

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:

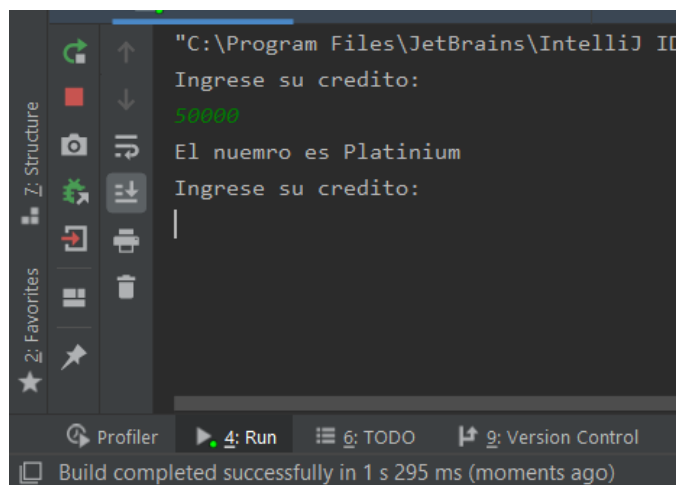
```
System.out.println("Ingrese su credito:");
while(leer.hasNextInt()) {

    int credit_number = leer.nextInt();

    if (credit_number <= 50000) {
        System.out.println("El nuemro es Platinum");
    }
    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:



```
"C:\Program Files\JetBrains\IntelliJ ID
Ingrese su credito:
50000
El nuemro es Platinum
Ingrese su credito:
|
```

Profiler 4: Run 6: TODO 9: Version Control

Build completed successfully in 1 s 295 ms (moments ago)

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:

```
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; }
}
```

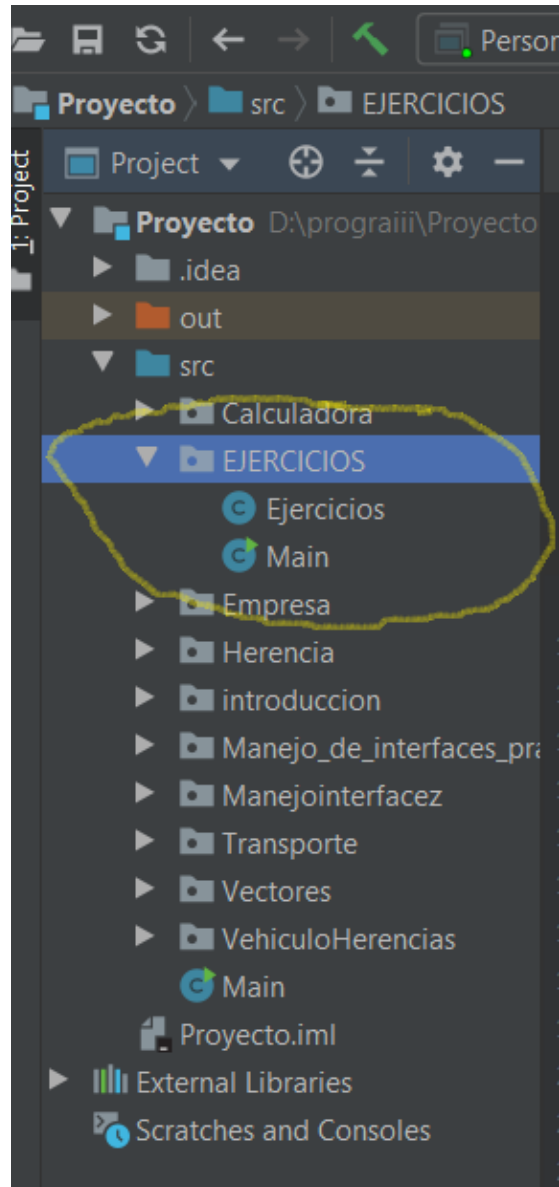
Elaboración: Propia

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("Cadenaigual a 10");  
        System.out.println("Crear un array con 10 elementos enteros");  
        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 "Cadenaigual a 10":  
                    ejercicios.metwhile();  
                    System.out.println("Metodo finalizado");  
                    System.out.println("Ingrese el metodo a ejecutar");  
  
                    break;  
                case "Crear un array con 10 elementos enteros":  
                    ejercicios.leerVector();  
                    ejercicios.vectorespar();  
                    break;  
                default:  
                    System.out.println("Metodo no valido");  
                    salir = false;  
            }  
        }  
    }  
}
```

```

    }
  }
}

```

Elaboración: Propia

Ejecución:

```

"C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.2\jbr\bin\j
Ejercicios
SerieFibonacci
MetodosdeOrdenamiento
Cadenaigualal0
Crearunarraycon10elementosenteros
Ingrese el metodo a ejecutar

SerieFibonacci
Ingrese la cantidad de elementos de la serie Fibonacci:
13
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
Ingrese el metodo a ejecutar
|

```

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 metsseleccion() {
    //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("Cadenaigual a 10");
        System.out.println("Crear un array con 10 elementos enteros");
        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 "Cadenaigual a 10":
                    ejercicios.metwhile();
                    System.out.println("Metodo finalizado");
                    System.out.println("Ingrese el metodo a ejecutar");

                    break;
                case "Crear un array con 10 elementos enteros":
                    ejercicios.leerVector();
                    ejercicios.vectorespar();
                    break;
                default:
                    System.out.println("Metodo no valido");
                    salir = false;
            }
        }
    }
}
```


Elaboración: Propia

Ejecución:

```
Run: Persona x Main (1) x
"C:\Program Files\JetBrains\IntelliJ IDEA 2019.
Ejercicios
SerieFibonacci
MetodosdeOrdenamiento
Cadenaigualal0
Crearunarraycon10elementosenteros
Ingrese el metodo a ejecutar
MetodosdeOrdenamiento
Ingrese tamaño del arreglo
5
Ingrese las posiciones de los arreglos 0
1
Ingrese las posiciones de los arreglos 1
5
Ingrese las posiciones de los arreglos 2
7
Ingrese las posiciones de los arreglos 3
9
Ingrese las posiciones de los arreglos 4
66
Metodo de la burbuja: [1, 5, 7, 9, 66]
Metodo por Selección : [1, 5, 7, 9, 66]
Ingrese el metodo a ejecutar
```

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("Cadenaigual a 10");
        System.out.println("Crear un array con 10 elementos enteros");
        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.met Burbuja();
                    ejercicios.met seleccion();
                    System.out.println("Ingrese el metodo a ejecutar");
                    break;
                case "Cadena igual a 10":
                    ejercicios.met while();
                    System.out.println("Metodo finalizado");
                    System.out.println("Ingrese el metodo a ejecutar");

                    break;
                case "Crear un array con 10 elementos enteros":
                    ejercicios.leerVector();
                    ejercicios.vectores par();
                    break;
                default:
                    System.out.println("Metodo no valido");
                    salir = false;
            }
        }
    }
}
```

```

    }
  }
}

```

Elaboración: Propia

Ejecución:

```

Run: Persona x Main (1) x
"C:\Program Files\JetBrains\IntelliJ IDEA
  Ejercicios
  SerieFibonacci
  MetodosdeOrdenamiento
  Cadenaigualala10
  Crearunarraycon10elementosenteros
  Ingrese el metodo a ejecutar
  Cadenaigualala10
  Ingrese la cadena:
  hola
  Error cadena no es de tamaño 10
  Ingrese otra cadena:
  holacomo es
  Cadena Encontrada
  Metodo finalizado
  Ingrese el metodo a ejecutar

```

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("Cadenaigual10");
    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();

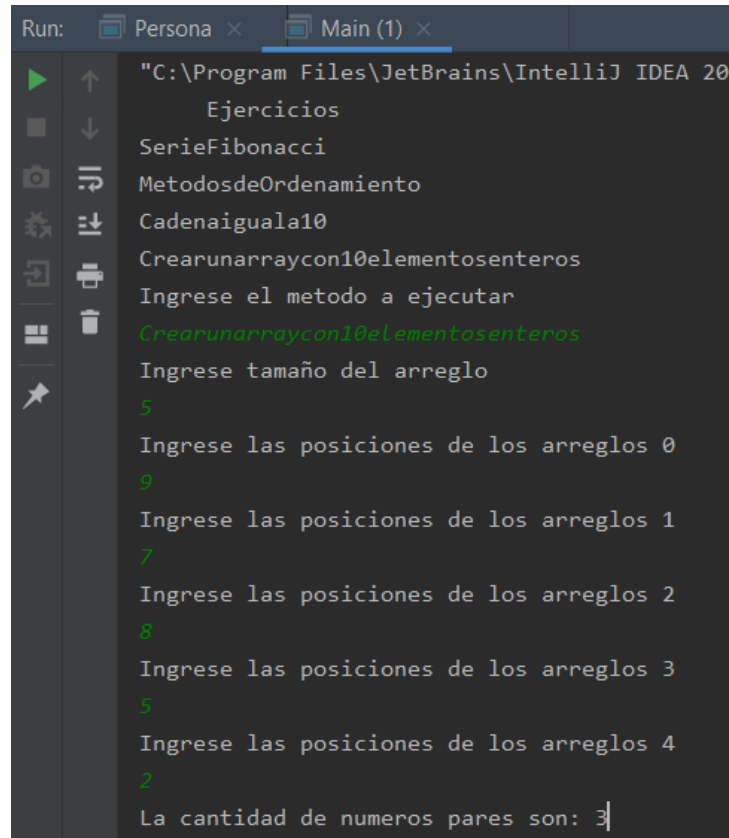
        switch (Metodo) {
            case "SerieFibonacci":
                System.out.println("Ingrese la cantidad de elementos de la serie Fibonacci: ");
                int n = leer.nextInt();
                ejercicios.serieFibonacci(n);
                System.out.println("Ingrese el metodo a ejecutar");
                break;
            case "MetodosdeOrdenamiento":
                ejercicios.leerVector();
                ejercicios.metburbuja();
                ejercicios.metsleccion();
                System.out.println("Ingrese el metodo a ejecutar");
                break;
            case "Cadenaigual10":
                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:



```
Run: Persona x Main (1) x
"C:\Program Files\JetBrains\IntelliJ IDEA 20
Ejercicios
SerieFibonacci
MetodosdeOrdenamiento
Cadenaigualal0
Crearunarraycon10elementosenteros
Ingrese el metodo a ejecutar
Crearunarraycon10elementosenteros
Ingrese tamaño del arreglo
5
Ingrese las posiciones de los arreglos 0
9
Ingrese las posiciones de los arreglos 1
7
Ingrese las posiciones de los arreglos 2
8
Ingrese las posiciones de los arreglos 3
5
Ingrese las posiciones de los arreglos 4
2
La cantidad de numeros pares son: 3
```

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(){

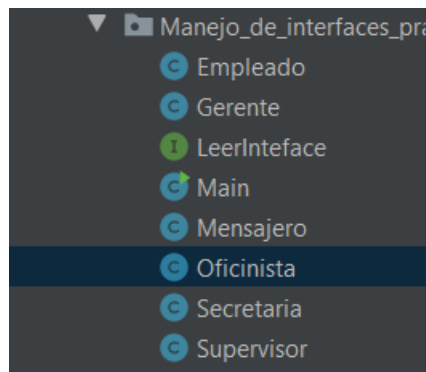
        System.out.printf("Nombre : %s, Apellido: %s, Numero CI: %d, Extension CI: %s",this.primerNombre,this.primerApellido,this.ciNumero,this.ciExtension);
    }
}
```

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 nroId;
    public String codArea;
    public int sueldoBasico;

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

        System.out.printf("Ingrese nro de ID: ");
        nroId = LEER.nextInt();
        System.out.printf("IngreseCodigo 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.nroId,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 Antigüedad : ");
    antiguedad = LEER.nextInt();
}
public void Mostrar(){
    super.Mostrar();
    System.out.printf("Sueldo: %d, Antigüedad: %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 Código del Área: ");
        codArea = LEER.next();
        System.out.printf("Ingresar Sueldo Básico: ");
        sueldoBasico = LEER.nextInt();
    }
    public void MostrarOF(){
        super.Mostrar();
        System.out.printf("CodArea: %s,Sueldo Básico: %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:

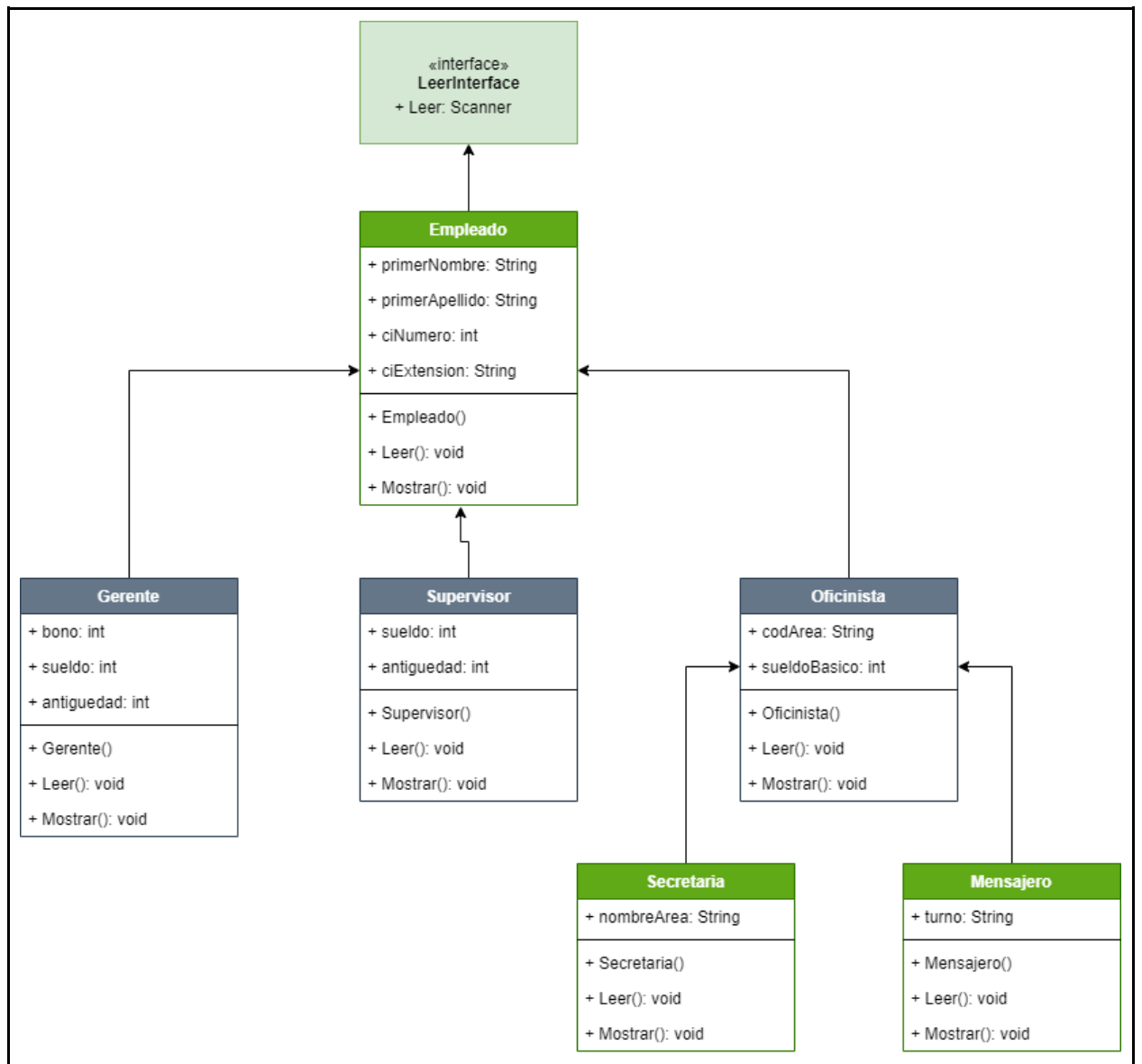
```
package Manejo_de_interfaces_practica_hito2;

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

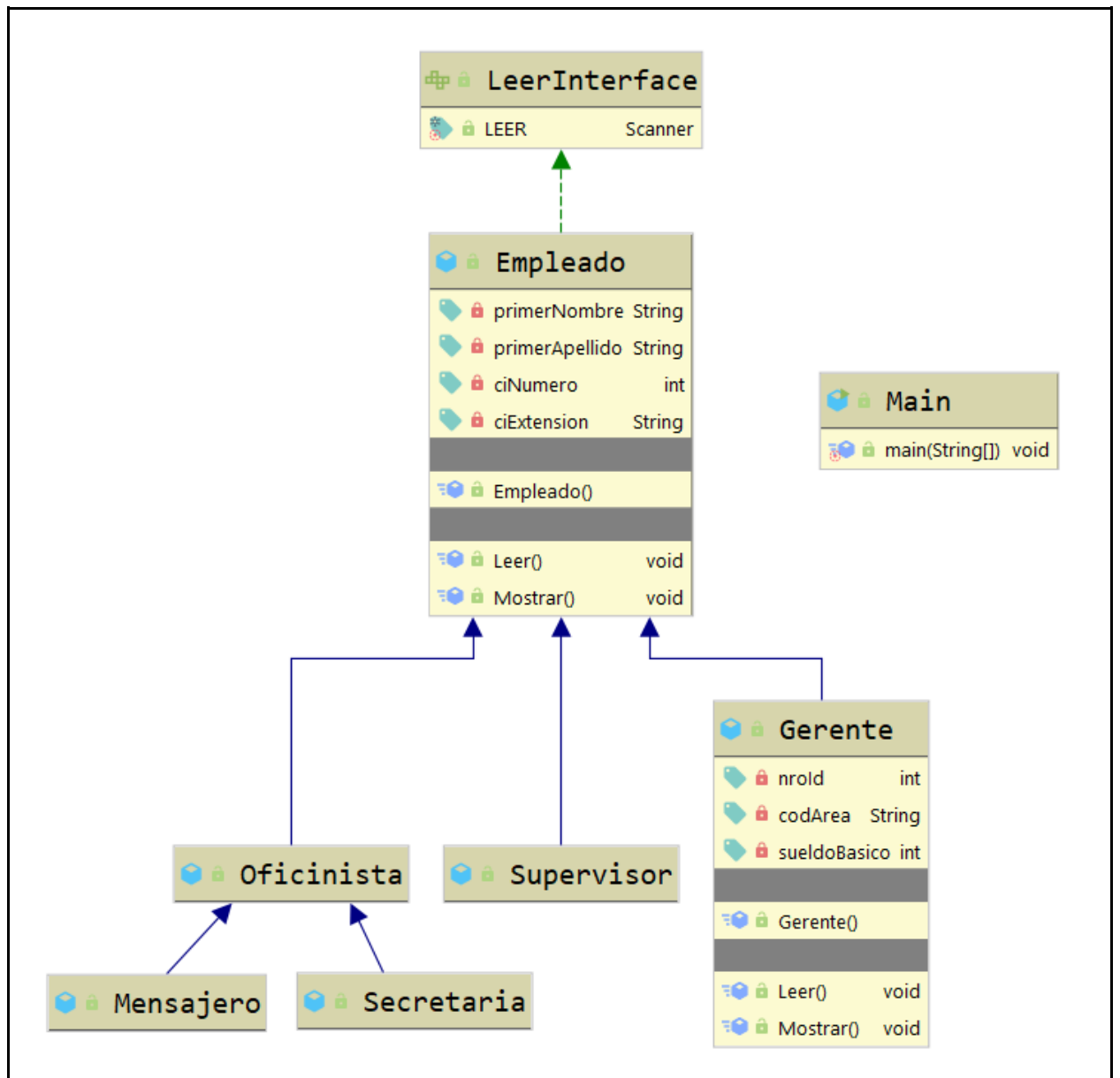
Elaboración: Propia

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



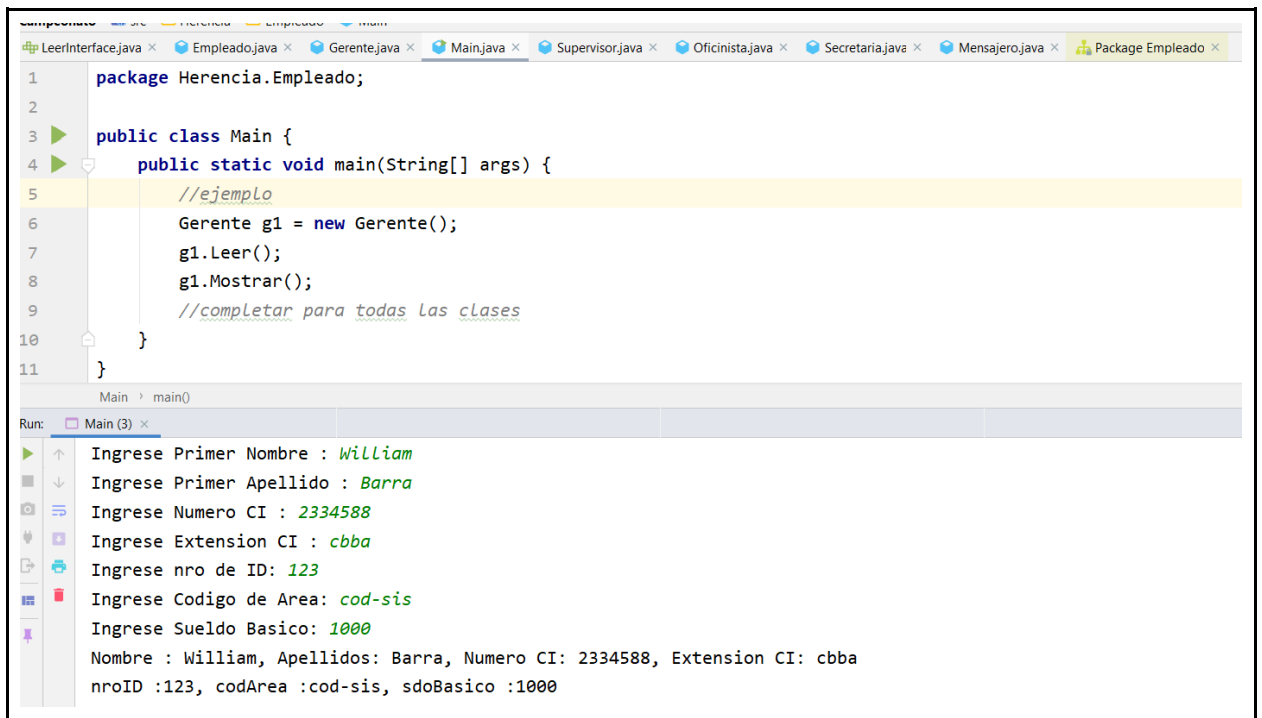
Posible diseño. (debe de completar todo el diseño)

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.



The screenshot shows an IDE with several open files: LeerInterface.java, Empleado.java, Gerente.java, Main.java, Supervisor.java, Oficinista.java, Secretaria.java, Mensajero.java, and Package Empleado. The Main.java file is active, showing the following code:

```
1 package Herencia.Empleado;
2
3 public class Main {
4     public static void main(String[] args) {
5         //ejemplo
6         Gerente g1 = new Gerente();
7         g1.Leer();
8         g1.Mostrar();
9         //completar para todas las clases
10    }
11 }
```

Below the code editor, the Run window shows the output of the program:

```
Run: Main (3) x
Ingrese Primer Nombre : William
Ingrese Primer Apellido : Barra
Ingrese Numero CI : 2334588
Ingrese Extension CI : cbba
Ingrese nro de ID: 123
Ingrese Codigo de Area: cod-sis
Ingrese Suelo Basico: 1000
Nombre : William, Apellidos: Barra, Numero CI: 2334588, Extension CI: cbba
nroID :123, codArea :cod-sis, sdoBasico :1000
```

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");

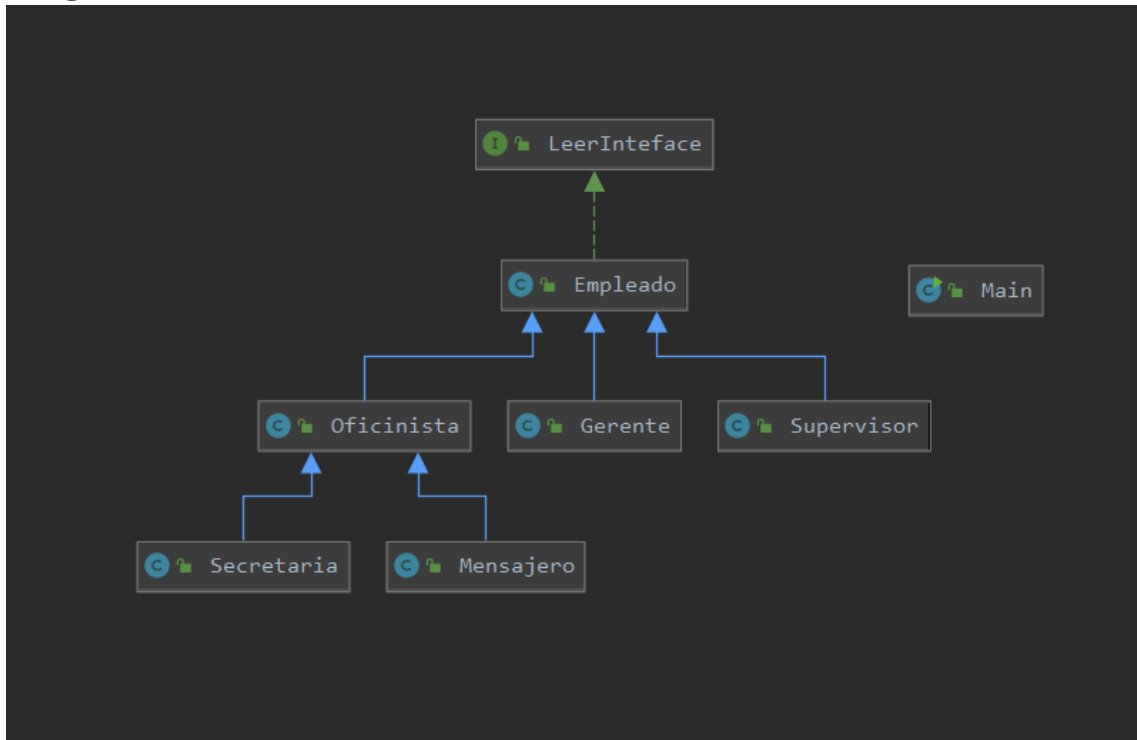
        break;
    case "Oficinista":
        of1.Leer();
        of1.MostrarOF();
        System.out.println("\nIngresado");
        System.out.println("Ingresar empleado a ingresar");
        break;
    case "Secretaria":
        sec.Leer();
        sec.Mostrar();
        System.out.println("\nIngresado");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

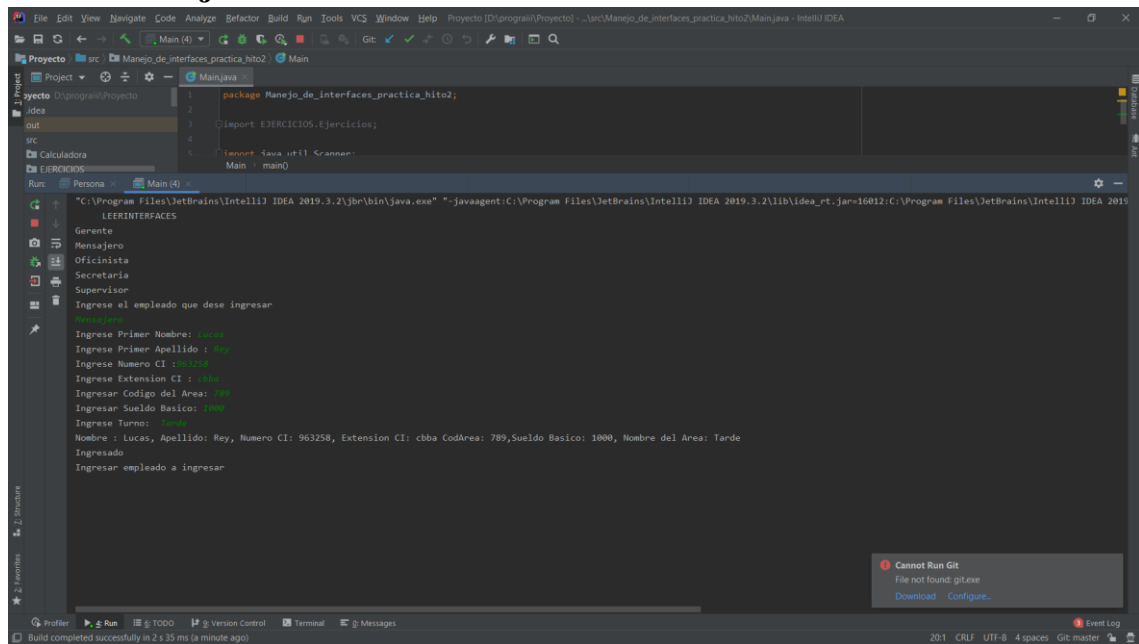
Ejecución:

Para gerente:

```
Run: Persona x Main (4) x
"C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.2\jbr\bin\java.exe" "-java
LEERINTERFACES
Gerente
Mensajero
Oficinista
Secretaria
Supervisor
Ingrese el empleado que dese ingresar
Gerente
Ingrese Primer Nombre: Diego
Ingrese Primer Apellido : Rivera
Ingrese Numero CI :5900277
Ingrese Extension CI : cbba
Ingrese nro de ID: 4566
IngreseCodigo de Area: 123
Ingrese Sueldo Basico: 500000
Nombre : Diego, Apellido: Rivera, Numero CI: 5900277, Extension CI: cbba
nroID : 4566, codArea: 123, sdoBasico: 500000 Ingresado
Ingresar empleado a ingresar
|
```

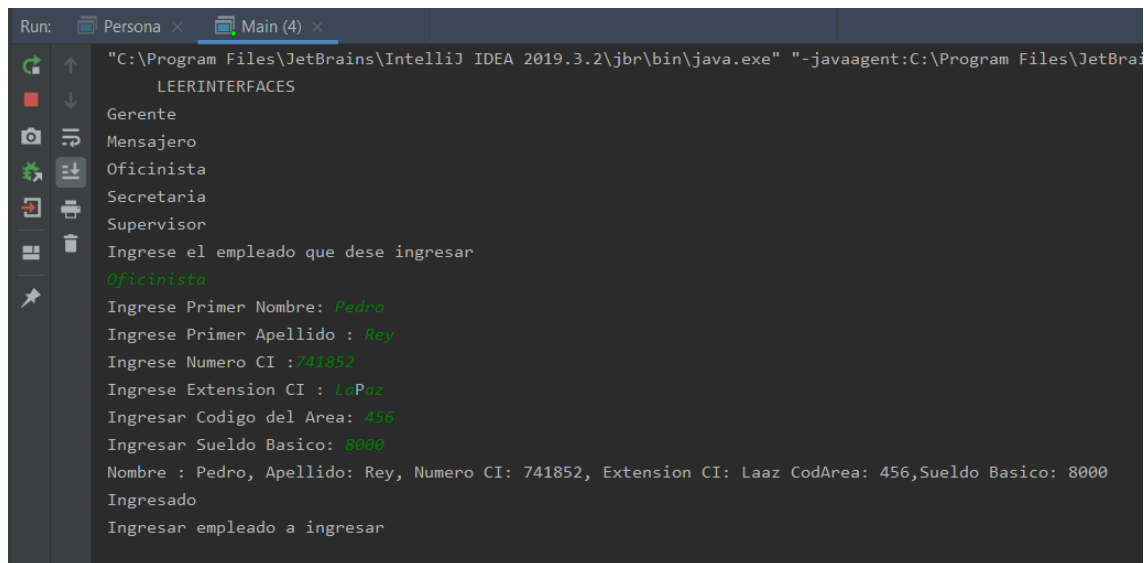
Elaboración: Propia

Para Mensajero:



Elaboración: Propia

Para Oficinista:



Elaboración: Propia

Para Secretaria:

```
Run: Persona x Main (4) x
"C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.2\jbr\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.2\lib\idea
LEERINTERFACES
Gerente
Mensajero
Oficinista
Secretaria
Supervisor
Ingrese el empleado que dese ingresar
Secretaria
Ingrese Primer Nombre: Adriana
Ingrese Primer Apellido : Pereira
Ingrese Numero CI : 789654
Ingrese Extension CI : cbba
Ingresar Codigo del Area: 1233
Ingresar Sueldo Basico: 1050
Ingrese Nombre del Area: Secretaria
Nombre : Adriana, Apellido: Pereira, Numero CI: 789654, Extension CI: cbba CodArea: 1233,Sueldo Basico: 1050, Nombre del Area: Secretaria
Ingresado
Ingresar empleado a ingresar
```

Elaboración: Propia

Para Supervisor:

```
Run: Persona x Main (4) x
"C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.2\jbr\bin\java.exe" "-javaagent:C:\Program Files\JetBrain
LEERINTERFACES
Gerente
Mensajero
Oficinista
Secretaria
Supervisor
Ingrese el empleado que dese ingresar
Supervisor
Ingrese Primer Nombre: Laura
Ingrese Primer Apellido : Rivera
Ingrese Numero CI : 5900278
Ingrese Extension CI : cbba
Ingrese Sueldo: 50000
Ingrese la Antigüedad : 22
Nombre : Laura, Apellido: Rivera, Numero CI: 5900278, Extension CI: cbba Sueldo: 50000, Antigüedad: 22
Ingresado
Ingresar empleado a ingresar
```

Elaboración: Propia