



|   |   |   |
|---|---|---|
|  | <p align="center"><b>Departamento Informática</b><br/> <b>Ciclo G.S. DAW-E-Learning</b><br/> <b>Módulo: PROGRAMACIÓN</b><br/> <b>Examen Presencial 1ª Convocatoria (UT6 a UT11)</b></p> |  |
|---|---|---|

|               |  |
|---------------|--|
| <b>NOMBRE</b> |  |
|---------------|--|

|  |  |                                    |  |
|--|--|------------------------------------|--|
| <b>NOTA PARTE I:<br/>Cuestionarios</b> |  | <b>NOTA PARTE II:<br/>Practica</b> |  |
|--|--|------------------------------------|--|

**PUNTUACIÓN: PARTE I : CUESTIONARIOS: 2 puntos . PARTE II: PRÁCTICA: 8 puntos**  
Para poder **sumar** ambas notas en el **CUESTIONARIOS** debe obtener como **mínimo 0,8** y en la **PRÁCTICA** como **mínimo un 3,2** puntos

**PARTE I: CUESTIONARIO ( 2 puntos- mínimo 0,8)**  
**UNIDADES: 6,7,8,9,10 y 11**

**Cada pregunta acertada contará como 0,1. Cada pregunta errónea restará 0,04 . No contestado: 0 puntos.**

**RESPUESTAS DEL CUESTIONARIO**

| Pregunt<br>a | Respuest<br>a | Pregunt<br>a. | Respuest<br>a. | Pregunt<br>a. | Respuest<br>a. | Pregunt<br>a. | Respuest<br>a |
|--------------|---------------|---------------|----------------|---------------|----------------|---------------|---------------|
| 1            |               | 6             |                | 11            |                | 16            |               |
| 2            |               | 7             |                | 12            |                | 17            |               |
| 3            |               | 8             |                | 13            |                | 18            |               |
| 4            |               | 9             |                | 14            |                | 19            |               |
| 5            |               | 10            |                | 15            |                | 20            |               |

**1. Las clases y métodos de E/S son las mismas independientemente del dispositivo con el que se quiera tratar. ¿Verdadero o falso?**

Seleccione **una**:

- a)** Verdadero
- b)** Falso

**2. Si al intentar acceder a un fichero, no existe se generará una:**

Seleccione **una**:

- a)** RMIException.
- b)** IOException.
- c)** SQLException.

3. Empleamos FileWriter para flujos de caracteres, pues para datos binarios se utiliza FileOutputStream. ¿Verdadero o falso?  
Seleccione **una**:  
a) Verdadero  
b) Falso
4. Señala las correctas. Cuando trabajamos con ficheros de acceso aleatorio en Java, el modo de apertura puede ser:  
Seleccione **una o más de una**:  
a) "r" para sólo lectura.  
b) "w".  
c) "rw" para lectura y escritura.  
d) Todas son correctas.
5. El método setVisible(false) con un JFrame permite que una aplicación deje de ocupar memoria.  
Seleccione **una**:  
a) Verdadero  
b) Falso
6. Swing es:  
Seleccione **una**:  
a) un componente de SWT.  
b) una librería de Java para la generación del GUI en aplicaciones.  
c) Una librería de NetBeans.  
d) Ninguna afirmación es correcta
7. Las casillas de verificación en Swing están implementadas para Java por la clase:  
Seleccione **una**:  
a) JCheckBox.  
b) JScrollPane.  
c) JSeparador.  
d) JButton.
8. La capacidad de las estructuras denominadas dinámicas...  
Seleccione **una**:  
a) Es infinita.  
b) Se establece en el momento de la creación.  
c) Crece conforme insertamos nuevos elementos.  
d) Depende de los elementos que se inserten.
9. Entre la siguientes creaciones de array hay una que es errónea, ¿sabrías decir cuál?  
Seleccione **una**:  
a) int t1 = new int[20];  
b) long t2; t2=new int[20];  
c) float t3=new float[20];  
d) double t4; t4=new double[100];

10. ¿Cuáles de los siguientes métodos nos permiten insertar elementos de un TreeSet?

Seleccione una:

- a) append()
- b) insert()
- c) add()
- d) offer()

11. En Programación Orientada a Objetos, ¿con qué nombre es conocido el mecanismo que permite crear clases basadas en otras existentes?

Seleccione una:

- a) Polimorfismo.
- b) Derivación.
- c) Herencia.
- d) Encapsulación

12. ¿Con qué nombre son conocidas aquellas clases cuya única función es la de ser superclase en una jerarquía, sin que llegue a haber nunca instancias de ellas?

Seleccione una:

- a) Clases básicas.
- b) Clases abstractas.
- c) Clases jerárquicas.
- d) Ese tipo de clases no tienen sentido y no existen en Java.

13. ¿Cómo podrías acceder al constructor de la superclase de una determinada clase?

Seleccione una:

- a) Mediante la referencia builder.
- b) Mediante la referencia superbuilder.
- c) Mediante la referencia super.
- d) Mediante la referencia this.

14. En Oracle, para declarar un objeto hay que realizar CREATE TYPE tipo\_objeto AS OBJECT ... y además CREATE TYPE BODY tipo\_objeto AS....

Seleccione una:

- a) Verdadero
- b) Falso

15. ¿Cuál de las siguientes declaraciones de métodos de Oracle es un procedimiento?

Seleccione una:

- a) MEMBER PROCEDURE ESCRIBIR(N NUMBER) RETURN NUMBER
- b) MEMBER PROCEDURE ESCRIBIR ()
- c) MEMBER FUNCTION ESCRIBIR () RETURN NULL
- d) MEMBER FUNCTION ESCRIBIR (N NUMBER) RETURN NULL

16. Para que en Oracle se pueda heredar tipo de dato objeto, ¿qué hay que poner?

Seleccione **una**:

- a) `CREATE OBJECT TIPO_OBJETO AS TYPE (T INTEGER) ABSTRACT;`
- b) `CREATE TYPE TIPO_OBJETO AS OBJECT (T INTEGER) NOT FINAL;`
- c) `CREATE TYPE TIPO_OBJETO AS OBJECT (T INTEGER) ABSTRACT;`
- d) `CREATE OBJECT TIPO_OBJETO AS TYPE (T INTEGER) NOT FINAL;`

17. Si se hace referencia al parámetro **SELF** dentro del cuerpo de un método, realmente se está haciendo referencia al objeto que ha invocado a dicho método.

Seleccione **una**:

- a) Verdadero
- b) Falso

18. Un `ResultSet`:

Seleccione **una o más de una**:

- a) Sirve para contener el resultado del comando `SELECT`.
- b) Es un comando de SQL estándar.
- c) Permite procesar el resultado de una consulta `SELECT`.
- d) Ninguna es cierta

19. La consulta: `s.executeUpdate("UPDATE CLIENTE SET teléfono='968610009' WHERE idCLIENTE=3")`

Seleccione **una**:

- a) Elimina datos.
- b) Actualiza datos.
- c) Añade datos.
- d) Ninguna es correcta.

20. El código siguiente:

```
Connection con = DriverManager.getConnection ( "jdbc:odbc:miBD", "miLogin", "miPassword");
```

Seleccione **una**:

- a) Ejecuta una consulta.
- b) Procesa los resultados de una consulta.
- c) Establece una conexión.
- d) Todas son correctas.

## Ejercicio práctico 8 puntos

- Completaréis el proyecto de NetBeans llamado:

### ExProg\_23\_24\_Mayo\_Apellidos\_Nombre

- Lo tenéis que renombrar y poner vuestros apellidos y el nombre como se indica.
- Dentro del proyecto hay 3 paquetes: **controladores**, **modelos** y **ejercicios**.
- **Paquete controladores** está la clase:
  - **Operaciones**: en la que vamos a incluir métodos para operar con el fichero EMPLEADOS.DAT y métodos para controlar los patrones.
- **Paquete modelos**: están las clases:
  - **Empleado**, con sus atributos, getters y setters
  - **comparadorEmpleado**, clase para poder comparar y ordenar los objetos Empleado
- **Paquete ejercicios**: en este paquete hay que modificar las clases de los ejercicios propuestos para realizar lo que se pide en cada ejercicio (renombrad las clases y añadirle vuestro nombre al final):
  - **Ejercicio1\_FicheroObjEmpleadosNombre**
  - **MiObjectOutputStream**: clase para eliminar las cabeceras de un fichero serializado.

## ENUNCIADO

- Tendremos dentro del paquete **ejercicios** un formulario o JFrame llamado **Ejercicio1\_FicheroObjEmpleados** que gestiona las nóminas de los empleados en un **fichero de objetos serializado**, denominado **EMPLEADOS.DAT**. El nombre del fichero será una constante.
- Los datos que tendrá cada empleado serán:
  - **idEmpleado** : numero único que tendrá cada empleado (int). Nº dígitos enteros: 5
  - **nombre** : nombre del empleado (String). Longitud: 50
  - **salario** : salario del empleado (float): 6 enteros y 2 decimales
  - **nroHijos** : nº de hijos (int). Nº dígitos enteros: 2
  - **tipoEmpleado** : posibles valores: **I**: Ingeniero o **V**: Vendedor(String)
  - **turno** : posibles valores: **1**: Mañana, **2**: Tarde, **3**: Nocturno, **4**: Mitad (int)
- Dentro del paquete **controladores**, tendremos una clase llamada **Operaciones**, que tendrá los siguientes métodos:
  - **buscaEmpleado**: que se le pasará como entrada el idEmpleado, buscará en el fichero EMPLEADOS.DAT y nos devolverá como salida un objeto de tipo Empleado cuyo idEmpleado le demos como entrada.

```
public static Empleado buscaEmpleado(int id)
```

- **existeEmpleado:** le pasaremos un idEmpleado y nos devolverá true si existe en el empleado y false si no existe.

*public static boolean existeEmpleado(id id)*

- **grabarEmpleado:** le daremos como entrada un objeto de tipo Empleado y lo grabará el fichero. Hay que tener en cuenta que si el fichero ya existe hay que utilizar **MiObjectOutputStream**, para eliminar las cabeceras.

*public static boolean grabarEmpleado(Empleado v)*

- **correcto:** método para comprobar que una cadena se corresponda con el patrón
- **listarNominas:** visualizará por consola todos los empleados ordenados por nombre.

Para ello declararemos un ArrayList de objetos Empleado.

Pasaremos todos los objetos del fichero a la lista.

Tendremos una clase llamada comparadorEmpleado que nos va a servir para ordenar el ArrayList de Empleados.

```
Collections.sort(listaEmpleados,new comparadorEmpleados());
```

Una vez esté ordenado visualizaremos todos los datos del ArrayList por consola.

| NOMBRE                | NRO.HIJOS | TIPO EMP. | TURNO  | SALARIO | SALARIO NETO |
|-----------------------|-----------|-----------|--------|---------|--------------|
| =====                 | =====     | =====     | =====  | =====   | =====        |
| XXXXXXXXXX..XX        | 99        | INGENIERO | MAÑANA | 2100    | 2000         |
| XXXXXXXXXX..XX        | 99        | VENDEDOR  | TARDE  | 2600    | 2200         |
|                       | .....     |           |        |         |              |
| TOTAL NÓMINAS: 999999 |           |           |        |         |              |

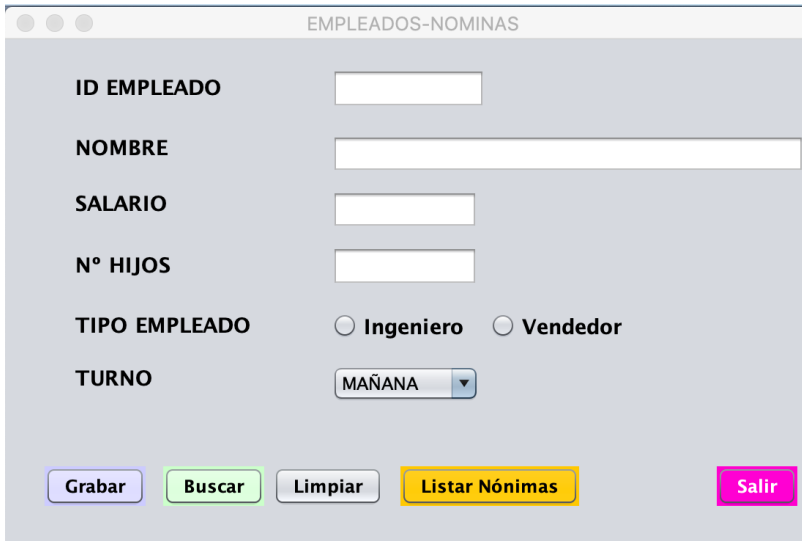
El **total nóminas** es la suma de los salarios netos.

El **salario neto:** para calcularlo se le aplicará un porcentaje de IRPF al SALARIO.

| NRO HIJOS | PORCENTAJE |
|-----------|------------|
| <2        | 15         |
| >=2       | 10         |

- Crearemos los siguientes **patrones** (Públicos):
  - **PatronCadena:** que nos sirva para controlar que solo le tecleen letras, números y espacios en blanco. Como máximo 50.
  - **PatronReal:** que nos sirva para controlar que solo le tecleen: 6 dígitos enteros como máximo , seguido de un punto y después 2 dígitos decimales como máximo.

- **PatronEntero:** que nos sirva para controlar que solo le tecleen: máximo 5 dígitos enteros.
- Tendremos un array de 4 elementos llamado **arrayTurno** con las descripciones del Turno (Mañana,Tarde, Nocturno, Mitad)
- Utilizaremos la clase **Empleados** para poder almacenar objetos.
- El formulario tendrá este formato:



- Los campos del formulario serán:
  - **Id Empleado.:** será un **TextField**. (nombre :txtIdEmpleado)
  - **Nombre:** será un **TextField**. (nombre : txtNombre)
  - **Salario:** será un **TextField** (nombre:txtSalario)
  - **Nº Hijos:** será un **TextField**. (nombre : txtNroHijos)
  - **Tipo Empleado:** será dos **JRadioButton** (**jRadioIngeniero**, **jRadioVendedor**), agrupados ambos en un **ButtonGroup** (**grupoTipoEmpleado**). En el fichero almacenaremos I o V
  - **Turno.:** será un **JComboBox** llamado **cboTurno**), en el que se podrá elegir: Mañana/Tarde/Nocturno/Mitad. En el fichero almacenaremos 1,2,3 o 4
  - Botones:
    - **Grabar:** nombre: **btnGrabar**
    - **Buscar:** nombre: **btnBuscar**
    - **Listar:** visualiza en consola los datos de los empleados.
    - **Limpiar:** nombre: **btnLimpiar**
    - **Salir:** nombre: **btnSalir**

- **Botón Grabar:**

- Crearemos un método: **algunoVacio**: que controle si algún campo está vacío, nos devolverá true si alguno está vacío y false si no está vacío ninguno. Todos los campos deben tener valor.
- Llamaremos al método y visualizaremos con JOptionPane el mensaje: LOS CAMPOS NO PUEDEN ESTAR VACÍOS y vuelve el foco a txtIdEmpleado.

**JOptionPane.showMessageDialog**(null, "Los campos no puede estar vacíos");

- Comprobaremos si el empleado existe llamando al método: **Operaciones.existeEmpleado**, si devuelve TRUE, visualizaremos con JOptionPane el mensaje: EL EMPLEADO EXISTE y vuelve el foco a txtIdEmpleado.
- Controlaremos que los tipos de los campos sean correctos mediante **Operaciones.correcto(valor,patrón)**. Para el idEmpleado y el nº de hijos el patrón: **PatronEntero**, para nombre, el patrón: **PatronCadena** y para las unidades: **PatronReal**. Si alguno no coincide con el patrón, visualizaremos con JOptionPane un mensaje indicando cuál es el campo erróneo y vuelve el foco a txtIdVenta.
- Grabaremos el empleado en el fichero mediante el método: **Operaciones.grabarEmpleado**, dándole como entrada un objeto de tipo Empleado. Si se graba correctamente, visualizaremos el mensaje "REGISTRO GRABADO" mediante un JOptionPane.showMessageDialog. Si no se graba, visualizaremos el mensaje "ERROR AL GRABAR REGISTRO" mediante un JOptionPane.showMessageDialog.
- Limpiaremos los campos del formulario, para ello crearemos el método **limpiarDatos**.
- El foco volverá al txtIdVenta.

- **Botón Buscar:**

- Si el IdEmpleado está vacío: visualizaremos un mensaje con JOptionPane.showMessageDialog: "ID NO PUEDE ESTAR VACÍO" y el foco volverá al txtIdEmpleado.
- Comprobaremos si el empleado existe, pasándole el idEmpleado al método: **Operaciones.existeEmpleado**, si devuelve FALSE, visualizaremos con JOptionPane el mensaje: "REGISTRO NO EXISTE" y el foco volverá al txtIdEmpleado
- Si el idEmpleado existe, visualizaremos los datos del empleado. Para ello realizaremos un método llamado **visualizarEmpleado**, al que le daremos como entrada un objeto de tipo Empleado y visualizará cada campo en el formulario.

- **Botón Limpiar:** limpiará todos los campos del vino que están en el formulario.

- **Botón Listar:** llamaremos al método **Operaciones.listarNominas** que visualizará el contenido del fichero de objetos ordenado por nombre. Para el campo tipoEmpleado, saldrá INGENIERO o VENDEDOR. Para el campo turno, saldrá MAÑANA/TARDE/NOCTURNO/MITAD

- **Botón Salir:** Finaliza el programa.

- El código fuente Java debe incluir **comentarios** del tipo Javadoc en cada indicando su utilidad.