PAC 3. UF5.

Estructuras de almacenamiento.

Aníbal Santos Gómez

Para trabajar con las estructuras de almacenamiento vamos a trabajar con los datos de la clínica VetIlerna:

En cliente añadiremos un ArrayList de la clase Mascota donde iniciamos el ArrayList sin introducir datos ni recibir parámetros. Crearemos un método para añadir mascotas.

```java
import java.util.ArrayList;

public class Cliente {
    private String nombre;
    private String dni;
    private ArrayList<Mascota> mascotas;

    Cliente (String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
        this.mascotas = new ArrayList<>();
    }

    Cliente () {

        this.mascotas = new ArrayList<>();
    }

    public void anadirMascota(Mascota mascota) {
        this.mascotas.add(mascota);
    }

    public String getNombre () {
        return this.nombre;
    }

    public String getDni () {
        return this.dni;
    }

    //metodo para añadir mascotas

    public ArrayList<Mascota> getMascotas () {
        return this.mascotas;
    }

    public void setNombre (String nombre) {
        this.nombre = nombre;
    }
```

```java
    public void setMascota (ArrayList<Mascota> mascota) {
        this.mascotas = mascota;
    }

    public String toString () {
        String cadena = this.nombre + " con DNI " + this.dni + " tiene como mascotas: ";

        /*for(Mascota m : mascotas){
            cadena+=m.getNombre();

        } */

        for (int i = 0; i < this.mascotas.size(); i++) {
            cadena += mascotas.get(i).getNombre();
        }
        return cadena;
    }
}
```

En la mascota añadiremos un HashMap para almacenar tratamientos (fecha y tratamiento) String. En los constructores iniciamos el HashMap sin datos. Además crearemos un método para añadir mascotas al HashMap.

```java
import java.util.HashMap;

public class Mascota {
    private String nombre;
    private static int codigo = 0; //valor inicial, estático para que cada vez que creemos mascota nueva, y recuperamos el ultimo valor que teniamos
    private Genero genero;

    //Hashmap (tiene clave: fecha ,,,,, y valor: tratamiento)

    private HashMap<String, String> tratamientos;

    //cada vez que creemos Mascota vacio o no vacio suma un codigo (por la variable estática

    Mascota () {

        this.codigo++;
        this.tratamientos = new HashMap<String,String>(); //inicializamos el hashmap
    }
```

```java
    Mascota (String nombre, Genero genero) {

        this.nombre = nombre;
        this.codigo++;
        this.genero = genero;
        this.tratamientos = new HashMap<String, String>();
    }

    //añadimos tratamiento al hashmap, recibimos fecha y tratamiento

    public void addTratamiento(String fecha, String tratamiento) {
        tratamientos.put(fecha,  tratamiento); //como trabajamos con mapas no utilizamos add sino put
    }
```

```java
    public String getNombre () {
        return this.nombre;
    }

    public int getCodigo () {
        return this.codigo;
    }

    public Genero getGenero () {
        return this.genero;
    }

    public HashMap<String, String> getTratamientos () {
        return this.tratamientos;
    }

    public void setNombre (String nombre) {
        this.nombre = nombre;
    }

    public void setGenero (Genero genero) {
        this.genero = genero;
    }

    public void setTratamientos (HashMap<String, String> tratamientos) {
        this.tratamientos = tratamientos;
    }

    public String toString () {
        String cadenaTratamientos = "";

        for (String fecha: tratamientos.keySet()) { //el keyset nos devuelve la coleccion de claves (al ser coleccion podemos recorrerla)
            cadenaTratamientos.concat(fecha + " : " + tratamientos.get(fecha) + "\n");
        }


        return this.nombre + " con el codigo " + this.codigo + " es " + this.genero + ". Ha recibido los siguientes tratamientos: " + tratamientos;
    }
}
```

En esta parte hemos utilizado código como un valor estático iniciándolo a cero para que cada vez que creemos una nueva mascota recuperemos el último valor que teniamos.

El HashMap tiene como clave la fecha, y como valor el tratamiento.

Para añadir el tratamiento al HashMap recibimos fecha y tratamiento como String. Como estamos trabajando con mapas en el método no utilizaremos .add sino .put.

En Registro cuando se pulse el botón de insertar crearemos un objeto de la clase Mascota con sus datos, además de un objeto de la clase Cliente con sus datos y se añadirá la mascota al cliente.

```
        insertar = new JButton("Insertar");
        insertar.setBounds(150, 500, 100, 20);
        insertar.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent arg0) {
                // TODO Auto-generated method stub

                //Creamos cliente

                Cliente cliente = new Cliente(nombreC.getText(), dni.getText());


                Genero genero;
                if(macho.isSelected()) {
                    genero= Genero.macho;
                }
                else {
                    genero= Genero.hembra;
                }

                //declaramos variable mascota, para hacer la creación más genérica

                Mascota mascota = null;

                if(perro.isSelected()) {
                    mascota = new Perro(nombreM.getText(), genero, raza.getText());
                }
                else if(gato.isSelected()) {
                    mascota = new Gato(nombreM.getText(), genero, color.getText());
                }
                else if(roedor.isSelected()) {
                    mascota = new Roedor(nombreM.getText(), genero, tipo.getText());
                }

                //se añade mascota al cliente

                cliente.anadirMascota(mascota);
                clientes.add(cliente);

                //mostramos info por consola

                System.out.println(cliente.toString());
            }

        });
```

En añadir mascota cuando pulsemos buscar, se buscará el DNI y añadiremos en el JTextField, y cuando pulsemos el botón insertar se añadirá la mascota en el DNI, si no existe mostraremos mensaje por consola.

```
insertar = new JButton("Insertar");
insertar.setBounds(150, 500, 100, 20);
insertar.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        // TODO Auto-generated method stub

        Mascota mascota = null;

        for(Cliente c: clientes) {
            if(c.getDni().equalsIgnoreCase(dni.getText())) {

                Genero genero;
                if(macho.isSelected()) {
                    genero= Genero.macho;
                }
                else {
                    genero= Genero.hembra;
                }

                if(perro.isSelected()) {
                    mascota = new Perro(nombreM.getText(), genero, raza.getText());
                }
                else if(gato.isSelected()) {
                    mascota = new Gato(nombreM.getText(), genero, color.getText());
                }
                else if(roedor.isSelected()) {
                    mascota = new Roedor(nombreM.getText(), genero, tipo.getText());
                }

                c.anadirMascota(mascota);
                System.out.println(c.getDni() + " " + c.getNombre() + " se le ha añadido la mascota " + mascota);

                break; //añadimos un break para que cuando lo encuentre no siga recorriendo
            }
        }
```

Acabando el for como vemos en la siguiente imagen, y sin encontrar mascota sacaremos el mensaje de error pertinente:

```
        //si termina el for y no hemos encontrado mascota sacamos mensaje de error

        if(mascota==null) {

            //error no ha encontrado al cliente

            System.out.println("ERROR: NO SE HA ENCONTRADO AL CLIENTE");
        }
    }

});
```

En tratamientos cuando pulsemos el botón buscar se buscará el DNI en el ArrayList de clientes poniéndolo si lo encuentra en el JComboBox. Y cuando se pulse añadir se añadirá en el HashMap de tratamientos de la mascota seleccionada y mostraremos por consola.

```
buscarC.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        // TODO Auto-generated method stub

        for(Cliente c : clientes) { //paraca cada cliente comprueba dni, si son iguales (sin mayus y minus), cogemos el nombre del cliente
            if(c.getDni().equalsIgnoreCase(dni.getText())) {
                nombreC.setText(c.getNombre()); //


                //jComboBox
                //recuperamos las mascotas desde cliente
                for(Mascota m : c.getMascotas()) {
                    mascotas.addItem(m.getNombre()); //añadimos item, cadena del nombre.
                }

                break; //añadimos un break para que cuando lo encuentre no siga recorriendo


            }
        }
```

```
anadir = new JButton("Anadir");
anadir.setBounds(700, 150, 100, 20);
anadir.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) { //
        // TODO Auto-generated method stub
        boolean chivato = false;
        for(Cliente c: clientes) {
            if(c.getDni().equalsIgnoreCase(dni.getText())){
                for(Mascota m : c.getMascotas()) {
                    if(m.getNombre().equalsIgnoreCase( (String) mascotas.getSelectedItem())) { //casteamos el objeto del combobox para que
                        //despues se compara con el nombre de la mascota

                        m.addTratamiento(fecha.getText(), tratamientoM.getText());

                        //Mostramos por consola:

                        System.out.println(m.getCodigo() + " " + m.getNombre() + " : " + fecha.getText() + " " + tratamientoM.getText());

                        chivato = true;
                        break;

                    }
                    if(chivato==true) {
                        break;
                    }
                }
            }
        }
```