

CORTAZAR DE LA CRUZ MANUEL GIOVANNI

Matrícula: 2193001368

Carrera: Ing. Computación

Docente: Dra. MARICELA CLAUDIA BRAVO CONTRERAS

Descripción breve

proyecto final

Programación Orientada a Objetos

Se desarrolla un sistema para la gestión de empleados, los cuales pueden clasificarse en jefe, secretaria, intendente y empleado general. Dicho sistema de gestión deberá ser capaz de realizar el registro de empleados, realizar la eliminación de un empleado proporcionando su clave, mostrar el listado de todos los empleados, y realizar la serialización y deserialización indicando desde consola el nombre del archivo.

Alumno: CORTAZAR DE LA CRUZ MANUEL GIOVANNI

Matrícula:2193001368

**CODIGO FUENTE**

**GestionEmpleados.java**

import java.io.File;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.Scanner;

public class GestionEmpleados

{

public static void main(String[] args)

{

UtileriasArchivos util = new UtileriasArchivos();

List<Empleado> listaEmpleados = new ArrayList<Empleado>();

String validar = null;

int opcion = 0;

System.out.println(" ================================= ");

do

{

System.out.println(" M E N U P R I N C I P A L ");

System.out.println("1. Deserializar Empleados");

System.out.println("2. Registrar Empleado");

System.out.println("3. Eliminar Empleado");

System.out.println("4. Listado de Empleados");

System.out.println("5. Serializar");

System.out.println("SALIR. Cualquier otro numero");

System.out.println("Escribe el numero de la opcion: ");

validar = leeCadena();

while(validarNumero(validar) == false){

System.out.println("Escribe el numero de la opcion: ");

validar = leeCadena();

}

opcion = ConNumero(validar);

switch(opcion)

{

case 1:

System.out.println("Nombre del archivo para deserializar");

String arch = leeCadena();

if(existe(arch))

listaEmpleados = util.deserializaListaEmpleados(arch);

break;

case 2:

System.out.println("Registro de nuevo Empleado");

System.out.println("Nombre: ");

//verificar si nombre ya existe

String nombre = leeCadena();

while(util.verNombre(nombre, listaEmpleados) == true){

System.out.println("ingrese otra nombre");

nombre = leeCadena();

}

System.out.println("Direccion: ");

String direccion = leeCadena();

System.out.println("Numero de SSN: ");

validar = leeCadena();

int ssn = 0;

while(validarNumero(validar) == false){

System.out.println("Escribe el numero de SSN: ");

validar = leeCadena();

}

ssn = ConNumero(validar);

System.out.println("Numero de clave: ");

validar = leeCadena();

int clave = 0;

while(validarNumero(validar) == false){

System.out.println("Escribe el numero de la clave: ");

validar = leeCadena();

}

clave = ConNumero(validar);

//verificar si clave ya existe

while(util.verClave(clave, listaEmpleados) == true){

System.out.println("ingrese otra clave");

clave = leeEntero();

}

System.out.println("Correo: ");

String correo = leeCadena();

System.out.println("Tipos de empleados \n1 Gerente, 2)secretaria, 3)Intendente, 4)Empleado");

validar = leeCadena();

int tipo = 0;

while(validarNumero(validar) == false){

System.out.println("Ingresa con numero tipo de mepleado: ");

validar = leeCadena();

}

tipo = ConNumero(validar);

Empleado nuevoEmpleado = null;

if(tipo == 2 && util.quienEselGerente(listaEmpleados) != null ){//secretaria

System.out.println("Oficina");

String oficina = leeCadena();

Gerente jefe = util.quienEselGerente(listaEmpleados);

nuevoEmpleado = new Secretaria(nombre, direccion, ssn, clave, correo, oficina, jefe);

}

else if(tipo == 3 && util.quienEselGerente(listaEmpleados) != null){//intendente

System.out.println("Asignacion");

String asignacion = leeCadena();

nuevoEmpleado = new Intendente(nombre, direccion, ssn, clave, correo, asignacion);

}else if(tipo == 4 && util.quienEselGerente(listaEmpleados) != null){//Empleado

nuevoEmpleado = new Empleado(nombre, direccion, ssn, clave, correo);

}

else if(tipo == 1 && (util.quienEselGerente(listaEmpleados) == null)){//gerente

System.out.println("Departamento");

String departamento = leeCadena();

nuevoEmpleado = new Gerente(nombre, direccion, ssn, clave, correo, departamento);

}

else if(util.quienEselGerente(listaEmpleados) == null){

System.out.println("No hay gerente, registra uno antes");

}else{

System.out.println(" ya hay gerente, no puedes registar otro");

}

if(tipo>4 || tipo<1){

System.out.println("Tipo no valido");

}

//se añade el usuario a ala lista

if(nuevoEmpleado != null){

listaEmpleados.add(nuevoEmpleado);

}

break;

case 3:

//Eliminar Empleado

System.out.println("Eliminar un empleado");

System.out.println("Buscar por (N)nombre o por (C)clave: ");

String criterio = leeCadena();

if(criterio.equalsIgnoreCase("N")){

System.out.println("proporciona nombre de empleado a eliminar");

String nomEliminar = leeCadena();

util.eliminaEmpleadoNombre(nomEliminar, listaEmpleados);

}else if(criterio.equalsIgnoreCase("C")){

System.out.println("proporciona clave de empleado a eliminar: ");

int claveEliminar = leeEntero();

util.eliminaEmpleadoClave(claveEliminar, listaEmpleados);

}else{

System.out.println("opcion incorrecta");

}

//M�todo para eliminar a un empleado por su clave

break;

case 4:

System.out.println("Listado de Empleados");

Iterator<Empleado> iter = listaEmpleados.iterator();

while(iter.hasNext())

{

Empleado elemento = iter.next();

if(listaEmpleados != null){

System.out.println(elemento.toString());

}

}

break;

case 5:

if(util.existenDatos(listaEmpleados)){

System.out.println("Nombre del archivo para serializar");

arch = leeCadena();

util.serializaListaEmpleados(listaEmpleados, arch);

}else{

System.out.println("Registra empleados");

}

break;

default: System.out.println("saliendo");

break;

}

}while(opcion >0 && opcion<6);

}

public static String leeCadena()

{

Scanner scan = new Scanner(System.in);

String cadena = scan.nextLine();

return cadena;

}

public static int leeEntero()

{

Scanner scan = new Scanner(System.in);

int numero = scan.nextInt();

return numero;

}

public static boolean existe(String archivo)

{

File archVerificado = new File(archivo);

if (archVerificado.exists())

{

System.out.println("Si existe");

return true;

}

else

{

System.out.println("Archivo no existe");

return false;

}

}

public static boolean validarNumero(String dato)

{

boolean esNumero = false;

if (dato.matches("[0-9]\*")){

esNumero = true;

}else{

System.out.println("no es numero");

esNumero = false;

}

return esNumero;

}

public static int ConNumero(String dato)

{

int numero = Integer.parseInt(dato);

return numero;

}

}

**UtileriasArchivos.java**

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.List;

public class UtileriasArchivos

{

// M�todo para serializar un arreglo de Empleados

public void serializaEmpleado(Empleado [] empleado, String archivo)

{

try

{

FileOutputStream fileOut = new FileOutputStream(archivo);

ObjectOutputStream out = new ObjectOutputStream(fileOut);

out.writeObject(empleado);

out.close();

fileOut.close();

System.out.printf("Datos serializados en " + archivo);

}

catch (IOException e)

{

System.out.println("Error de I O");

e.printStackTrace();

}

}

// M�todo para deserializar un arreglo de Empleados

public Empleado [] deserializaEmpleado(String archivo)

{

Empleado [] empleado = null;

try

{

FileInputStream fileIn = new FileInputStream(archivo);

ObjectInputStream in = new ObjectInputStream(fileIn);

empleado = (Empleado[]) in.readObject();

in.close();

fileIn.close();

System.out.printf("Datos deserializados");

}

catch (IOException e)

{

System.out.println("Error de I O");

e.printStackTrace();

}

catch (ClassNotFoundException e)

{

System.out.println("Clase de referencia no encontrada");

e.printStackTrace();

}

return empleado;

}

// M�todo para serializar una Lista de Empleados

public void serializaListaEmpleados(List<Empleado> empleado, String archivo)

{

try

{

FileOutputStream fileOut = new FileOutputStream(archivo);

ObjectOutputStream out = new ObjectOutputStream(fileOut);

out.writeObject(empleado);

out.close();

fileOut.close();

System.out.printf("Datos serializados en " + archivo);

}

catch (IOException e)

{

System.out.println("Error de I O");

e.printStackTrace();

}

}

// M�todo para deserializar una lista de Empleados

@SuppressWarnings("unchecked")

public List<Empleado> deserializaListaEmpleados(String archivo)

{

List<Empleado> empleado = null;

try

{

FileInputStream fileIn = new FileInputStream(archivo);

ObjectInputStream in = new ObjectInputStream(fileIn);

empleado = (List<Empleado>) in.readObject();

in.close();

fileIn.close();

System.out.printf("Datos deserializados");

}

catch (IOException e)

{

System.out.println("Error de I O");

e.printStackTrace();

}

catch (ClassNotFoundException e)

{

System.out.println("Clase de referencia no encontrada");

e.printStackTrace();

}

return empleado;

}

public boolean existenDatos(List<Empleado> lista){

if(lista != null && lista.size() != 0){

System.out.println("Hay empleados en la lista");

return true;

}else{

System.out.println("No hay empleados");

return false;

}

}

//Suponiendo que solo hay un gerente

public Gerente quienEselGerente(List<Empleado> lista)

{

Gerente gerente = null;

for(Empleado empleado:lista)

if(empleado instanceof Gerente)

{

System.out.println("Gerente encontrado");

gerente = (Gerente) empleado;

}

return gerente;

}

public void eliminaEmpleadoClave(int clave, List<Empleado> lista)

{

Empleado aEliminar = new Empleado();

for(Empleado emp:lista)

if(emp.getNumero()== clave)

aEliminar = emp;

//Imprimir datos del empleado a eliminar

if(aEliminar.getNombre() != null){

System.out.println("Datos del empleado a eliminar");

System.out.println(aEliminar.toString());

}else{

System.out.println("El usuario no existe en la base");

}

lista.remove(aEliminar);

}

public void eliminaEmpleadoNombre(String nombre, List<Empleado> lista)

{

Empleado aEliminar = new Empleado();

for(Empleado emp:lista)

if(emp.getNombre().equals(nombre))

aEliminar = emp;

//Imprimir datos del empleado a eliminar

if(aEliminar.getNombre() != null){

System.out.println("Datos del empleado a eliminar");

System.out.println(aEliminar.toString());

}else{

System.out.println("El usuario no existe en la base");

}

lista.remove(aEliminar);

}

public boolean verClave(int clave, List<Empleado> lista)

{

Empleado aBuscar = new Empleado();

boolean encontrado = false;

for(Empleado emp:lista){

if(emp.getNumero()== clave){

aBuscar = emp;

System.out.println("Clave ya existe");

encontrado = true;

}

}

return encontrado;

}

public boolean verNombre(String nombre, List<Empleado> lista)

{

Empleado aBuscar = new Empleado();

boolean encontrado = false;

for(Empleado emp:lista){

if(emp.getNombre().equals(nombre)){

aBuscar = emp;

System.out.println("Nombre ya existe");

encontrado = true;

}

}

return encontrado;

}

}

**Empleado.java**

import java.io.Serializable;

public class Empleado implements Serializable

{

private String nombre;

private String direccion;

private int SSN;

private int numero;

private String correo;

public Empleado()

{

}

public Empleado(String nombre, String direccion, int sSN, int numero, String correo)

{

super();

this.nombre = nombre;

this.direccion = direccion;

SSN = sSN;

this.numero = numero;

this.correo = correo;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getDireccion() {

return direccion;

}

public void setDireccion(String direccion) {

this.direccion = direccion;

}

public int getSSN() {

return SSN;

}

public void setSSN(int sSN) {

SSN = sSN;

}

public int getNumero() {

return numero;

}

public void setNumero(int numero) {

this.numero = numero;

}

public String getCorreo() {

return correo;

}

public void setCorreo(String correo) {

this.correo = correo;

}

@Override

public String toString() {

return "\nNombre=" + nombre + ", \nDireccion=" + direccion + ", \nSSN=" + SSN + ", \nNumero=" + numero

+ ", \nCorreo=" + correo;

}

}

**Gerente.java**

public class Gerente extends Empleado

{

private String departamento;

public Gerente(String nombre, String direccion, int sSN, int numero, String correo, String departamento) {

super(nombre, direccion, sSN, numero, correo);

this.departamento = departamento;

}

public String getDepartamento() {

return departamento;

}

public void setDepartamento(String departamento) {

this.departamento = departamento;

}

@Override

public String toString() {

return "Gerente [departamento=" + departamento + super.toString() + "]";

}

}

**Secretaria.java**

public class Secretaria extends Empleado

{

private String oficina;

private Gerente jefe;

public Secretaria(String nombre, String direccion, int sSN, int numero, String correo, String oficina,

Gerente jefe)

{

super(nombre, direccion, sSN, numero, correo);

this.oficina = oficina;

this.jefe = jefe;

}

public String getOficina() {

return oficina;

}

public void setOficina(String oficina) {

this.oficina = oficina;

}

public Gerente getJefe() {

return jefe;

}

public void setJefe(Gerente jefe) {

this.jefe = jefe;

}

@Override

public String toString() {

return "Secretaria [oficina=" + oficina + ", jefe=" + jefe + super.toString() + "]";

}

}

**Intendente.java**

public class Intendente extends Empleado

{

private String asignacion;

public Intendente(String nombre, String direccion, int sSN, int numero, String correo, String asignacion)

{

super(nombre, direccion, sSN, numero, correo);

this.asignacion = asignacion;

}

public String getAsignacion() {

return asignacion;

}

public void setAsignacion(String asignacion) {

this.asignacion = asignacion;

}

@Override

public String toString() {

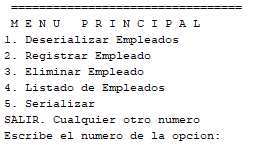
return "Intendente [asignacion=" + asignacion + super.toString() + "]";

}

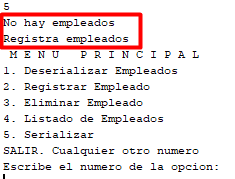
}

**TESTING DEL PROGRAMA**

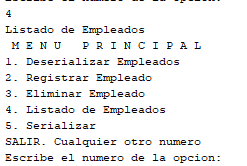
Ejecuto el programa:



Haremos cosas al revés, trataré de serializar primero, antes de crear cualquier lista o empleados:



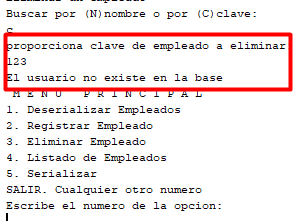
Opción 4 para que me imprima la lista de empleados:

pero no hay registros, así que no imprime nada y devuelve el menú principal.

Opción 3 eliminaremos un empleado:

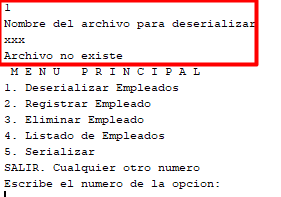


Ya sea que lo busquemos o por nombre, si no hay registro de algún empleado nos dirá lo siguiente:



Y nos devuelve la pantalla principal.

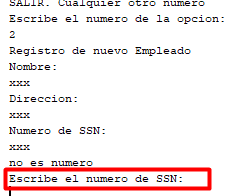
Opción 1:



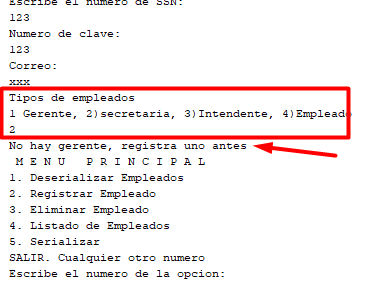
Pero como no se ha creado ningún archivo, no podemos deserializar nada, y nos dice que el archivo a deserializar no existe, y nos da el menú principal

AHORA HAGAMOS LA COSAS BIEN, se creará una lista de empleados con todo lo necesario.

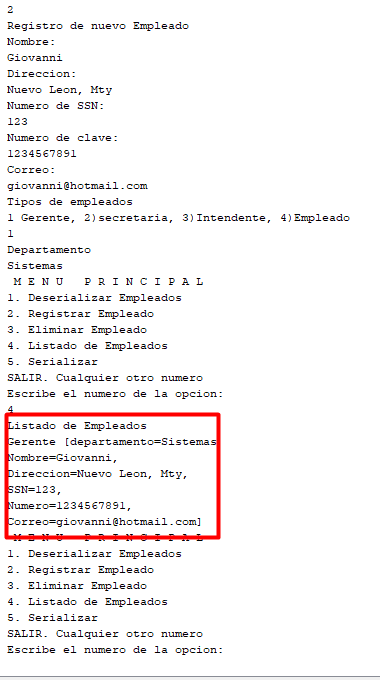
En las opciones donde se pide número, si no se agrega un número, no dejará avanzar hasta que se agregue un número.



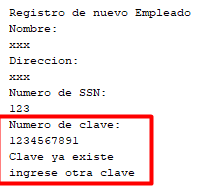
Si intentamos crear cualquier empleado antes de hacer el o que exista un gerente, no nos dejará crearlo y nos mandará al menú de nuevo



CREAMOS UN GERENTE

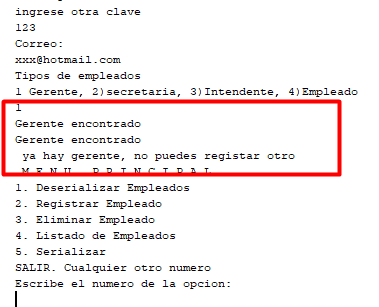


Intentamos crear otro empleado con la misma clave

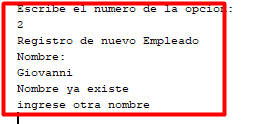


Hasta que no ingresemos una nueva clave, no nos dejará registrar

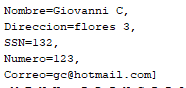
Intentamos hacer otro gerente:



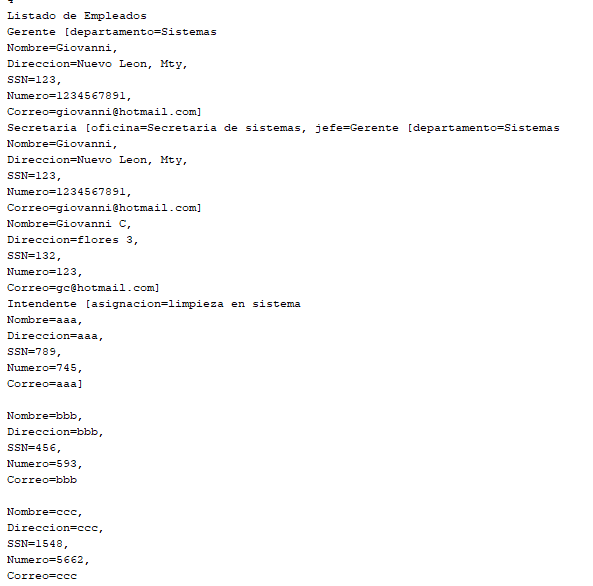
Intentaremos crear un nuevo empleado con un nombre similar al gerente:



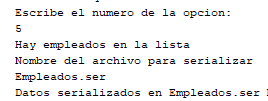
No nos dejará ingresar el nombre, a menos que se cambie con una pequeña modificación

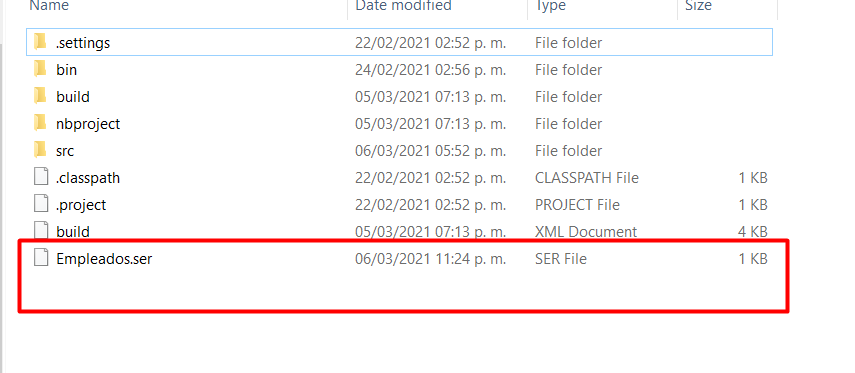


Creamos varios empleados



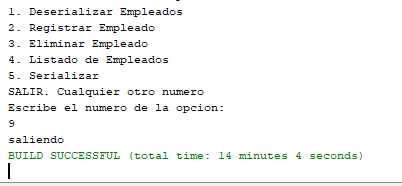
Serializamos en un archivo



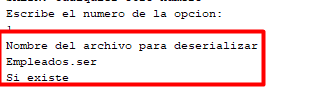


Salimos del programa

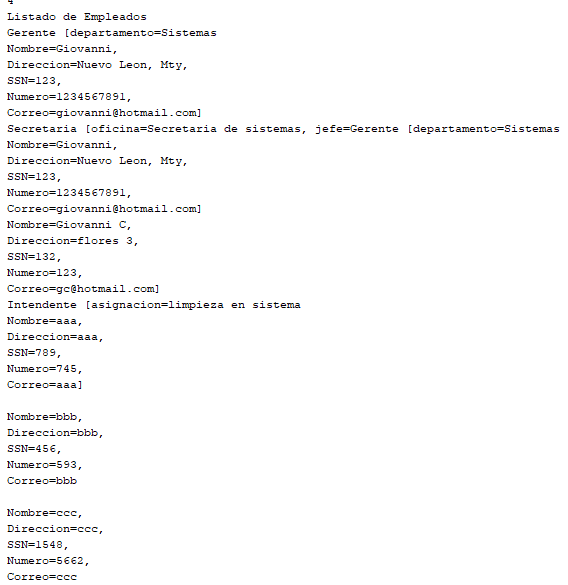
Apretamos cualquier otro número:



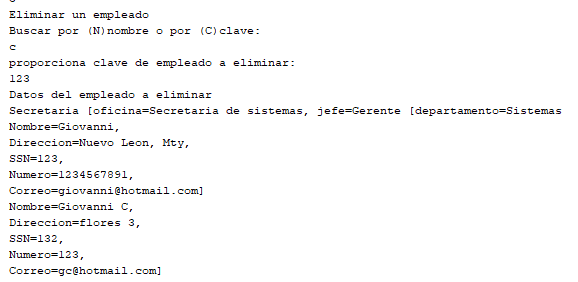
Volvemos a ejecutar y deserializamos empleados



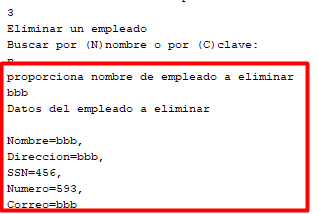
Nos dice que si existe y con la opción 4 mostramos a los empleados:



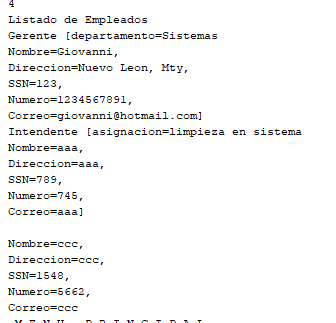
Procederemos a eliminar empleados:



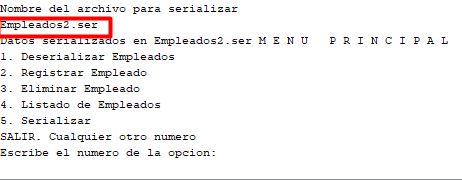
Cuando muestra los datos significa que se eliminó, eliminamos otro empleado, pero ahora por nombre y mostramos a los empleados restantes:

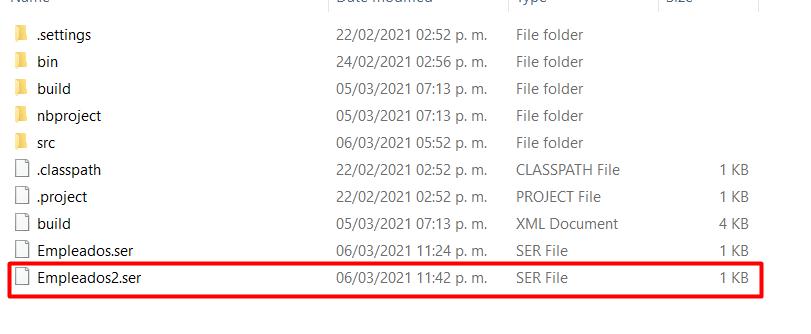


Empleados restantes:

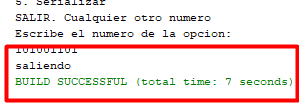


Volvemos a serializar con otro nombre:





Y Salimos del programa



FIN