Ejercicio U7_B5_E1: Repite el ejercicio anterior pero utilizando la clase LinkedHashSet

Solo hay que sustituir la clase HashSet por LinkedHashSet

```
import java.util.LinkedHashSet;
public class App {
    public static void main(String[] args) {
        LinkedHashSet<String> conjuntoVehiculos=new LinkedHashSet<String>();
        conjuntoVehiculos.add("moto");
        conjuntoVehiculos.add("coche");
        conjuntoVehiculos.add("bici");
        conjuntoVehiculos.add("moto");
        conjuntoVehiculos.add("bici");
        String sVehiculos="";

        for(String s: conjuntoVehiculos)
            System.out.print(s+" ");
    }
}
```

Ejercicio U7_B5_E2:
Solo hay que sustituir la clase LinkedHashSet por TreeSet

```
import java.util.TreeSet;
public class App {
    public static void main(String[] args) {
        TreeSet<String> conjuntoVehiculos=new TreeSet<>();
        conjuntoVehiculos.add("moto");
        conjuntoVehiculos.add("coche");
        conjuntoVehiculos.add("bici");
        conjuntoVehiculos.add("moto");
        conjuntoVehiculos.add("bici");
        String sVehiculos="";

        for(String s: conjuntoVehiculos)
            System.out.print(s+" ");
    }
}
```

CONCLUSIÓN: Hashset**,** LinkedHashSet y TreeSet almacenan un conjunto y ofrecen al usuario de estas clases las operaciones matemáticas de conjunto (observa por ejemplo, que en los tres casos no admite almacenar elementos repetidos). Desde el punto de vista de su uso las tres se comportan igual. La diferencia entre unas y otras es interna y esto condiciona que cada una de ellas sea más o menos eficiente en un contexto dado.

Ejercicio U7_B5_E3:

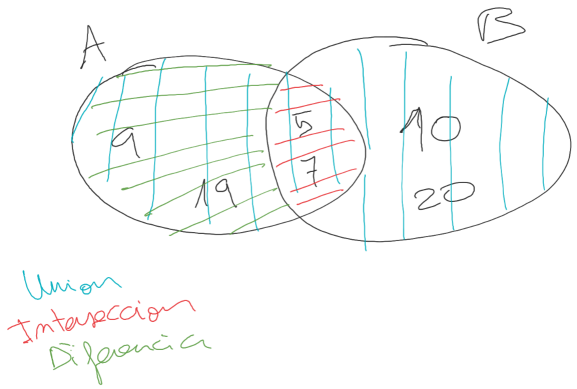
```
import java.util.Set;
import java.util.TreeSet;
public class App {
    public static void main(String[] args) {
        Set<String> conjuntoVehiculos=new TreeSet<>();
        conjuntoVehiculos.add("moto");
        conjuntoVehiculos.add("coche");
        conjuntoVehiculos.add("bici");
        conjuntoVehiculos.add("moto");
        conjuntoVehiculos.add("bici");
```

```
        String sVehiculos="";

        for(String s: conjuntoVehiculos)
            System.out.print(s+" ");
    }
}
```

Mismas consideraciones que cuando hicimos un ejercicio similar con el interface List

Ejercicio U7_B5_E4:



Cómo A y B los declaramos inmutables, utilizamos un tercer conjunto C para tener un Set modificable.

```
import java.util.Set;
import java.util.TreeSet;

class App {
    /*
     * A:[5, 7, 9, 19]
     * B:[5, 7, 10, 20]
     * A union B: [5, 7, 9, 10, 19, 20]
     * A diferencia B: [9, 19]
     * A intersección B: [5, 7]
     *
     */
    public static void main(String[] args) {
        Set<Integer> A = Set.of(5, 7, 9, 19);
        Set<Integer> B = Set.of(5, 7, 10, 20);
        Set<Integer> C;

        System.out.println(A);
        System.out.println(B);
        // UNION
        C = new TreeSet<>(A);
        C.addAll(B);// los repetidos no se agregan
        System.out.println("UNION:" + C);

        C = new TreeSet<>(A);
        C.retainAll(B);
        System.out.println("INTERSECCION: " + C);

        C = new TreeSet<>(A);
        C.removeAll(B);
        System.out.println("DIFERENCIA: " + C);

    }
}
```