Marcar como hecha

3.2.1 Collections

Ejemplo de uso de métodos

En todos recordar:

```
import java.util.*;
```

01ArrayList

01.1 Uso de método add()

```
        public static void main(String[] args) {
                  // Crea un ArrayList vacío
                   LinkedList<String> l_list = new LinkedList<String>();
                 // uso de método add() añade
                        l_list.add("Red");
                        l_list.add("Green");
                        l_list.add("Black");
                        l_list.add("White");
                        l_list.add("Pink");
                       // imprime
                for (String element : l_list) {
                  System.out.println(element);
                  }
        }
```

01.2 Uso de set()

```
    public static void main(String[] args) {
         // Crea un ArrayList vacío
          List<String> list_Strings = new ArrayList<String>();
          list_Strings.add("Red");
          list_Strings.add("Green");
          list_Strings.add("Orange");
          list_Strings.add("White");
          list_Strings.add("Black");
          // imprime
          System.out.println(list_Strings);
          // Actualiza el tercero con "Yellow"
          list_Strings.set(2, "Yellow");
          // imprime
          System.out.println(list_Strings);
         }
```

01.3 Uso de contains()

```
      public static void main(String[] args) {
            // Crea un ArrayList vacío
          List<String> list_Strings = new ArrayList<String>();
          list_Strings.add("Red");
          list_Strings.add("Green");
          list_Strings.add("Orange");
          list_Strings.add("White");
          list_Strings.add("Black");
          // Buscamos el rojo
            if (list_Strings.contains("Red")) {
            System.out.println("Encontrado elemento");
```

?

```
            } else {
            System.out.println("no hay ese elemento");
            }
          }
```

01.4 Uso de sort()

```
      public static void main(String[] args) {
            // Crea un ArrayList vacío
          List<String> list_Strings = new ArrayList<String>();
          list_Strings.add("Red");
          list_Strings.add("Green");
          list_Strings.add("Orange");
          list_Strings.add("White");
          list_Strings.add("Black");
          System.out.println("List antes sort: "+list_Strings);
          Collections.sort(list_Strings);
          System.out.println("List despues sort: "+list_Strings);
         }
```

02 LinkedList

02.1 Uso de getFirst() y getLast()

```
    public static void main(String[] args) {
        // crea un LinkedList vacío
         LinkedList<String> l_list = new LinkedList<String>();
       // use add() method to add values in the linked list
              l_list.add("Red");
              l_list.add("Green");
              l_list.add("Black");
              l_list.add("Pink");
              l_list.add("orange");

          // imprime original
       System.out.println("Original :" + l_list);

       // encuentra primera usa Iterator;
        Object first_element = l_list.getFirst();
        System.out.println("Primer elemento es: "+first_element);

        // encuentra última usa Iterator;
        Object last_element = l_list.getLast();
        System.out.println("Ultimo elemento es: "+last_element);
     }
```

02.2 Uso de clear()

```
   public static void main(String[] args) {
            // crea linked list
             LinkedList<String> l_list = new LinkedList<String>();
           // use add() method to add values in the linked list
                 l_list.add("Red");
                 l_list.add("Green");
                 l_list.add("Black");
                 l_list.add("Pink");
                 l_list.add("orange");

             // imprimo
           System.out.println("The Original linked list: " + l_list);
```

```
        // elimino valores
          l_list.clear();

          System.out.println("The New linked list: " + l_list);
        }
```

03 HashSet

03.1 Uso de toArray()

```
    public static void main(String[] args) {
        // Crea hash set
    HashSet<String> h_set = new HashSet<String>();
    // add() y añade elementos también aquí
         h_set.add("Red");
         h_set.add("Green");
         h_set.add("Black");
         h_set.add("White");
         h_set.add("Pink");
         h_set.add("Yellow");
     System.out.println("Original Hash Set: " + h_set);
     // Crea  Array
     String[] new_array = new String[h_set.size()];
     h_set.toArray(new_array);

    // imprime Array
    System.out.println("Array elements: ");
     for(String element : new_array){
       System.out.println(element);
    }
  }
```

03.2 Uso de raiz común en transformación de hashSet a treeSt

```
        public static void main(String[] args) {
        // Crea hash set
    HashSet<String> h_set = new HashSet<String>();
  // use add() method to add values in the hash set
         h_set.add("Red");
         h_set.add("Green");
         h_set.add("Black");
         h_set.add("White");
         h_set.add("Pink");
         h_set.add("Yellow");
     System.out.println(" Hash Set Original: " + h_set);

    // Crea TreeSet de los  elementos del HashSet
    Set<String> tree_set = new TreeSet<String>(h_set);

    // muestra elementos TreeSet
    System.out.println(" elementos TreeSet: ");
    for(String element : tree_set){
       System.out.println(element);
    }
  }
```

04 TreeSet

04.1 Uso del toString() y add()

```
public static void main(String[] args) {
```

```
 TreeSet<String> tree_set = new TreeSet<String>();
 tree_set.add("Red");
 tree_set.add("Green");
 tree_set.add("Orange");
 tree_set.add("White");
 tree_set.add("Black");
 System.out.println("Tree set: ");
 System.out.println(tree_set);
}
```

04.2 Uso de Iterator() y iterator.hasNext()

```
public static void main(String[] args) {
```

```
// crea TreeSet
  TreeSet <Integer>tree_num = new TreeSet<Integer>();
  TreeSet <Integer>treeheadset = new TreeSet<Integer>();

  // agnado numeros
  tree_num.add(1);
  tree_num.add(2);
  tree_num.add(3);
  tree_num.add(5);
  tree_num.add(6);
  tree_num.add(7);
  tree_num.add(8);
  tree_num.add(9);
  tree_num.add(10);

  // Encuentra números menores que  7
treeheadset = (TreeSet)tree_num.headSet(7);

  // crea iterator
  Iterator iterator;
  iterator = treeheadset.iterator();

  //imprime datos de treeset
  System.out.println("Datos en TreeSet: ");
  while (iterator.hasNext()){
  System.out.println(iterator.next() + " ");
  }
  }
```

04.3 Uso de remove()

```
 public static void main(String[] args) {
// crea TreeSet
  TreeSet <Integer>tree_num = new TreeSet<Integer>();
  TreeSet <Integer>treeheadset = new TreeSet<Integer>();

  // Agnado numeros
  tree_num.add(10);
  tree_num.add(22);
  tree_num.add(36);
  tree_num.add(25);
  tree_num.add(16);
  tree_num.add(70);
  tree_num.add(82);
  tree_num.add(89);
  tree_num.add(14);
  System.out.println("tree set original: "+tree_num);
  System.out.println("Eliminamos el 70 de la lista: "+tree_num.remove(70));
  System.out.println("Tree set after despues de quitar el 70: "+tree_num);
  }
```

05 PriorityQueue

05.1 Uso general

```
public static void main(String[] args) {
```

```
 PriorityQueue<String> queue=new PriorityQueue<String>();
 queue.add("Red");
 queue.add("Green");
 queue.add("Orange");
 queue.add("White");
 queue.add("Black");
 System.out.println("Elementos Priority Queue: ");
 System.out.println(queue);
}
```

06 HashMap

06.1 Uso de put() getKey, getValue()

```
        public static void main(String args[]) {
               HashMap<Integer,String> hash_map= new HashMap<Integer,String>();
               hash_map.put(1, "Red");
               hash_map.put(2, "Green");
               hash_map.put(3, "Black");
               hash_map.put(4, "White");
               hash_map.put(5, "Blue");
               for(@SuppressWarnings("rawtypes") Map.Entry x:hash_map.entrySet()){
                System.out.println(x.getKey()+" "+x.getValue());
               }
              }
```

06.2 Uso de clone()

```
    public static void main(String args[]){
         HashMap<Integer,String> hash_map= new HashMap<Integer,String>();
         hash_map.put(1, "Red");
         hash_map.put(2, "Green");
         hash_map.put(3, "Black");
         hash_map.put(4, "White");
         hash_map.put(5, "Blue");
           // print the map
          System.out.println("map original: " + hash_map);
          HashMap<Integer,String> new_hash_map= new HashMap<Integer,String>();
          new_hash_map = (HashMap)hash_map.clone();
          System.out.println("Clonado : " + new_hash_map);
            }
```

07 TreeMap

07.1 uso de entrySet()

```
        public static void main(String args[]){
              // Crea tree map
              TreeMap<Integer,String> tree_map=new TreeMap<Integer,String>();
             // Put elements to the map
             tree_map.put(1, "Red");
             tree_map.put(2, "Green");
             tree_map.put(3, "Black");
             tree_map.put(4, "White");
             tree_map.put(5, "Blue");

              for (Map.Entry<Integer,String> entry : tree_map.entrySet())
              {
               System.out.println(entry.getKey() + "=>" + entry.getValue());
              }
            }
```

07.2 Uso de containsKey()

```
    public static void main(String args[]){
         // Crea tree map
          TreeMap<String,String> tree_map1=new TreeMap<String,String>();
         // coloca elementos map
         tree_map1.put("C1", "Red");
         tree_map1.put("C2", "Green");
         tree_map1.put("C3", "Black");
         tree_map1.put("C4", "White");
         System.out.println(tree_map1);
               if(tree_map1.containsKey("C1")){
                   System.out.println("el TreeMap contiene C1 como clave");
               } else {
                   System.out.println("el TreeMap NO contiene C1 como clave");
               }
               if(tree_map1.containsKey("C5")){
                   System.out.println("el TreeMap contiene C5 como clave");
               } else {
                   System.out.println("el TreeMap NO contiene C5 como clave");
               }
           }
```

07.3 Buscar la clave mayor o igual que sea inferior a una clave dada con floorKey

```
    public static void main(String args[]) {
// Crea tree map
TreeMap < Integer, String > tree_map1 = new TreeMap < Integer, String > ();
// Put elements to the map
tree_map1.put(10, "Red");
tree_map1.put(20, "Green");
tree_map1.put(40, "Black");
tree_map1.put(50, "White");
tree_map1.put(60, "Pink");

System.out.println("Contenido original TreeMap: " + tree_map1);
System.out.println("Buscando 10: ");
System.out.println("La clave encontrada es: " + tree_map1.floorKey(10));
System.out.println("Buscando 30: ");
System.out.println("La clave encontrada es: " + tree_map1.floorKey(30));
System.out.println("Buscando 70: ");
System.out.println("La clave encontrada es: " + tree_map1.floorKey(70));
}
```

07.4 uso de subMap:

```
    public static void main(String args[]) {
     // Declaro tree maps
     TreeMap < Integer, String > tree_map = new TreeMap < Integer, String > ();
     SortedMap < Integer, String > sub_tree_map = new TreeMap < Integer, String > ();
     // uso Put
     tree_map.put(10, "Red");
     tree_map.put(20, "Green");
     tree_map.put(30, "Black");
     tree_map.put(40, "White");
     tree_map.put(50, "Pink");
     System.out.println(" TreeMap Orginal: " + tree_map);
     sub_tree_map = tree_map.subMap(20, true, 40, true);
     System.out.println("Sub map nuevo: " + sub_tree_map);
    }
```

07.5 uso de tailMap()

```
public static void main(String args[]) {
// Declo tree maps
TreeMap < Integer, String > tree_map = new TreeMap < Integer, String > ();
// Put
tree_map.put(10, "Red");
tree_map.put(20, "Green");
tree_map.put(30, "Black");
tree_map.put(40, "White");
tree_map.put(50, "Pink");
System.out.println(" TreeMap Orginal: " + tree_map);
System.out.println("grupo con claves mayores o iguales a 20: " + tree_map.tailMap(20));
}
```