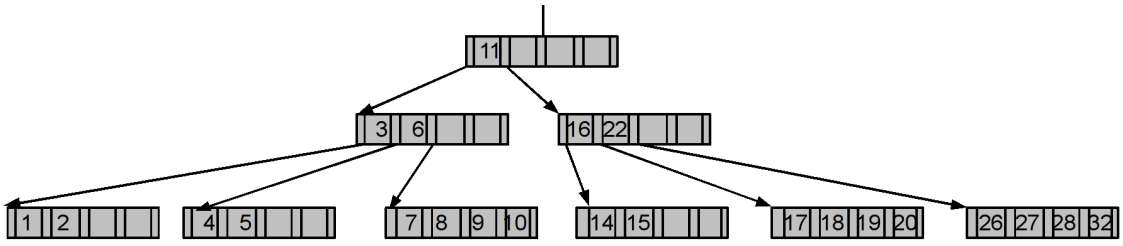
Ejercicio 1 [3.0 puntos].

a) **[1.0 punto]** Representar el árbol rojo-negro resultante de la siguinete secuencia de inserciones [60, 3, 12, 23, 51, 80, 14, 59, 15, 9, 13, 1].

b) **[1.0 punto]** Explique brevemente qué es el clustering primario, el clustering secundario y qué mecanismos existen en mapas para evitarlo.

c) **[1.0 punto]** Dado el árbol B de orden 5 mostrado en la figura, representar gráficamente dicho árbol después de la inserción de la clave 33 y, posteriormente, la eliminación de la clave 2.



Ejercicio 2 [3.5 puntos]

a) **[0.5 punto]** Definir el tipo de dato (atributos de la clase) que considere más adecuado para implementar un mapa donde las colisiones se resuelven por encadenamiento separado.

b) **[1.5 puntos]** Implementar la funcionalidad necesaria para construir un iterador para dicha clase que recorra todos los elementos almacenados en el mapa.

c) **[1.5 puntos]** Implementar el método `remove`, que reciba un clave y elimine aquél elemento identificado por dicha clave. Este método deberá además considerar que si el mapa tiene un nivel de ocupación por debajo del nivel que se considere típico, se deberá reducir su tamaño. Se asume que el resto de métodos dados en la interfaz `Map` están implementados

Ejercicio 3 [3.5 puntos]

La codificación de Huffman es un mecanismo que permite comprimir información mediante la reducción del número de bits empleado en los caracteres más usados. Para cada mensaje, se utiliza una codificación distinta que se basa en la frecuencia. Por ejemplo, dado el mensaje SUSIE SAYS IT IS EASY, tendríamos la siguiente tabla de frecuencias

Carácter	Frecuencia
A	2
E	2
I	3
S	6
T	1
U	1
Y	2
sp - Espacio	4
lf - Retorno de carro	1

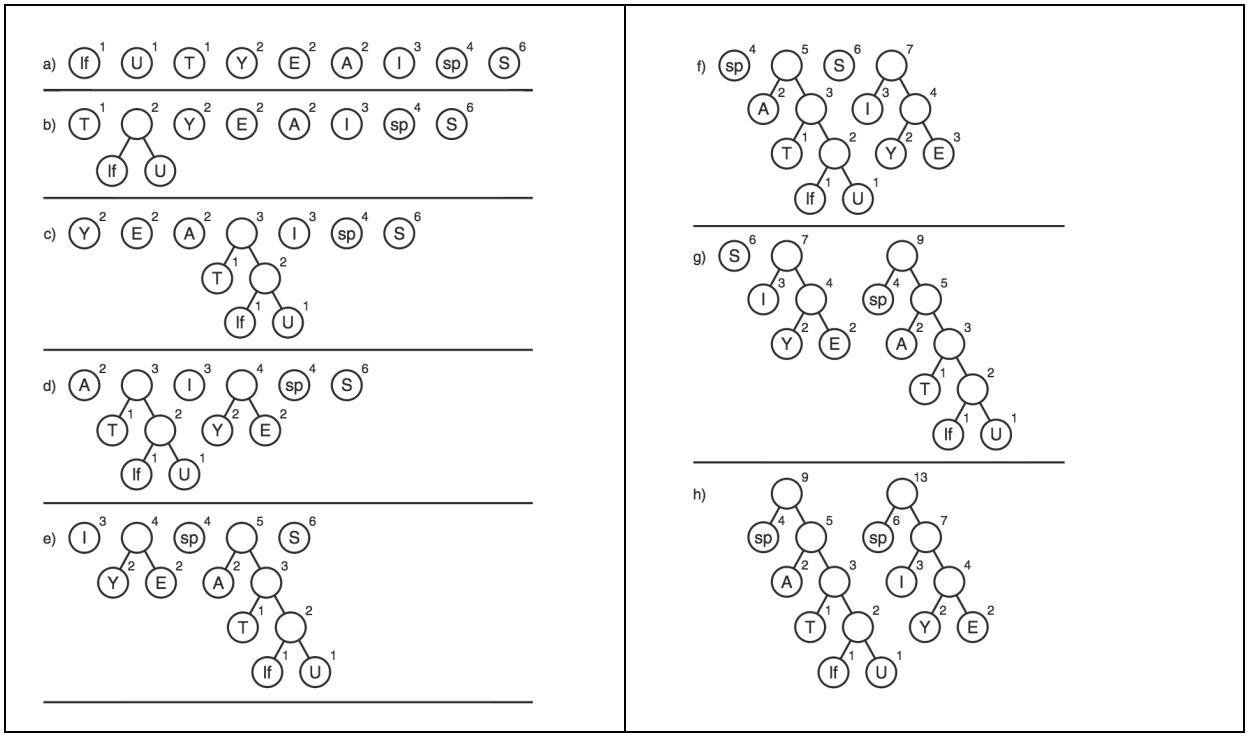
El método Huffman crea los códigos en tres pasos. El primer paso precisa de evaluar la frecuencia de ocurrencia de cada carácter. En el segundo paso, se construye un árbol iterativamente. Inicialmente, se almacena en una lista ordenada tantos árboles como caracteres distintos, de tal forma que cada árbol tiene un único nodo con la frecuencia de ocurrencia de cada carácter y el propio carácter. En el siguiente paso, se unen los dos árboles con menor frecuencia. Como resultado de cada unión se crea un nuevo árbol cuya raíz es un nodo con frecuencia igual a la suma de los nodos unidos y sin información de carácter. Este proceso continua hasta que sólo queda un árbol. Finalmente, en la tercera etapa se recorre de manera inversa el árbol utilizando un criterio de asignación de códigos (por ejemplo cero a la izquierda y uno a la derecha). En la figura siguiente se muestra un ejemplo de cómo se construiría.

Se pide implementar en Java la clase `Huffman` en que tenga la siguientes características.

a) **[1.0 punto]** Un constructor que recibe una cadena (tipo `String`), que determina la frecuencia de aparición de cada carácter y lo almacena esta información en la estructura de datos que considere más adecuada. Finalmente, llama al método `análisis` (descrito en el siguiente apartado).

b) **[1.5 punto]** Implementar un método privado `análisis` que construya el árbol mediante el proceso iterativo descrito anteriormente.

c) **[1.0 punto]** Implementar un método público `codificar` que reciba un carácter y devuelva un `String` con el código binario de Huffman que lo identifica.



NOTA: Para crear la clase `Huffman` se podrá agregar en su interior cualquiera de las estructuras de datos estudiadas durante el curso. No es necesario proporcionar el código de aquellas estructuras de datos desarrolladas durante el curso. En el caso de la lista ordenada, se podrán utilizar todos los métodos que se consideren razonables (por ejemplo, insertar ordenado, crear lista, borrar lista, etc.).

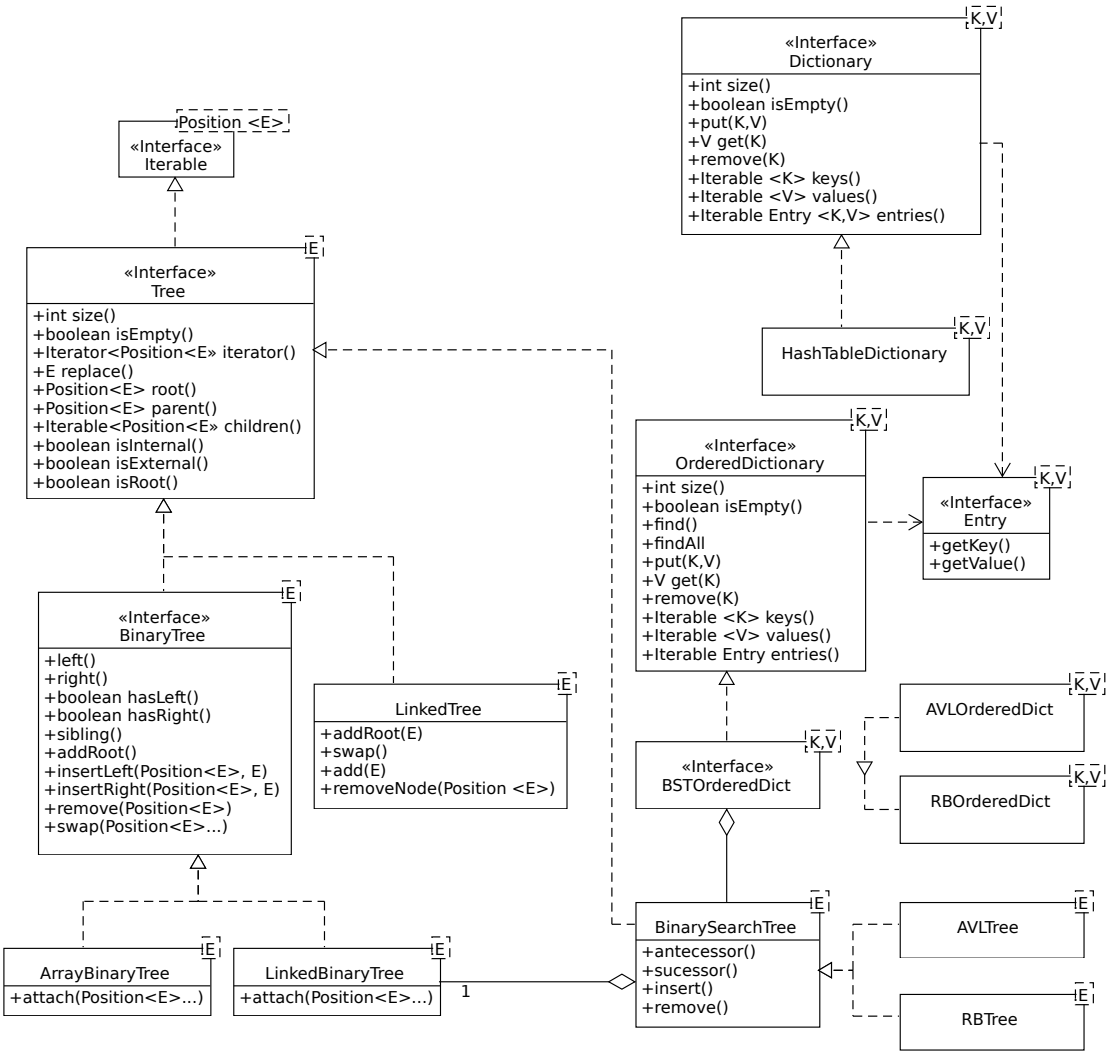


Diagrama de Clases correspondiente a las estructuras presentadas en clase

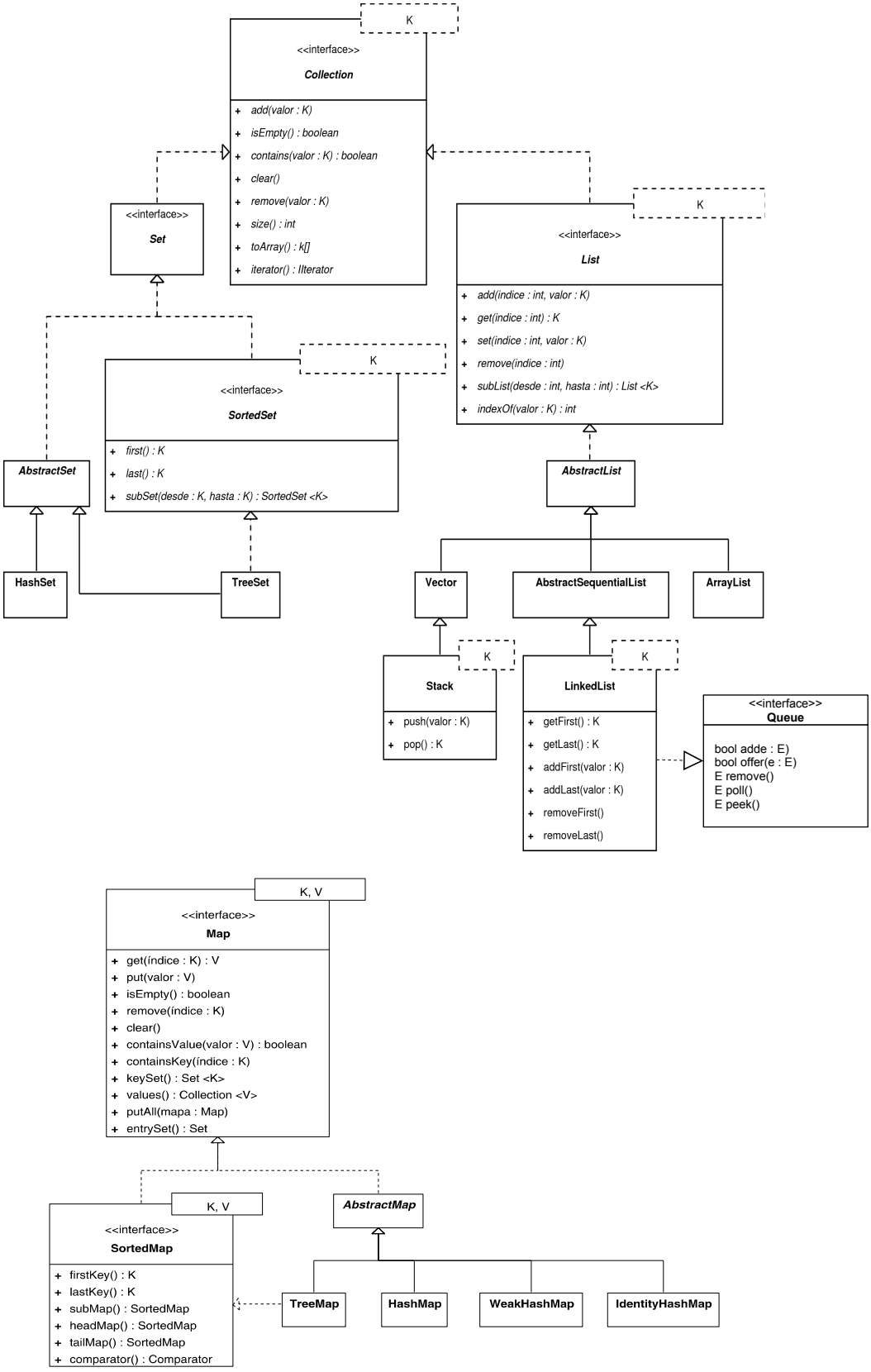


Diagrama de clases correspondiente a las estructuras propias de Java