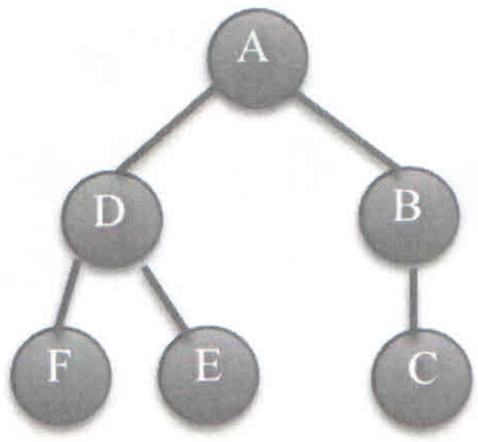
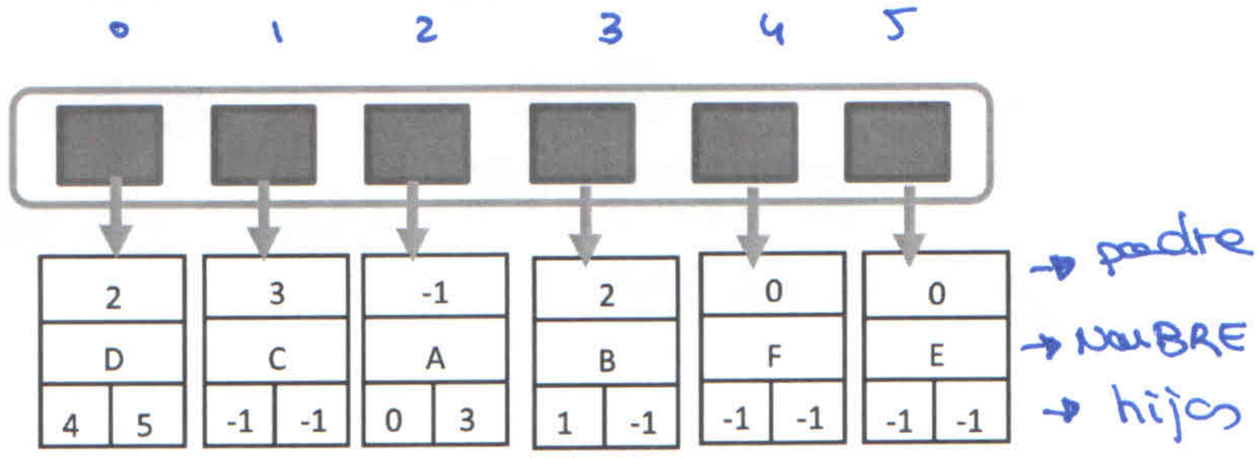
Ejercicio 1 [3.5 puntos]

Un árbol binario se puede almacenar en un array (implementado mediante un `ArrayList`) donde cada posición del array contiene una referencia a un objetivo `BTPos`. En la figura siguiente se muestra un ejemplo de árbol binario y una posible representación mediante un array.





Se pide:

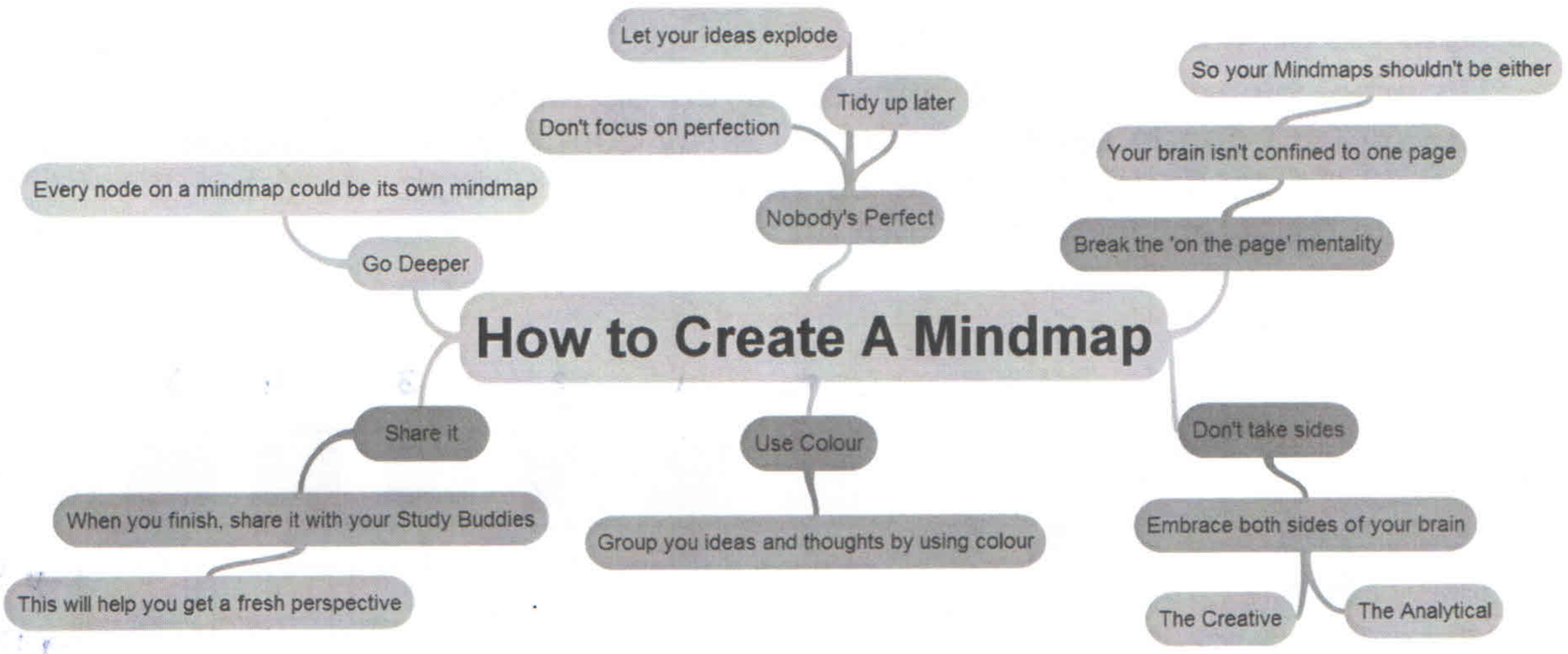
a) **[0.5 puntos]** Definir los tipos de datos (clases en Java) `ArrayListBinaryTree` y `BTPos` que permitan la creación de árboles. Se requiere que las nuevas clases hereden de donde corresponda para que se incorporen a la jerarquía de clases desarrollada durante el curso.

b) **[2.0 puntos]** Implementar el método `removeSubtree` que, dado un `Position`, elimine el subárbol enraizado en dicho `Position`. También se requiere que el método respete la jerarquía de clases desarrollada durante el curso. Se recomienda el uso de estructuras de datos auxiliares para evitar complejidades lineales al actualizar el árbol.

c) **[1.0 punto]** Implementar el método `insertLeft` en la clase que se considere más adecuada para añadir un nuevo nodo al árbol. También se requiere que el método respete la jerarquía de clases desarrollada durante el curso.

Ejercicio 2 [3.5 puntos].

Se desea crear un software para generar y almacenar esquemas de tipo Mapa Mental (`MindMap`) como el de la figura adjunta. Los mapas mentales son unos diagramas muy eficaces para extraer y memorizar información. Un mapa mental consiste en una serie de etiquetas ligadas y dispuestas radialmente alrededor de una etiqueta clave central. Desde la etiqueta central se puede acceder a cualquier otra etiqueta del diagrama por un camino único que puede atravesar otras etiquetas.



Para ello, se desea disponer de una clase `MindMap` que permita crear, editar y almacenar en memoria estos diagramas. En particular, al menos se deberá poder realizar operaciones de: **inserción** de nueva etiqueta (`insertLabel`), **borrado** de etiqueta (`removeLabel`), **consulta** de la etiqueta central (`centralLabel`) y **recorrido** de las etiquetas (`labels`) a las que se pueda acceder directamente desde una etiqueta que se le pase por parámetro.

Cada **etiqueta**, además del **texto** correspondiente, deberá permitir la edición y consulta de información sobre el **tamaño de fuente** (mediante un valor entero), la **posición** espacial de la etiqueta (mediante unas coordenadas enteras x e y) y el **color** de la misma (definido mediante 3 enteros correspondientes a sus componentes roja, verde y azul).

Para crear la clase `MindMap` se podrá agregar en su interior cualquiera de las estructuras de datos estudiadas durante el curso, valorándose especialmente lo adecuado de la elección que se realice.

Como resultado, el alumno debe:

a) **[1.0 punto]** Presentar un Diagrama Estático de Clases (preferiblemente en UML) con todas las clases que considere necesarias, incluyendo los métodos y propiedades (o atributos) de las mismas. Además, explicará brevemente cada una de las clases con un párrafo de menos de 40 palabras

b) **[2.5 puntos]** Implementar en Java el código de la clase `MindMap` y de las clases que ésta necesite. En particular, la clase `MindMap` deberá incluir los métodos ya descritos: `insertLabel`, `removeLabel`, `centralLabel` y `labels`. No es necesario proporcionar el código de aquellas estructuras de datos desarrolladas durante el curso o de aquellas que pertenezcan al API de Java.

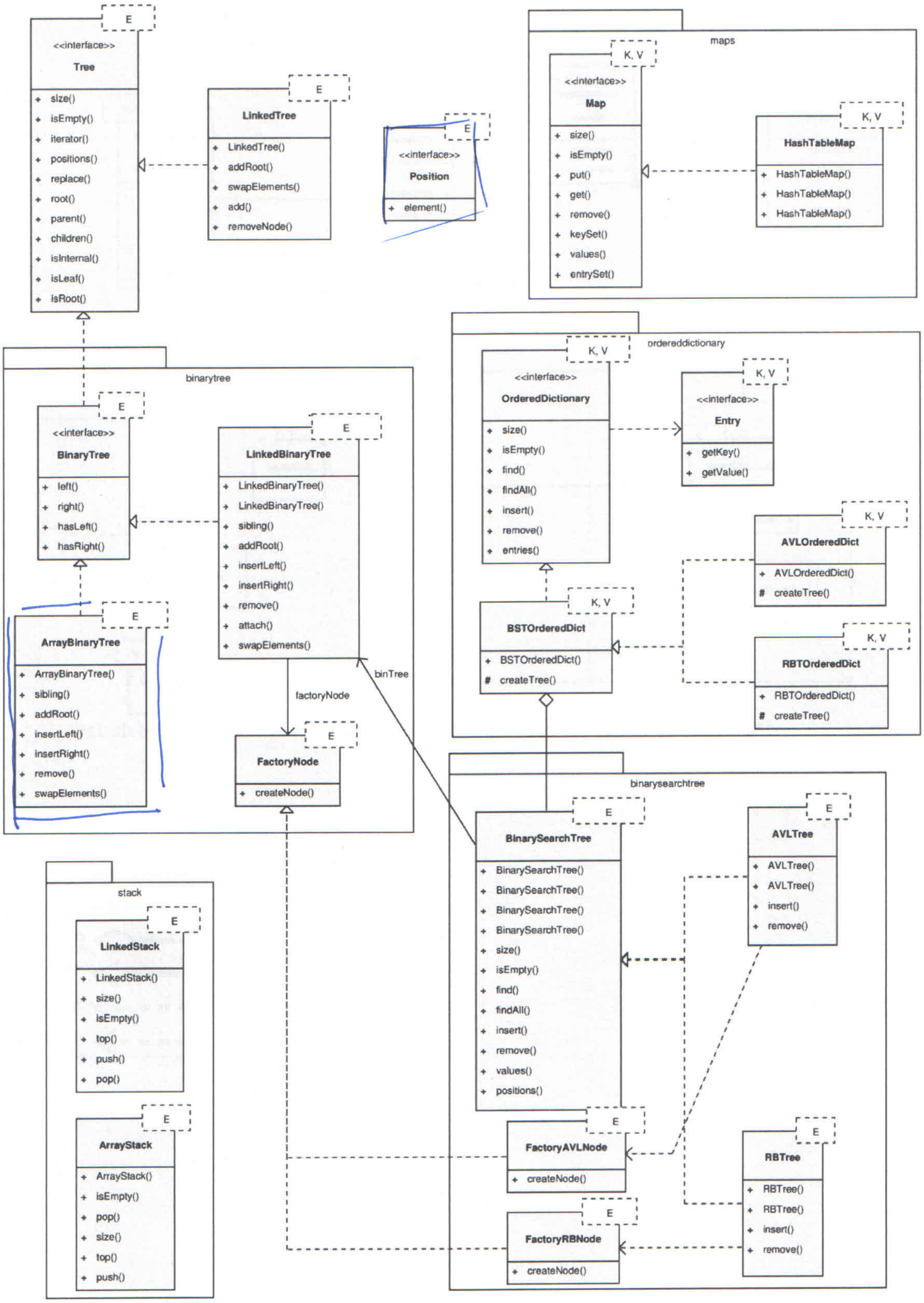


Diagrama de Clases correspondiente a las estructuras presentadas en clase

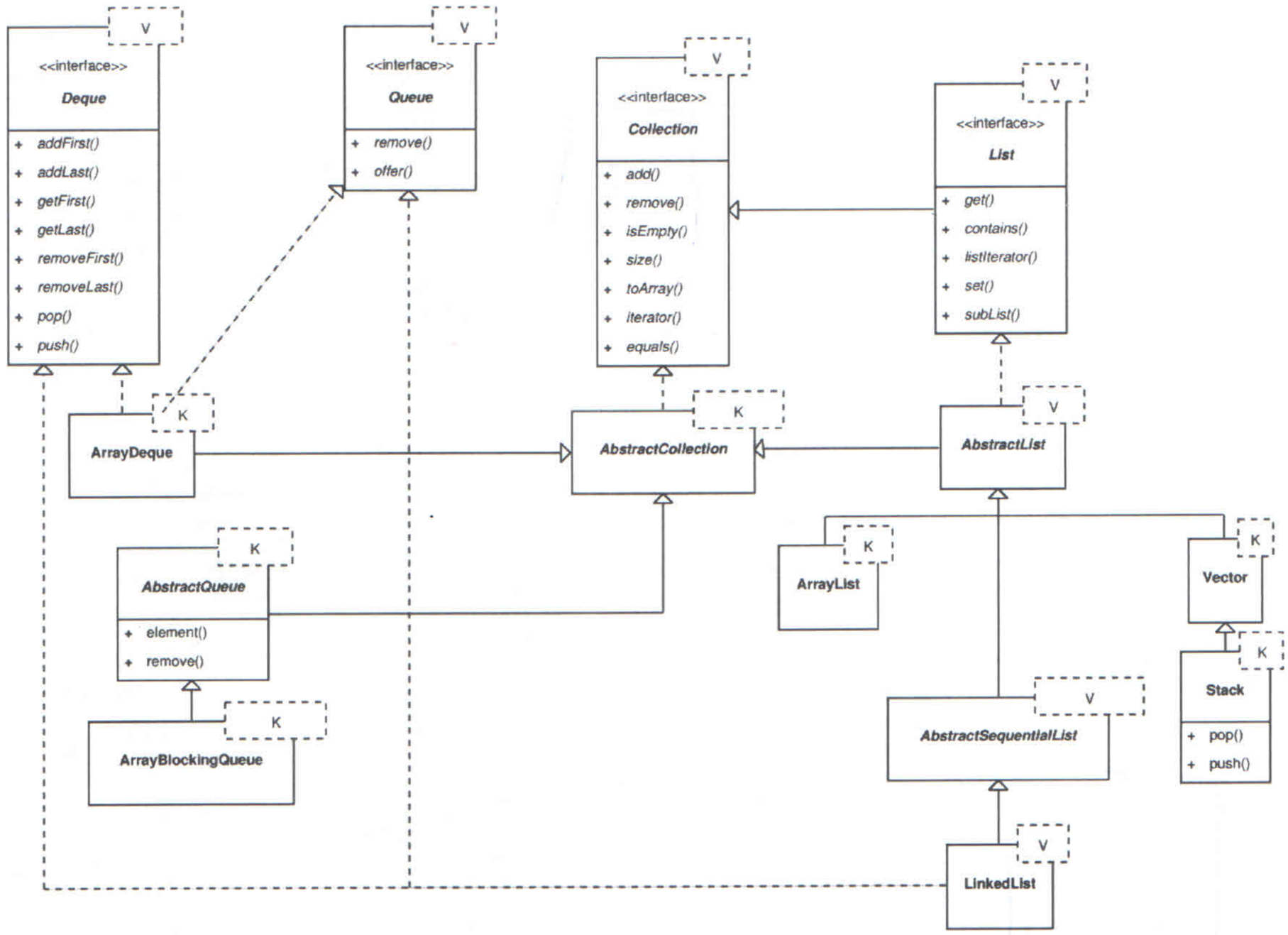


Diagrama de clases correspondiente a las estructuras de datos lineales de Java