Examen del 2º Trimestre

Modelo 2022_2023_A

1. Instrucciones

Para el presente examen es necesario estar en disposición de los siguientes conocimientos y destrezas:

- E/S por consola
- Manejo de tipos de datos básicos
- Manejo de cadenas de caracteres
- Estructuras de control de flujo
- Arrays
- Colecciones y diccionarios
- Fundamentos de POO
- Herencia y abstracción

Debes seguir las siguientes instrucciones para realizar el examen. El incumplimiento de una de ellas puede implicar suspender el examen.

- Debes seguir las normas y buenas prácticas recomendadas como programador.
- **El alumno que copie o se deje copiar, tendrá una calificación de cero**.
- **Debes subir al aula virtual un fichero zip** con el proyecto creado. El nombre del fichero comprimido debe tener el siguiente formato: *Apellido1Apellido2_Nombre.zip*. Por ejemplo: `GarcíaPérez_Ana.zip`
- El anterior fichero deberá poder importarse sin problemas en el IDE (más allá de tener que cambiar las versiones del entorno de ejecución).
- Se debe obtener, como mínimo, un 50% de la calificación de los ejercicios 3 y 4.
- **El código debe poder compilarse y ejecutarse.**
- Si tienes dudas sobre cómo implementar alguna funcionalidad, toma la decisión que te parezca más adecuada.
- Tras finalizar el examen y subirlo **debes borrar del equipo del aula** todo código, proyecto, PDF y resto de ficheros y documentación que hayas utilizado para realizar el examen.

2. Ejercicios

Ejercicio 1 (*3,7 puntos*)

Implementa las tres siguientes clases en un paquete denominado ***ejercicio1.modelo***. En cada una de las clases, se definen una serie de características que deben cumplirse. Estas clases ayudan a gestionar los trabajadores de una empresa. Dichos trabajadores pueden ser **fijos** o **eventuales**. Aunque este ejercicio te pueda llevar un tiempo extenso de implementación, dedícale el tiempo oportuno sabiendo que los demás son más concretos y se basan en estas clases.

✔ Clase **Trabajador**: no debemos poder crear objetos de esta clase.

- Tiene que tener los siguientes atributos: **`nombre`**, **`apellidos`**, **`fechaAlta`** (define la fecha de alta del trabajador en la empresa). Estos atributos deben ser accesibles de forma directa **únicamente** (es decir, con **`this.`***`atributo`*) desde la propia clase Trabajador, desde las clases que deriven de ella o desde clases que estén en el mismo paquete.
- Debe tener un constructor con todos los atributos.
- Debe tener un constructor únicamente con el nombre y los apellidos del trabajador. En ese caso, la fecha de alta se seteará al día de mañana. Por ejemplo, si en el momento de creación del objeto es 10/03/2000, se seteará a 11/03/2000.
- Una vez construido el objeto, se podrá acceder desde cualquier clase a los valores de todos sus atributos.
- Una vez construido el objeto, NO se podrá modificar el valor de ningún atributo ni aunque sea desde la propia clase Trabajador.
- Además, debemos tener los siguientes métodos en la clase:
  - **`getNombreCompleto()`**: devolverá la concatenación del nombre y los apellidos separados por un espacio.

- **`mesesTrabajados()`**: devolverá el número de meses que lleva el trabajador en la empresa.
- **`getSalario()`**: deberá ser implementado por las hijas que hereden de esta clase y devolverá el salario que gana el Trabajador según lógica propia de la clase hija. Podrá ser accedido desde cualquier clase.
- **`toString()`**: por cada Trabajador mostrará el siguiente texto (con los datos personalizados de nombre, apellidos y fecha de alta). La fecha tiene el formato español de día, mes y año separado por guiones. Por ejemplo: 10-03-2000.

```
Ibai Llanos Garatea trabaja en la empresa desde el 10-01-2020
```

- Dos trabajadores son el mismo trabajador (son iguales) si tienen los mismos apellidos y fecha de alta (independientemente de mayúsculas, minúsculas y tildes).
- Todo Trabajador tiene un sueldo base de 1000 €.

✔ Clase **Fijo**: es un Trabajador que además tiene las siguientes características.

- No se debe poder heredar de esta clase.
- Tiene un **`bonusCategoria`** de tipo numérico decimal que únicamente debe ser accesible directamente desde la propia clase. Debe ser igual o superior a 300 €. Si fuese inferior a 300 €, el bonus se establecerá a 300 € directamente.
- Debe tener un constructor con todos los atributos.
- El bonus de categoría no se puede obtener mediante *getter* pero sí se puede modificar desde cualquier clase.
- Además, debemos tener los siguientes métodos en la clase:
  - **`getTrienios()`**: devolverá el número de trienios que ha acumulado el trabajador Fijo. Cada trienio son 3 años completo de trabajo en la empresa. Este método únicamente se puede llamar desde la propia clase Fijo.
  - **`getSalario()`**: se calcula mediante la siguiente fórmula:

    ```
    Sueldo Base + bonus de categoría + trienios * 100;
    ```

  - **`toString()`**: incluirá el mensaje de todo trabajador junto con el salario.

```
Ibai Llanos Garatea trabaja en la empresa desde el 10-01-2020. Tiene un salario de 2.200,00 euros
```

✔ Clase **Eventual**: es un Trabajador que además tiene las siguientes características.

- Tiene que tener los siguientes tres atributos: **`fechaFin`** (cuándo finaliza el contrato), **`email`** (de tipo texto) y **`tiempoContrato`** (solamente puede ser uno de los siguientes valores: trimestral, semestral y anual). Todos los atributos únicamente deben ser accesibles directamente desde la propia clase.
- Debe tener un constructor que reciba por parámetro el nombre, los apellidos, la fecha de alta, el email y el tiempo de contrato. La **fecha de fin** se seteará según la siguiente lógica:
  - Si el tiempo de contrato es trimestral, equivaldrá a 3 meses después de la fecha de alta. Por ejemplo, si la fecha de alta es el 10/03/2020, se seteará a 10/06/2020.
  - Si el tiempo de contrato es semestral, equivaldrá a 6 meses después de la fecha de alta.
  - Si el tiempo de contrato es anual, equivaldrá a 1 año después de la fecha de alta.
- La fecha de fin y el email se pueden obtener mediante getter pero el tiempo de contrato no se puede consultar ni modificar una vez creado el objeto desde el exterior.
- Además, debemos tener los siguientes métodos en la clase:
  - **`getSalario()`**: se calcula aplicando un 10% de subida sobre el sueldo base. Todos los eventuales cobran lo mismo.
  - **`toString()`**: debe ser igual al siguiente con los datos personalizados.

```
Rafael Nadal Parera tiene un contrato temporal de tipo trimestral hasta el 2018-08-01 y cobra 1.210,00 euros.
```

- De forma natural, predeterminada, los trabajadores eventuales se ordenan por fecha de fin de contrato. Primero los que acaban antes el contrato y después los que terminan más tarde el contrato.

Ejercicio 2 (*1,2 puntos*)

En este ejercicio, dentro del paquete ***ejercicio2***, hay que crear una clase repositorio, llamado `TrabajadorRepositorio`, que nos va a permitir crear trabajadores fijos y eventuales, así como recuperarlos para usarlos en otros ejercicios. Partiendo del código proporcionado, completa los métodos con el código fuente necesario para conseguir lo que deseamos.

```
public class TrabajadorRepositorio {
	private static List<Fijo> fijos;
	private static List<Eventual> eventuales;

	public static List<Fijo> getFijos() {
		//...
	}

	public static List<Eventual> getEventuales() {
		//...
	}

	public static List<Trabajador> getTrabajadores() {
		//...
	}
}
```

- El método `getFijos()` debe permitir obtener una lista con los siguientes trabajadores **fijos**:

Nombre	Apellidos	Fecha de alta	Bonus
Lydia	Valentín Pérez	01/05/2000	600
Margarita	Salas Falguera	08/10/1985	1000,5
Ibai	Llanos Garatea	10/01/2020	1000
Luka	Modric	01/06/2006	800

- El método `getEventuales()` debe permitir obtener una lista con los siguientes trabajadores **eventuales**:

Nombre	Apellidos	Fecha de alta	Email	Tiempo de contrato
Cristina	López Pérez	11/05/2020	cristina@twitch.es	Anual
Rafael	Yuste	01/08/2021	rafael@ciencia.es	Semestral
Nadia	Comaneci	01/09/2018	nadia@deporte.es	Anual
Rafael	Nadal Perera	01/05/2018	rafael@deporte.es	Trimestral

- El método `getTrabajadores()` debe devolver una lista que incluya **todos** los Trabajadores (fijos y eventuales) que acabas de crear (todos los trabajadores en una única lista).

Ejercicio 3 (*3,6 puntos*)

Debes crear una clase llamada `AppEjercicio3` dentro del paquete ***ejercicio3***. Esta clase nos va a permitir mostrar la información de los distintos trabajadores. Debes implementar los métodos que se indican con la funcionalidad pedida. Para ello, debes partir del siguiente método `main`:

```
public static void main(String[] args) {
	List<Trabajador> trabajadores = // Recupera todos los trabajadores del repositorio

	mostrarEstadisticas(trabajadores);
	mostrarFijosOrdenados(trabajadores);
	mostrarEventualesOrdenados(trabajadores, 2018);
	mostrarTrabajadorConSueldoSuperior(trabajadores);
}
```

- El método `mostrarEstadisticas()` debe indicar la suma del salario de todos los trabajadores y la media de los salarios de todos los trabajadores. La salida debe ser la siguiente (con los valores correctos).

```
SALARIOS
********
El total invertido en salarios es de 15.140,50 euros
La media salarial es de 1.892,56 euros
```

- El método `mostrarFijosOrdenados()` debe mostrar la información de los trabajadores fijos (únicamente) **ordenados** según los siguientes criterios: primero de más a menos salario y, en caso de igualdad en el salario, por orden alfabético de nombre y apellidos (concatenación del nombre con los apellidos). No debes modificar la clase Fijo para implementar esta funcionalidad.

```
TRABAJADORES FIJOS
******************
Margarita Salas Falguera trabaja en la empresa desde el 08-10-1985. Tiene un salario de 3.300,50 euros
Luka Modric trabaja en la empresa desde el 01-06-2006. Tiene un salario de 2.400,00 euros
```

- El método `mostrarEventualesOrdenados()` debe mostrar la información de los trabajadores eventuales (únicamente) **ordenados** según la fecha de fin del contrato. Únicamente se deben mostrar aquellos que terminen contrato después del año que se le pasa por parámetro (el 2018 en nuestro ejemplo de código). Se indica un ejemplo de salida por pantalla (los datos no tienen porqué coincidir con la imagen).

```
TRABAJADORES EVENTUALES
**********************
Nadia Comaneci tiene un contrato temporal de tipo anual hasta el 2019-09-01 y cobra 1.210,00 euros.
Cristina López Pérez tiene un contrato temporal de tipo anual hasta el 2021-05-11 y cobra 1.210,00 euros.
```

- El método `mostrarTrabajadorConSueldoSuperior()` debe mostrar el trabajador (que puede ser fijo o eventual) con el mayor sueldo. En caso de igualdad de sueldos, se mostrará el primero que esté en la lista.

```
TRABAJADOR CON EL MAYOR SUELDO
******************************
Margarita Salas Falguera trabaja en la empresa desde el 08-10-1985. Tiene un salario de 3.300,50 euros
```

Ejercicio 4 (*1,5 puntos*)

Implementa una clase llamada `AppEjercicio4`, en el paquete ***ejercicio4***, que obtenga todas las direcciones de **email** de los trabajadores **Eventuales** y muestre un resumen de cuántas veces aparece cada dominio. Entendemos por dominio todo lo que aparece detrás de la arroba (@). Se indica un ejemplo de salida por pantalla de la información calculada en este ejercicio.

```
DOMINIOS
********
ciencia.es: 1
twitch.es: 1
deporte.es: 2
```