` debe verificar la salida establecida en el *test*.
11. El método `public boolean` **`equals`**`(Object obj)` devuelve `true` si los dos arrays de Fracciones son exactamente iguales en contenido (elemento a elemento), `false` en caso contrario.
12. Se utilizará el valor de `Fraccion` correspondiente a la suma de todas las Fracciones que hay almacenadas en el array (`public Fraccion` **`getSumaArray`**`()`) como criterio para la comparación de array de Fracciones (`public int` **`compareTo`**`(Object o)`). Es decir, un array es mayor que otro si la `Fraccion` suma es mayor.

Conforme se vaya implementado dicha clase se deberá ir pasando el test (**ArrayFraccionesTestJUnit5**) que permita comprobar su correcto funcionamiento. Dicho test se proporcionará con el material de esta sesión y se deberá almacenar en el paquete **test.org.ip.sesion08**. Recordemos que hay que crear un nuevo paquete (sino está creado ya) en la carpeta de fuentes **test**, éste será **org.ip.sesion08**. A continuación, debemos copiar el archivo de test **ArrayFraccionesTestJUnit5.java** y a partir de este momento hay que implementar todas las operaciones en el archivo **ArrayFracciones.java** que deberá estar ubicado en la carpeta de fuentes **src** en el paquete **org.ip.sesion08**.

2. Implementa una clase Java `OrdenacionArray` **(OrdenacionArray)** que contenga los métodos que permitan hacer una ordenación de un array por cualquiera de los métodos que se describen en el diagrama de clases UML que se muestra a continuación.



Para el adecuado funcionamiento de este test hay que tener muy bien implementada la clase **`Fraccion`** de la **sesión 07** (`org.ip.sesion07`), sobre todo la función `compareTo`.

Conforme se vaya implementado dicha clase se deberá ir pasando el test (**OrdenacionArrayTestJUnit5**) que permita comprobar su correcto funcionamiento. Dicho test se proporcionará con el material de esta sesión y se deberá almacenar en el paquete **test.org.ip.sesion08**. Recordemos que hay que crear, si no lo ha creado ya, un nuevo paquete en la carpeta de fuentes **test**, éste será **org.ip.sesion08**. A continuación, debemos copiar el archivo de test **OrdenacionArrayTestJUnit5.java** y a partir de este momento hay que implementar todas las operaciones de ordenación en el archivo **OrdenacionArray.java** que deberá estar ubicado en la carpeta de fuentes **src** en el paquete **org.ip.sesion08**.