UNIVERSIDAD PRIVADA FRANZ TAMAYO

DEFENSA HITO 2 - TAREA FINAL

Nombre Completo: Unifranz. Diego Emiliano Rivera Tapia
Asignatura: PROGRAMACIÓN III
Carrera: INGENIERÍA DE SISTEMAS
Paralelo: PROG (1)
Docente: Lic. William R. Barra Paredes
fecha: 29/03/2020
github:

Parte Teórica.

1. Preguntas.

Responda de manera breve y clara posible.

- Defina y muestre ejemplos de la clase Scanner.

  Respuesta:

  Son métodos para leer los valores de entrada de varios tipos de variables.

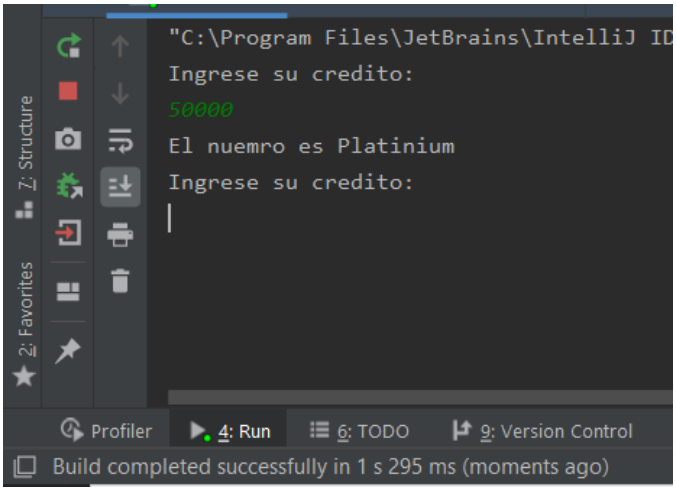
  Ejemplo:

  Código:

```
System.out.println("Ingrese su credito:");
while(leer.hasNextInt()) {

   int credit_number = leer.nextInt();

   if (credit_number <= 50000) {
      System.out.println("El nuemro es Platinium");
   }
   else if (credit_number >= 10000) {
      System.out.println("El nuemro es GOLD");
   }
    else if (credit_number < 10000) {
      System.out.println("El nuemro es SILVER");
   } else {
      System.out.println("Error");
   }
   System.out.println("Ingrese su credito:");
}
```

Elaboración: Propia

Ejecución del código:



Elaboración: Propia

- Que es la programación orientada a objetos(POO).

Respuesta:
Es un paradigma de programación capaz de resolver una problemática a partir de clases y objetos.

- Cuál es la diferencia entre interfaz y herencia.
  Respuesta:
  La herencia es una forma de extender las funcionalidades y atributos de una clase,
  y la interfaz obliga a una clase a dar cuerpo a los métodos declarados
- Qué elementos crees que definen a un objeto.
  Respuesta:
  Son entidades con características o rasgos diferentes a otros objetos.
- Que es unas clases abstracta y muestre un ejemplo.

  Respuestas:
  Es una clase que declara la existencia de métodos, pero no la implementación de dichos métodos

  Ejemplo:

```java
package Transporte;

import java.util.Scanner;

public class Auto implements Vehiculo {
    private int nroRuedas;
    private String color;


    public void nroRuedas() {
        System.out.printf("Numero de ruedas del auto: %d\n",this.getNroRuedas());
    }

    public void color() {
        System.out.printf("Color del auto: %s\n",this.getColor());
    }

    public void setNroRuedas(int nroRuedas) { this.nroRuedas = nroRuedas; }
    public int getNroRuedas() { return this.nroRuedas; }
    public void setColor(String color) { this.color = color; }
    public String getColor() { return this.color; }

}
```

Auto

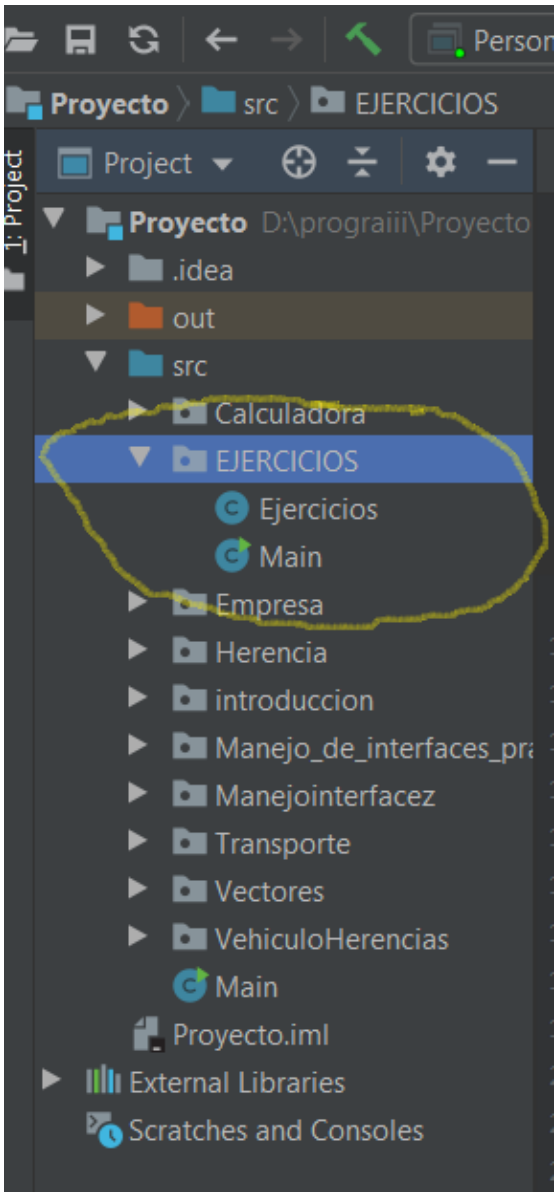
Elaboración: Propia

2. Ejercicios.

Debe de trabajarlos en el package EJERCICIOS.

- Crear un PACKAGE de nombre EJERCICIOS.
- Crear una clase de nombre Ejercicios.java.
- Crear la clase MAIN para mostrar las soluciones.

Solución:



Elaboración: Propia

- ## Preguntas ejercicios.

- Generar la serie **fibonacci** hasta un valor n leído por teclado

Solución:

Método para la Serie:

```
public void serieFibonacci(int n){
   int cont = 0;
   int x  = 1;
   int aux = 0;
   while (cont<n){
      for(int i = 0;i<n;i++) {
         System.out.printf("%d,",cont);
         aux=cont+x;
         cont=x;
         x=aux;
      }
```

```
    }
}
```

Elaboración: Propia

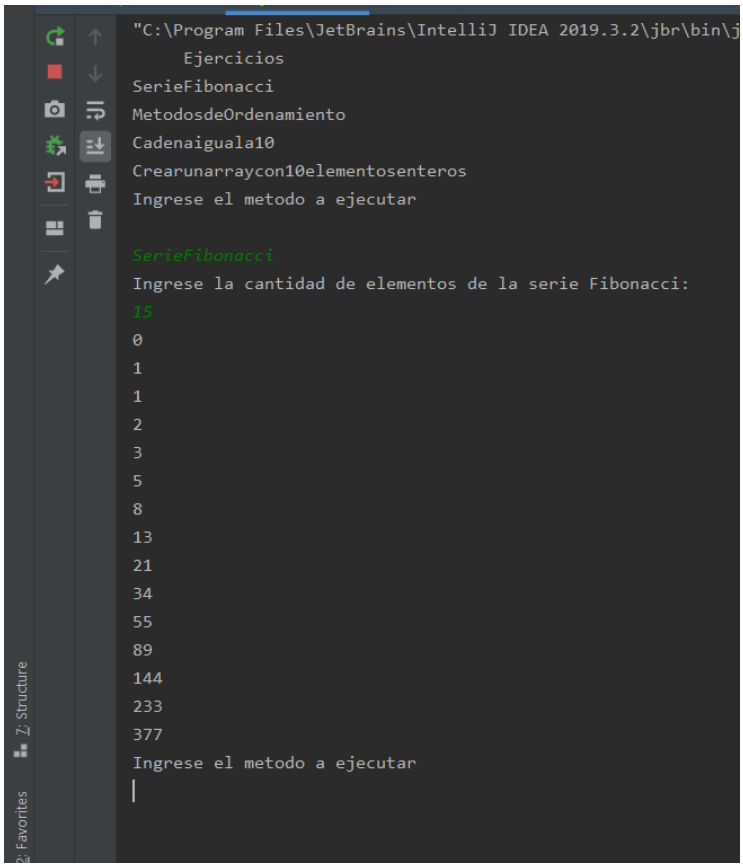
Main:

```
package EJERCICIOS;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        System.out.println("\t Ejercicios");
        System.out.println("SerieFibonacci");
        System.out.println("MetodosdeOrdenamiento");
        System.out.println("Cadenaiguala10");
        System.out.println("Crearunarraycon10elementosenteros");
        System.out.println("Ingrese el metodo a ejecutar");
        Scanner escribir = new Scanner(System.in);
        Scanner leer = new Scanner(System.in);
        Ejercicios ejercicios = new Ejercicios();
        boolean salir = true;
        String Metodo;

        while(salir) {
            Metodo = escribir.next();

            switch (Metodo) {
                case "SerieFibonacci":
                    System.out.println("Ingrese la cantidad de elementos de la serie Fibonacci: ");
                    int n = leer.nextInt();
                    ejercicios.serieFibonacci(n);
                    System.out.println("\n Ingrese el metodo a ejecutar");
                    break;
                case "MetodosdeOrdenamiento":
                    ejercicios.leerVector();
                    ejercicios.metburbuja();
                    ejercicios.metseleccion();
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "Cadenaiguala10":
                    ejercicios.metwhile();
                    System.out.println("Metodo finalizado");
                    System.out.println("Ingrese el metodo a ejecutar");

                    break;
                case"Crearunarraycon10elementosenteros":
                    ejercicios.leerVector();
                    ejercicios.vectorespar();
                    break;
                default:
                    System.out.println("Metodo no valido");
                    salir = false;
```

```
        }
    }
}

}
```

Elaboración: Propia

Ejecución:



Elaboración: Propia

- Mostrar 2 métodos de ordenación de vectores.
  - Puede ser burbuja el método por selección.

Solución:

Método para el vector:

```
private Integer vector[];
private Scanner leervec;
private Scanner leer;

public Ejercicios(){
    leervec = new Scanner(System.in);
    leer = new Scanner(System.in);
}
```

```
public void leerVector() {
    System.out.println("Ingrese tamaño del arreglo");
    int n = leervec.nextInt();
    vector = new Integer[n];

    for (int i =0;i<n;i++){
        System.out.printf("Ingrese las posiciones de los arreglos %d\n",i);
        int numleido= leervec.nextInt();
        vector[i]= numleido;
    }
}
```

Elaboración: Propia

Método burbuja:

```
public void metburbuja() {
    int aux;
    //int[]  vector;
    for(int i=0;i < vector.length; i++){
        for(int j=i+1;j<vector.length;j++){
            if(vector[i] > vector[j]){
                aux= vector[i];
                vector[i]=vector[j];
                vector[j]= aux;

            }
        }

    }
    System.out.println("Metodo de la burbuja: "+ Arrays.toString(vector));
}
```

Elaboración: Propia

Método selección:

```
public void metseleccion() {
    //int[]  vector = new int[]{55,66,44,11};

    for (int i = 0; i < vector.length - 1; i++){
        int min = i;
        for (int j = i + 1; j < vector.length; j++){

            if (vector[j] < vector[min]){
                min = j;
            }
        }
        if (i != min){
            int aux= vector[i];
            vector[i] = vector[min];
            vector[min] = aux;
        }
    }
    System.out.println("Metodo por Selección : " + Arrays.toString(vector));
}
```

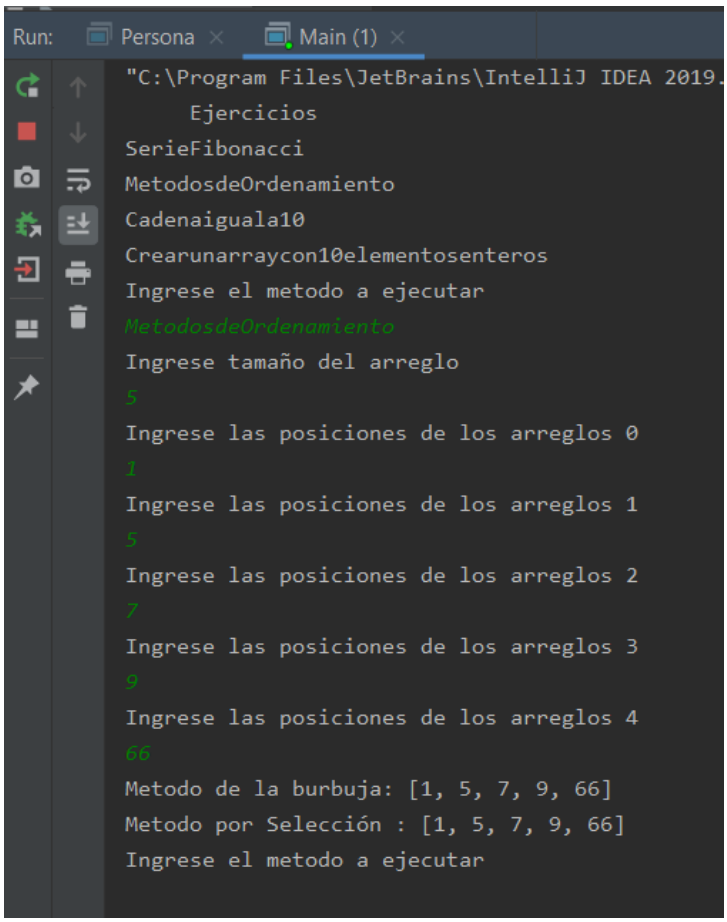
Elaboración: Propia

Main:

```
package EJERCICIOS;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        System.out.println("\t Ejercicios");
        System.out.println("SerieFibonacci");
        System.out.println("MetodosdeOrdenamiento");
        System.out.println("Cadenaiguala10");
        System.out.println("Crearunarraycon10elementosenteros");
        System.out.println("Ingrese el metodo a ejecutar");
        Scanner escribir = new Scanner(System.in);
        Scanner leer = new Scanner(System.in);
        Ejercicios ejercicios = new Ejercicios();
        boolean salir = true;
        String Metodo;

        while(salir) {
            Metodo = escribir.next();

            switch (Metodo) {
                case "SerieFibonacci":
                    System.out.println("Ingrese la cantidad de elementos de la serie Fibonacci: ");
                    int n = leer.nextInt();
                    ejercicios.serieFibonacci(n);
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "MetodosdeOrdenamiento":
                    ejercicios.leerVector();
                    ejercicios.metburbuja();
                    ejercicios.metseleccion();
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "Cadenaiguala10":
                    ejercicios.metwhile();
                    System.out.println("Metodo finalizado");
                    System.out.println("Ingrese el metodo a ejecutar");

                    break;
                case"Crearunarraycon10elementosenteros":
                    ejercicios.leerVector();
                    ejercicios.vectorespar();
                    break;
                default:
                    System.out.println("Metodo no valido");
                    salir = false;

            }
        }
    }
}
```

Elaboración: Propia

Ejecución:



Elaboración: Propia

- Usando while y el método **hasNext**() de la clase Scanner, leer N cadenas hasta encontrar una cadena que tenga una cantidad de caracteres igual a 10.
- Si la cadena ingresada tiene un número igual a 10 caracteres mostrar un mensaje indicando **"Cadena Encontrada"** y salir del while.

Solución:

Código:

```
public void metwhile() {

        System.out.println("Ingrese la cadena: ");
        while (leer.hasNextLine()){

            String cadena_tam = leer.nextLine();

            if (cadena_tam.length() == 10) {
                System.out.println("Cadena Encontrada");
                break;

            } else {
                System.out.println("Error cadena no es de tamaño 10");
            }
```

```
            System.out.println("Ingrese otra cadena:");
        }

}
```

Elaboración: Propia

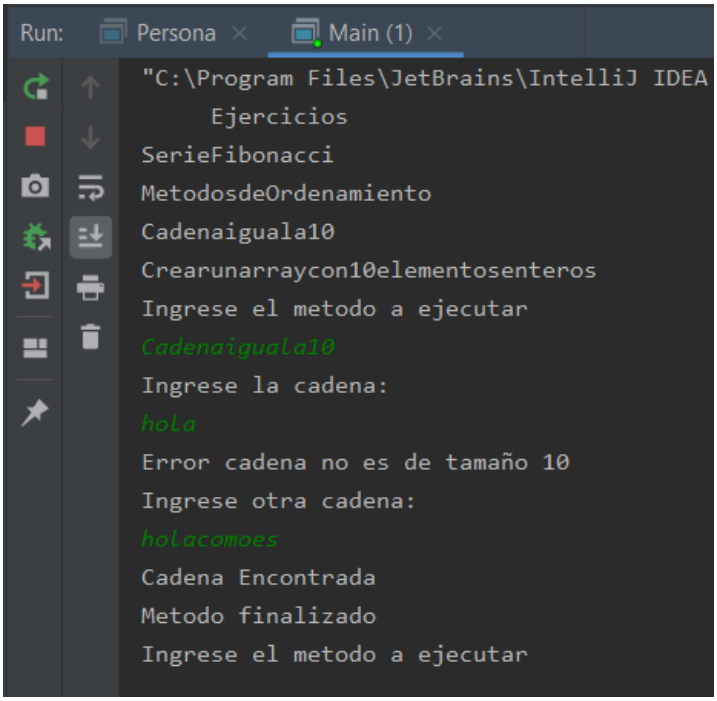
Main:

```
package EJERCICIOS;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        System.out.println("\t Ejercicios");
        System.out.println("SerieFibonacci");
        System.out.println("MetodosdeOrdenamiento");
        System.out.println("Cadenaiguala10");
        System.out.println("Crearunarraycon10elementosenteros");
        System.out.println("Ingrese el metodo a ejecutar");
        Scanner escribir = new Scanner(System.in);
        Scanner leer = new Scanner(System.in);
        Ejercicios ejercicios = new Ejercicios();
        boolean salir = true;
        String Metodo;

        while(salir) {
            Metodo = escribir.next();

            switch (Metodo) {
                case "SerieFibonacci":
                    System.out.println("Ingrese la cantidad de elementos de la serie Fibonacci: ");
                    int n = leer.nextInt();
                    ejercicios.serieFibonacci(n);
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "MetodosdeOrdenamiento":
                    ejercicios.leerVector();
                    ejercicios.metburbuja();
                    ejercicios.metseleccion();
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "Cadenaiguala10":
                    ejercicios.metwhile();
                    System.out.println("Metodo finalizado");
                    System.out.println("Ingrese el metodo a ejecutar");

                    break;
                case"Crearunarraycon10elementosenteros":
                    ejercicios.leerVector();
                    ejercicios.vectorespar();
                    break;
                default:
                    System.out.println("Metodo no valido");
                    salir = false;
```

```
        }
    }
}

}
```

Elaboración: Propia

Ejecución:



Elaboración: Propia

- Crear un array con 10 elementos enteros.
- Determinar cuántos elementos de ese array son **pares**.

Solución:

Código:

```
public void vectorespar(){
    int aux = 0;
    //int[]  vectoraux;
    for(int i=0;i < vector.length; i++){
            if(i % 2 == 0){
                aux++;

            }
    }
    System.out.println("La cantidad de numeros pares son: "+ aux);

}
```

Elaboración: Propia

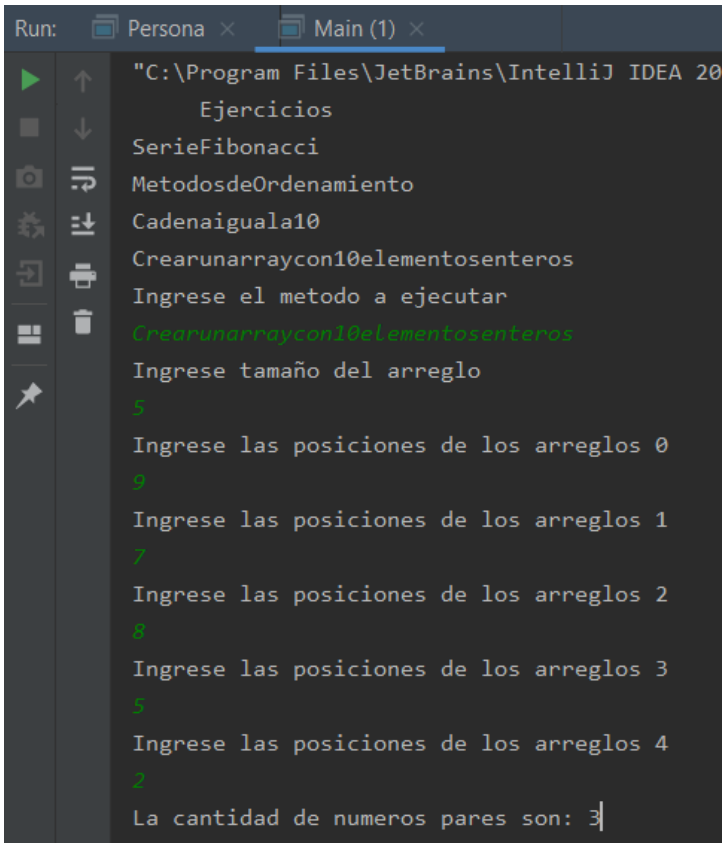
Main:

```
package EJERCICIOS;

import java.util.Scanner;

public class Main {
```

```
    public static void main(String[] args) {
        System.out.println("\t Ejercicios");
        System.out.println("SerieFibonacci");
        System.out.println("MetodosdeOrdenamiento");
        System.out.println("Cadenaiguala10");
        System.out.println("Crearunarraycon10elementosenteros");
        System.out.println("Ingrese el metodo a ejecutar");
        Scanner escribir = new Scanner(System.in);
        Scanner leer = new Scanner(System.in);
        Ejercicios ejercicios = new Ejercicios();
        boolean salir = true;
        String Metodo;

        while(salir) {
            Metodo = escribir.next();

            switch (Metodo) {
                case "SerieFibonacci":
                    System.out.println("Ingrese la cantidad de elementos de la serie Fibonacci: ");
                    int n = leer.nextInt();
                    ejercicios.serieFibonacci(n);
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "MetodosdeOrdenamiento":
                    ejercicios.leerVector();
                    ejercicios.metburbuja();
                    ejercicios.metseleccion();
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "Cadenaiguala10":
                    ejercicios.metwhile();
                    System.out.println("Metodo finalizado");
                    System.out.println("Ingrese el metodo a ejecutar");

                    break;
                case"Crearunarraycon10elementosenteros":
                    ejercicios.leerVector();
                    ejercicios.vectorespar();
                    break;
                default:
                    System.out.println("Metodo no valido");
                    salir = false;

            }
        }
    }

}
```

Elaboración: Propia

Ejecución:



Elaboración: Propia

Nota. Todos los valores deben ser leídos por teclado (Para todos los ejercicios).

Parte Práctica.

I. Manejo de Interfaces: **LeerInteface.java**

Esta **interface** tiene declarado una instancia del objeto **Scanner(System.in)**. Esta **interfaz** debe ser implementada en la **clase Empleado.**

El nombre de esta variable es LEER.

- **Scanner LEER = new Scanner(System.in);**

Solución:

```
package Manejo_de_interfaces_practica_hito2;

import java.util.Scanner;

public interface LeerInteface {
   Scanner LEER = new Scanner(System.in);
}
```

Elaboración: Propia

II. Manejo de Herencia: **Empleado.java**

Esta es la clase **padre** del cual heredan otras subclases. Esta clase implementa la interfaz **LeerInterface.java.**

Solución:

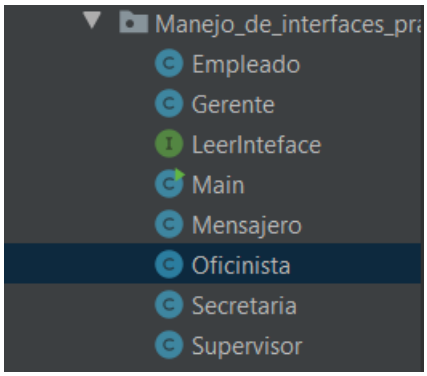
```
package Manejo_de_interfaces_practica_hito2;

public class Empleado implements LeerInteface {

   public String primerNombre;
   public String primerApellido;
   public int ciNumero;
   public String ciExtension;

   public void Leer(){
      //System.out.println("Ingrese numero de empleados");
      //int n = LEER.nextInt();
      //for(int i=0;i<n;i++) {

         System.out.printf("Ingrese Primer Nombre: ");
         primerNombre = LEER.next();
         System.out.printf("Ingrese Primer Apellido : ");
         primerApellido = LEER.next();
         System.out.printf("Ingrese Numero CI :");
         ciNumero = LEER.nextInt();
      System.out.printf("Ingrese Extension CI : ");
      ciExtension = LEER.next();

      //}

   }
   public void Mostrar(){

      System.out.printf("Nombre : %s, Apellido: %s, Numero CI: %d, Extension CI: %s
",this.primerNombre,this.primerApellido,this.ciNumero,this.ciExtension);
   }

}
```

Elaboración: Propia

III. Manejo de Clases: **Gerente.java, Supervisor.java y Oficinista.java**

Todas estas clases heredan de la clase **Empleado.java**

Solución:



Gerente:

```
package Manejo_de_interfaces_practica_hito2;

public class Gerente extends Empleado {

   public int nrold;
   public String codArea;
   public int sueldoBasico;


   public void Leer(){
      super.Leer();

      System.out.printf("Ingrese nro de ID: ");
      nrold = LEER.nextInt();
      System.out.printf("Ingrese Codigo de Area: ");
      codArea = LEER.next();
      System.out.printf("Ingrese Sueldo Basico: ");
      sueldoBasico = LEER.nextInt();

   }
   public void Mostrar(){
      super.Mostrar();
      System.out.println("\t");
      System.out.printf("nroID : %d, codArea: %s, sdoBasico: %d
",this.nrold,this.codArea,this.sueldoBasico);
   }

}
```

Elaboración: Propia

Supervisor:

```
package Manejo_de_interfaces_practica_hito2;

public class Supervisor extends Empleado {
```

```
    public int sueldo;
    public int antiguedad;

    public void Leer(){
        super.Leer();
        System.out.printf("Ingrese Sueldo: ");
        sueldo = LEER.nextInt();
        System.out.printf("Ingrese la Antiguedad : ");
        antiguedad = LEER.nextInt();
    }
    public void Mostrar(){
        super.Mostrar();
        System.out.printf("Sueldo: %d, Antiguedad: %d",this.sueldo,this.antiguedad);
    }
}
```

Elaboración: Propia

Oficinista:

```
package Manejo_de_interfaces_practica_hito2;

public class Oficinista extends Empleado {
    public String codArea;
    public int sueldoBasico;


    public void Leer(){
        super.Leer();
        System.out.printf("Ingresar Codigo del Area: ");
        codArea = LEER.next();
        System.out.printf("Ingresar Sueldo Basico: ");
        sueldoBasico = LEER.nextInt();

    }
    public void MostrarOF(){
        super.Mostrar();
        System.out.printf("CodArea: %s,Sueldo Basico: %d",this.codArea,this.sueldoBasico);
    }
}
```

Elaboración: Propia

IV. Manejo de Interfaces, Herencia y Clases: **Secretaria.java y Mensajero.java**

Estas clases heredan de la clase **Oficinista.java.**

Secretaria:

```
package Manejo_de_interfaces_practica_hito2;

public class Secretaria extends Oficinista {
   public String nombreArea;

   public void Leer() {
      super.Leer();
      System.out.printf("Ingrese Nombre del Area: ");
      nombreArea = LEER.next();
   }
   public void Mostrar(){
      super.MostrarOF();
      System.out.printf(", Nombre del Area: %s",this.nombreArea);

   }
}
```

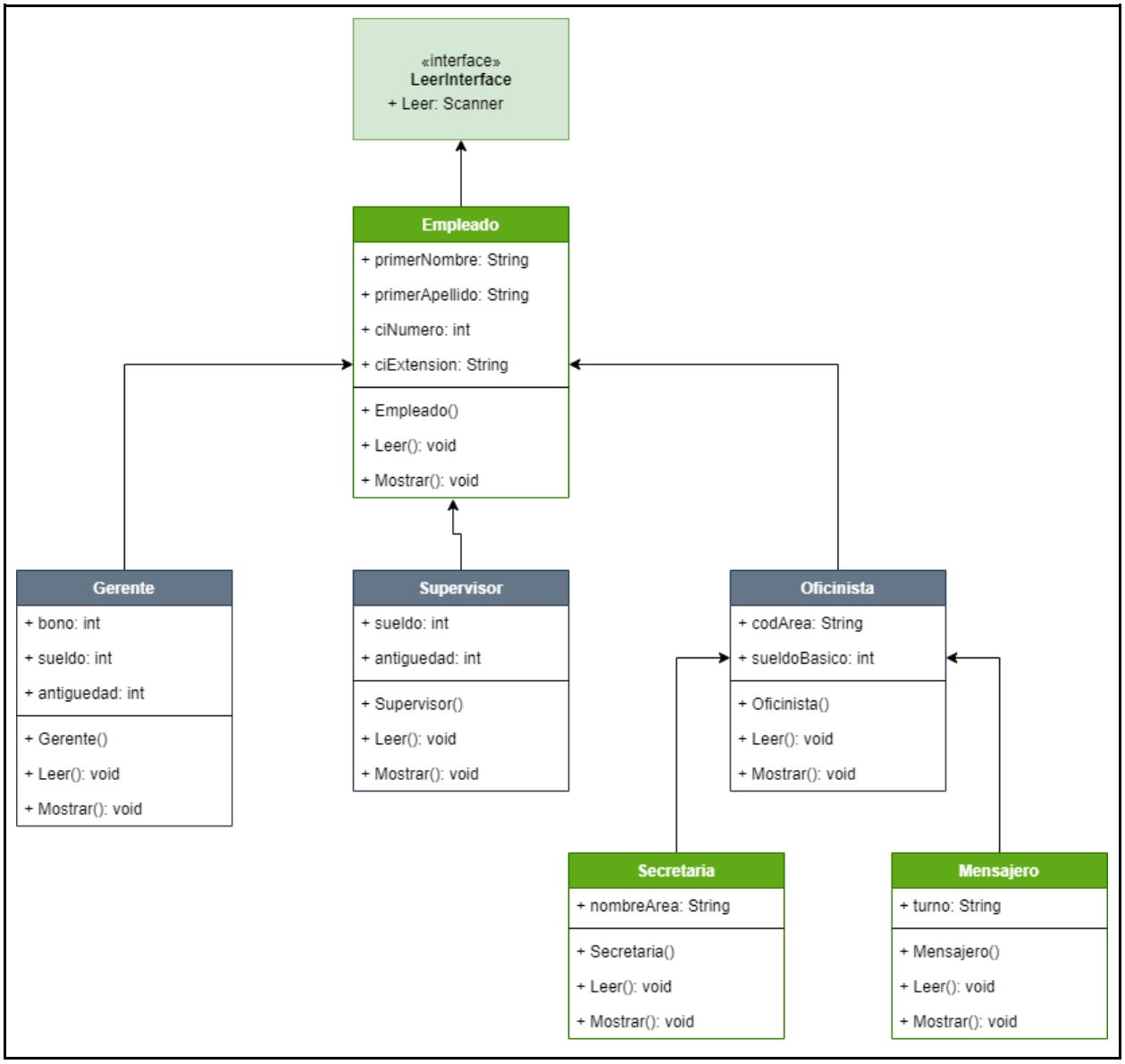
Elaboración: Propia

Mensajero:

```
package Manejo_de_interfaces_practica_hito2;

public class Supervisor extends Empleado {
   public int sueldo;
   public int antiguedad;

   public void Leer(){
      super.Leer();
      System.out.printf("Ingrese Sueldo: ");
      sueldo = LEER.nextInt();
      System.out.printf("Ingrese la Antiguedad : ");
      antiguedad = LEER.nextInt();
   }
   public void Mostrar(){
      super.Mostrar();
      System.out.printf("Sueldo: %d, Antiguedad: %d",this.sueldo,this.antiguedad);
   }
}
```

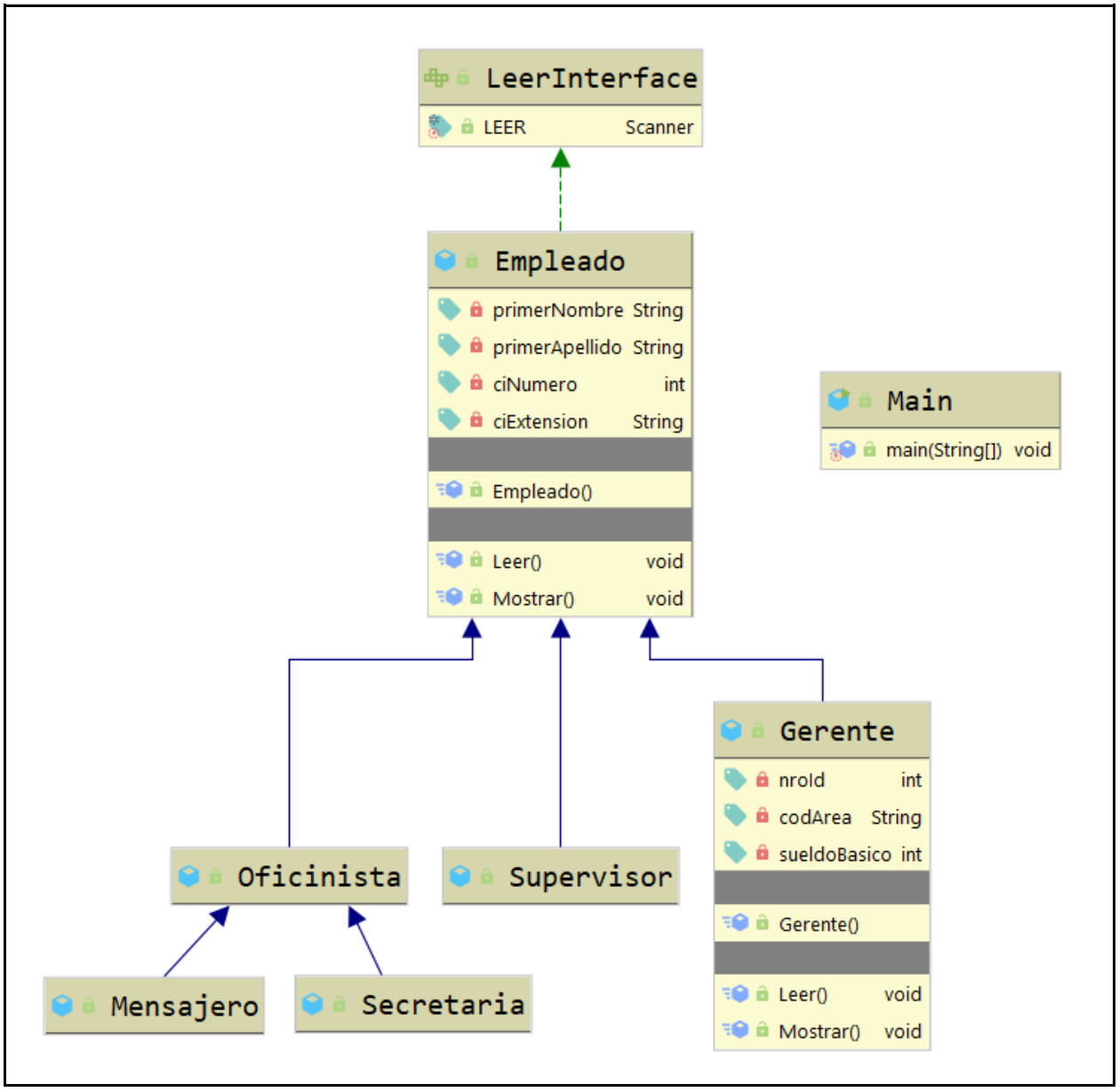
Elaboración: Propia

Para dar solución a la parte práctica deberá de basarse en el siguiente diseño.



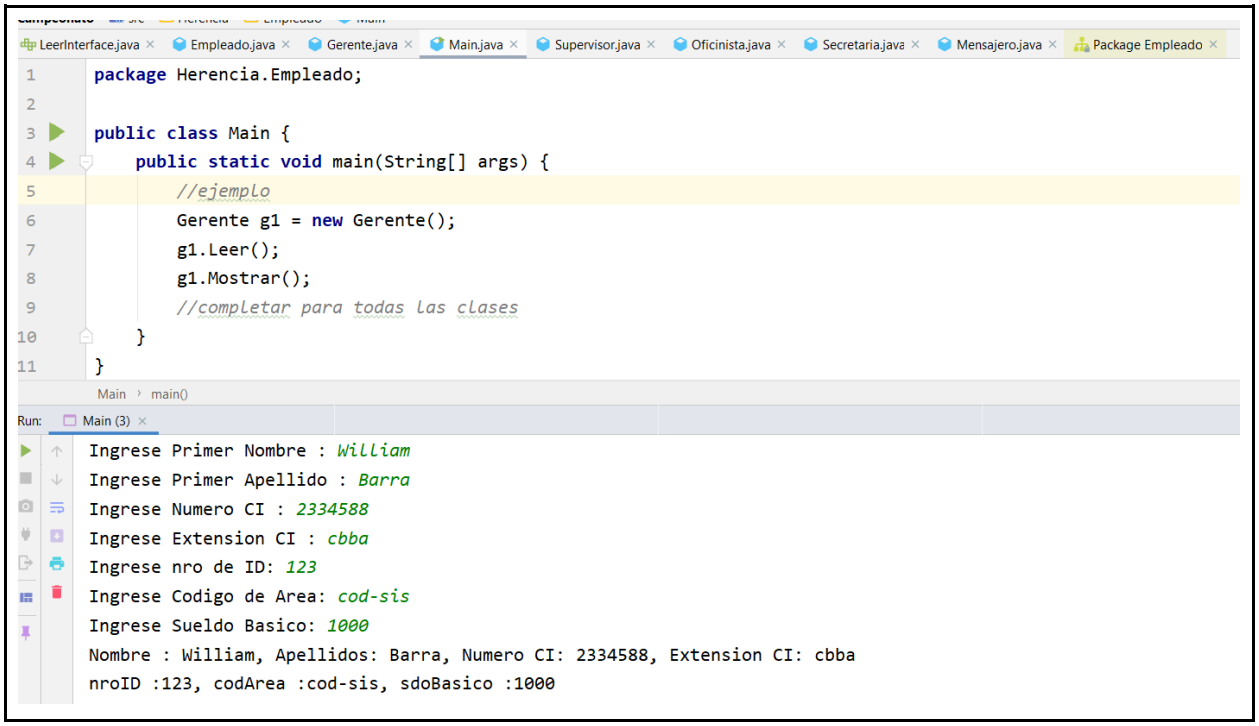
Posible diseño. (debe de completar todo el diseño)

Note que en todas las clases existe los métodos **Leer y Mostrar**, para poder leer **cadenas,** enteros, etc. Debe de utilizar la variable **LEER** que se encuentra declarada en la **Interface**.



Comportamiento esperado (Ejemplo class:Oficinista.java)

La solución planteada deberá de mostrar los datos de manera similar que el de la imagen.



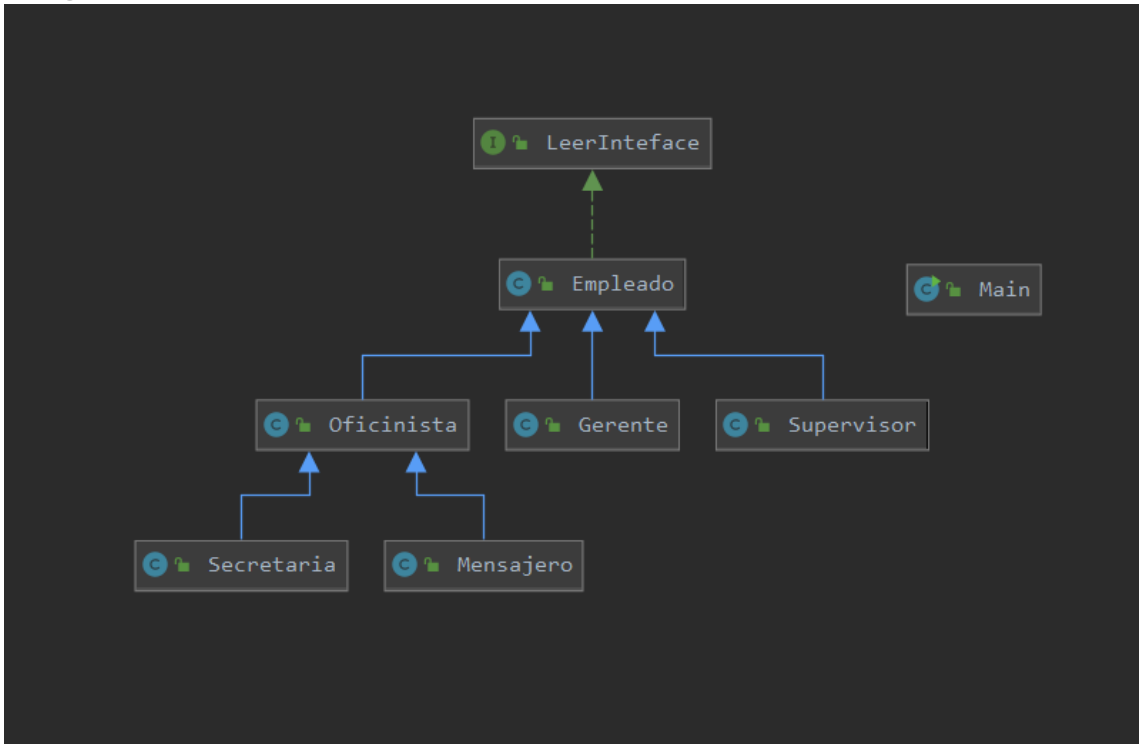
Main:

```
package Manejo_de_interfaces_practica_hito2;

import EJERCICIOS.Ejercicios;

import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      System.out.println("\t LEERINTERFACES");
      //System.out.println("Empleado");
      System.out.println("Gerente");
      System.out.println("Mensajero");
      System.out.println("Oficinista");
      System.out.println("Secretaria");
      System.out.println("Supervisor");
      System.out.println("Ingrese el empleado que dese ingresar");
      Scanner escribir = new Scanner(System.in);
     // Empleado emp = new Empleado();
      Gerente g1 = new Gerente();
      Mensajero men = new Mensajero();
      Oficinista of1 = new Oficinista();
      Secretaria sec = new Secretaria();
      Supervisor sup = new Supervisor();

      boolean salir = true;
      String Metodo;

      while (salir) {
```

```
        Metodo = escribir.next();

        switch (Metodo) {
          case "Gerente":
            g1.Leer();
            g1.Mostrar();
            System.out.println("\nIngresado");
            System.out.println("Ingresar empleado a ingresar");
            break;
          case "Mensajero":
            men.Leer();
            men.Mostrar();
            System.out.println("\nIngresado");
            System.out.println("Ingresar empleado a ingresar");

            break;
          case "Oficinista":
            of1.Leer();
            of1.MostrarOF();
            System.out.println("\nIngresado");
            System.out.println("Ingresar empleado a ingresar");
            break;
          case "Secretaria":
            sec.Leer();
            sec.Mostrar();
            System.out.println("\nIngresado");
            System.out.println("Ingresar empleado a ingresar");

            break;
          case "Supervisor":
            sup.Leer();
            sup.Mostrar();
            System.out.println("\nIngresado");
            System.out.println("Ingresar empleado a ingresar");
            break;
          default:
            System.out.println("\nEmpleado no valido");
            salir = false;

        }
      }
    }
}
```

Elaboración: Propia

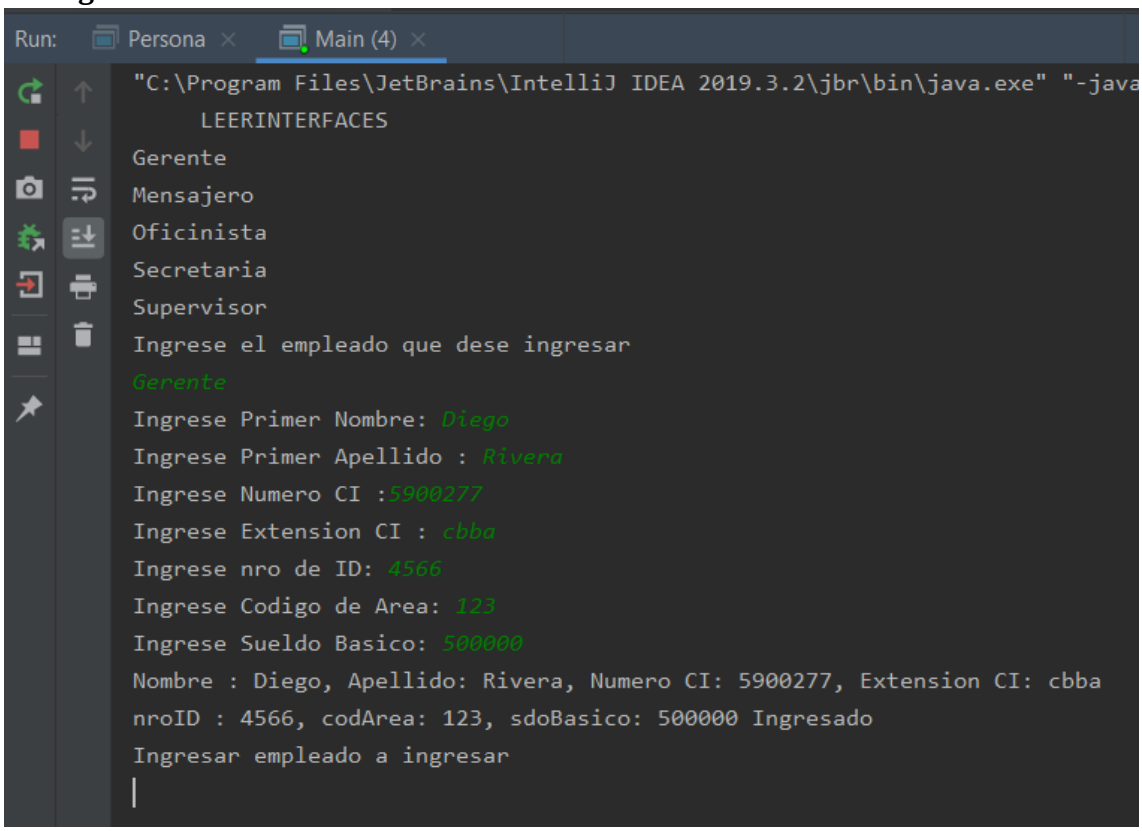
Diagrama:



Elaboración: Propia

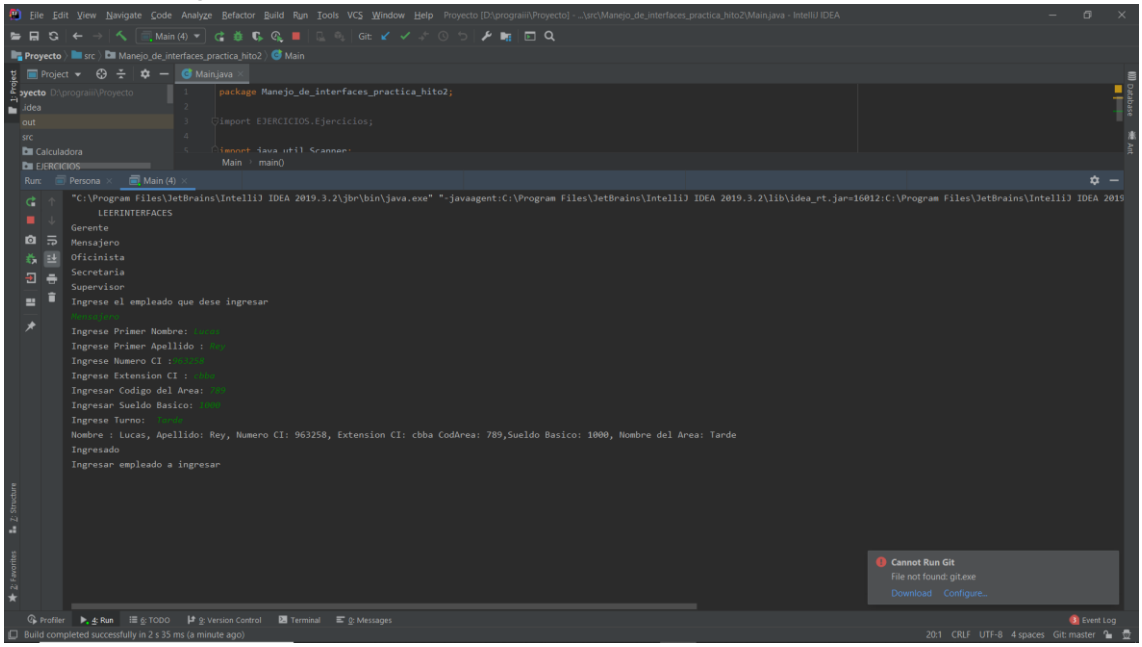
Ejecución:

Para gerente:



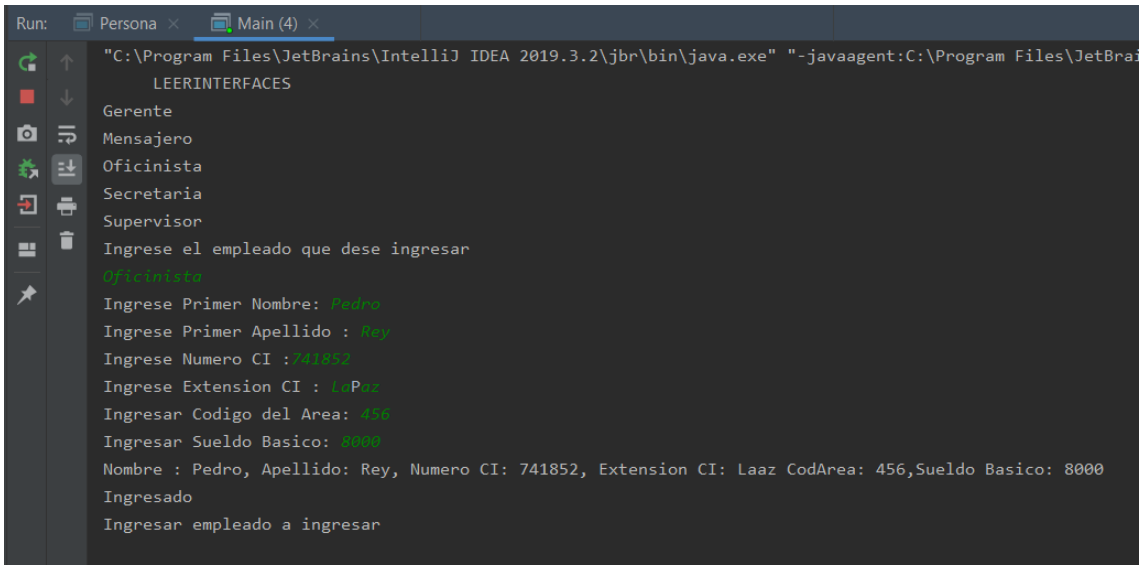
Elaboración: Propia

Para Mensajero:



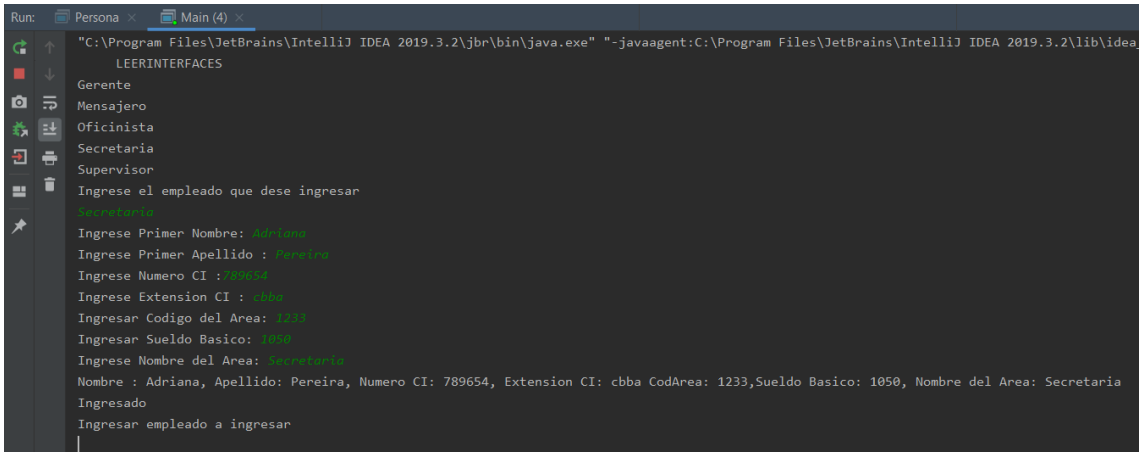
Elaboración: Propia

Para Oficinista:



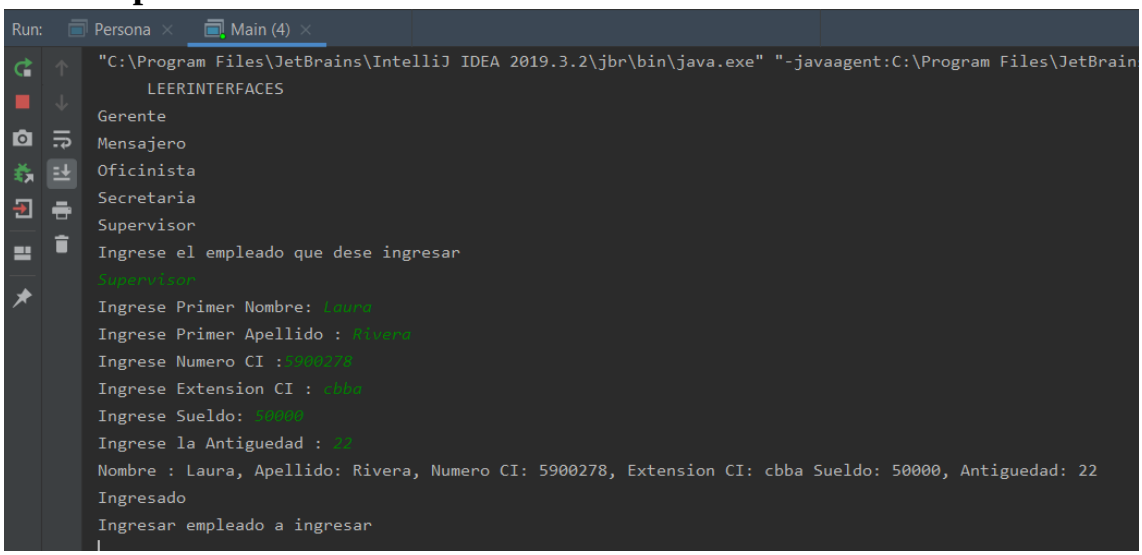
Elaboración: Propia

Para Secretaria:



Elaboración: Propia

Para Supervisor:



Elaboración: Propia