MEMORIA FPGS DAM 1 MARCOS MORALES ARAGÓN

Práctica iniciada el 6/04 y terminada el 16/06

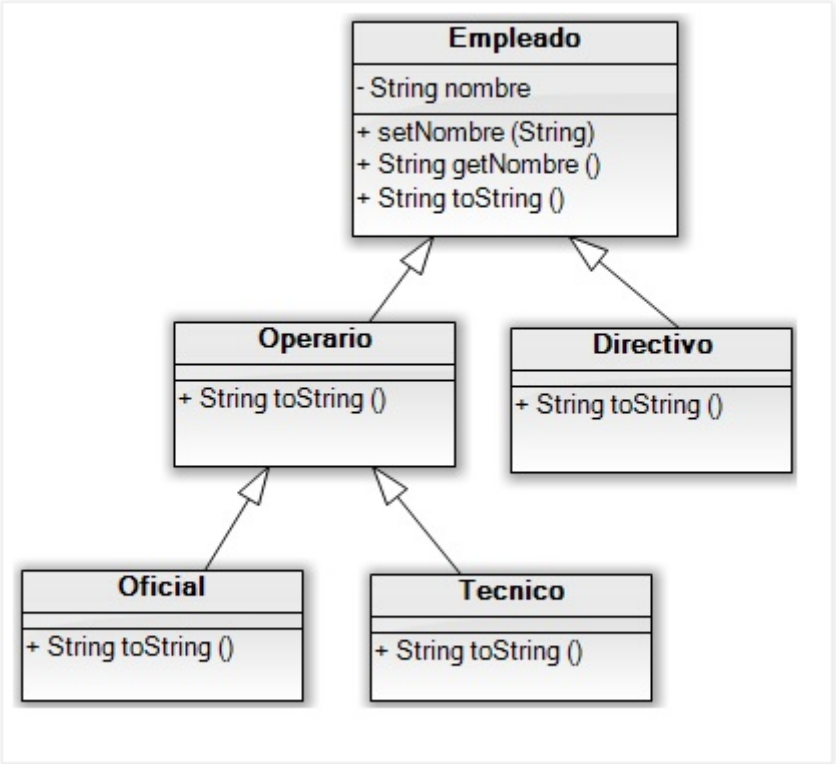
→ ÍNDICE ←

→ 1. Práctica

Portal de Gestión

El Portal de Gestión consiste en una aplicación web para gestión de personal de una compañía. En dicha aplicación se podrá dar de alta usuarios de distinto tipo, listar los usuarios almacenados, realizar búsqueda de usuarios, modificar usuarios, indicar información adicional de usuarios, etc.

La estructura base de los empleados, a nivel de jerarquía de clases, viene descrita por el siguiente diagrama UML:



La clase base es la clase Empleado. Esta clase contiene:

- Un atributo privado nombre de tipo String que heredan el resto de clases.
- Un constructor por defecto.
- Un constructor con parámetros que inicializa el nombre con el String que recibe.

- Método set y get para el atributo nombre.
- Un método toString() que devuelve el String: "Empleado " + nombre.

Este modelo de será la base para futuras prácticas, con el objetivo de conseguir una aplicación de gestión con una interfaz web o frontend desarrollada en Angular, un backend desarrollado en Java y un almacenamiento de datos en BBDD relacional con Oracle.

Las tareas para la construcción del Portal serían:

1. Instalación de IntelliJ IDEA como IDE de desarrollo. https://www.jetbrains.com/es-es/idea/download/#section=windows
2. Creación de programa Java con versión 1.8 donde se definirán las distintas clases de y objetos según la definición del UML. Crear métodos equals y hascode.
3. Creación de métodos para crear, listar, borrar y modificar los distintos elementos definidos. (*3.1)
4. Creación de métodos de validación de datos en las distintas funciones.
5. Construir un usuario de BBDD y crear las tablas que apliquen a las entidades Java creadas.
6. Configurar conexión de BBDD con la aplicación Java para utilizar la BBDD como almacén
7. Realizar los métodos necesarios para poder dar de alta y listar por consola los valores almacenados en BBDD
8. Ampliación de propiedades a las clases ya definidas (apellidos, fecha de nacimiento, nacionalidad), realizando script de actualización de tablas de BBDD y actualización de código correspondiente.

(*3.1)

Todo empleado debe ser identificado por un código alfanumérico de 6 caracteres. Además, por cada empleado será necesario informar nombre y apellidos, DNI, fecha de nacimiento, y dirección. La dirección debe estar compuesta por calle, número, bloque, piso, puerta, código postal, localidad y provincia.

Del personal técnico deberá almacenarse los títulos de las distintas especialidades que pueda tener, así como el año finalización de la especialidad.

Del personal directivo es necesario conocer el departamento o departamentos que tiene a su cargo.

Un empleado puede estar registrado en el sistema según tres estados: alta, baja, en tramite.

En relación con la empresa, es necesario conocer la antigüedad de los empleados en la empresa. Es posible que el mismo empleado, estuviera en la empresa un tiempo, finalizara su contrato y posteriormente volviera a ser contratado.

Es necesario contar con la información de cada periodo de trabajo de cada empleado.

El objetivo del Portal es almacenar toda la información descrita, aplicar las comprobaciones correspondientes y poder consultar información

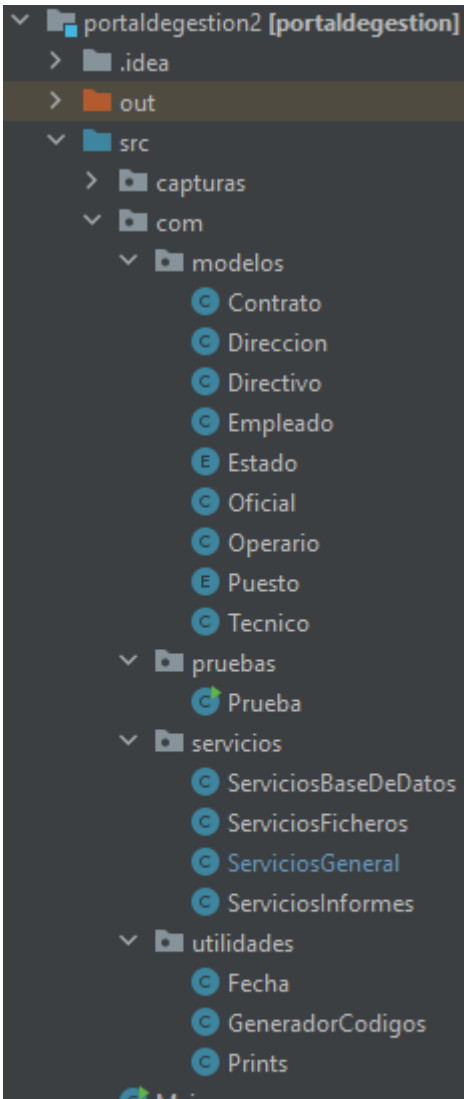
respondiendo a preguntas u opciones de menú:

- ¿Cuál es el empleado actual de mayor edad?
- ¿Cuál es el empleado actual de menor edad?
- ¿Cuántos empleados tiene la empresa actualmente?
- ¿Cuántos empleados se han dado de baja el año actual?
- ¿Qué directivo es responsable de más departamentos?
- ¿Cuál es el directivo responsable del departamento X?
- ¿Qué empleado tiene más titulaciones en especialidades?
- ¿Cuál es la titulación de la especialidad más reciente y a quién pertenece?

→ 2. Funciones y posibilidades

La función principal del programa es la gestión de empleados de una empresa, mediante creación de empleados y sus contratos en la empresa. El objetivo es el almacenamiento de los datos de los datos de los empleados y sus contratos. En caso de borrar a algún empleado sus datos serán mandados a un fichero backup , pero los contratos se mantendrán en la base de datos con el código que tenía asignado en ese momento.

→ 3. Estructura del programa



La estructura del proyecto es la siguiente. Ha sufrido varios cambios a lo largo de todo el desarrollo sobretodo debido a la falta de experiencia y conocimientos sobre como agrupar y nombrar las clases.

La carpeta modelos contiene la estructura de objetos. La clase empleados es la principal y tiene sus ramificaciones (Directivo, Técnico, Operario Oficial) aunque estas no han sido puestas en uso en la aplicación. Contrato y dirección son atributos de la clase empleado que tienen sus propios campos. Y por último tenemos las clases Estado y Puesto que son clases enumeradas.

La carpeta pruebas contiene un ejecutable donde se ha ido probando código previo a su implementación en el proyecto. Sobre todo la he usado para aprender a utilizar bien las nuevas cosas que se querían implementar y asegurar que fallos podían producir y de los resultados que estos pueden dar.

La carpeta servicios contiene la mayor parte del proyecto. En ella encontramos :

- Los java.class encargados de implementar de manera correcta y hacer un uso óptimo de algunas funcionalidades (ficheros y base de datos)
- Los java.class encargados de la lógica de creación de empleados y uso de las funciones de los ficheros y base de datos (general y informes)

→ 4. Cómo usar el programa y de los datos de entrada, opciones y resultados

El programa por ahora está pensado para ser utilizado desde consola, es decir, a la hora de crear nuevas cosas se usa la consola a través de los métodos leer, el programa está pensado y diseñado para que esta función sea la única que se tiene que cambiar a la hora de querer cambiar la entrada de datos.

El programa al iniciarse cargará la base de datos a memoria y desde ese punto hasta que se utilice algunas de las funciones guardar , ninguno de los cambios surtirá efecto.

A continuación dividiré el funcionamiento de las partes del código:

- La base de datos:

Principalmente la base de datos tiene el problema de la conexión, es decir , si queremos usar este programa en otra base de datos deberemos cambiar la conexión que se realiza en el programa

```
public Connection cargarBaseDeDatos(String palabra){
    String frase = "";
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
    } catch (ClassNotFoundException e) {
        prints.escribir( frase: "Fallo en la declaracion del driver");
    }
    Connection conexion = null;
    try  {
        conexion = DriverManager.getConnection(
                url: "jdbc:oracle:thin:@//10.206.110.3:1521/pdbcmdb.at4wireless.com", user: "TD_CMDB_V1", password: "bg4ve8l8");
        if (conexion != null) {
            frase = "¡ Conexion exitosa !";
        }
        if (palabra.equals("")){
            prints.escribir(frase);
        }
    } catch (Exception e) {
        prints.escribir( frase: "Error al realizar la conexion");
    }
    return conexion;
}
```

Aparte de esto a la hora de guardar, borrar, cargar o modificar ,los nombres de las tablas deben de estar ajustados a las creadas para ello. Un ejemplo es que yo ahora mismo cargo los datos de los empleados desde la tabla “FPM_EMPLEADOS”, para utilizar el programa de manera correcta tenemos dos soluciones:

1. Crear tablas idénticas a las de mis base de datos (se adjuntara una captura ahora) replicando los campos y nombre
2. Crear tablas con campos idénticos, cambiar el nombre de la tabla y a su vez alterar en el código el nombre de referencia (cambiar FPM_EMPLEADOS por su nuevo nombre)

Estas son las tablas creadas en la base de datos y sus campos.

Siempre deben respetarse la estructura del orden y tipo de formato de las tablas que se han mostrado arriba, ya que a la hora de realizar las funciones este orden es importante.

Estos cambios deben ser realizados por alguien que entienda cómo realizar la modificación de manera correcta.

- Los ficheros:

Los ficheros son utilizados para la copia de seguridad, que es donde se van los empleados borrados.

Este archivo siempre ha de llamarse copiaDeSeguridad.txt o en caso de querer cambiarlo buscar sus usos y establecer el nuevo nombre.

En el fichero los datos se establecerán separados por #, todo el proceso esta explicado en el readme de mi proyecto.

Explicacion de funcionamiento de los archivos :

Es importante que se sigan estos parametros sino te dara error al cargar tu archivo, el programa esta preparado para aunque de fallo al cargar el archivo puedas trabajar en el pero es recomendable que trabajes con el archivo bien cargado para poder usar todas las funciones de la aplicacion

- El archivo donde se guardaran a los empleados se debe llamar empleados.txt.
- El archivo donde se guardaran los datos de los empleados eliminados se debe llamar copiaDeSeguridad.txt.
- En caso de querer introducirlos escribiendo en el fichero deberás seguir el siguiente formato

Todos estos datos los separaremos con #

Código	Nombre	Primer Apellido	Segundo Apellido	DNI	Fecha nacimiento	Nacionalidad	Estado	Calle	Numero	Bloque	Piso	Puerta	Codigo postal	Localidad	Provincia	Fecha Creacion	Contrato
Admite NULL																Puede ser NULL (se le asignara la fecha del momento del ordenador)	Debe ser []

Código	Nombre	Primer Apellido	Segundo Apellido	DNI	Fecha nacimiento	Nacionalidad	Estado	Calle	Numero	Bloque	Piso	Puerta	Codigo postal	Localidad	Provincia	Fecha Creacion	Contrato
Admite NULL																Puede ser NULL (se le asignara la fecha del momento del ordenador)	Debe ser []


```
7h01x8#Prueba#Ape1#Ape2#45612318Q#15-06-2000#Española#Baja#1517795#no#0#no#no#no#29004#Málaga#Málaga#15-06-2021#[]
```

El archivo empleados.txt sigue creado pero está en desuso completo, ya que ahora los datos de los empleados son cargados desde la base de datos.

- Servicios General:

A continuación explicaré las funciones más relevantes que aparecen en el código y su lógica, a excepción de las funciones que son llamadas desde el main ya que esas son opciones de menú que serán explicadas en el punto siguiente

generarCodigo→ Este código tiene el propósito de evitar que se puedan generar códigos idénticos dentro de una misma lista, es decir, evitar que existan 2 direcciones con el mismo código. Para ello hace uso de un bucle donde se crean los códigos y luego se comprueba que estos no existen haciendo uso de la función codigoExiste.

codigoExiste → Recorre las listas seleccionando el código y comparándolo con el creado. Esta función es sumamente importante ya que los códigos son las pk de las tablas en las bases de datos y es importante que estos no se repitan de ninguna de las maneras

transformarStringAIntDevuelveInt → Al usar scanner para leer por consola, todos los datos son leídos en formato nextLine(), el cual devuelve un String. Esto nos lleva a el uso de esta función, la cual lee el dato en string y lo cambia a int.

leerDNI → El DNI también está controlado y aunque no se comprueba si se repite si controla que esté bien escrito, que sus 8 primeros componentes sean un números y el último una letra. Para ello nos aprovechamos del error que da intentar convertir una letra en número. Si intentas convertir una letra en número nos dará una excepción, por lo cual, lo que se hace es que se intenta convertir el último dígito del dato introducido en número. En caso de ser una letra nos dará una excepción (lo cual indicaría que el último dato se trata de una letra y no de un número.

datosEmpleados → Esta es la función mas grande que tiene el proyecto y es la encargada de procesar la creación de los empleados independientemente de donde se quieran usar: fichero, base de datos , teclado o papelera (la papelera es un mapa), lo único es indicar de donde ha sido llamada la función, si por ejemplo ha sido llamada desde una función de base de datos en el campo “deDondeVieneElDato” se pondrá bbdd.

El resto de funciones creo que con el JavaDoc que tienen se entienden bien cuales son sus funciones y acciones, solo quería mencionar las más relevantes

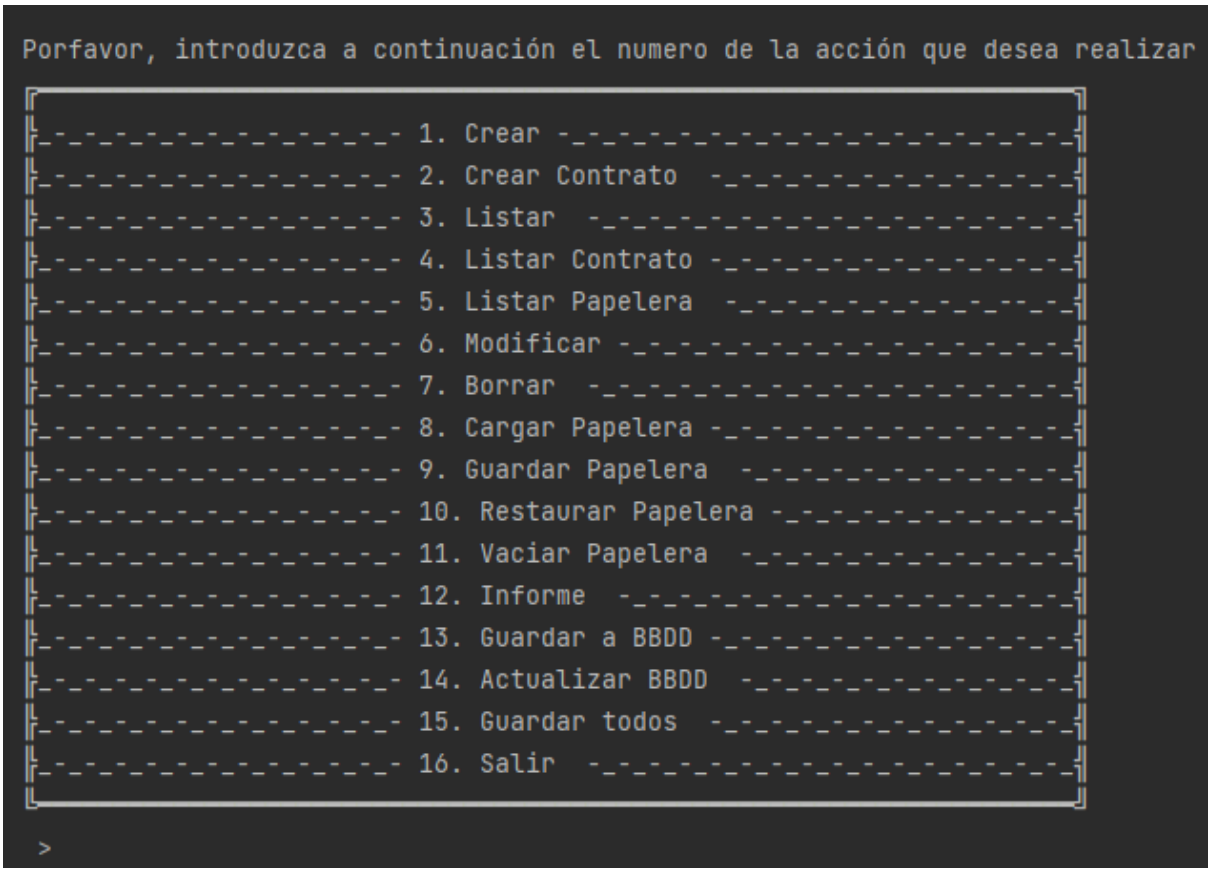
En este apartado también son creadas y determinadas las listas que se usan durante todo el programa las cuales son :

```
public static ArrayList<Empleado> empleados = new ArrayList<>();
public static ArrayList<Empleado> empleadosNuevos = new ArrayList<>();
public static ArrayList<Empleado> empleadosModificados = new ArrayList<>();
public static ArrayList<Empleado> empleadosBorradosNuevos = new ArrayList<>();

public static ArrayList<Contrato> contratos = new ArrayList<>();
public static ArrayList<Contrato> contratosNuevos = new ArrayList<>();

public static HashMap<String, Empleado> empleadosBorrados = new HashMap<>();
```

→ 5. Opciones de menú



Este es el menú de mi aplicación, voy a explicar qué hace cada una de sus opciones.

1. Crear → Crea un empleado nuevo junto a su dirección. Se debe de tener cuidado al introducir el dni porque en caso de hacerlo mal el número de veces establecidas (3 aunque se pueden cambiar a las que se quiera) el proceso de creación se parará y serás enviado al menú principal perdiendo todo lo que hubieras creado de ese empleado.
2. Crear contrato → Para ello es necesario tener empleados creados. Se pedirá el código de uno y una vez entregado el código se pasará a la creación de los contratos. En caso de ser el primer contrato del empleado el programa

automáticamente detectara eso y establecerá como fecha de inicio su fecha de creación (establecida al crear al empleado), en caso de que no sea el primero la fecha deberá de ser establecida a mano.

3. Listar → Muestra por pantalla los datos de los empleados.
4. Listar Contratos → Muestra por pantalla únicamente todos los contratos creados
5. Listar papelera → Muestra los datos de los empleados que se encuentren en la papelera. Para ello deben existir empleados borrados, por lo cual o borramos un empleado o cargamos la papelera, sino, no saldrá nada.
6. Modificar → Permite cambiar los datos de un empleado (los contratos no se pueden modificar)
7. Borrar → Elimina a un empleado y este es añadido a el mapa papelera
8. Cargar papelera → Carga el archivo papelera y mete en memoría los datos de los empleados que se encuentren borrados
9. Guardar papelera → Borra el archivo copiaDeSeguridad.txt , una vez borrado crea uno nuevo y introduce los datos que se encuentre en memoria del mapa papelera
10. Restaurar papelera → Introduce a memoria los datos de la papelera, restaurando los datos que se encuentran en ella.
11. Vaciar papelera → Borra todos los datos en la papelera
12. Informe → Responde a las siguientes preguntas : ¿Cuál es el empleado actual de mayor edad?, ¿Cuál es el empleado actual de menor edad? y ¿Cuántos empleados se han dado de baja el año actual?.
13. Guardar a BBDD → Guarda los datos de los empleados en la base de datos
14. Actualizar BBDD → Guarda los cambios realizados, mediante modificar (la opción de menú 6) , de los empleados ya existentes
15. Guardar todos → Guarda la papelera, los contratos, los empleados , actualiza y guarda la papelera. Es el método más recomendable de utilizar siempre antes de cerrar el programa

Es importante mencionar el uso de las listas :

empleadosNuevos → Al crear un empleado nuevo en el programa será añadido a la lista empleados y también a esta lista, de esta manera al guardar a bbdd se usará esta lista para conocer cuáles han sido creados durante la ejecución del programa , una vez guardados esta lista se vacía puesto que ya figuran en la bbdd.
empleadosModificados → Nos proporciona una lista de los empleados que han sido modificados para posteriormente ser sometidos a un update. Una vez sometidos a el update se vacía la lista
empleadosBorrados → Cuando borramos un empleado aparte de ser introducidos en el mapa de papelera también son introducidos en esta lista ya que es importante que sean borrados de la base de datos. Después su lista es vaciada
contratosNuevos → Aquí se introducen los contratos nuevos ya que al igual que los empleados es necesario saber cuáles han sido creados durante la ejecución del programa. Después su lista es limpiada ya que no serán contratos nuevos.

Es necesario mencionar que no existe direccionesNuevas ya que la creación de estas están vinculadas con los empleados, por lo cual al usar empledaosNuevos también sabemos las direccione nuevas creadas.

→ 6. Conclusión

Primero que todo aportaré una autocrítica de mi trabajo, ya que considero que es lo primordial.

Creo que podría haber sido mejor, noto que pueden llegar a faltar algunas cosas y no todo está tal y como a mi me gustaría. He aprendido que tengo todavía mucho trabajo por delante y sobre todo aprender a optimizar mejor mis acciones y ser capaz de ser más productivo. Dentro de mis planes de verano ya entran aprender a mejorar esas características y el año que viene ser capaz de ser un mejor desarrollador. El programa en si no me gusta al 100% pero también he de decir que estoy contento con casi todo lo que he hecho.

Ahora bien, la experiencia es algo que sí me ha encantado. He disfrutado todos los días de las prácticas y ha sido algo único, creo que la brecha entre los conocimientos que tenía al entrar y al salir es muy grande. Siento que he aprendido mucho pero también estoy motivado porque me he dado cuenta de lo grande que es el horizonte y de todas las cosas que todavía me quedan por aprender. Estoy entusiasmado con todo lo que he llegado a aprender y con lo que me queda por descubrir.

Desde el primer día me he sentido muy cómodo y he disfrutado de la experiencia al máximo. Espero poder volver a hacer aquí las prácticas el año que viene y poder volver a vivir la experiencia.

A nivel del programa, me he encontrado con incontables dificultades pero mi intención con respecto a las prácticas era afrontarlas como un desafío y desde el primer al último error he disfrutado de todos ellos. Algunos han sido más complicados que otros pero todos han sido interesantes y muy nutritivos.

También agradecer el trabajo de nuestro “mentor” Francisco. He decidido llamarle mentor porque siento que me ha enseñado mucho y ha compartido con nosotros muchos de sus conocimientos. Siempre que ha podido nos ha ayudado y ha sido siempre muy comprensivo con nosotros, al igual que les dije a mis profesores durante la exposición de las prácticas, creo que es un gran profesional y me ha gustado aprender de él. Desde el primer momento nos ha ido guiando en la práctica y siempre ha sido muy paciente con nosotros. Que haya disfrutado tanto de la experiencia también es mérito suyo y creía que esto merecía una mención.