

Ejemplo de Utilización de Collection Sort

Es muy útil cuando tenemos una colección de Objetos y necesitamos ordenarla. La forma de utilización es la siguiente:

```
import java.util.Collections;
...
Collections.sort(Vector);
    o bien
Collections.sort(Vector, Comparador);
```

Para poder utilizarlo de la primer forma el vector tiene que ser de Objetos que tengan implementado el método compareTo. Por ejemplo un vector de Integer o de String:

```
Vector<Integer> conjunto;
conjunto = new Vector<Integer>();
conjunto.add(new Integer(3));
conjunto.add(new Integer(4));
conjunto.add(new Integer(3));
conjunto.add(new Integer(5));
conjunto.add(new Integer(8));
conjunto.add(new Integer(2));
conjunto.add(new Integer(1));
conjunto.add(new Integer(20));
conjunto.add(new Integer(19));
Collections.sort(conjunto);
```

En el caso de tener un Vector de Objetos propios debemos implementar el método de la siguiente manera teniendo en cuenta que debe retornar un int, -1 si el primero es menor que el segundo, 0 si son iguales y 1 en el caso de que el primero sea mayor al segundo:

```
public class Persona implements Comparable{
    private String name;
    private String apellido;
    private int age;
    private int dni;
    private String mail;

    public Persona(String n, String a, int ag, int d, String m){
        name = n;
        apellido = a;
        age = ag;
        dni = d;
        mail = m;
    }

    public void mostrar(){
        System.out.println("- -- -- -- -- -");
        System.out.println("Nombre: " + name);
        System.out.println("Apellido: " + apellido);
        System.out.println("Edad: " + age);
        System.out.println("D.N.I.: " + dni);
        System.out.println("Mail: " + mail);
        System.out.println("- -- -- -- -- -" + "\n");
    }

    public int compareTo(Object o){
        Persona P2 = (Persona)o;
        if(this.apellido.compareTo(P2.apellido) == 0){
            if(this.name.compareTo(P2.name) == 0){
                if(this.dni < P2.dni)
                    return -1;
                else
                    return 1;
            }
            else
                return this.name.compareTo(P2.name);
        }
        else
            return this.apellido.compareTo(P2.apellido);
    }
}

import java.util.Collections;
public class Principal {

    public static void mostrarVector(Vector v){
        for(int i = 0; i < v.size(); i++){
            ((Persona)v.elementAt(i)).mostrar();
        }
    }

    public static void main(String[] args) {
        Vector v = new Vector();
        v.add(new
Persona("Rodrigo","Ibañez",21,33719996,"rsi_el_loco@hotmail.com"));
        v.add(new
Persona("Claudia","Suizan",48,14170590,"claudia@hotmail.com"));
        v.add(new Persona("Juana","Ibañez",6,48123456,"juana@hotmail.com"));
        mostrarVector(v);
        Collections.sort(v);
        mostrarVector(v);
    }
}
```

Si lo que necesitamos es poder comparar mas dinamicamente por multiples atributos entonces tenemos que crear un comparador e implementar el metodo compare entre los 2 objetos que queremos comparar de la siguiente manera:

```
import java.util.Comparator;

public class Comparador implements Comparator{
    Comparador sig;
    String id;

    public Comparador(String i, Comparador s){
        id = i;
        sig = s;
    }

    public int compare(Object P, Object R) {
        Persona P1 = (Persona)P;
        Persona P2 = (Persona)R;
        int res = ((Comparable)P1.get(id)).compareTo(P2.get(id));
        if(res == 0 && sig != null)
            return sig.compare(P, R);
        else
            return res;
    }
}

import java.util.Enumeration;

public class Persona {
    Hashtable h;

    public Persona(String n, String a, Integer d){
        h = new Hashtable();
        this.add("Nombre", n);
        this.add("Apellido", a);
        this.add("D.N.I.", d);
    }

    public void add(String id, Object valor){
        h.put(id, valor);
    }

    public Object get(String id){
        return h.get(id);
    }

    public void mostrar(){
        System.out.println("-  --  --  --  --  -");
        for(Enumeration e = h.keys(); e.hasMoreElements(); ){
            String id = (String)e.nextElement();
            System.out.println(id + " " + this.get(id));
        }
        System.out.println("-  --  --  --  --  -" + "\n");
    }
}
```

```
import java.util.Collections;

public class Principal {

    public static void mostrarVector(Vector v){
        for(int i = 0; i < v.size(); i++){
            ((Persona)v.elementAt(i)).mostrar();
        }
    }

    public static void main(String[] args) {
        Vector v = new Vector();
        v.add(new Persona("Rodrigo", "Ibañez", new Integer(33719996)));
        v.add(new Persona("Claudia", "Suizan", new Integer(14170590)));
        v.add(new Persona("Juana", "Ibañez", new Integer(48123456)));

        ((Persona)v.elementAt(0)).add("Edad", new Integer(21));
        mostrarVector(v);

        Comparador C3 = new Comparador("D.N.I.", null);
        Comparador C2 = new Comparador("Nombre", C3);
        Comparador C = new Comparador("Apellido", C2);

        Collections.sort(v, C);
        mostrarVector(v);
    }
}
```

Tener en cuenta las siguientes diferencias de sintaxis:

Primer ejemplo:

```
public class Persona implements Comparable{
    ...
    public int compareTo(Object o){
    ...
}
```

Segundo ejemplo:

```
public class Comparador implements Comparator{
    ...
    public int compare(Object P, Object R) {
    ...
}
```