**IREP**

**TAD Libro**{

private String isbn;

private String nombre;

private String categoria;

private int ancho;

1) isbn, nombre, categoría distintos a vacío o null.

2) ancho>0 //representa el espesor del libro en cm.

}

**TAD Estanteria**{

private String rotulo;

private int tamanio;

private int copiaTamanio;

private int cantidaddelibros;

private ArrayList<Libro> librosEnEstanteria;

\*1) rotulo distinto a null

\*2) librosEnEstanteria.length == cantidaddeLibros

\*3) para todo elem Libro en librosEnEstanteria, elem.getCategoria==rotulo //todos los libros tienen la misma categoría y la categoría==rotulo.

\*copiaTamanio – tamanio == Espacio Libre.

\*copiaTamanio == suma(tamanio, estanteria.espacioLibre).

\*0<=tamanio<=copiaTamanio //revisar los signos xdxd

\*

**int suma=0;**

**int cantidad=0;**

**for(Libro libros : librosEnEstanteria){**

**suma= libros.getAncho();**

**cantidad++;**

**}**

**this.tamanio= this.tamanio-suma;**

**this.cantidaddelibros= cantidad;**

//

-Cuando se agrega un libro, se agrega únicamente si el rotulo de la estantería es igual a la categoría del libro. Si la estantería no está rotulada, lanza una excepción.

-La estantería solo se puede Rotular si la misma no contiene libros, es decir, se encuentra vacía.

}

**TAD BDUNGS**{

private ArrayList<Estanteria> todasLasEstanterias;

private Conjunto<String> conjuntoISBN;

private Diccionario<Libro, Integer> cantDeEjemplares;

\*

int librosTotales=0

for(Estanterias est : todasLasEstanterias){

est.getCopiaTamanio()==tamaniodeEstanterias;

librosTotales+= est.getCantidadDeLibros; //suma de todos los libros que hay en las estanterias

}

librosTotales == sumaDeSignificado(this.cantDeEjemplares)

\*for(Estanterias est: todasLasEstanterias){

for(Libro libro: est.getLibrosEnEstanteria() ){

libro.getISBN()∈ conjuntoISBN;

cantDeEjemplares.existe(libro); //el libro existe en el diccionario

}

}

\*cantDeEjemplares.tamanio() == conjuntoISBN.tamanio()

//

1) todasLasEstanterias.length >0 y va a ser == a la cantidad de estanterías (la cual se pide en el constructor).

2) conjuntoISBN, String que representan el ISBN de un libro. Todos los ISBN de este conjunto, son ISBN de libros que hay en la biblioteca.

3) cantDeEjemplares es un Diccionario<Libro, Integer> la Clave es el objeto Libro y el Significado la cantidad de ejemplares de ese libro (va a ser >=1).

-Un libro es igual a otro libro, por su ISBN.

-Todas las Estanterías se inicializan con el Rotulo vacío. Para ingresar un libro, primero se necesita rotularla.

-Al ingresar un libro, se crea el libro únicamente si no se creó previamente.

-Cuando se ingresa un libro, busca una estantería con el Rotulo igual a la Categoría del Libro, si la estantería tiene espacio disponible (y este es menor al ancho del libro) lo agrega.

En el caso que el libro no puede ingresar a una estantería, lanza una excepción.

-Para eliminar un libro se requiere el ISBN, quita el libro únicamente si el mismo pertenece a alguna Estantería. Elimina todos sus ejemplares. Si el libro no se encuentra en ninguna estantería, lanza una excepción.

-Para rotular una estantería, se pide el numero de la estantería (tiene que ser mayor o igual a 1 y menor a la cantidadDeEstanteria.

}

**Métodos:**

A continuación, se va a realizar un breve resumen de los métodos mas importantes de la biblioteca, en los cuales se pasa a explicar cómo la estructura de datos escogida cumple con los requisitos pedidos.

**public int reacomodarCategoria(String categoria):**

Primero se comprueba si existe esa categoría.

Creamos una ArrayList<Libros> cajaDeLibros, utilizamos la función auxiliar quickSortMayorAMenor(ArrayList<Libros> libros), le pasamos como parámetro una ArrayList<Libros> con todos los libros dentro de la biblioteca con la categoría que se pide (utilizando una función auxiliar).

Entonces, cajaDeLibros contiene todos los libros con la misma categoría, ordenados de mayor a menor (respecto al tamaño del libro). Luego vaciamos las estanterías de la categoría que se pide reacomodar.

Con las estanterías ya vacías, por cada una recorro cajaDeLibros (de atrás hacia adelante, por ende, los libros van a ingresar de menor a mayor) y se comprueba si se puede ingresar el libro a la estantería, si se puede ingresarlo se agrega a la estantería y se elimina de la caja de libros. Si un libro no ingresa a la estantería, va a ingresar en la próxima (porque está vacía), siempre va a ingresar en una estantería.

Luego, para saber cuántas estanterías se liberaron se cuenta las que estanterías estén vacías (y que tengan esta misma categoría) y se retorna ese valor.

**HashMap<String,Integer> verLibrosCategoria(String categoria):**

Primero se comprueba si existe esa categoría

Se crea un HashMap<String,Integer> diccDeCategoria para ISBN (Clave) y el significado Integer para la cantidad de ejemplares. Nosotros utilizamos el TAD Diccionario con el que trabajamos en clases, porque era más declarativo.   
Teniendo el Diccionario<Libro, Integer> cantDeEjemplares el cual utilizamos para contar la cantidad de ejemplares de un Libro, recorremos el conjunto del diccionario cantDeEjemplares y si libro.getCategoria es igual a la categoría pasada como parámetro. Entonces, se agrega a HashMap<String,Integer> diccDeCategoria con la clave libro.getISBN y el significado cantDeEjemplares.obtener(Libro). Al final, retorna diccDeCategoria.

**int espacioLibre(int estanteria):**

Primero se comprueba si el int es válido (las estanterías se enumeran a partir del 1), si es así, le resto uno y lo guardo en un int pos.

Se pidió que este método tenga complejidad O(1), lo que hicimos fue crear el método espacioLibre en la clase de estantería. Este método retorna el tamaño de la estantería, el cual va cambiando a medida que se agregan o quitan libros de ella. En la clase BDUNGS tenemos una ArrayList<Estanterias>todasLasEstanterias, este método recibe un int el cual es el número de la estantería al cual se le consulta el espacio disponible. Accede a todasLasEstanterias.get(pos) y se llama a la función espacioDisponible(). En código es: todasLasEstanterias.get(pos).espacioDisponible();

El método retorna todasLasEstanterias.get(pos).espacioDisponible();