```java
/**
 * @ (#) GestionTienda.java
 *
 * Clase GestionTienda.
 * La clase GestionTienda desarrolla todos los métodos que dan vida, que dan funcionalidad,
 * a toda instancia que vayamos a crear de las clases Electrodomestico, Cliente y Promo.
 * Para almacenar una colección de objetos tenemos los HashMap electrodomesticos, clientes y promos,
 * y el ArrayList tecnicos.
 * Estas colecciones tienen la función de registro general, es decir, en cada mapa o lista se guarda
 * respectivamente todo objeto creado que necesite almacenar el sistema de gestión, y a su vez estar disponibles,
 * ya que diferentes métodos necesitaran acceder a dichas colecciones con frecuencia.
 *
 * @author Yassine Marroun
 * @version 1.00 2018/05/04
 */
package gestion;
import modelo.bd.Electrodomestico;

public class GestionTienda {

    public static Map<String, Electrodomestico> electrodomesticos = new HashMap<String, Electrodomestico>();
    public static Map<String, Cliente> clientes = new HashMap<String, Cliente>();
    public static List<Usuario> tecnicos = new ArrayList<Usuario>();
    public static Map<String, Promo> promos = new HashMap<String, Promo> ();


    /*
    El método crearDatosInicialesPruebas realiza la función de banco de pruebas, es decir,
    en él creamos una serie de objetos que se incluyen en los correspondientes mapas o lista,
    mencionados anteriormente. Con ello tenemos una base sobre la que realicemos pruebas
    una vez se inicia el programa.
     */
    public void crearDatosInicialesPruebas() {

        Hogar electro1 = new Hogar("Bosch", "SMS58M18", "Inox", 499, 8, 0, "A+", "845x600x600", 56);
        Imagen electro2 = new Imagen("Samsung", "UE65MU6105KXXC", "Negro", 951, 12, 0, 65, "Ultra HD 4K", 1300);
        Informatica electro3 = new Informatica("Lenovo", "Ideapad 320-15ISK", "Negro Onyx", 399, 24, 0, "Intel Core i3-6006U", "8 GB", "1 TB");
        Sonido electro4 = new Sonido("Bose", "SoundLink Mini II", "Perla", 199, 18, 0, "100 - 240 V", "Bluetooth", "microUSB, AUX");
        Telefonia electro5 = new Telefonia("Sony", "Xperia XA2 Ultra", "Negro", 429, 32, 0, "Full HD (1080p)", "16 MP - 23 MP", "3580 mAh");

        electrodomesticos.put("SMS58M18", electro1);
        electrodomesticos.put("UE65MU6105KXXC", electro2);
        electrodomesticos.put("Ideapad 320-15ISK", electro3);
        electrodomesticos.put("SoundLink Mini II", electro4);
        electrodomesticos.put("Xperia XA2 Ultra", electro5);

        Cliente cliente1 = new Cliente("Pedro", "Sanchez", "2453175S", "Calle Los Alpes, 23.", "632012587");
        Cliente cliente2 = new Cliente("Isabel", "Romero Diaz", "3785425A", "Calle Arcillas, 37.", "693214758");
        Cliente cliente3 = new Cliente("Daniel", "Escalante", "41236984H", "Avenida de las Retamas, 11.", "630247852");
```

```
66          Cliente cliente4 = new Cliente("Juan", "Silva", "75216235P", "Calle Los Arces,
    72.", "916325748");
67
68          clientes.put("2453175S", cliente1);
69          clientes.put("3785425A", cliente2);
70          clientes.put("41236984H", cliente3);
71          clientes.put("75216235P", cliente4);
72
73          Empleado tecnico1 = new Empleado (Enumerados.TipoEmpleado.TECNICO, "Jorge",
    "Garrido", "45786329S");
74          Empleado tecnico2 = new Empleado (Enumerados.TipoEmpleado.TECNICO, "Carlos",
    "Bueno", "85147956Y");
75          Empleado tecnico3 = new Empleado (Enumerados.TipoEmpleado.TECNICO, "Sergio",
    "Ibaiz", "54359624N");
76          tecnicos.add(tecnico1);
77          tecnicos.add(tecnico2);
78          tecnicos.add(tecnico3);
79
80
81          Promo promo1 = new Promo("Navidades", "Descuento aplicable del 23/12 al 08/01.",
    10);
82          Promo promo2 = new Promo("DiasEspeciales", "Descuento aplicable en fechas
    señaladas.", 15);
83          Promo promo3 = new Promo("RebajasEnero", "Descuento aplicable del 09/01 al 08/02.",
    25);
84          Promo promo4 = new Promo("RebajasJulio", "Descuento aplicable del 01/07 al 31/07.",
    25);
85          Promo promo5 = new Promo("BlackFriday", "Descuento aplicable del 20/11 al 25/11.",
    18);
86
87          promos.put("Navidades", promo1);
88          promos.put("DiasEspeciales", promo2);
89          promos.put("RebajasEnero", promo3);
90          promos.put("RebajasJulio", promo4);
91          promos.put("BlackFriday", promo5);
92
93      }
94
95
96      // Métodos para gestionar ELECTRODOMESTICOS.
97
98      /*
99      El método crearElectrodomestico crea una nueva instancia de Electrodomestico,
100     le asigna los datos de la instancia que devuelve el método datosElectrodomestico
101     y por último llama al método guardarElectrodomestico, pasándole como parámetro el
    objeto
102     Electrodomestico creado.
103      */
104     public void crearElectrodomestico(){
105
106         Electrodomestico electrodomestico = new Electrodomestico();
107         electrodomestico = electrodomestico.datosElectrodomestico(true, null);
108         guardarElectrodomestico(electrodomestico);
109
110     }
111
112
113     /*
114     El método guardarElectrodomestico, mediante el campo modelo, comprueba si ya existe un
    objeto igual en el mapa
115     electrodomesticos, si no es así, incluye en el mapa el nuevo objeto recibido como
    parámetro.
```

```
116     */
117    public void guardarElectrodomestico(Electrodomestico electrodomestico){
118        Electrodomestico electro = electrodomesticos.get(electrodomestico.getModelo());
119        if (electro != null) {
120            System.out.println("No se puede introducir este Electrodomestico. Ya esta
   registrado.");
121        } else {
122            electrodomesticos.put(electrodomestico.getModelo(), electrodomestico);
123        }
124    }
125
126
127    /*
128    El método actualizarElectrodomestico solicita un String para una variable modelo, se la
   pasa al método
129    buscarElectrodomestico para recuperar el objeto que tenga el mismo campo modelo,
130    una vez tiene el objeto electrodomestico que necesitamos, invoca con dicha instancia al
   método
131    datosElectrodomestico para que se introduzca de nuevo todos los datos para esa
   instancia.
132    Finalmente, la guarda en el mapa electrodomesticos.
133     */
134    public void actualizarElectrodomestico(){
135        String modelo;
136        System.out.println("Introduzca el MODELO de electrodomestico para actualizar sus
   datos: ");
137        modelo = Utiles.sc.nextLine();
138        Electrodomestico electrodomestico = buscarElectrodomestico(modelo);
139        if (electrodomestico!=null){
140            electrodomestico = electrodomestico.datosElectrodomestico(false,
   electrodomestico.getModelo());
141            electrodomesticos.put(electrodomestico.getModelo(), electrodomestico);
142        } else{
143            System.out.println("No existe dicho MODELO.");
144        }
145    }
146
147
148    /*
149    El método buscarElectrodomestico recibe una variable modelo y devuelve la instancia
   electrodomestico
150    que contenga el mismo campo modelo.
151     */
152    public Electrodomestico buscarElectrodomestico(String modelo){
153        return electrodomesticos.get(modelo);
154    }
155
156
157    /*
158    El método buscarElectrodomesticoPantalla solicita que se introduzca un String para una
   variable modelo.
159    Se crea un objeto electrodomestico al que se asigna una instancia que recuperamos con
160    buscarElectrodomestico pasándole modelo.
161    Si existe, finalmente, imprime todos los datos de dicho electrodomestico.
162     */
163    public void buscarElectrodomesticoPantalla(){
164        String modelo;
165        System.out.println("Buscar Electrodomestico por MODELO: ");
166        modelo = Utiles.sc.nextLine();
167        Electrodomestico electrodomestico = buscarElectrodomestico(modelo);
168        if (electrodomestico!=null){
169            System.out.println(electrodomestico.toString());
```

```
170         } else{
171             System.out.println("No existe dicho modelo.");
172         }
173     }
174 
175 
176     /*
177     El método listarElectrodomesticos recorre el mapa electrodomesticos mediante un bucle for extendido o for each,
178     e imprime los datos que contiene todo objeto o instancia electrodomestico que tenemos almacenada en el mapa.
179     Con este método visualizaremos, por ejemplo, el inventario de electrodomésticos que tenemos en tienda.
180      */
181     public void listarElectrodomesticos(){
182         for (Electrodomestico electro : electrodomesticos.values()) {
183             System.out.println(electro.toStringInventario());
184         }
185     }
186 
187 
188     // Métodos para gestionar CLIENTES.
189 
190     /*
191     El método crearCliente crea una nueva instancia de Cliente, le asigna los datos de la instancia
192     que devuelve el método datosCliente y por último llama al método guardarCliente,
193     pasándole como parámetro el objeto Cliente creado.
194      */
195     public void crearCliente(){
196 
197         Cliente cliente = new Cliente();
198         cliente = cliente.datosCliente();
199         guardarCliente(cliente);
200     }
201 
202 
203     /*
204     El método guardarCliente, mediante el campo dni, comprueba si ya existe un objeto igual en el mapa clientes,
205     si no es así, incluye en el mapa el nuevo objeto recibido como parámetro.
206      */
207     public void guardarCliente(Cliente cliente){
208         Cliente cl = clientes.get(cliente.getDni());
209         if (cl != null) {
210             System.out.println("No se puede introducir el Cliente. Ya esta registrado.");
211         } else {
212             clientes.put(cliente.getDni(), cliente);
213         }
214     }
215 
216 
217     /*
218     El método actualizarCliente solicita un String para una variable dni, se la pasa al método buscarCliente
219     para recuperar el objeto que tenga el mismo campo dni, una vez tiene el objeto cliente que necesitamos,
220     invoca con dicha instancia al método datosCliente para que se introduzca de nuevo todos los datos
221     para esa instancia. Finalmente, la guarda en el mapa clientes.
222      */
223     public void actualizarCliente(){
```

```
224         String dni;
225         System.out.println("Introduzca el DNI del cliente para actualizar sus datos: ");
226         dni = Utiles.sc.nextLine();
227         Cliente cliente = buscarCliente(dni);
228         if (cliente!=null){
229             cliente = cliente.datosCliente();
230             clientes.put(cliente.getDni(), cliente);
231         } else{
232             System.out.println("Cliente no existente.");
233         }
234     }
235 
236 
237     /*
238     El método buscarCliente recibe una variable dni y devuelve la instancia cliente que
    contengo el mismo campo dni.
239      */
240     public Cliente buscarCliente(String dni){
241         return clientes.get(dni);
242     }
243 
244 
245     /*
246     El método buscarClientePantalla solicita que se introduzca un String para una variable
    dni.
247     Se crea un objeto cliente al que se asigna una instancia que recuperamos con
    buscarCliente pasándole dni.
248     Si existe, finalmente, imprime todos los datos de dicho cliente.
249      */
250     public void buscarClientePantalla(){
251         String dni;
252         System.out.println("Buscar Cliente por DNI: ");
253         dni = Utiles.sc.nextLine();
254         Cliente cliente = buscarCliente(dni);
255         if (cliente!=null){
256             System.out.println(cliente.toString());
257         } else{
258             System.out.println("Cliente no existente.");
259         }
260     }
261 
262 
263     // Métodos para gestionar PROMOCIONES.
264 
265     /*
266     El método buscarPromo devuelve una instancia de Promo en la que coincida el campo
    nombre
267     que se ha pasado al método como parámetro.
268      */
269     public Promo buscarPromo(String nombre){
270         return promos.get(nombre);
271     }
272 
273 
274     /*
275     El método noOfertada, recibe por parámetro un cliente y una promo, comprueba, mediante
    el campo nombre,
276     si dicho cliente tiene una instancia con los mismos datos en su lista promos, si es
    así,
277     devuelve dicha instancia, si no retorna null.
278      */
279     private Promo noOfertada(Cliente cl, Promo promo){
```

```
280         if (cl.getPromos()==null){
281             return null;
282         }
283         for (Promo promoCli: cl.getPromos()){
284             if (promoCli.getNombre().equals(promo.getNombre())){
285                  return promoCli;
286             }
287         }
288         return null;
289     }
290 
291 
292     /*
293     El método de tipo Boolean haPasadounAnno recibe una instancia promo, mediante su campo fechaPresentada
294     comprueba si ha sido presentada, esta instrucción devuelve true en caso de que fechaPresentada contenga null
295     o una fecha que no alcance al año respecto a la fecha actual.
296     Si no cumple ninguna de las dos condiciones retorna false.
297      */
298     private Boolean haPasadounAnno(Promo yaOfertada){
299         Calendar haceUnAnno = Calendar.getInstance();
300         haceUnAnno.add(Calendar.YEAR, -1);
301         if (yaOfertada.getFechaPresentada() == null || yaOfertada.getFechaPresentada().before(haceUnAnno)){
302             return true;
303         }
304         return false;
305     }
306 
307 
308     /*
309     El objetivo del método enviarPromo es incluir una instancia de la promo que se quiera en un momento dado,
310     en la lista promos que tiene cada objeto cliente.
311     Primero solicita un String para la variable nombrePro, crea una instancia promo con los datos de la instancia
312     que devuelve buscarPromo tras la búsqueda que ha realizado con nombrePro.
313     Con un bucle for each recorre el mapa clientes, comprueba en cada cliente, mediante los métodos
314     noOfertada y haPasadounAnno, si contiene una instancia de la promo que tenemos en su correspondiente
315     lista promos, si no es así, la incluye.
316      */
317     public void enviarPromo() {
318 
319         Boolean enviada = false;
320         System.out.println("Introduce el nombre de la PROMOCION a enviar: ");
321         String nombrePro = Utiles.sc.nextLine();
322         Promo promo = buscarPromo(nombrePro);
323         if (promo!=null){
324             for (Cliente cl : clientes.values()) {
325                 Promo yaOfertada = noOfertada(cl, promo);
326                 if (yaOfertada == null){
327                     if (cl.getPromos()==null){
328                         cl.setPromos(new ArrayList<Promo>());
329                     }
330                     promo.setFechaPresentada(Calendar.getInstance());
331                     cl.getPromos().add(promo);
332                     enviada = true;
333                 } else{
334                     if (haPasadounAnno(yaOfertada)){
```

```
335                         promo.setFechaPresentada(Calendar.getInstance());
336                         cl.getPromos().add(promo);
337                         enviada = true;
338                     }
339                 }
340             }
341             if (enviada){
342                 System.out.println("\nPromocion enviada correctamente.");
343             } else {
344                 System.out.println("\nOferta ya presentada hace menos de un año a todos los
    clientes.");
345             }
346         } else{
347             System.out.println("\nNo existe Promocion.");
348         }
349     }
350 
351 
352     /*
353     El método promosCliente solicita un String para una variable dni. Localiza al cliente
    que tiene
354     el mismo campo dni. Imprime su nombre, apellidos y las promos que tiene en su lista.
355     Este método nos valdría para enviarle un correo electrónico al cliente con sus datos y
    promociones,
356     tal como indica el enunciado.
357      */
358     public void promosCliente() {
359         System.out.println("Introduzca el DNI del cliente para enviarle sus PROMOCIONES:
    ");
360         String dni = Utiles.sc.nextLine();
361         Cliente cl = clientes.get(dni);
362         if (cl != null){
363             System.out.println("\nEstimado/a cliente: " + cl.getNombre() + " "+
    cl.getApellidos() +
364                                ", tiene a su disposicion las siguientes PROMOCIONES:");
365             for(Promo promo: cl.getPromos()){
366                 System.out.println(promo.toString());
367             }
368         }
369     }
370 }
```