UNIVERSIDAD DE LAS AMERICAS

FACULTAD DE INGENIERA Y CIENCIAS APLICADAS

Taller diseño avanzado de clases

Autor:

Alejandro Taipicaña

Docente:

Bernarda Sandoval

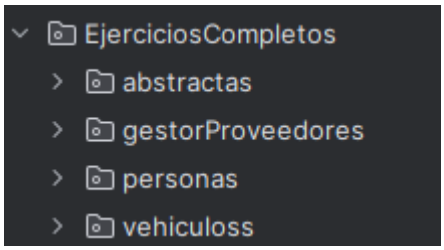
Link:

https://github.com/AlejAnDroXK/EjerciciosCompletos

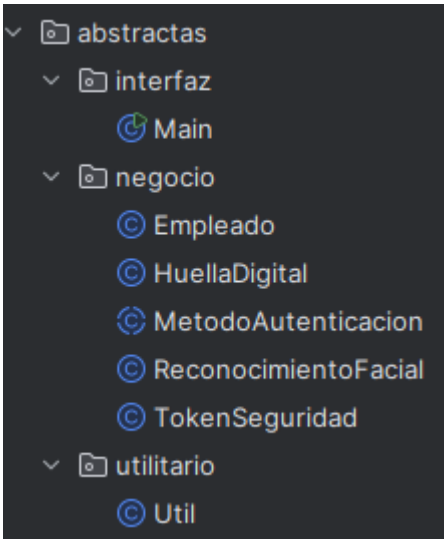
2025

Índice

1. **Vista General de los ejercicios**



2. **Ejercicio Abstractas**



2.1 Interfaz

Main

```
1  package EjerciciosCompletos.abstractas.interfaz;
2
3  import EjerciciosCompletos.abstractas.negocio.Empleado;
4  import EjerciciosCompletos.abstractas.negocio.MetodoAutenticacion;
5  import EjerciciosCompletos.abstractas.utilitario.Util;
6
7  import java.util.ArrayList;
8  import java.util.List;
9  import java.util.Scanner;
10
11 public class Main {
12     public static void main(String[] args) {
13         Util u = new Util();
14         Scanner sc = new Scanner(System.in);
15         String cedula, nombre, patronhuella,token,patronRostro;
16         int opc=0,nivelseguridad;
17         do {
18             u.menu();
19             opc = Integer.parseInt(sc.nextLine());
20             switch (opc){
21                 case 1: {//Agregar Empleado
22                     System.out.println("----------------");
```

```
                System.out.println("Ingrese cedula: ");
                cedula= sc.nextLine();
                System.out.println("Ingrese su nombre: ");
                nombre = sc.nextLine();
                u.agregarEmpleado(cedula, nombre);
                pausa();
            }break;
            case 2: {//Agregar token seguridad
                System.out.println("----------------");
                System.out.println("Ingrese su cedula: ");
                cedula= sc.nextLine();
                if(u.buscarEmpleado(cedula) != -1){
                    System.out.println("Ingrese token: ");
                    token= sc.nextLine();
                    System.out.println("Ingrese nivel de seguridad: ");
                    nivelseguridad=Integer.parseInt(sc.nextLine());
                    u.agregarAutenticacionToken(cedula,token,nivelseguridad);
                }else {
                    System.out.println("Empleado No Existe");
                }
                pausa();
            }break;
            case 3:{//Agregar reconocimiento facial
                System.out.println("----------------");
                System.out.println("Ingrese su cedula: ");
                cedula= sc.nextLine();
                if(u.buscarEmpleado(cedula) != -1){
                    System.out.println("Ingrese rostro: ");
                    patronRostro= sc.nextLine();
                    System.out.println("Ingrese nivel de seguridad: ");
                    nivelseguridad=Integer.parseInt(sc.nextLine());
                    u.agregarAutenticacionRostro(cedula,patronRostro,nivelseguridad);
                }else {
                    System.out.println("Empleado No Existe");
                }
                pausa();
            }break;
            case 4:{//Agregar huella digital
                System.out.println("----------------");
                System.out.println("Ingrese su cedula: ");
                cedula= sc.nextLine();
                if(u.buscarEmpleado(cedula) != -1){
                    System.out.println("Ingrese huella: ");
                    patronhuella= sc.nextLine();
                    System.out.println("Ingrese nivel de seguridad: ");
                    nivelseguridad=Integer.parseInt(sc.nextLine());
                    u.agregarAutenticacionHuella(cedula,patronhuella,nivelseguridad);
                }else {
                    System.out.println("Empleado No Existe");
                }
                pausa();
            }break;
            case 5: {//Mostrar datos de empleados (todos
                System.out.println("----------------");
                u.listarEmpleados();
                pausa();
```

```
79              }break;
80              case 6: {//Buscar empleado y mostrar datos
81                  System.out.println("-----------------");
82                  System.out.println("Ingrese cedula del empleado: ");
83                  cedula = sc.nextLine();
84                  if (u.buscarEmpleado(cedula) != -1) {
85                      Empleado e = u.getEmpleado(cedula);
86                      System.out.println("Empleado encontrado:");
87                      System.out.println("Nombre: " + e.getNombre());
88                      System.out.println("Cédula: " + e.getCedula());
89                      System.out.println("Métodos de autenticación: ");
90                      for (MetodoAutenticacion ma : e.getAutenticaciones()) {
91                          System.out.println(ma);
92                      }
93                  } else {
94                      System.out.println("Empleado no encontrado");
95                  }
96                  pausa();
97              }break;
98              case 7: {//Total métodos autenticación empleado
99                  System.out.println("-----------------");
100                 System.out.println("Ingrese cedula del empleado: ");
101                 cedula = sc.nextLine();
102                 if (u.buscarEmpleado(cedula) != -1) {
103                     Empleado e = u.getEmpleado(cedula);
104                     System.out.println("Métodos de autenticación: " + e.getAutenticaciones().size());
105                 } else {
106                     System.out.println("Empleado no encontrado");
107                 }
108                 pausa();
109             }break;
110             case 8: {//Total métodos huella empleado
111                 System.out.println("-----------------");
112                 System.out.println("Ingrese cedula del empleado: ");
113                 cedula = sc.nextLine();
114                 if (u.buscarEmpleado(cedula) != -1) {
115                     int cantidadHuella = u.cantidadAutenticacionHuella(cedula);
116                     System.out.println("Cantidad de métodos de huella digital registrados: " + cantidadHuella);
117                 } else {
118                     System.out.println("Empleado no encontrado");
119                 }
120                 pausa();
121             } break;
122             case 9:{//Total métodos token empleado
123                 System.out.println("-----------------");
124                 System.out.println("Ingrese cedula del empleado: ");
125                 cedula = sc.nextLine();
126                 if (u.buscarEmpleado(cedula) != -1) {
127                     int cantidadToken = u.cantidadAutenticacionToken(cedula);
128                     System.out.println("Cantidad de métodos de token registrados: " + cantidadToken);
129                 } else {
130                     System.out.println("Empleado no encontrado");
131                 }
132                 pausa();
133             }break;
134             case 10:{//Total métodos facial empleado
```

```
                System.out.println("-----------------");
                System.out.println("Ingrese cedula del empleado: ");
                cedula = sc.nextLine();
                if (u.buscarEmpleado(cedula) != -1) {
                    int cantidadRostro = u.cantidadAutenticacionRostro(cedula);
                    System.out.println("Cantidad de métodos de reconocimiento facial registrados: " + cantidadRostro);
                } else {
                    System.out.println("Empleado no encontrado");
                }
                pausa();
            }break;
            case 11:{//Métodos mayor a umbral de empleado
                System.out.println("-----------------");
                System.out.println("Ingrese cedula del empleado: ");
                cedula = sc.nextLine();
                System.out.println("Ingrese el umbral de seguridad: ");
                int umbral = Integer.parseInt(sc.nextLine());
                if (u.buscarEmpleado(cedula) != -1) {
                    String autenticaciones = u.autenticaciionMayorSeguridad(cedula, umbral);
                    System.out.println("Métodos de autenticación mayores al umbral de seguridad: ");
                    System.out.println(autenticaciones);
                } else {
                    System.out.println("Empleado no encontrado");
                }
                pausa();
            }break;
            case 12:{//Autenticar empleado
                System.out.println("-----------------");
                System.out.println("Ingrese cedula del empleado: ");
                cedula = sc.nextLine();
                System.out.println("Ingrese el tipo de autenticación (huella, rostro, token): ");
                String tipo = sc.nextLine();
                System.out.println("Ingrese el dato a autenticar: ");
                String dato = sc.nextLine();
                String result = u.autenticarUtil(dato, tipo, cedula);
                System.out.println(result);
                pausa();
            }break;
            case 13: {
                System.out.println("-----------------");
                System.out.println("Saliendo del sistema...");
                System.exit(0);
            } break;
            default: {
                System.out.println("Opcion Invalida. Ingrese Nuevamente");
            }
          }
        } while (opc != 13);
    }

    public static void pausa() {
        System.out.println("___________Presione enter para continuar___________");
        Scanner sc = new Scanner(System.in);
        sc.nextLine();
    }
}
```

2.2 Negocio

Empleado

```java
package EjerciciosCompletos.abstractas.negocio;
//siempre metodo agregar y algo para cpnsultar la lista
import java.util.ArrayList;
import java.util.List;

public class Empleado {
    private String cedula;
    private String nombre;
    private List<MetodoAutenticacion> autenticaciones;

    public Empleado(String cedula, String nombre) {
        this.cedula = cedula;
        this.nombre = nombre;
        autenticaciones = new ArrayList<>();
    }

    public List<MetodoAutenticacion> getAutenticaciones() {return autenticaciones;}
    public String getCedula() {return cedula;}
    public String getNombre() {return nombre;}
    public void setNombre(String nombre) {this.nombre = nombre;}

    public void adicionarAutenticacion(MetodoAutenticacion me){
        autenticaciones.add(me);
    }

    public int cantidadRegistroHuella(){
        int cont=0;
        for(MetodoAutenticacion me: autenticaciones){
            if (me instanceof HuellaDigital){
                cont++;
            }
        }
        return cont;
    }
    public int cantidadRegistroRostro(){
        int cont=0;
        for(MetodoAutenticacion me: autenticaciones){
            if (me instanceof ReconocimientoFacial){
                cont++;
            }
        }
        return cont;
    }
    public int cantidadRegistroToken(){
        int cont=0;
        for(MetodoAutenticacion me: autenticaciones){
            if (me instanceof TokenSeguridad){
                cont++;
            }
        }
        return cont;
    }
```

```
53
54    public boolean autenticar (String dato, String tipo){
55        for(MetodoAutenticacion me: autenticaciones){
56            if(me.getTipo().equalsIgnoreCase(tipo)){
57                if(me.autenticar(dato))
58                    return true;
59            }
60        }
61        return false;
62    }
63
64    public String seguridadMayorUmbral(int valor){
65        String seguridad="";
66        for(MetodoAutenticacion me: autenticaciones){
67            if(me.getNivelseguridad() >= valor){
68                seguridad += me.getTipo();
69                seguridad += "\n";
70            }
71        }
72        return seguridad;
73    }
74 }
```

HuellaDigital

```
1  package EjerciciosCompletos.abstractas.negocio;
2
3  public class HuellaDigital extends MetodoAutenticacion{
4      private  String patronHuella;
5
6      public HuellaDigital(int nivelseguridad, String patronHuella) {
7          super(nivelseguridad, "huella");
8          this.patronHuella = patronHuella;
9      }
10
11     public String getPatronHuella() {return patronHuella;}
12     public void setPatronHuella(String patronHuella) {this.patronHuella = patronHuella;}
13
14     @Override
15     public String toString(){
16         return super.toString()+",  Patrón huella: " + patronHuella;
17     }
18     @Override
19     public boolean autenticar(String dato){
20         if (patronHuella.contains(dato))
21             return true;
22         return false;
23     }
24
25 }
```

MetodoAutenticacion

```java
package EjerciciosCompletos.abstractas.negocio;

public abstract class MetodoAutenticacion {
    private int nivelseguridad;
    private String tipo;

    public MetodoAutenticacion(int nivelseguridad, String tipo) {
        this.nivelseguridad = nivelseguridad;
        this.tipo = tipo;
    }

    public int getNivelseguridad() {return nivelseguridad;}
    public String getTipo() {return tipo;}
    public void setNivelseguridad(int nivelseguridad) {this.nivelseguridad = nivelseguridad;}
    public void setTipo(String tipo) {this.tipo = tipo;}

    @Override
    public String toString() {
        return "Tipo: "+ tipo + "\nNivel de seguridad=" + nivelseguridad;
    }


    public abstract boolean autenticar(String dato);
}
```

ReconocimientoFacial

```java
package EjerciciosCompletos.abstractas.negocio;

public class ReconocimientoFacial extends MetodoAutenticacion{
    private String patronRostro;

    public ReconocimientoFacial(int nivelseguridad, String patronRostro) {
        super(nivelseguridad, "rostro");
        this.patronRostro = patronRostro;
    }

    public String getPatronRostro() {return patronRostro;}
    public void setPatronRostro(String patronRostro) {this.patronRostro = patronRostro;}

    @Override
    public String toString(){
        return super.toString()+", Patrón rostro: " + patronRostro;
    }
    @Override
    public boolean autenticar(String dato){
        if (patronRostro.contains(dato))
            return true;
        return false;
    }
}
```

TokenSeguridad

```java
package EjerciciosCompletos.abstractas.negocio;

public class TokenSeguridad extends MetodoAutenticacion{
    private String token;

    public TokenSeguridad(int nivelseguridad, String token) {
        super(nivelseguridad, "token");
        this.token = token;
    }

    public void setToken(String token) {this.token = token;}

    @Override
    public String toString(){
        return super.toString()+", Patrón token: "+token;
    }
    @Override
    public boolean autenticar(String dato){
        if (token.equals(dato))
            return true;
        return false;
    }
}
```

2.3 Utilitario

Util

```java
package EjerciciosCompletos.abstractas.utilitario;

import EjerciciosCompletos.abstractas.negocio.*;

import java.util.ArrayList;
import java.util.List;

public class Util {
    private List<Empleado> empleados;
    public Util(){
        empleados= new ArrayList<>();
    }
    public List<Empleado> getEmpleados() {
        return empleados;
    }

    public void agregarEmpleado(String cedula, String nombre){
        int indice=buscarEmpleado(cedula);
        if(indice==-1){
            empleados.add(new Empleado(cedula,nombre));
        }else{
            System.out.println("Empleado ya existe");
        }
    }

    public int buscarEmpleado(String cedula){
```

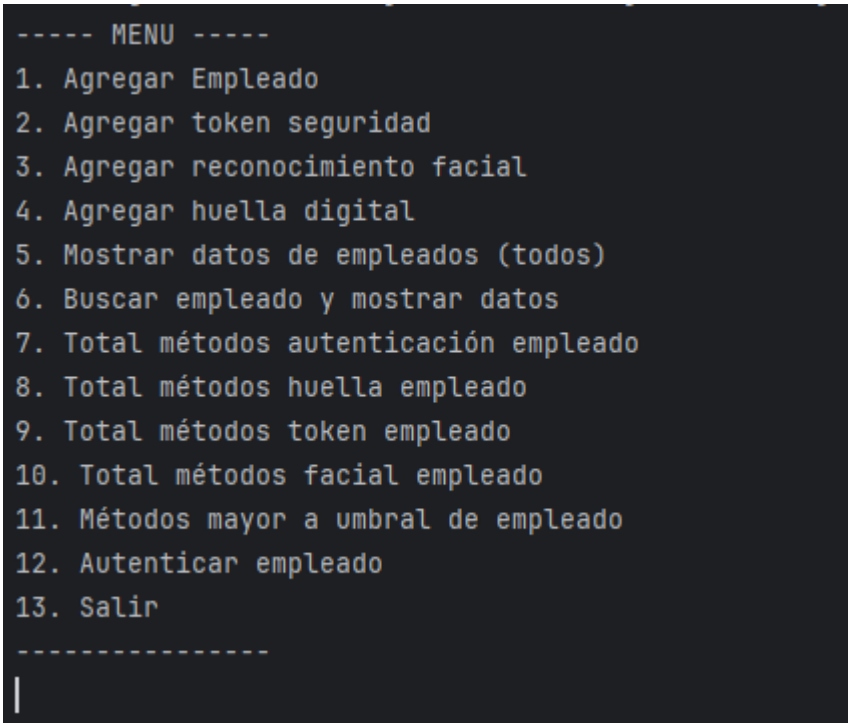
```
27        for(int i=0; i <empleados.size();i++){
28            if(empleados.get(i).getCedula().equals(cedula)){
29                return i;
30            }
31        }
32        return -1;
33    }
34
35    public Empleado getEmpleado(String cedula) {
36        int indice = buscarEmpleado(cedula);
37        if (indice != -1) {
38            return empleados.get(indice);
39        } else {
40            return null;
41        }
42    }
43
44    public void agregarAutenticacionRostro(String cedula, String patronRostro, int nivelSeguridad){
45        int indice =buscarEmpleado(cedula);
46        if(indice != -1){
47            if(cantidadAutenticacionRostro(cedula)==0){
48                empleados.get(indice).adicionarAutenticacion(new ReconocimientoFacial(nivelSeguridad,patronRostro));
49            }else{
50                System.out.println("Ya tiene un rostro");
51            }
52        }else{
53            System.out.println("Empleado no existe");
54        }
55    }
56    public void agregarAutenticacionToken(String cedula, String token, int nivelSeguridad) {
57        int indice = buscarEmpleado(cedula);
58        if (indice != -1) {
59            empleados.get(indice).adicionarAutenticacion(new TokenSeguridad(nivelSeguridad,token));
60        } else {
61            System.out.println("Empleado no existe");
62        }
63    }
64    public void agregarAutenticacionHuella(String cedula, String huella, int nivelSeguridad){
65        int indice =buscarEmpleado(cedula);
66        if(indice != -1){
67            empleados.get(indice).adicionarAutenticacion(new HuellaDigital(nivelSeguridad,huella));
68        }else{
69            System.out.println("Empleado no existe");
70        }
71    }
72
73    public int cantidadAutenticacionHuella(String cedula){
74        int indice =buscarEmpleado(cedula);
75        if(indice != -1){
76            return empleados.get(indice).cantidadRegistroHuella();
77        }else{
78            System.out.println("Empleado no existe");
79            return -1;
80        }
81    }
82    public int cantidadAutenticacionRostro(String cedula){
```

```
        int indice =buscarEmpleado(cedula);
        if(indice != -1){
            return empleados.get(indice).cantidadRegistroRostro();
        }else{
            System.out.println("Empleado no existe");
            return -1;
        }
    }
    public int cantidadAutenticacionToken(String cedula){
        int indice =buscarEmpleado(cedula);
        if(indice != -1){
            return empleados.get(indice).cantidadRegistroToken();
        }else{
            System.out.println("Empleado no existe");
            return -1;
        }
    }

    public String autenticaciionMayorSeguridad(String cedula, int nivel){
        int indice =buscarEmpleado(cedula);
        if(indice != -1){
            return empleados.get(indice).seguridadMayorUmbral(nivel);
        }else{
            System.out.println("Empleado no existe");
            return null;
        }
    }
    public String autenticarUtil (String dato,String tipo,String cedula){
        int indice=buscarEmpleado(cedula);
        if(indice!=-1){
            Empleado empleado = empleados.get(indice);
            boolean autenticado = empleado.autenticar(dato, tipo);
            if (autenticado) {
                return "Acceso concedido";
            } else {
                return "Acceso denegado";
            }
        } else {
            return "Empleado no encontrado";
        }
    }
    public void listarEmpleados() {
        System.out.println("-----  EMPLEADOS  -----");
        for (Empleado e : empleados) {
            System.out.println("Nombre: " + e.getNombre() + ", Cédula: " + e.getCedula());
        }
    }

    public void menu(){
        System.out.println("-----  MENU -----");
        System.out.println("1. Agregar  Empleado");
        System.out.println("2. Agregar  token seguridad");
        System.out.println("3. Agregar  reconocimiento facial");
        System.out.println("4. Agregar  huella digital");
        System.out.println("5. Mostrar datos de empleados (todos)");
        System.out.println("6. Buscar empleado y mostrar datos");
```

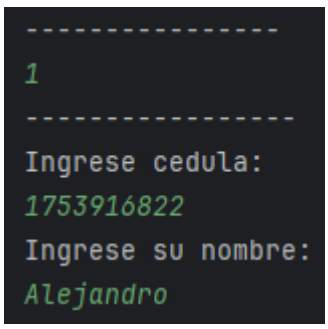
```
139         System.out.println("7. Total métodos autenticación empleado");
140         System.out.println("8. Total métodos huella empleado");
141         System.out.println("9. Total métodos token empleado");
142         System.out.println("10. Total métodos facial empleado");
143         System.out.println("11. Métodos mayor a umbral de empleado");
144         System.out.println("12. Autenticar empleado");
145         System.out.println("13. Salir");
146         System.out.println("----------------");
147     }
148 }
```

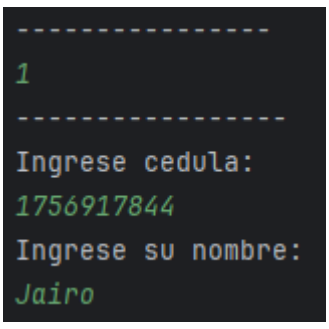
2.4 Pruebas funcionamiento ejercicio Abstractas

Menú

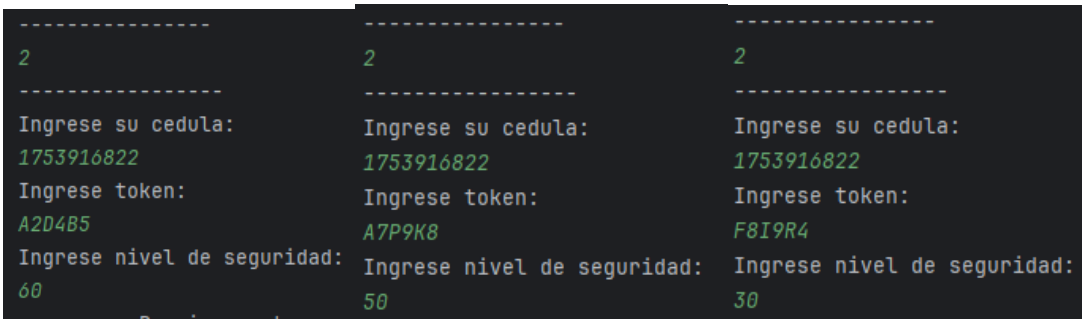


1. Agregar Empleado

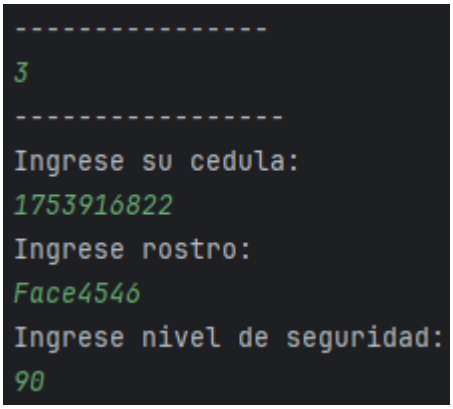


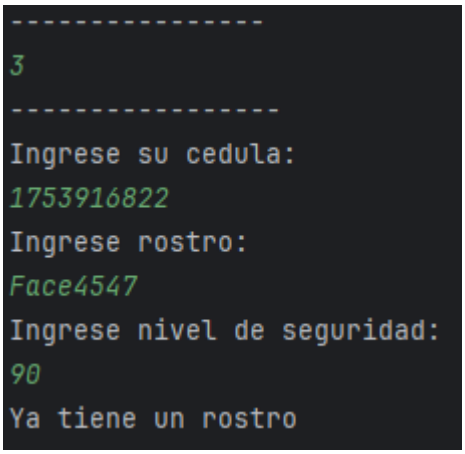


2. Agregar token seguridad

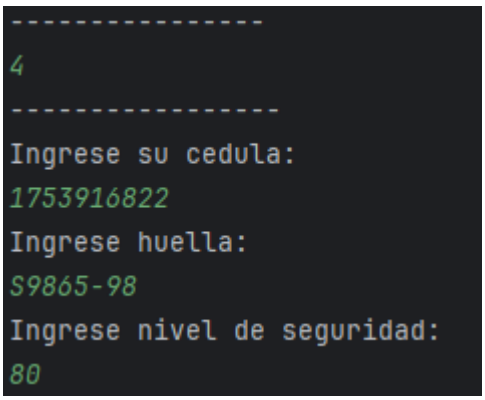


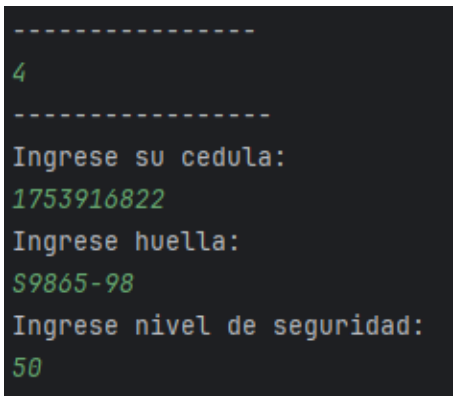
3. Agregar reconocimiento facial



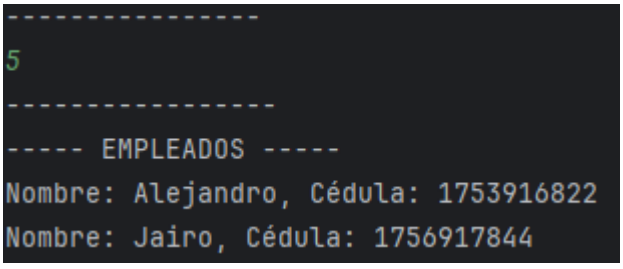


4. Agregar huella digital

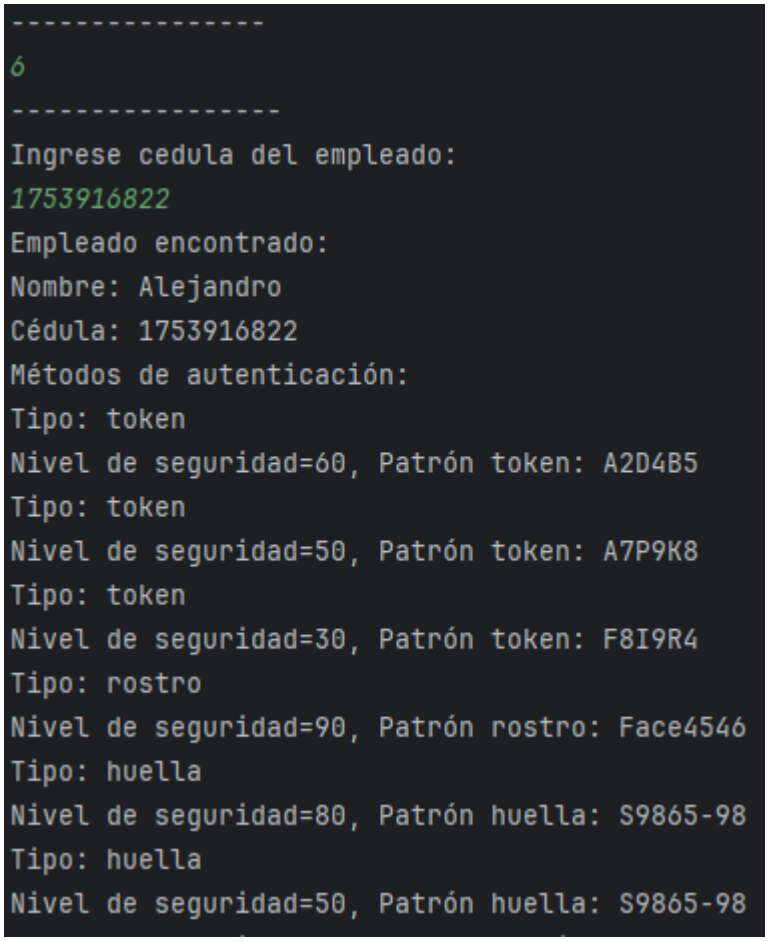




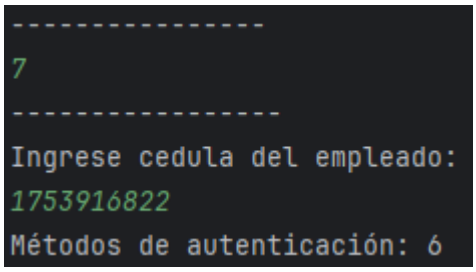
5. Mostrar datos de empleados (todos)



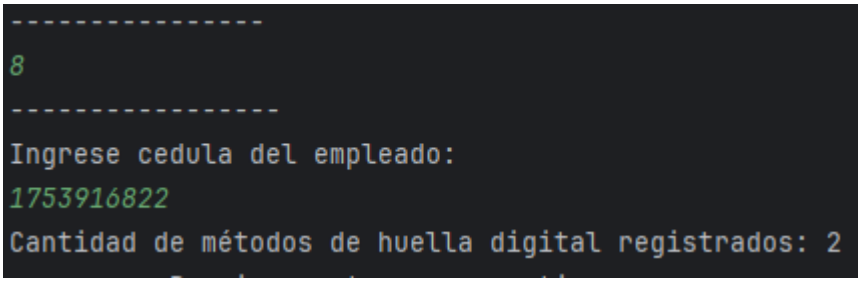
6. Buscar empleado y mostrar datos



7. Total métodos autenticación empleado



8. Total métodos huella empleado



9. Total métodos token empleado

```
----------------
9
-----------------
Ingrese cedula del empleado:
1753916822
Cantidad de métodos de token registrados: 3
```

10. Total métodos facial empleado

```
----------------
10
-----------------
Ingrese cedula del empleado:
1753916822
Cantidad de métodos de reconocimiento facial registrados: 1
```

11. Métodos mayor a umbral de empleado

```
----------------
11
-----------------
Ingrese cedula del empleado:
1753916822
Ingrese el umbral de seguridad:
40
Métodos de autenticación mayores al umbral de seguridad:
token
token
rostro
huella
huella
```

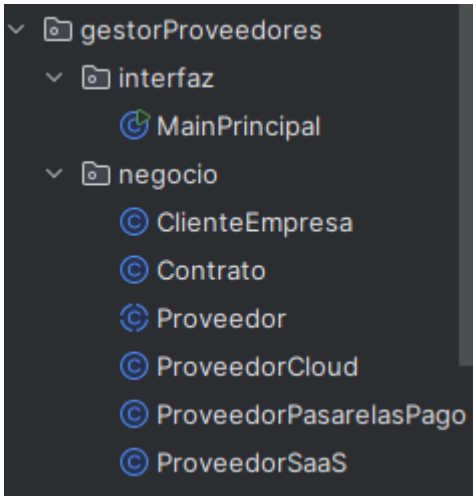
12. Autenticar empleado

```
----------------
12
-----------------
Ingrese cedula del empleado:
1753916822
Ingrese el tipo de autenticación (huella, rostro, token):
rostro
Ingrese el dato a autenticar:
Face4546
Acceso concedido
```

13. Salir

```
----------------
13
-----------------
Saliendo del sistema...
```

3. Ejercicio Gestor de Proveedores



3.1 Interfaz

MainPrincipal

```java
package EjerciciosCompletos.gestorProveedores.interfaz;

import EjerciciosCompletos.gestorProveedores.negocio.*;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

```

```java
public class MainPrincipal {
    static List<Proveedor> proveedores = new ArrayList<>();
    static List<ClienteEmpresa> clientes = new ArrayList<>();

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int opc;
        do {
            menu();
            opc = Integer.parseInt(sc.nextLine());
            switch (opc) {
                case 1: { // Crear proveedor
                    System.out.println("-----------------");
                    System.out.println("Nombre del proveedor:");
                    String nombre = sc.nextLine();
                    System.out.println("País:");
                    String pais = sc.nextLine();
                    System.out.println("Tipo (Cloud / SaaS / Pago):");
                    String tipo = sc.nextLine();

                    if (tipo.equalsIgnoreCase("Cloud")) {
                        proveedores.add(new ProveedorCloud(nombre, pais));
                    } else if (tipo.equalsIgnoreCase("SaaS")) {
                        proveedores.add(new ProveedorSaaS(nombre, pais));
                    } else if (tipo.equalsIgnoreCase("Pago")) {
                        proveedores.add(new ProveedorPasarelasPago(nombre, pais));
                    } else {
                        System.out.println("El proveedor es invalido");
                    }
                    pausa();
                }
                break;
                case 2: { // Crear cliente empresarial
                    System.out.println("-----------------");
                    System.out.println("Nombre de la empresa cliente:");
                    String nombre = sc.nextLine();
                    clientes.add(new ClienteEmpresa(nombre));
                    pausa();
                }
                break;
                case 3: { // Asociar proveedor a cliente
                    System.out.println("-----------------");
                    if (clientes.isEmpty()) {
                        System.out.println("No hay clientes registrados, ingrese un cliente previamente");
                        pausa();
                        break;
                    }
                    if (proveedores.isEmpty()) {
                        System.out.println("No hay proveedores registrados, ingrese un proveedor previamente");
                        pausa();
                        break;
                    }
                    ClienteEmpresa cliente = buscarCliente(sc);
                    Proveedor proveedor = buscarProveedor(sc);

                    if (cliente != null && proveedor != null) {
```

```
65                  cliente.contratarProveedor(proveedor);
66                  System.out.println("Proveedor asociado correctamente.");
67              }
68              pausa();
69          }
70          break;
71          case 4: { // Crear contrato entre proveedor y cliente
72              System.out.println("-----------------");
73              if (clientes.isEmpty())  {
74                  System.out.println("No hay clientes registrados, ingrese un cliente previamente");
75                  pausa();
76                  break;
77              }
78              if (proveedores.isEmpty()) {
79                  System.out.println("No hay proveedores registrados, ingrese un proveedor previamente");
80                  pausa();
81                  break;
82              }
83              ClienteEmpresa cliente = buscarCliente(sc);
84              if (cliente == null) {
85                  pausa();
86                  break;
87              }
88              Proveedor proveedor = buscarProveedor(sc);
89              if (proveedor == null) {
90                  pausa();
91                  break;
92              }
93              boolean asociado = false;
94              for (Proveedor p : cliente.getProveedores())  {
95                  if (p.equals(proveedor)) {
96                      asociado = true;
97                      break;
98                  }
99              }
100             if (!asociado) {
101                 System.out.println("El cliente NOOOO tiene asociado este proveedor");
102                 System.out.println("Debe asociarlo antes de crear un contrato en la opcion 3");
103                 pausa();
104                 break;
105             }
106             System.out.println("Precio:");
107             double precio = Double.parseDouble(sc.nextLine());
108             System.out.println("Duración en meses:");
109             int meses = Integer.parseInt(sc.nextLine());
110             proveedor.agregarContrato(new Contrato(precio, meses));
111             System.out.println("Contrato creado correctamente.");
112             pausa();
113         }
114         break;
115
116         case 5: { // Verificar si un cliente posee proveedores de un tipo determinado
117             System.out.println("-----------------");
118             if (clientes.isEmpty())  {
119                 System.out.println("No hay clientes registrados, ingrese un cliente previamente");
120                 pausa();
```

```
                    break;
                }
                ClienteEmpresa cliente = buscarCliente(sc);
                if (cliente != null) {
                    System.out.println("Ingrese tipo de proveedor: (Cloud / SaaS / Pago)");
                    String tipo = sc.nextLine();
                    boolean tiene = cliente.tieneProveedorTipo(tipo);
                    if (tiene) {
                        System.out.println("El cliente si cuenta con proveedores tipo:  " + tipo);
                    } else {
                        System.out.println("El cliente no cuenta con proveedores tipo: " + tipo);
                    }
                }
                pausa();
            }
            break;
            case 6: { // Listar contratos activos
                System.out.println("---- CONTRATOS  ACTIVOS DEL SISTEMA ----");
                for (ClienteEmpresa c : clientes) {
                    System.out.println(c.listarContratosActivos());
                }
                pausa();
            }
            break;
            case 7: {
                System.out.println("-----------------");
                System.out.println("Saliendo del sistema...");
                System.exit(0);
            }
            break;
            default:
                System.out.println("Opción inválida");
        }

    } while (opc != 7);
}

public static ClienteEmpresa buscarCliente(Scanner sc) {
    System.out.println("Nombre del cliente:");
    String nombre = sc.nextLine();
    for (ClienteEmpresa c : clientes) {
        if (c.getNombre().equalsIgnoreCase(nombre)) {
            return c;
        }
    }
    System.out.println("Cliente no encontrado");
    return null;
}

public static Proveedor buscarProveedor(Scanner sc) {
    System.out.println("Nombre del proveedor:");
    String nombre = sc.nextLine();
    for (Proveedor p : proveedores) {
        if (p.getNombre().equalsIgnoreCase(nombre)) {
            return p;
        }
```

```
        }
        System.out.println("Proveedor no encontrado");
        return null;
    }

    public static void menu() {
        System.out.println("------ MENU GESTOR DE PROVEEDORES ------");
        System.out.println("1. Crear proveedor");
        System.out.println("2. Crear cliente empresarial");
        System.out.println("3. Asociar proveedor a cliente");
        System.out.println("4. Crear contrato");
        System.out.println("5. Verificar tipo de proveedor");
        System.out.println("6. Listar contratos activos");
        System.out.println("7. Salir");
        System.out.println("-------------------------------------");
    }

    public static void pausa() {
        System.out.println("__________Presione enter para continuar__________");
        Scanner sc = new Scanner(System.in);
        sc.nextLine();
    }
}
```

3.2 Negocio

ClienteEmpresa

```
package EjerciciosCompletos.gestorProveedores.negocio;

import java.util.ArrayList;
import java.util.List;

public class ClienteEmpresa {
    private String nombre;
    private List<Proveedor> proveedores;

    public ClienteEmpresa(String nombre) {
        this.nombre = nombre;
        proveedores = new ArrayList<>();
    }

    public String getNombre() {return nombre;}
    public List<Proveedor> getProveedores() {return proveedores;}

    public void contratarProveedor(Proveedor proveedor) {
        proveedores.add(proveedor);
    }

    public boolean tieneProveedorTipo(String tipo) {
        for (Proveedor p : proveedores) {
            if (p.getTipoProveedor().equalsIgnoreCase(tipo)) {
                return true;
            }
```

```
        }
        return false;
    }

    public String listarContratosActivos() {
        StringBuilder sb = new StringBuilder();

        for (Proveedor p : proveedores) {
            for (Contrato c : p.getContratos()) {
                if (c.estaActivo()) {
                    sb.append("Cliente: ").append(nombre).append("\n");
                    sb.append("Proveedor: ").append(p.getNombre()).append("\n");
                    sb.append("Tipo: ").append(p.getTipoProveedor()).append("\n");
                    sb.append("Precio: ").append(c.getPrecio()).append("\n");
                    sb.append("Duración: ").append(c.getDuracionMeses()).append(" meses\n");
                    sb.append("----------------------------\n");
                }
            }
        }
        return sb.toString();
    }
}
```

Contrato

```
package EjerciciosCompletos.gestorProveedores.negocio;

public class Contrato {
    private double precio;
    private int duracionMeses;

    public Contrato(double precio, int duracionMeses) {
        this.precio = precio;
        this.duracionMeses = duracionMeses;
    }

    public double getPrecio() {return precio;}
    public int getDuracionMeses() {return duracionMeses;}

    public boolean estaActivo() {
        return duracionMeses > 0;
    }
}
```

Proveedor

```
package EjerciciosCompletos.gestorProveedores.negocio;

import java.util.ArrayList;
import java.util.List;

public abstract class Proveedor {
```

```
7
8     protected String nombre;
9     protected String pais;
10
11    protected List<Contrato> contratos;
12
13    public Proveedor(String nombre, String pais) {
14        this.nombre = nombre;
15        this.pais = pais;
16        contratos = new ArrayList<>();
17    }
18
19    public String getNombre() {return nombre;}
20    public String getPais() {return pais;}
21    public List<Contrato> getContratos() {return contratos;}
22
23    public void agregarContrato(Contrato contrato) {
24        contratos.add(contrato);
25    }
26
27    public abstract String getTipoProveedor();
28 }
```

ProveedorCloud

```
1  package EjerciciosCompletos.gestorProveedores.negocio;
2
3  public class ProveedorCloud extends Proveedor {
4
5      public ProveedorCloud(String nombre, String pais) {
6          super(nombre, pais);
7      }
8
9      @Override
10     public String getTipoProveedor() {
11         return "Cloud";
12     }
13 }
```

ProveedorPasarelasPago

```
1  package EjerciciosCompletos.gestorProveedores.negocio;
2
3  public class ProveedorPasarelasPago extends Proveedor {
4
5      public ProveedorPasarelasPago(String nombre, String pais) {
6          super(nombre, pais);
7      }
8
9      @Override
10     public String getTipoProveedor() {
11         return "Pago";
```

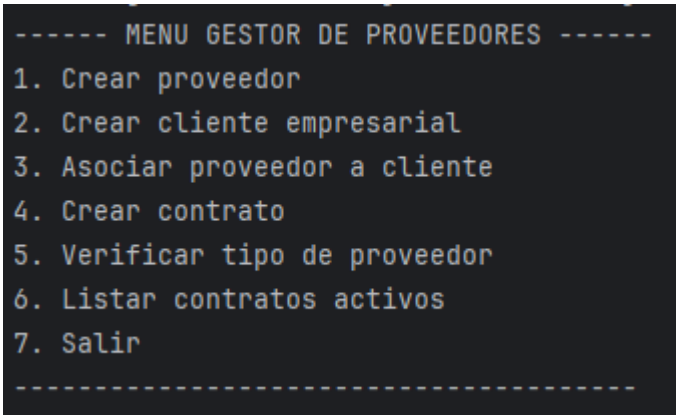
```
    }
}
```

ProveedorSaas

```
package EjerciciosCompletos.gestorProveedores.negocio;

public class ProveedorSaaS extends Proveedor {

    public ProveedorSaaS(String nombre, String pais) {
        super(nombre, pais);
    }

    @Override
    public String getTipoProveedor() {
        return "SaaS";
    }
}
```

3.3 Pruebas funcionamiento ejercicio Gestor Proveedores

Menú



1. Crear proveedor

2. Crear cliente empresarial

```
2
----------------
Nombre de la empresa cliente:
AlejandroCorp
```

3. Asociar proveedor a cliente

```
3
----------------
Nombre del cliente:
AlejandroCorp
Nombre del proveedor:
Claro
Proveedor asociado correctamente.
```

```
3
----------------
Nombre del cliente:
Alejandrocorp
Nombre del proveedor:
twenti
Proveedor asociado correctamente.
```

4. Crear contrato

```
4
----------------
Nombre del cliente:
alejandrocorp
Nombre del proveedor:
claro
Precio:
56000
Duración en meses:
50
Contrato creado correctamente.
```

```
4
----------------
Nombre del cliente:
alejandrocorp
Nombre del proveedor:
twenti
Precio:
30000
Duración en meses:
70
Contrato creado correctamente.
```

5. Verificar tipo de proveedor

```
5
-----------------
Nombre del cliente:
alejandrocorp
Ingrese tipo de proveedor: (Cloud / SaaS / Pago)
cloud
El cliente si cuenta con proveedores tipo:  cloud
```

```
5
-----------------
Nombre del cliente:
alejandrocorp
Ingrese tipo de proveedor: (Cloud / SaaS / Pago)
saas
El cliente no cuenta con proveedores tipo: saas
```

```
5
-----------------
Nombre del cliente:
alejandrocorp
Ingrese tipo de proveedor: (Cloud / SaaS / Pago)
pago
El cliente si cuenta con proveedores tipo:  pago
```

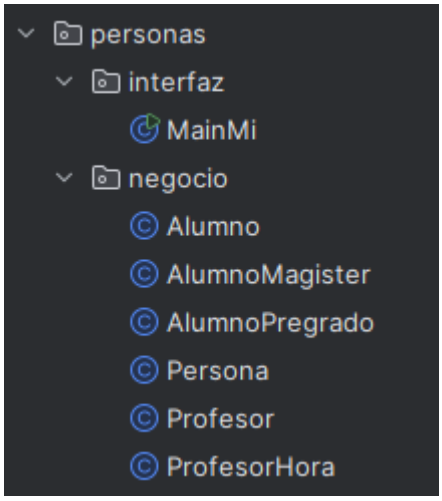
6. Listar contratos activos

```
6
---- CONTRATOS ACTIVOS DEL SISTEMA ----
Cliente: AlejandroCorp
Proveedor: Claro
Tipo: Pago
Precio: 56000.0
Duración: 50 meses
----------------------------
Cliente: AlejandroCorp
Proveedor: Twenti
Tipo: Cloud
Precio: 30000.0
Duración: 70 meses
----------------------------
```

7. Salir

```
7
-----------------
Saliendo del sistema...
```

4. Ejercicio Personas



4.1 Interfaz

MainMi

```java
package EjerciciosCompletos.personas.interfaz;

import EjerciciosCompletos.personas.negocio.AlumnoMagister;
import EjerciciosCompletos.personas.negocio.AlumnoPregrado;
import EjerciciosCompletos.personas.negocio.Persona;
import EjerciciosCompletos.personas.negocio.ProfesorHora;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class MainMi {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int opc, horas;
        String nom,universidad,carrera,cedula,tesis,especialidad;
        List<Persona> perso = new ArrayList<Persona>();
        do{
            menu();
            opc = Integer.parseInt(sc.nextLine());
            switch (opc){
                case 1:{
                    System.out.println("--------------------LISTADO--------------------");
                    listarAlumnosPregrado(perso);
                    listarAlumnosMagister(perso);
                    listarProfes(perso);
                    pausa();
```

```
28              }break;
29              case 2:{
30                  System.out.println("----------------");
31                  System.out.println("Ingrese el nombre");
32                  nom=sc.nextLine();
33                  System.out.println("Ingrese el cedula");
34                  cedula=sc.nextLine();
35                  if (existeCedula(perso, cedula)) {
36                      System.out.println("La cédula ya existe");
37                      pausa();
38                      break;
39                  }
40                  System.out.println("Ingrese el universidad");
41                  universidad=sc.nextLine();
42                  System.out.println("Ingrese el carrera");
43                  carrera=sc.nextLine();
44                  perso.add(new AlumnoPregrado(nom,cedula,universidad,carrera));
45                  pausa();
46              }break;
47              case 3:{
48                  System.out.println("----------------");
49                  System.out.println("Ingrese el nombre");
50                  nom=sc.nextLine();
51                  System.out.println("Ingrese el cedula");
52                  cedula=sc.nextLine();
53                  if (existeCedula(perso, cedula)) {
54                      System.out.println("La cédula ya existe");
55                      pausa();
56                      break;
57                  }
58                  System.out.println("Ingrese el universidad");
59                  universidad=sc.nextLine();
60                  System.out.println("Ingrese la tesis");
61                  tesis=sc.nextLine();
62                  perso.add(new AlumnoMagister(nom,cedula,universidad,tesis));
63                  pausa();
64              }break;
65              case 4:{
66                  System.out.println("----------------");
67                  System.out.println("Ingrese el nombre");
68                  nom=sc.nextLine();
69                  System.out.println("Ingrese el cedula");
70                  cedula=sc.nextLine();
71                  if (existeCedula(perso, cedula)) {
72                      System.out.println("La cédula ya existe");
73                      pausa();
74                      break;
75                  }
76                  System.out.println("Ingrese la especialidad");
77                  especialidad=sc.nextLine();
78                  System.out.println("Ingrese las horas");
79                  horas=Integer.parseInt(sc.nextLine());;
80                  perso.add(new ProfesorHora(nom,cedula,especialidad,horas));
81                  pausa();
82              }break;
83              case 5:{
```

```
84                  System.out.println("-----------------");
85                  listarAlumnosPregrado(perso);
86                  pausa();
87              }break;
88              case 6:{
89                  System.out.println("-----------------");
90                  listarAlumnosMagister(perso);
91                  pausa();
92              }break;
93              case 7:{
94                  System.out.println("-----------------");
95                  listarProfesores(perso);
96                  pausa();
97              }break;
98              case 8:{
99                  System.out.println("-----------------");
100                 System.out.println("Saliendo del sistema...");
101                 System.exit(0);
102             }break;
103             default:
104                 System.out.println("No es una opcion valida");
105         }
106     }while(opc != 8);
107 }
108 public static void menu() {
109     System.out.println("\n--------------------OPCION--------------------");
110     System.out.println("1.- Mostrar listado");
111     System.out.println("2.- Ingresar Alumno pregrado");
112     System.out.println("3.- Ingresar Alumno magister");
113     System.out.println("4.- Ingresar Profesor hora");
114     System.out.println("5.- Listar Alumnos Pregrado");
115     System.out.println("6.- Listar Alumnos Magister");
116     System.out.println("7.- Listar Profesores (Cedula, Horas, Sueldo) ");
117     System.out.println("8.- SALIR");
118     System.out.println("--------------------INGRESE  UNA OPCION--------------------\n");
119 }
120
121 public static void listarAlumnosPregrado(List<Persona> perso){
122     System.out.println("-----  ALUMNOS  DE PREGRADO  -----");
123
124     for(Persona p : perso){
125         if (p instanceof AlumnoPregrado){
126             System.out.println("-"+p);
127         }
128     }
129 }
130
131 public static void listarAlumnosMagister(List<Persona> perso){
132     System.out.println("-----  ALUMNOS  DE MAGISTER  -----");
133
134     for(Persona p : perso){
135         if (p instanceof AlumnoMagister){
136             System.out.println("-"+p);
137         }
138     }
139 }
```

```
140
141     public static void listarProfes(List<Persona> perso){
142         System.out.println("-----  PROFESORES  -----");
143
144         for(Persona p : perso){
145             if (p instanceof ProfesorHora){
146                 System.out.println("-"+p);
147             }
148         }
149     }
150
151     public static void listarProfesores(List<Persona> perso){
152         System.out.println("-----  PROFESORES  SUELDO  -----");
153         for(Persona p: perso){
154             if (p instanceof ProfesorHora){
155                 ProfesorHora ph= (ProfesorHora) p;
156                 System.out.println("Nombre: "+ph.getNombre()+
157                         "\n     Cedula: "+ph.getCedula()+
158                         "\n     Horas: "+ph.getHoras()+
159                         "\n     Sueldo: "+ph.getHoras()*20);
160             }
161         }
162     }
163     public static void pausa() {
164         System.out.println("_____Presione enter para continuar_____");
165         Scanner sc = new Scanner(System.in);
166         sc.nextLine();
167     }
168
169     public static boolean existeCedula(List<Persona> perso, String cedula) {
170         for (Persona p : perso) {
171             if (p.getCedula().equals(cedula)) {
172                 return true;
173             }
174         }
175         return false;
176     }
177 }
```

4.2 Negocio

Alumno

```
1  package EjerciciosCompletos.personas.negocio;
2
3  public class Alumno extends Persona {
4      private String universidad;
5
6      public Alumno(String nombre, String cedula, String universidad) {
7          super(nombre, cedula);
8          this.universidad = universidad;
9      }
10
11     public String getUniversidad() {
```

```
        return universidad;
    }

    public void setUniversidad(String universidad) {
        this.universidad = universidad;
    }

    @Override
    public String toString() {
        return super.toString() + "\n Universidad: "+ universidad;
    }
}
```

AlumnoMagister

```
package EjerciciosCompletos.personas.negocio;

public class AlumnoMagister extends Alumno {
    private String tesis;

    public AlumnoMagister(String nombre, String cedula, String universidad, String tesis) {
        super(nombre, cedula, universidad);
        this.tesis = tesis;
    }

    public String getTesis() {
        return tesis;
    }

    public void setTesis(String tesis) {
        this.tesis = tesis;
    }

    @Override
    public String toString() {
        return super.toString() + "\nTesis=" + tesis;
    }
```

AlumnoPregrado

```
package EjerciciosCompletos.personas.negocio;

public class AlumnoPregrado extends Alumno {
    private String carrera;

    public AlumnoPregrado(String nombre, String cedula, String universidad, String carrera) {
        super(nombre, cedula, universidad);
        this.carrera = carrera;
    }

    public String getCarrera() {
```

```
        return carrera;
    }

    public void setCarrera(String carrera) {
        this.carrera = carrera;
    }

    @Override
    public String toString() {
        return super.toString() + "\n Carrera=" + carrera;
    }
}
```

Persona

```
package EjerciciosCompletos.personas.negocio;

public class Persona {
    private String nombre, cedula;

    public Persona(String nombre, String cedula) {
        this.nombre = nombre;
        this.cedula = cedula;
    }

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

    public String getCedula() {
        return cedula;
    }

    @Override
    public String toString() {
        return  "Nombre=" + nombre + ", Cedula=" + cedula;
    }
}
```

Profesor

```
package EjerciciosCompletos.personas.negocio;

public class Profesor extends Persona {
    private String especialidad;

    public Profesor(String nombre, String cedula, String especialidad) {
        super(nombre, cedula);
        this.especialidad = especialidad;
```

```
    }

    public String getEspecialidad() {
        return especialidad;
    }

    public void setEspecialidad(String especialidad) {
        this.especialidad = especialidad;
    }
    @Override
    public String toString() {
        return super.toString() + "\n Especialidad=" + especialidad;
    }
}
```

ProfesorHora

```
package EjerciciosCompletos.personas.negocio;

public class ProfesorHora extends Profesor {
    private int horas;

    public ProfesorHora(String nombre, String cedula, String especialidad, int horas) {
        super(nombre, cedula, especialidad);
        this.horas = horas;
    }

    public int getHoras() {
        return horas;
    }

    public void setHoras(int horas) {
        this.horas = horas;
    }

    @Override
    public String toString() {
        return super.toString() + "\n Horas=" + horas;
    }
```

4.3 Pruebas funcionamiento ejercicio Personas

Menú

```
---------------------OPCION---------------------
1.- Mostrar listado
2.- Ingresar Alumno pregrado
3.- Ingresar Alumno magister
4.- Ingresar Profesor hora
5.- Listar Alumnos Pregrado
6.- Listar Alumnos Magister
7.- Listar Profesores (Cedula, Horas, Sueldo)
8.- SALIR
---------------------INGRESE UNA OPCION---------------------
```

1.- Mostrar listado

2.- Ingresar Alumno pregrado

```
2
-----------------
Ingrese el nombre
Alejandro
Ingrese el cedula
1753684533
Ingrese el universidad
UDLA
Ingrese el carrera
Ing. ciberseguridad
```

```
2
-----------------
Ingrese el nombre
Johan
Ingrese el cedula
1798456733
Ingrese el universidad
UDLA
Ingrese el carrera
Enfermeria
```

3.- Ingresar Alumno magister

```
3
-----------------
Ingrese el nombre
Jairo
Ingrese el cedula
1765983455
Ingrese el universidad
Catolica
Ingrese la tesis
Inv. de subespecies
```

```
3
-----------------
Ingrese el nombre
Ramon
Ingrese el cedula
1709345622
Ingrese el universidad
Salesiana del Norte
Ingrese la tesis
Problema del viajero
```

4.- Ingresar Profesor hora

```
4
-----------------
Ingrese el nombre
Ana
Ingrese el cedula
1789893455
Ingrese la especialidad
Calculo
Ingrese las horas
50
```

```
4
-----------------
Ingrese el nombre
Leslie
Ingrese el cedula
1745654633
Ingrese la especialidad
Fisica
Ingrese las horas
30
```

5.- Listar Alumnos Pregrado

```
5
-----------------
----- ALUMNOS DE PREGRADO -----
-Nombre=Alejandro, Cedula=1753684533
 Universidad: UDLA
 Carrera=Ing. ciberseguridad
-Nombre=Johan, Cedula=1798456733
 Universidad: UDLA
 Carrera=Enfermeria
```

6.- Listar Alumnos Magister

```
6
-----------------
----- ALUMNOS DE MAGISTER -----
-Nombre=Jairo, Cedula=1765983455
 Universidad: Catolica
Tesis=Inv. de subespecies
-Nombre=Ramon, Cedula=1709345622
 Universidad: Salesiana del Norte
Tesis=Problema del viajero
```

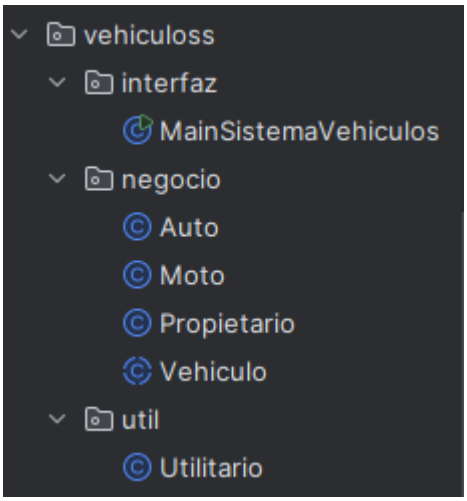
7.- Listar Profesores (Cedula, Horas, Sueldo)

```
7
----------------
----- PROFESORES SUELDO -----
Nombre: Ana
     Cedula: 1789893455
     Horas: 50
     Sueldo: 1000
Nombre: Leslie
     Cedula: 1745654633
     Horas: 30
     Sueldo: 600
```

8.- SALIR

```
8
----------------
Saliendo del sistema...
```

5. Ejercicio Vehiculos



5.1 Interfaz

MainSistemaVehiculos

```java
package EjerciciosCompletos.vehiculoss.interfaz;

import EjerciciosCompletos.vehiculoss.negocio.Propietario;
import EjerciciosCompletos.vehiculoss.negocio.Vehiculo;
import EjerciciosCompletos.vehiculoss.util.Utilitario;

import java.util.List;
import java.util.Scanner;
```

```java

public class MainSistemaVehiculos {
    public static void main(String[] args) {
        Utilitario u = new Utilitario();
        Scanner sc = new Scanner(System.in);
        String marca, modelo;
        int anio;
        String traccion;
        String tipo;

        //Propietario
        String cedula, nombre, telefono;

        //Moto
        double altura;
        String arranque;

        int opc=0;

        do {
            u.menu();
            opc = Integer.parseInt(sc.nextLine());
            switch (opc){
                case 1: {//Agregar Propietario
                    System.out.println("-----------------");
                    System.out.println("Ingrese cedula: ");
                    cedula= sc.nextLine();
                    System.out.println("Ingrese su nombre: ");
                    nombre = sc.nextLine();
                    System.out.println("Ingrese su telefono: ");
                    telefono= sc.nextLine();

                    u.agregarPropietario(cedula, nombre, telefono);
                    pausa();
                }break;
                case 2: {//Asignar Propietario a Auto
                    System.out.println("-----------------");
                    System.out.println("Ingrese su cedula: ");
                    cedula= sc.nextLine();
                    Propietario p = u.burscarPropietario(cedula);
                    if(p != null){
                        System.out.println("Ingrese marca: ");
                        marca= sc.nextLine();
                        System.out.println("Ingree modelo: ");
                        modelo=sc.nextLine();
                        System.out.println("Ingrese Tipo: ");
                        tipo=sc.nextLine();
                        System.out.println("Ingrese traccion: ");
                        traccion=sc.nextLine();
                        System.out.println("Ingrese anio: ");
                        anio= Integer.parseInt(sc.nextLine());

                        u.agregarAuto(marca,modelo,anio,p,traccion,tipo);
                    }else {
                        System.out.println("Propietario No Existe");
                    }
```

```
65                  pausa();
66              }break;
67              case 3:{//Asignar Propietario a Moto
68                  System.out.println("-----------------");
69                  System.out.println("Ingrese su cedula: ");
70                  cedula= sc.nextLine();
71                  Propietario p= u.buscarPropietario(cedula);
72                  u.buscarPropietario(cedula);
73                  if(p!=null){
74                      System.out.println("Ingrese marca: ");
75                      marca= sc.nextLine();
76                      System.out.println("Ingrese modelo: ");
77                      modelo= sc.nextLine();
78                      System.out.println("Ingrese año: ");
79                      anio = Integer.parseInt(sc.nextLine());
80                      System.out.println("Ingrese altura: ");
81                      altura= Double.parseDouble(sc.nextLine());
82                      System.out.println("Ingrese arranque: ");
83                      arranque=sc.nextLine();
84
85                      u.agregarMoto(marca,modelo,anio,p,altura,arranque);
86                  }else {
87                      System.out.println("Propietario No Existe");
88                  }
89                  pausa();
90              }break;
91              case 4:{//Buscar Vehiculos por Marca
92                  System.out.println("-----------------");
93                  System.out.println("Ingrese la marca de vehiculos que quiere buscar: ");
94                  marca=sc.nextLine();
95                  List<Vehiculo> lista= u.buscarVehiculoMarca(marca);
96                  if (lista.isEmpty()) {
97                      System.out.println("No se encontraron vehiculos de esa marca.");
98                  } else {
99                      System.out.println("\n-------------------------------\n");
100                     for (Vehiculo v : lista) {
101                         System.out.println(v);
102                         System.out.println("-------------------------------");
103                     }
104                 }
105                 pausa();
106             }break;
107             case 5: {//Listar Vehiculos
108                 System.out.println("-----------------");
109                 System.out.println(u.listarVehiculos());
110                 pausa();
111             }break;
112             case 6: {//Listar Propietarios
113                 System.out.println("-----------------");
114                 System.out.println(u.listaPropietarios());
115                 pausa();
116             }break;
117             case 7: {//Listar Automoviles
118                 System.out.println("-----------------");
119                 System.out.println(u.listarAutomoviles());
120                 pausa();
```

```
            }break;
            case 8: {//Listar Motos
                System.out.println("-----------------");
                System.out.println(u.listarMoto());
                pausa();
            }break;
            case 9: {//Mostrar Motos por Marca
                System.out.println("-----------------");
                System.out.print("Ingrese la marca de moto a buscar: ");
                marca = sc.nextLine();
                System.out.println(u.buscarMotoMarca(marca));
                pausa();
            } break;
            case 10:{//Matricular
                System.out.println("-----------------");
                System.out.println("Escriba la cedula: ");
                cedula= sc.nextLine();
                System.out.println("Marca: ");
                marca= sc.nextLine();
                System.out.println("Año: ");
                anio= Integer.parseInt(sc.nextLine());
                int valor= u.matricular(cedula,marca,anio);
                if (valor!=-1){
                    System.out.println("Valor "+valor+"\nMatriculado");
                }else{
                    System.out.println("Si estan bien los datos esta mattriculado ya, si estan mal vuelva a ingresar los datos");
                }
                pausa();
            }break;
            case 11: {//salir
                System.out.println("-----------------");
                System.out.println("Saliendo del sistema...");
                System.exit(0);
            } break;
            default: {
                System.out.println("Opcion Invalida. Ingrese Nuevamente");
            }
          }
      } while (opc != 11);
    }
    public static void pausa() {
        System.out.println("_____Presione enter para continuar_____");
        Scanner sc = new Scanner(System.in);
        sc.nextLine();
    }
}
```

5.2 Negocio

Auto

```
package EjerciciosCompletos.vehiculoss.negocio;

import java.util.GregorianCalendar;
```

```java

public class Auto extends Vehiculo{
    private String traccion;
    private String tipo;

    public Auto(String marca, String modelo, int anio, Propietario duenio, String traccion, String tipo) {
        super(marca, modelo, anio, duenio);
        this.traccion = traccion;
        this.tipo = tipo;
    }

    public String getTraccion() {
        return traccion;
    }

    public void setTraccion(String traccion) {
        this.traccion = traccion;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    @Override
    public String toString() {
        return super.toString() + "\n      Traccion: " + traccion +
                                  "\n      Tipo: " + tipo;
    }

    @Override
    public int matricula() {
        if (matriculado) {
            return -1; // ya matriculado
        }
        GregorianCalendar gc=new GregorianCalendar();
        int anio=gc.get(GregorianCalendar.YEAR);
        int calculo= 300-((anio-getAnio())*10);
        matriculado = true;
        if (calculo>0){
            return calculo;
        }else {
            return 10;
        }
    }
}
```

Moto

```java
package EjerciciosCompletos.vehiculoss.negocio;

```

```java
import java.util.GregorianCalendar;

public class Moto extends Vehiculo {
    private double altura;
    private String arranque;

    public Moto(String marca, String modelo, int anio, Propietario duenio, double altura, String arranque) {
        super(marca, modelo, anio, duenio);
        this.altura = altura;
        this.arranque = arranque;
    }

    public double getAltura() {
        return altura;
    }

    public void setAltura(double altura) {
        this.altura = altura;
    }

    public String getArranque() {
        return arranque;
    }

    public void setArranque(String arranque) {
        this.arranque = arranque;
    }

    @Override
    public String toString(){
        return super.toString() + "\n      Altura: " + altura +
                                  "\n      Arranque: " + arranque;
    }

    @Override
    public int matricula() {
        if (matriculado) {
            return -1; // ya matriculado
        }
        GregorianCalendar gc=new GregorianCalendar();
        int anio=gc.get(GregorianCalendar.YEAR);
        int calculo= 200-((anio-getAnio())*10);
        matriculado = true;
        if (calculo>0){
            return calculo;
        }else {
            return 10;
        }
    }

```

Propietario

```java
package EjerciciosCompletos.vehiculoss.negocio;
```

```java


public class Propietario {
    private String cedula;
    private String nombre;
    private String telefono;

    public Propietario(String cedula, String nombre, String telefono) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.telefono = telefono;
    }

    public String getCedula() {
        return cedula; //Set no, la cedula no cambia
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    @Override
    public String toString() {
        return "\n           Cedula:  " + cedula +
               "\n           Nombre: " + nombre +
               "\n           Telefono: " + telefono;
    }
}
```

Vehiculo

```java
package EjerciciosCompletos.vehiculoss.negocio;

public abstract class Vehiculo{
    private String marca;
    private String modelo;
    private int anio;

    private Propietario duenio; //Creacion de objeto de la clase (tipo de dato) Propietario

    public Vehiculo(String marca, String modelo, int anio, Propietario duenio) {
        this.marca = marca;
```

```
12          this.modelo = modelo;
13          this.anio = anio;
14          this.duenio = duenio;
15      }
16
17      public String getMarca() {
18          return marca;
19      }
20
21      public void setMarca(String marca) {
22          this.marca = marca;
23      }
24
25      public String getModelo() {
26          return modelo;
27      }
28
29      public void setModelo(String modelo) {
30          this.modelo = modelo;
31      }
32
33      public int getAnio() {
34          return anio;
35      }
36
37      public void setAnio(int anio) {
38          this.anio = anio;
39      }
40
41      public Propietario getDuenio() {
42          //Retorna un objeto de la clase propietario
43          return duenio;
44      }
45
46      public void setDuenio(Propietario duenio) {
47          //Recibe parametro de la clase propietario
48          this.duenio = duenio;
49      }
50
51      @Override
52      public String toString (){
53          return "Marca: " + marca +
54                  "\n     Modelo: " + modelo +
55                  "\n     Anio: " + anio +
56                  "\n     Propietario: " + duenio;
57      }
58      protected boolean matriculado = false;
59      public abstract int matricula();
60  }
```

5.3 Util

Utilitario

```java
package EjerciciosCompletos.vehiculoss.util;

import EjerciciosCompletos.vehiculoss.negocio.Auto;
import EjerciciosCompletos.vehiculoss.negocio.Moto;
import EjerciciosCompletos.vehiculoss.negocio.Propietario;
import EjerciciosCompletos.vehiculoss.negocio.Vehiculo;

import java.util.ArrayList;
import java.util.List;

//Creacion de listas necesarias para la aplicacion
public class Utilitario {
    private List<Propietario> propietarios;
    private List<Vehiculo> vehiculos;

    //NO tiene constructor por parametros, es por defectp (inicializa las listas)
    public Utilitario (){
        propietarios = new ArrayList<>();
        vehiculos = new ArrayList<>();
    }

    //Requerimientos aca - Metodos para guardar datos
    public void agregarPropietario(String cedula, String nombre, String telefono){
        Propietario pr = burscarPropietario(cedula);
        if(pr == null){
            propietarios.add(new Propietario(cedula, nombre, telefono));
        }
        else {
            System.out.println("El propietario ya existe");
        }
    }


    //Metodo para devolver datos - Busqueda por parametro
    public Propietario buscarPropietario (String cedula){
        for (Propietario p: propietarios) {
            if (p.getCedula().equals(cedula)){
                return p; //Si lo encontro, devuelve el objeto
            }
        }
        return null; //En caso de que no haya encontrado
    }

    public void agregarAuto(String marca, String modelo, int anio, Propietario duenio, String traccion, String tipo){
        vehiculos.add(new Auto(marca, modelo, anio, duenio,traccion,tipo));
    }
    public void agregarMoto(String marca, String modelo, int anio, Propietario duenio, double altura, String arranque){
        vehiculos.add(new Moto(marca, modelo, anio, duenio, altura, arranque));
    }

    //Metodo para devolver datos - Busqueda por parametro
    public Propietario burscarPropietario (String cedula){
        for (Propietario p: propietarios) {
            if (p.getCedula().equals(cedula)){
                return p; //Si lo encontro, devuelve el objeto
            }
```

```
57          }
58          return null; //En caso de que no haya encontrado
59      }
60
61      public List<Vehiculo> buscarVehiculoMarca(String marca) {
62          List<Vehiculo> buscar = new ArrayList<>();
63          for (Vehiculo v : vehiculos) {
64              if (v.getMarca().equalsIgnoreCase(marca)) {
65                  buscar.add(v);
66              }
67          }
68          return buscar;
69      }
70
71      public String buscarMotoMarca(String marca){
72          StringBuilder sb = new StringBuilder();
73          for(int i = 0; i < vehiculos.size(); i++){
74              if (vehiculos.get(i).getMarca().equalsIgnoreCase(marca)) {
75                  if (vehiculos.get(i) instanceof Moto){
76                      Moto m = (Moto) vehiculos.get(i); //For normal - Casting (Cosas que solo estan en una clase se hace Casting)
77                      sb.append("\n Nombre: " + m.getDuenio().getNombre());
78                      sb.append("\n    Modelo: " + m.getModelo());
79                      sb.append("\n    Anio: " + m.getAnio());
80                      sb.append("\n    Arranque: " + m.getArranque());
81                      sb.append("\n    Altura: " + m.getAltura());
82                      sb.append("\n");
83                  }
84              }
85          }
86          return sb.toString(); //sb es un OBJETO
87      }
88
89      //metodo q muestra de todos los vehiculos (listas)
90      public String listarVehiculos(){
91          String aux = "-";
92          for(int i = 0; i < vehiculos.size(); i++){
93              aux += vehiculos.get(i);
94              aux += "\n";
95              aux += "\n-";
96          }
97          return aux;
98      }
99      public String listarAutomoviles(){
100         StringBuilder sb = new StringBuilder();
101         for (Vehiculo v: vehiculos){
102             if (v instanceof Auto){
103                 Auto a = (Auto) v;
104                 sb.append(a);
105                 sb.append("\n");
106             }
107         } return sb.toString();
108     }
109
110     public String listarMoto(){
111         StringBuilder sb = new StringBuilder();
112         for (Vehiculo v: vehiculos){
```

```
113             if (v instanceof  Moto){
114                 Moto a = (Moto) v;
115                 sb.append(a);
116                 sb.append("\n");
117             }
118         } return sb.toString();
119     }
120 
121     public String listaPropietarios(){
122         StringBuilder  sb =new StringBuilder();
123         for(int i = 0; i < propietarios.size(); i++){
124             sb.append(propietarios.get(i).toString());
125             sb.append("\n");
126         }return sb.toString();
127     }
128 
129     public int matricular(String cedula,String marca, int anio){
130         for (Vehiculo v: vehiculos){
131             if (v.getMarca().equalsIgnoreCase(marca)&& v.getDuenio().getCedula().equals(cedula) && v.getAnio()==anio){
132                 return v.matricula();
133             }
134         }
135         return -1;
136     }
137 
138     public void menu(){
139         System.out.println("-----  MENU -----");
140         System.out.println("1. Agregar  Propietario");
141         System.out.println("2. Asignar Propietario a Auto");
142         System.out.println("3. Asignar Propietario a Moto");
143         System.out.println("4. Buscar Vehiculos por Marca");
144         System.out.println("5. Listar Vehiculos");
145         System.out.println("6. Listar Propietarios");
146         System.out.println("7. Listar Automoviles");
147         System.out.println("8. Listar Motos");
148         System.out.println("9. Mostrar Motos por Marca");
149         System.out.println("10. Matricular");
150         System.out.println("11. Salir");
151         System.out.println("----------------");
152     }
153 }
```

5.4 Pruebas funcionamiento ejercicio Vehiculos

Menú

```
----- MENU -----
1. Agregar Propietario
2. Asignar Propietario a Auto
3. Asignar Propietario a Moto
4. Buscar Vehiculos por Marca
5. Listar Vehiculos
6. Listar Propietarios
7. Listar Automoviles
8. Listar Motos
9. Mostrar Motos por Marca
10. Matricular
11. Salir
---------------
```

1. Agregar Propietario

```
1
----------------
Ingrese cedula:
1753916822
Ingrese su nombre:
Alejandro
Ingrese su telefono:
0994659854
```

```
1
----------------
Ingrese cedula:
1795685422
Ingrese su nombre:
Jairo
Ingrese su telefono:
0996558899
```

2. Asignar Propietario a Auto

```
2
----------------
Ingrese su cedula:
1753916822
Ingrese marca:
Suzuki
Ingree modelo:
SUV
Ingrese Tipo:
Familiar
Ingrese traccion:
4x4
Ingrese anio:
2022
```

```
2
----------------
Ingrese su cedula:
1795685422
Ingrese marca:
Suzuki
Ingree modelo:
Explora
Ingrese Tipo:
Familiar
Ingrese traccion:
4x2
Ingrese anio:
2025
```

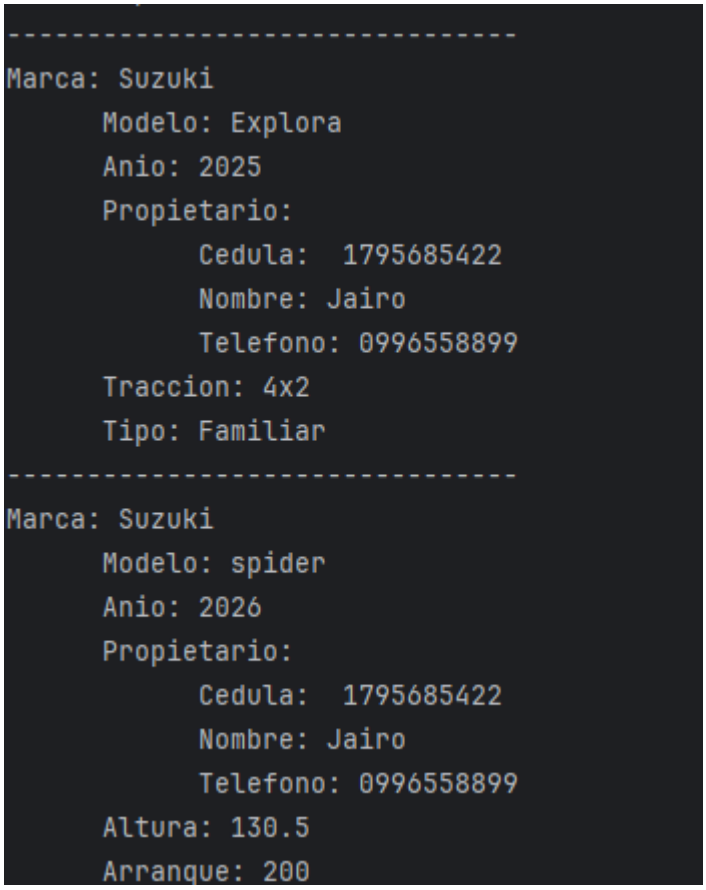
3. Asignar Propietario a Moto

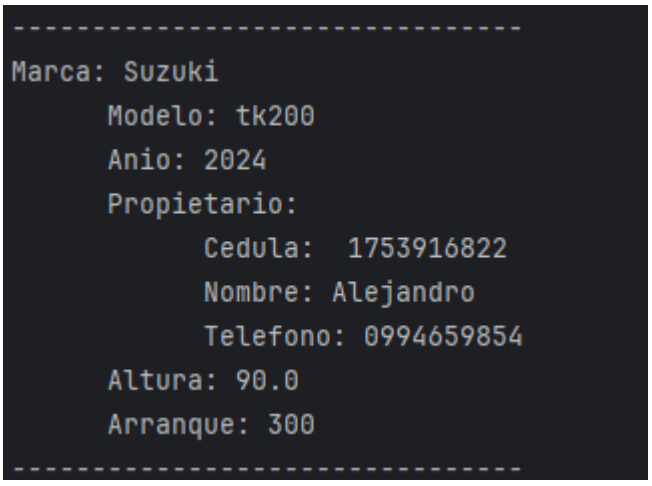
```
3
-----------------
Ingrese su cedula:
1795685422
Ingrese marca:
Suzuki
Ingrese modelo:
spider
Ingrese año:
2026
Ingrese altura:
130.5
Ingrese arranque:
200
```

```
3
-----------------
Ingrese su cedula:
1753916822
Ingrese marca:
Suzuki
Ingrese modelo:
tk200
Ingrese año:
2024
Ingrese altura:
90
Ingrese arranque:
300
```

4. Buscar Vehiculos por Marca

```
4
----------------
Ingrese la marca de vehiculos que quiere buscar:
suzuki

------------------------------

Marca: Suzuki
      Modelo: SUV
      Anio: 2022
      Propietario:
            Cedula:  1753916822
            Nombre: Alejandro
            Telefono: 0994659854
      Traccion: 4x4
      Tipo: Familiar
```





5. Listar Vehiculos

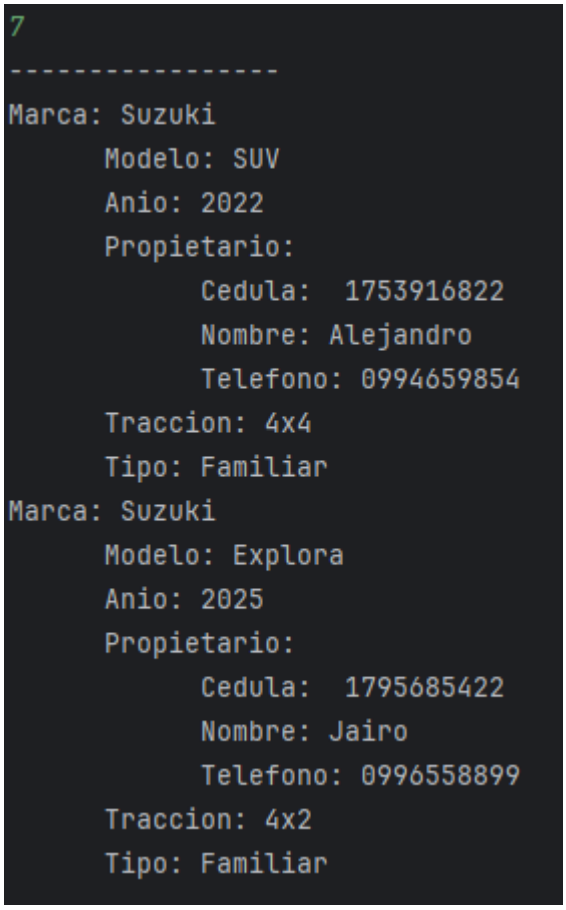
```
5
----------------
-Marca: Suzuki
      Modelo: SUV
      Anio: 2022
      Propietario:
            Cedula:  1753916822
            Nombre: Alejandro
            Telefono: 0994659854
      Traccion: 4x4
      Tipo: Familiar

-Marca: Suzuki
      Modelo: Explora
      Anio: 2025
      Propietario:
            Cedula:  1795685422
            Nombre: Jairo
            Telefono: 0996558899
      Traccion: 4x2
      Tipo: Familiar
```

```
-Marca: Suzuki
      Modelo: spider
      Anio: 2026
      Propietario:
            Cedula:  1795685422
            Nombre: Jairo
            Telefono: 0996558899
      Altura: 130.5
      Arranque: 200

-Marca: Suzuki
      Modelo: tk200
      Anio: 2024
      Propietario:
            Cedula:  1753916822
            Nombre: Alejandro
            Telefono: 0994659854
      Altura: 90.0
      Arranque: 300
```

6. Listar Propietarios

```
6
----------------

            Cedula:  1753916822
            Nombre: Alejandro
            Telefono: 0994659854

            Cedula:  1795685422
            Nombre: Jairo
            Telefono: 0996558899
```

7. Listar Automoviles



8. Listar Motos

```
8
-----------------
Marca: Suzuki
      Modelo: spider
      Anio: 2026
      Propietario:
            Cedula:  1795685422
            Nombre: Jairo
            Telefono: 0996558899
      Altura: 130.5
      Arranque: 200
Marca: Suzuki
      Modelo: tk200
      Anio: 2024
      Propietario:
            Cedula:  1753916822
            Nombre: Alejandro
            Telefono: 0994659854
      Altura: 90.0
      Arranque: 300
```

9. Mostrar Motos por Marca

```
9
-----------------
Ingrese la marca de moto a buscar: suzuki

 Nombre: Jairo
     Modelo: spider
     Anio: 2026
     Arranque: 200
     Altura: 130.5

 Nombre: Alejandro
     Modelo: tk200
     Anio: 2024
     Arranque: 300
     Altura: 90.0
```

10. Matricular

```
10
-----------------
Escriba la cedula:
1753916822
Marca:
suzuki
Año:
2024
Valor 190
Matriculado
```

```
10
-----------------
Escriba la cedula:
1753916822
Marca:
suzuki
Año:
2022
Valor 270
Matriculado
```

11. Salir

```
11
-----------------
Saliendo del sistema...
```