Interfaces y genericidad

1. (package conjunto)

a. Crear el un interfaz genérico Coleccion<T>, con los siguientes métodos. Se trata de que pienses cuáles serán las definiciones de los métodos. Recuerda que un interfaz únicamente contiene definiciones de métodos (sin su implementación) :

- agregar: Añade a la colección un elemento x
- eliminar: Elimina de la colección un elemento x
- estaVacia: Devuelve si la colección contiene elementos o no.
- talla: Devuelve el número de elementos de la colección.
- contiene: Devuelve si la colección contiene o no a determinado elemento x

b. Implementar la clase Cojunto <T> que implemente (implements) el interfaz Coleccion <T>. Para ello, la clase conjunto tendrá un atributo privado, de tipo ArrayList, llamado elementos. Se utilizará el ArrayList para almacenar los elementos del conjunto. Un conjunto no admite elementos duplicados, cosa que se tendrá en cuenta al implementar el método agregar. El intento de agregar un elemento que ha existe en el conjunto no tendrá ningún efecto. Además de los métodos que impone el interfaz, la clase tendrá que tener también un constructor y un método toString.

c. Implementar la clase TestConjunto que cree un conjunto de String y añada varios elementos y elimine alguno de ellos. Usar el método toString para comprobar los resultados.

2. (package genéricos) Crear una clase Genericos que incorpore los métodos genéricos que se indican a continuación. Se usará el método main para probar los métodos desarrollados.

   a. *T minimo (T o1, T o2),* que devuelva el mínimo de dos objetos.

   b. *T maximo (T[ ] v),* que devuelva el máximo de un array de objetos.

   c. *int numVeces(T[ ] v, T x)* que devuelva el número de apariciones de x en el array v de objetos.

   d. *int numVecesOrdenado(T[ ] v, T x)* que devuelva el número de apariciones de x en un array v **que está ordenado ascendentemente**.

   e. *int mayores(T v[],T x)* que devuelva el número de elementos de v que son mayores que x..

   f. *int mayoresOrdenado(T v[], T x)* que devuelva el número de elementos de v que son mayores que x, suponiendo que v está ordenado ascendentemente.

   g. *boolean estaEn(T v[],T x)* que devuelva true si x está en el array v.

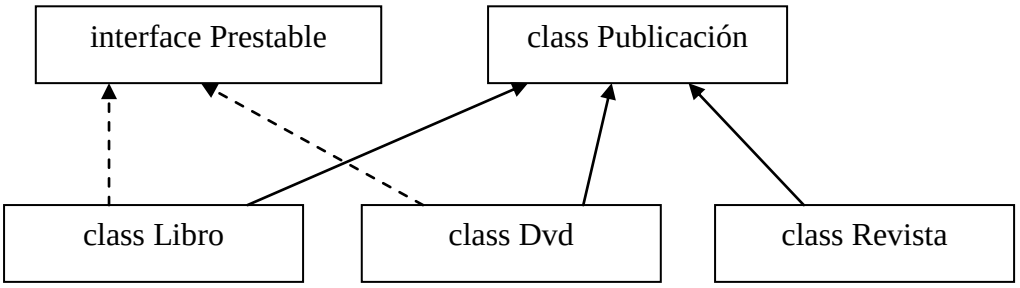
   h. *boolean estaEnOrdenado(T v[],T x)* que devuelva true si x está en v, suponiendo que v está ordenado ascendentemente.

   i. *int posiciónDe(T v[],T x),* que devuelva la posición que ocupa x dentro del array v, o -1 si x no está en v.

   j. *int posiciónDeOrdenado(T v[],T x),* que devuelva la posición que ocupa x dentro del array v **ordenado ascendentemente**, o -1 si x no está en v.

Interfaces y genericidad

3. Implementa la siguiente jerarquía de clases:



- Las características comunes a todas las publicaciones son su código, título y año de publicación. Estas tres características las recibirán todos los constructores que se definan.
- Además, las revístas tienen un número (nº de revista) y los Dvd tienen una duración. Estos atributos los recibirá el costructor correspondiente.
- Los Libros y los Dvd tienen un atributo *prestado* que indica si está prestado o no. Cuando se crea un objeto de uno de estos tipos, inicialmente estará como no prestado y el atributo no se recibe en el constructor.
- El interfaz Prestable contendrá dos métodos:
    - void prestar();
    - void devolver();
    - boolean getPrestado();
- Las clases Libro y Dvd deben implementar el interface Prestable. La implementación de los métodos *prestar* y *devolver* debera´modificar convenientemente el atributo *prestado.* El método getPrestado devolverá el valor del atributo.

Crear un programa de test que:

- Cree un ArrayList de publicaciones que contenga únicamente libros y dvd's.
- Hacer un recorrido del ArrayList que muestre los títulos.
- Hacer un recorrido del ArrayList que cuente el número de elementos que se encuentren prestados.

4. (Ordenación) Escribe un programa que cree un ArrayList de Publicaciones y añádele publicaciones de distinto tipo. A continuación, haciendo uso del interface Comparator:
    a. Implementa un método ***mostrarPorCodigo*** que reciba el ArrayList creado y lo muestre **ordenado por código.**
    b. *Implementa un método* ***mostrarPorTitulo*** que reciba el ArrayList creado y lo muestre **ordenado por título.**
    c. *Implementa un método* ***mostrarPorAnyo*** que reciba el ArrayList creado y lo muestre **ordenado por año.**
    d. *Implementa un método* ***mostrarPorAnyoDecreciente*** que reciba el ArrayList creado y lo muestre **ordenado de mayor a menor año.** A igual año, de menor a mayor título. A igual título, de menor a mayor código.
5. (Filtrado) Escribe un programa que cree un ArrayList de Publicaciones y añádele publicaciones de distinto tipo. A continuación, implementa métodos que reciban el ArrayList y lo muestren filtrado según los criterios que se indican en cada uno de los apartados que se indican a continuación. Para filtrar se usará el Inteface Predicate que ya se encuentra en Java. El interface Predicate<T>. Este interface contiene un método boolean test(T o).
    a. Mostrar las publicaciones cuyo código de publicación es par.
    b. Mostrar las publicaciones cuyo título tiene una longitud mayor que 5
    c. Mostrar las publicaciones del año actual
    d. Mostrar las publicaciones cuyo año está entre 2000 y 2005 y cuyo título contiene la palabra “casa”.
    e. Mostrar las publicaciones que son libros.