Apartados 2 y 3

Sugerencias de implementación

Apartado 2 - Índices

- El sistema tendrá **un índice por cada propiedad de cada tipo de dato**. El índice guardará la información ordenada por clave, para ello se sugiere usar un TreeMap<Object, Set<Long>> (Por ejemplo, en un libro habrá un índice por título, otro por autor y otro por editorial).
- Para poder asociar valores con la clave "null", se puede tratar este como un conjunto especial, o se puede crear el TreeMap con un comparator que admita valores nulos.

```
public class IndexImpl implements Index {
    SortedMap<Object, Set<Long>> data= new TreeMap<>(AllowNullComparator.Instance);
    @Override
    public void add(Object key, Long value) {
        Set<Long> ids= data.get(key);
        if (ids==null) {
            ids= new HashSet<>();
            data.put(key,  ids);
        }
        ids.add(value);
    }
```

```
class AllowNullComparator implements Comparator<Object>{
    public static final AllowNullComparator Instance= new AllowNullComparator();
    @SuppressWarnings("unchecked")
    @Override
    public int compare(Object a, Object b) {
        if (a == null) {
            return b == null ? 0 : -1;
        } else if (b == null) {
            return 1;
        } else return ((Comparable<Object>)a).compareTo(b);
    }

}
```

```
ndex.add("4", 41);
```

Asignación "Hash" a la clave "Key" le asigna el valor "value". Ejemplo "Editorial" *(ordena por key)*

Key	Value
"Alfaguara"	3
"Bruguera"	4, 7
"Edelvives"	1
"Mc Graw Hill"	10 , 9, 5

Apartado 2 – Índices (Tester)

- Programa que genere Pares Clave, Valor y luego pruebe funciones add, delete, search.

```
index.add("12", 11);
index.add("12", 21);
index.add("23", 21);
index.add("23", 31);
index.add("4", 41);
```

Key	Value
“12”	1,2
“23”	2,3
“4”	4

```
index.search("4").contains(11);
```
Sería Falso

```
index.delete("4", 41);
```
Borra de la tabla el par “4”,4

```
index.search("4").contains(41)
```
Sería Falso

```
index.search("26"), Collections.emptySet()
```
La comparación sería Verdadera

```
index.search("12", "23");
```
Devolverá la colección de ids 1, 2, 3 (key es string)

```
index.search("23", null)
```
Devolverá la colección de ids 2, 3, 4 (key es string)

- En el caso de métodos que devuelven colecciones, se recomienda que estas sean inmutables. Esto se puede conseguir usando los métodos de Collections, como por ejemplo para listas con el método :Collections.unmodifiableList(List<? extends T> list)

```
Collections.unmodifiableSet(ids);
```

Apartado 3 – Tablas de Datos

- Una tabla tiene su nombre (Descriptor), sus índices (tantos como propiedades), y sus datos.
- “la lista de pares propiedad-valor. Esto se puede hacer con un Map<Long, Map<String, Object>>, donde Map<String, Object> representa los pares propiedad-valor de una entidad.”

```
private final TypeDescriptor descriptor;
private final Map<String, Index> indexes;
private final Map<Long, Map<String, Object>> data;

private Long count=0;
```

- UpdateEntity será utilizado como mecanismo de incorporación de Entity a la tabla. (si existe borraremos la ocurrencia anterior para generar siempre el índice)

```
Map<String, Object> old = (e.getId()!=null) ? data.get(e.getId()) : null;
//para simplificar la actualización primero borramos los valores antiguos de los índices
if (old!=null) removeIndexes(old, e.getId());
```

- Posteriormente:
  - Asignamos nuevo id (o mantenemos el que existía)
  - Para cada una de las propiedades (bucle bajo getProperties)
    - Añadimos el Valor de la Nueva Entidad para esa propiedad
    - Añadimos el par “Valor”,id al índice de búsqueda de dicha propiedad

LIBRO (Table) **descriptor**

indexes
Título
Editorial
Autor

Key	Value
“Bruguera”	4, 7
“Edelvives”	1

data

Id	Titulo	Editorial	Autor
1	“Viven”	“Edelvives”	6
4	“La vuelta al mundo”	“Bruguera”	7
7	“Nueve Reinas”	“Bruguera”	6

Apartado 3 – Tablas de Datos (Tester)

```
type=Libro.getDescriptor();
table= new TableImpl(type);
```

Generamos una tabla de libros

```
table.updateEntity(l1);
```

Si "l1" es un libro, incorporará dicho libro a la tabla (de datos e índices)

```
Libro l=(Libro)table.getEntity(0L);
```

Si hemos incorporado el libro "0L" (el primero). Devolverá un objeto de tipo Libro (comparar con otro objeto creado en modo "Práctica 2")

```
table.delete(l2);
```

Si "l2" es un libro, lo borrará y excluirá dicho libro de la tabla (de datos e índices)

```
Collection<Long> r= table.search("autor", 1L);
```

Buscará todos los libros con autor "1L"

```
table.search("editorial", "The", "Z");
```

Buscará todos los libros con la editorial desde "The" hasta "Z"