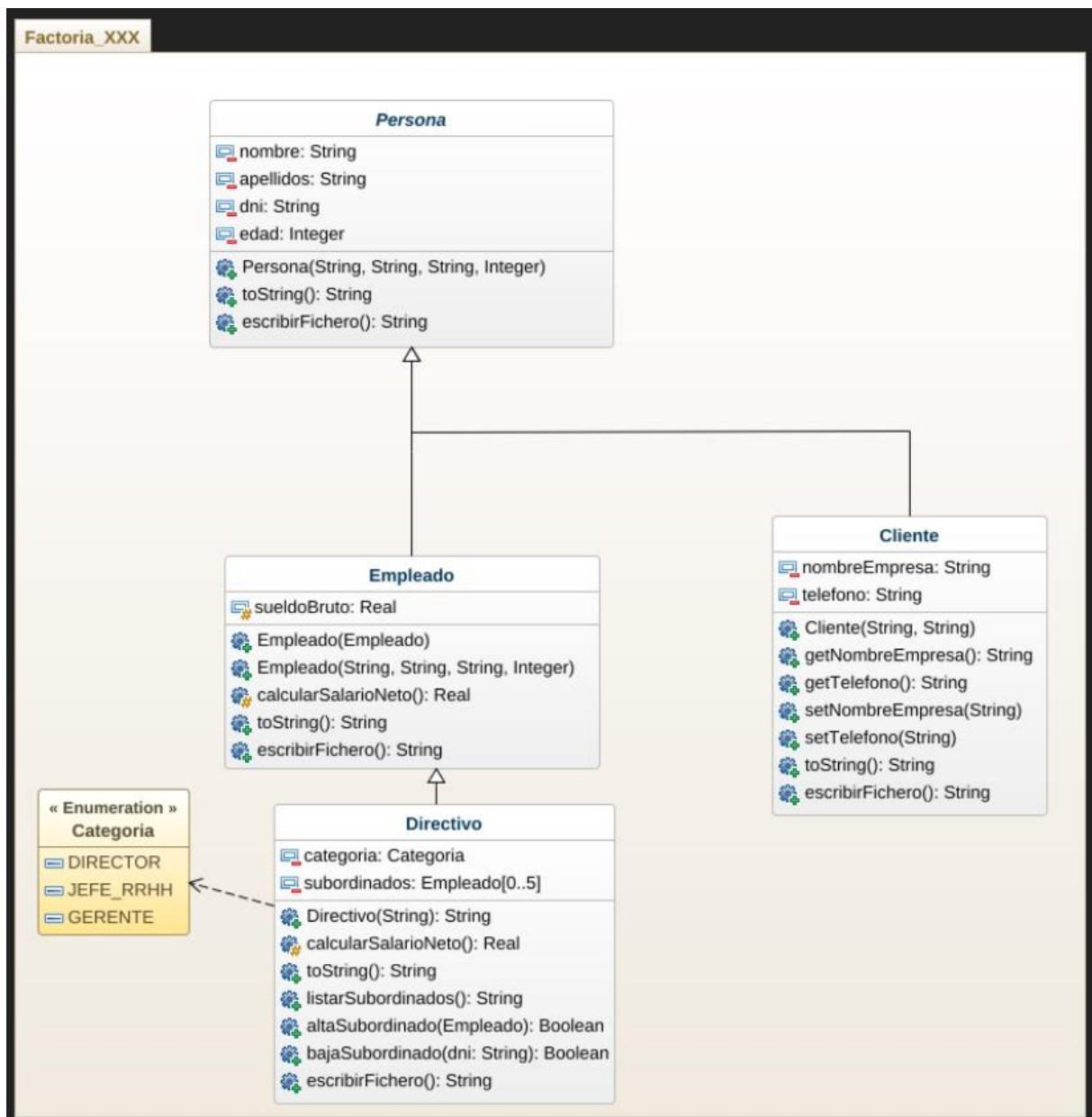
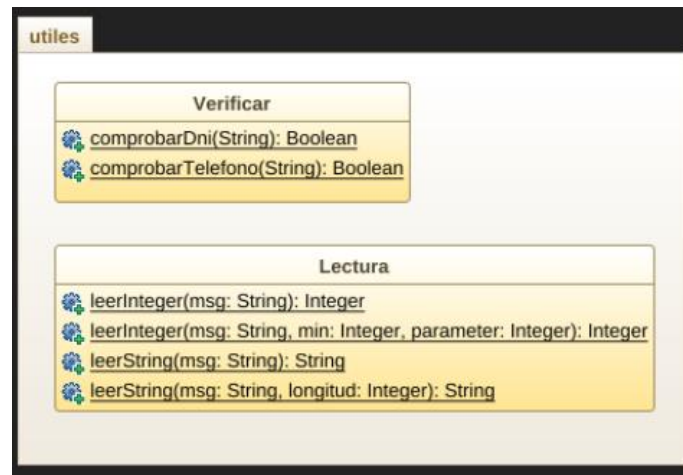


Programación		
Examen Segundo Trimestre		
Nombre:		

Se quiere llevar la gestión de los empleados y clientes de la empresa `Factoria_XXX`. La empresa tendrá como máximo 30 empleados (entre Empleado y Directivo) y 30 clientes.





1.- La clase principal se llamará **Factoria_XXX**, donde XXX corresponde a tus iniciales. Dicha clase llevará la gestión, tanto de los clientes como de los empleados con lo cual, el usuario de la aplicación podrá **añadir, eliminar y listar clientes**. También se podrá **añadir empleados, directivos y asociar empleados a los directivos, dar de baja a los empleados y a los directivos, listar empleados y directivos, y listar los empleados subordinados a un directivo**. Y contendrá el método **main**.

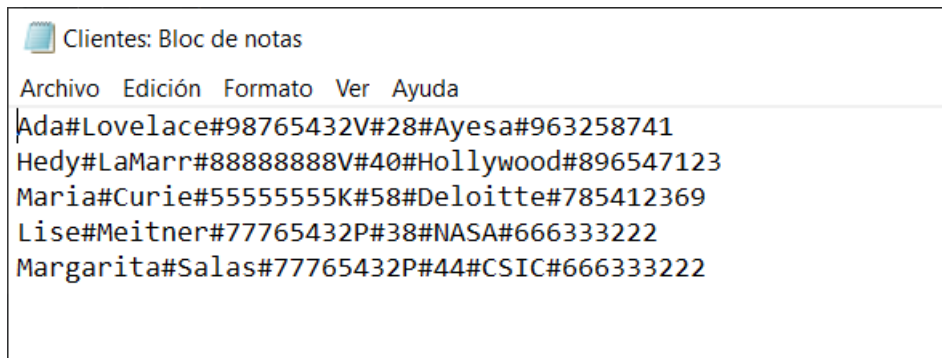
- Clase **Persona**: se trata de una clase abstracta que contiene los datos básicos de una persona: nombre, apellidos, dni y edad. Aunque en el diagrama no aparezca se debe realizar los `get` y `set`.
- Clase **Cliente**: hereda de la clase Persona y contiene el nombre de la empresa y el teléfono.
- Clase **Empleado**: hereda de Persona y contiene el sueldo bruto (el salario bruto será de 1387.34 euros).
 - **calcularSalarioNeto**: este método calculará el salario neto, que se le aplicará una retención del 19%.
- Clase **Directivo**: es una clase final que hereda de Empleado, un directivo puede tener a su cargo varios empleados (entre 0 a 5). El directivo tendrá una categoría tipo enumerado llamado Categoría.
 - **calcularSalarioNeto**: a parte de la retención del 19%, se le aplicará un complemento según la categoría:
 - **Director**: 950 euros
 - **Jefe de recursos humanos**: 550 euros
 - **Gerente**: 375 euros
- Clase **Verificar**: se trata de la clase estática que nos va a servir para comprobar si el teléfono y el dni tiene la estructura correcta.
 - **comprobarTelefono**: devolverá verdadero si el teléfono introducido por parámetro contiene 9 números, falso en caso contrario.
 - **comprobarDni**: devolverá verdadero si el DNI introducido por lo menos contiene 8 números y una letra, falso en caso contrario.
- Clase **Lectura**: se trata de una clase estática que nos servirá para realizar lecturas de datos por teclado. Si te hiciera falta algún otro método que no está especificado en la clase realízalo y razona porque lo necesitas.

2.- Realizar la estructura necesaria para poder guardar la información en 2 ficheros de texto, uno para los clientes y otro para los empleados. Y que dichos ficheros se carguen a la hora de ejecutar la aplicación.

En la clase principal (Factoria_XXX) tendrá 2 métodos:

- guardarDatos() y
- cargarDatos().

Ejemplo de fichero de clientes



```
Clientes: Bloc de notas
Archivo Edición Formato Ver Ayuda
Ada#Lovelace#98765432V#28#Ayesa#963258741
Hedy#LaMarr#88888888V#40#Hollywood#896547123
Maria#Curie#55555555K#58#Deloitte#785412369
Lise#Meitner#77765432P#38#NASA#666333222
Margarita#Salas#77765432P#44#CSIC#666333222
```

Como ayuda os paso los métodos que debéis incorporar a la clase Lectura:

○ leerArchivo

```
/**
 * leer todo el contenido de un archivo.
 *
 * @param rutaArchivo parámetro que recibe el método con la ruta (y nombre del archivo) del que
 * se quiere realizar la lectura
 * @return Devuelve todo el contenido como una cadena de caracteres
 */
public static String leerArchivo(String rutaArchivo) {
    String res = "";
    Scanner sc = null;
    File fichero = new File(rutaArchivo);

    try {
        System.out.println("Leyendo el contenido del fichero.....");
        sc = new Scanner(fichero);

        // leer línea a línea el fichero
```

```

        while (sc.hasNextLine()) {

            res += sc.nextLine() + "\n";

        }

        System.out.println("Lectura terminada.....");

    } catch (Exception e) {

        System.out.println("Mensaje:  " + e.getMessage());

    } finally {

        try {

            if (sc != null) {

                sc.close();

            }

        } catch (Exception e2) {

            System.out.println("Mensaje fichero:  " + e2.getMessage());

        }

    }

    return res;

}

```

○ **escribirArchivo**

```

/**
 * método estático que permite guardar la información que recibe por parámetro en un archivo de
 * texto
 *
 *
 * @param ruta Cadena de caracteres que indica la ruta y el nombre del archivo donde se quieren
 * guardar los datos.
 *
 * @param datos cadena de caracteres que contiene la información que se quiere guardar.
 *
 * @param sobreEscribir booleano que en función de su valor indica si se sobrescribe el contenido
 * del archivo (verdadero) o se añade al final del mismo (falso).
 *
 * @return verdadero si se ha guardado correctamente el archivo y falso si se ha dado algún error
 * y no se ha podido guardar
 */
public static boolean escribirArchivo(String ruta, String datos, boolean sobreEscribir) {

    boolean res;

    FileWriter fichero;

    try {

        fichero = new FileWriter(ruta, !sobreEscribir);

        fichero.write(datos);

    }

```

```
        fichero.close();

        res = true;

    } catch (FileNotFoundException ex) {

        System.out.println("Fichero no encontrado");

        res = false;

    } catch (Exception e) {

        System.out.println("Mensaje: " + e.getMessage());

        res = false;

    }

    return res;

}
```

3.- Realizar los comentarios oportunos para la documentación Javadoc.

Se valorará

- la adecuación de los nombres de constantes, atributos, métodos, variables, ...
- utilización de sangría
- código modutable, reutilizable, claro, conciso, limpio y elegante
- utilización de las estructuras apropiadas
- uso apropiado de los conceptos de Programación Orientada a Objetos

Ejemplos de pantallas del programa en ejecución

```
run:
-----
----  Bienvenido a Factoria_XXX  ----
1. Añadir cliente
2. Eliminar cliente
3. Listar cliente
-----
4. Añadir empleado
5. Eliminar empleado
6. Listar empleados
7. Listar subordinados
-----
10. ASOCIAR EMPLEADOS A DIRECTIVOS
11. LISTAR EMPLEADOS ASOCIADOS A DIRECTIVOS
-----
12. Guardar datos
0. SALIR-
-----

Ingrese Selección:
```

```
-----

Ingrese Selección: 6

-----

LISTADO DE EMPLEADOS

-----

Nombre: Pepe
Apellidos: Lopez
DNI: 12345678Z
Edad: 33
Sueldo: 1123.7454

-----

Nombre: Amapola
Apellidos: del Campo Gomez
DNI: 33345678Z
Edad: 34
Sueldo: 1123.7454

DIRECTOR
-----

Nombre: Robustiano
Apellidos: Jimenez
DNI: 87654321Z
Edad: 35
Sueldo: 1893.2452

-----

----  Bienvenido a Factoria_XXX  ----
1. Añadir cliente
```

```
LISTADO DE CLIENTES

-----

Nombre: Ada
Apellidos: Lovelace
DNI: 98765432V
Edad: 28
Nombre empresa: Ayesa
Telefono: 963258741

-----

Nombre: Hedy
Apellidos: LaMarr
DNI: 88888888V
Edad: 40
Nombre empresa: Hollywood
Telefono: 896547123

-----

Nombre: Maria
Apellidos: Curie
DNI: 55555555K
Edad: 58
Nombre empresa: Deloitte
Telefono: 785412369

-----

Nombre: Lise
Apellidos: Meitner
DNI: 77765432P
Edad: 38
Nombre empresa: NASA
Telefono: 666333222
```

```
LISTADO SUBORDINADO

-----

Los subordinados de: Robustiano Jimenez
-----

Nombre: Pepe
Apellidos: Lopez
DNI: 12345678Z
Edad: 33
Sueldo: 1123.7454

-----

---- Bienvenido a Factoria_XXX ----
1. Añadir cliente
2. Eliminar cliente
3. Listar cliente
-----
```

```
2. Eliminar cliente
3. Listar cliente
-----
4. Añadir empleado
5. Eliminar empleado
6. Listar empleados
7. Listar subordinados
-----
8. ASOCIAR EMPLEADOS A DIRECTIVOS
9. LISTAR EMPLEADOS ASOCIADOS A DIRECTIVOS
-----
10. Guardar datos
0. SALIR-
-----

Ingrese Selección: 10
Guardado de datos de CLIENTES:
----->Información guardada correctamente
Guardado de datos de EMPLEADOS:
----->Información guardada correctamente

-----
---- Bienvenido a Factoria_XXX ----
1. Añadir cliente
2. Eliminar cliente
3. Listar cliente
-----
4. Añadir empleado
5. Eliminar empleado
6. Listar empleados
7. Listar subordinados
-----
8. ASOCIAR EMPLEADOS A DIRECTIVOS
9. LISTAR EMPLEADOS ASOCIADOS A DIRECTIVOS
-----
10. Guardar datos
0. SALIR-
-----
```