

```

1  /**
2      FACTORY
3  */
4  public static Objeto factory(String[] data){
5      int atr1,atr2;
6      float atr3;
7
8      if(data.length() ≠ 5){
9          return null;
10     }else{
11         try{
12             atr1 = Integer.parseInt(data[0]);
13             atr2 = Integer.parseInt(data[1]);
14             atr3 = Float.parseFloat(data[3]);
15         }catch(NumberFormatException e){
16             return null;
17         }
18         return new Objeto(atr1,atr2,data[2],atr3,data[4]);
19     }
20 }
21
22 /**
23     ORDENAR
24 */
25
26 public void ordenarObjetos(){
27     //Ordenar por un parámetro
28     Collections.sort(listaObjetos,
29     Comparator.comparing(Objeto::getAtributo1));
30     //Ordenar por mas de 1 parámetro (2 en este caso)
31     Collections.sort(listaObjetos,
32     Comparator.comparing(Objeto::getAtributo1).thenComparing(Objeto::getAtributo2
33 ));
34 }
35
36 /**
37     DAR DE ALTA
38 */
39
40 public void altaObjeto(String[] data){
41     Objeto objeto = Objeto.factory(data);
42     if(objeto ≠ null){
43         listaObjetos.add(objeto);
44     }
45 }
46
47 /**
48     DAR DE BAJA
49 */
50
51 public void bajaObjeto(String eliminar){
52     for(Objeto objeto : listaObjetos){
53         if(objeto.getAtributo1().equals(eliminar)){
54             listaObjetos.remove(eliminar);
55         }
56     }
57 }
58
59 /**
60     MODIFICAR ATRIBUTOS
61 */

```

```

57 public void modificar(String Atributo, String newAtributo){
58     for(Objeto objeto : listaObjetos){
59         if(objeto.getAtributo1().equalsIgnoreCase(Atributo)){
60             objeto.setAtributo1(newAtributo);
61         }
62     }
63 }
64
65 /**
66     CONSULTAR OBJETOS
67 */
68 public String consultar(String Atributo){
69     String consulta="";
70     for(Objeto objeto : listaObjetos){
71         if(objeto.getAtributo1().equalsIgnoreCase(Atributo)){
72             for(String aux : objeto.auxiliar){
73                 consulta = consulta + aux + "\n";
74             }
75         }
76     }
77 }
78
79 /**
80     ARCHIVOS DELIMITADOS IMPORTAR
81 */
82 public void importar(){
83     String[][] data;
84     Path p = Rutas.pathToFileInFolderOnDesktop("FOLDER","archivo.txt");
85     File f = p.toFile();
86     try{
87         data = importFromDisk(f,"#");
88     }catch(IOException e){
89         e.printStackTrace();
90     }
91     Objeto objeto;
92     for(String[] x : data){
93         objeto = Objeto.factory(x);
94         if(objeto != null){
95             listaObjetos.add(objeto);
96         }
97     }
98 }
99
100 /**
101     ARCHIVOS DELIMITADOS EXPORTAR
102 */
103 public void exportar(){
104     Path p = Rutas.pathToFileInFolderOnDesktop("FOLDER","archivo.txt");
105     File f = p.toFile();
106     String[][] data = new String[listaObjetos.size()][];
107     for(int i=0;i<listaObjetos.size();i++){
108         data[i]=listaObjetos.get(i).toString().split("#");
109     }
110     try{
111         exportToDisk(data,f,"#");
112     }catch(IOException e){
113         e.printStackTrace();
114     }
115 }
116

```

```

117 /**
118     ARCHIVOS BINARIOS CARGAR
119 */
120 public void cargar(){
121     Path p = Rutas.pathToFileInFolderOnDesktop("FOLDER","archivo.bin");
122     File f = p.toFile();
123     try{
124         FileInputStream fis = new FileInputStream(f);
125         BufferedInputStream bis = new BufferedInputStream(fis);
126         ObjectInputStream ois = new ObjectInputStream(bis);
127         listaObjetos = (ArrayList<Objeto>) ois.readObject();
128         ois.close();
129     }catch(IOException e ){
130         e.printStackTrace();
131     }
132 }
133
134 /**
135     ARCHIVOS BINARIOS GUARDAR
136 */
137 public void guardar(){
138     Path p = Rutas.pathToFileInFolderOnDesktop("FOLDER","archivo.bin");
139     File f = p.toFile();
140     try{
141         FileOutputStream fos = new FileOutputStream(f);
142         BufferedOutputStream bos = new BufferedOutputStream(fos);
143         ObjectOutputStream oos = new ObjectOutputStream(bos);
144         oos.writeObject(listaObjetos);
145         oos.close();
146     } catch(IOException e ){
147         e.printStackTrace();
148     }
149 }
150
151 /**
152     TOSTRING
153 */
154 public void toString(){
155     return atr1 + "#" + atr2 + "#" + atr3 + "#";
156 }
157 public void toString(){
158     String data = "";
159     data = getAtributo1() + "#" + getAtributo2() + "#";
160     for(String aux : auxiliar){//auxiliar es un arraylist<string> por ejemplo
profesores, asignaturas
161         data = data + " " + aux;
162     }
163     return data;
164 }
165
166 /**
167     ARRAYLIST DE OBJETOS A MATRIZ DE STRING
168 */
169 public String[][] ArrayListToMatrix(){
170     String[][] data = new String[listaObjetos.size()][];
171     for(int i=0;i<listaObjetos.size();i++){
172         data[i]=listaObjetos.get(i).toString().split("#");
173     }
174     return data;
175 }

```

```
176 // MOSTRAR POR PANTALLA
177 public void mostrarObjetos(){
178     String[][] data = ArrayListToMatrix();
179     printToScreen3(data);
180 }
181
```