Nombre y Apellidos:

**Instrucciones.** La tarea constará de diferentes puntos que crearan un modelo de clases y un programa que desarrollaremos en la unidad siguiente.

La tarea se entregará en el aula virtual en un ZIP **tareaUnidad12NombreApellidos.Zip** conteniendo los ejercicios contestados. Crear un proyecto en eclipse **tareaUnidad12NombreApellidos** con el modelo que se ofrece en la práctica.

**RAE 7.** Desarrolla programas aplicando características avanzadas de los lenguajes orientados a objetos y del entorno de programación.

**Criterios de evaluación:**

a) Identificar los conceptos de herencia, superclase y subclase. Indicadores 76

d) Crear clases heredadas que sobrescriban la implementación de métodos de la superclase. Indicadores 71

e) Diseñar y aplicar jerarquías de clases. Indicadores 72

g) Realizar programas que implementen y utilicen jerarquías de clases. Indicadores 74 y 77

# Detalles de la tarea de esta unidad.

**Enunciado.**

En esta tarea vamos a aplicar lo aprendido en el tema 10, desarrollando dos patrones de diseño, Composite y Strategy sobre nuevas clases y patrones. La estructura del proyecto será la siguiente.

## Añadiendo nuevos patrones

### Patrón Strategy (4 puntos)

Indicador 68 , 71, 72, 74 y 77

Nombre y Apellidos:

Como **hemos avanzado en el apartado anterior**, vamos a **definir la función que calcula los sueldos** con un **Function<Trabajador,Double>**. Es **el primer paso para introducir el patrón Strategy funcional** en nuestra aplicación como hemos visto en este Tema.

## Sueldos

Recordamos **el cálculo de sueldo para los diferentes Trabajadores** del sistema:

Trabajador Freelance:

*Sueldo = sueldoHoras* numHoras*

Trabajador Anónimo

*Sueldo = SueldoBase*

EstudianteEnPracticas

*Sueldo= 0.0*

Resto de Trabajadores

*Sueldo del Trabajador = SueldoBase + %sobre el Sueldobase + importe por antigüedad.*

En esta **tabla para definir los porcentajes y los pluses**, que debemos introducir y obtener ahora de **CategoriaEnum y AntiguedadEnum** para realizar el **calculo de sueldos**.

Nombre y Apellidos:

| **Sueldo base** | 607 € |
|---|---|
| **EMPLEADO**<br>**ENCARGADO**<br>**DIRECTIVO**<br>**TÉCNICO**<br>FREELANCE<br>ANONIMO<br>ESTUDIANTE | +15% sueldo base<br>+35% sueldo base<br>+60% sueldo base<br>+40% sueldo base<br>+0% sueldo base<br>+0% sueldo base<br>+0% sueldo base |
| **NOVATO**<br>**MADURO**<br>**EXPERTO** | +150 €<br>+300 €<br>+600 € |

## Implementación de patrón Strategy funcional

**Para implementar el patrón Strategy funcional** usaremos una clase llamada **EstrategiaSueldos.java** en **el paquete modelo.sueldos**. En **esta clase definiremos dos propiedades estáticas de tipo Function<Trabajador,Double>** y **dos métodos que también implementarán en su signatura** este **interfaz function**, serán compatibles con el, para **realizar el cálculo de sueldos** de los **diferentes tipos de trabajador.**

**Propiedades estáticas: sueldoAnonimo, sueldoEstudiante** serán **dos expresiones lambda** siguiente el interfaz Function definido anteriormente.

**Métodos estáticos: sueldoTrabajador**, y **sueldoFreelance** que deben **ser compatibles con el interfaz Function** definido anteriormente.

**Usaremos estos métodos en la creación de los Trabajadores**, dependiendo del **tipo de categoría, en el builder**. Podríamos **añadirlos también dentro del método .categoria del builder**, **seleccionando el Function adecuado para el sueldo** de esa **categoría**. Pero prefiero que lo **dejéis en el builder porque es más visual como cambiamos el comportamiento de una clase** en tiempo de ejecución con una **Estrategia.**

**Por ejemplo**, si el empleado es un **estudiante en el builder añadimos su método adecuado**, en este caso una **propiedad estática** de la clase

Nombre y Apellidos:

**EstrategiaSueldos**. Lo debéis hacer dentro del Stream que genera la lista de empleados dentro del **maptoObject del int Stream**. Construid **una expresión lambda de bloque**. Seguimos **usando el IntStream para generar los trabajadores** porque es el **generador del Id**.

```
if (cat==CategoriaEnum.CATEGORIA_ESTUDIANTE ) {

builder.sueldoFunction(EstrategiaSueldos.sueldoEstudiante)
```

Indicador 68 , 71, 72, 74 y 77

## Patrón Composite (4 puntos)

Vamos **a añadir a nuestro modelo una nueva clase Empresa** que entre otras cosas va a **contener a los Empleados por agregación**, y **Empresas subsidiarias también por agregación y usando el patrón Composite** visto en el tema.

Nombre y Apellidos:

## Implementación del Patrón Composite para nuestra nueva clase Empresa.

Vamos a crear una **nueva clase Empresa que pueda ser una empresa simple**, ella misma, o **contener empresas subsidiarias**, y ser una **empresa matriz.** Por ejemplo, **Alphabet Inc.,** es la empresa Matriz de **Calico, CapitalG, DeepMind, Google, Google Fiber, GV, etc.**

## Datos

1. Guardaremos **la siguiente información** sobre la empresa creando **getters y setters**
   **(0,25 puntos)**

```
private Integer idEmpresa;
private String razonSocial;
private Double capitaSocial;
private String Direccion;
private String Telefono;
private Double gananciasBruto=0.0;
```

Además para controlar si la empresa es subsidiaria o no:

```
private boolean esSubsidiaria= false;
```

Ádemás **mantendremos un Set y un List**. Un **TreeSet** con **las empresas subsidiarias** llamada **subsidiarias**. **Un ArrayList con los trabajadores** llamada **trabajadores.** Recordar que **programamos para las abstracciones. Necesitareis que Empresa implemente Comparable<Empresa> por el TreeSet.** Haced la comparación por id.

Nombre y Apellidos:

2. Añadimos dos métodos **addTrabajador y removeTrabajador**, para quitar y añadir trabajadores de la empresa. **La condición importante:** no se puede añadir trabajadores a la empresa matriz, se escribe por consola el mensaje de error `"No se puede añadir trabajador a la empresa matriz"` **(0,5 puntos)**

3. **Añadimos dos métodos addSubsidiaria y removeSubsidiaria**, para añadir y quitar subsidiarias. . **La condición importante:** No se puede añadir una empresa subsidiaria a una subsidiaria se escribe por consola el mensaje de error `"No se puede añadir una empresa subsidiaria a una empresa subsidiaria".` Podéis marcar la **empresa subsidiaria** con una **propiedad privada booleana esSubsidiaria**.
**(0,5 puntos)**

## Calculos

1. **Si la empresa es simple** las **ganancias** se almacenarán en **su propia propiedad** `gananciasBruto.` Si **no se almacenara en las empresas subsidiarías**. Lo mismo para el **capital social**, que se calculara con `calculaCapitalSocial.` Estos **métodos los implementaremos haciendo uso de la API Stream**. **(0,25 puntos)**

2. Igualmente, **sólo se almacenan trabajadores en las empresas subsidiarias**, con lo que deberéis realizar un método **getListaTotalTrabajadores.** Si **la empresa es subsidiaria** me devolverá **su lista trabajadores**. Si la **empresa es matriz** debemos **recorrer todas las subsidiarias** y unir todas sus listas. **Se puede hacer con una solo stream() de subsidiarias y usando flatMap. (0,5 puntos)**

3. **No crearemos la Empresa** con un **new**, se creará con un **método estático creaEmpresa** con los siguientes parámetros. Si la empresa es matriz el parámetro trabajadores será un **Optional vacio**. Si **es subsidiaria un Optional** con una **lista** de tipo List<Trabajador> **(0,5 puntos)**

```
creaEmpresa  (Integer idEmpresa,String razonSocial, Double capitaSocial
```

Nombre y Apellidos:

```
                    , String direccion, String telefono, Double gananciasBruto,
Optional<List<Trabajador>> optTrabajadores))
```

4. Creamos **la clase GeneraEmpresa en Debug**, con el método generaAlphabet(), que nos generará la empresa **Alphabet Inc.,** y **tres subsidiarias DeepMind, Google, Google Fiber . Añadimos una lista con trabajadores aleatorios** a cada subsidiaria 30, 50 y 20 por ejemplo**. La dirección, los teléfonos están en internet. Podéis consultarlos en Internet si queréis.**
   **(0,5 puntos)**

   No introduzcais muchos empleados porque se llena el buffer de la consola y se borra parte de la información.

5. Crear el método **en la clase GeneraEmpresa, listarEmpresa.** El método listará la información **primero de la empresa y luego** de las **subsidiarias**. Necesitareis **un toString() en Empresa** que **no contenga lista de trabajadores ni de subsidiarias.** Si contendrá las **ganancias brutas** y el **capital social** calculado correctamente.

   **(0,5 puntos)**

## En App.java

Indicador 68 , 71, 72, 74 y 77

**(2 puntos)**

**Es la copia de MainTrabajador**, nuestra **nueva clase principal**. Mantenemos **el menú para los trabajadores y añadiremos alguna cosa más** que ahora indicaremos. Primero creamos la Empresa llamando al método empresa, generaGoogle().

Mantenemos **los elementos de menú anteriores y añadimos** estas **nuevas opciones**:

Nombre y Apellidos:

**Escriba MAXSTRAB para obtener el Trabajador con máximo sueldo**

**Escriba MINSTRAB para obtener el mínimo de Sueldo de Empleados**

Para estos dos siguiente elementos de menú debéis incluir un método estático imprimirEmpleadosAgrupados en GeneraEmpleados

**Escriba TRABCAT para obtener los trabajadores impresos pero agrupados por categoría**

**Escriba TRABANT para obtener los trabajadores impresos pero agrupados por antigüedad**

Para **estos dos siguiente elementos** de menú necesitareis métodos en GeneraEmpresa imprimeEmpresaResumen e imprimeEmpresa

**Escriba INFEMP para obtener un resumen de la información de la empresa listada sin subsidiarias.**

**Escriba INFEMPS para obtener la información de la empresa listada con todas sus subsidiarias**

Ejemplos de ejecucion

MAXSTRAB

```
El empleado com Máximo de sueldo  es:Trabajador [trabajador_id=27,
nombre=JUAN González González, edad=18, categoria=CATEGORIA_TECNICO,
antiguedad=ANTIGUEDAD_EXPERTO, fecha_alta=16 del mes de Noviembre de 2018,
sueldo=3569.4, direccion=Plaza de Caídos en la Guerra civil,
cuenta=CuentaBancaria [titular=27, Iban=IBANES2184998309124577705,
comisionMantemimiento=91.0, tipoInteres=0.02]]
```

MINSTRAB

```
El empleado con Mínimo de sueldo de trabajadores es:Trabajador
[trabajador_id=5, nombre=FRANCISCO Fernández González, edad=64,
categoria=CATEGORIA_ESTUDIANTE, antiguedad=ANTIGUEDAD_NOVATO, fecha_alta=3
del mes de Febrero de 2008, sueldo=0.0, direccion=Plaza de Beladíez,
cuenta=null]
```

Nombre y Apellidos:

TRABCAT

Listado de empleados agrupados por categoria

Su listado de trabajadores es:

**DIRECTIVO=[Trabajador [trabajador_id=10, nombre=FRANCISCA Martínez Rodríguez, edad=63, categoria=CATEGORIA_DIRECTIVO, antiguedad=ANTIGUEDAD_MADURO,**

**FREE_LANCE=[Trabajador [trabajador_id=2, nombre=DOLORES Fernández Fernández, edad=54, categoria=CATEGORIA_FREE_LANCE, antiguedad=ANTIGUEDAD_NOVATO, fecha_alta=15**

**Y así para todas las categorías**

**TRABANT**
l Mínimo de sueldo de trabajadores es:

Su listado de trabajadores es:

**EXPERTO=[Trabajador [trabajador_id=7, nombre=MARIA JOSE Fernández García, edad=59, categoria=CATEGORIA_FREE_LANCE, antiguedad=ANTIGUEDAD_EXPERTO**

**MADURO=[Trabajador [trabajador_id=5, nombre=JOSE LUIS Sánchez Rodríguez, edad=43, categoria=CATEGORIA_FREE_LANCE, antiguedad=ANTIGUEDAD_MADURO, fecha_alta=1 NOVATO=[Trabajador [trabajador_id=1, nombre=MANUEL Martínez González, edad=34, categoria=CATEGORIA_ENCARGADO, antiguedad=ANTIGUEDAD_NOVATO,**

INFEMP

Información Resumen de la empresa listada sin subsecciones.

Empresa [idEmpresa=1, razonSocial=Alphabet Inc, capitaSocial=3000000.0, Direccion=2100 N Lyon Ave, Springfield, MO 65803, United States, Telefono=+14174229126, gananciasBruto=4.32E9]

INFEMPS

Información de la empresa listada con todas sus subsidiarias

Empresa [idEmpresa=1, razonSocial=Alphabet Inc, capitaSocial=3000000.0, Direccion=2100 N Lyon Ave, Springfield, MO 65803, United States, Telefono=+14174229126, gananciasBruto=4.32E9]

Empresa [idEmpresa=1, razonSocial=Google Fiber, capitaSocial=1000000.0, Direccion=267 8th St, San Francisco, CA 94103, United States, Telefono=+18009327277, gananciasBruto=2.0E7]

Nombre y Apellidos:

```
Empresa [idEmpresa=1, razonSocial=Google Inc, capitaSocial=1000000.0,
Direccion=Mountain View, CA 94043, United States, Telefono=+650-253-0000,
gananciasBruto=4.0E9]

Empresa [idEmpresa=1, razonSocial=Deep Mind, capitaSocial=1000000.0,
Direccion=Mountain View, CA 94043, United States, Telefono=+650-253-0000,
gananciasBruto=3.0E8]
```

## Condiciones de Entrega

**Entregamos un zip que contendrá:**

1. El proyecto **final con**. El proyecto completo funcionando.

Nombre y Apellidos:

2. **Pantallazos probando las seis nuevas funcionalidades del menú**, de manera que se vea al menos **un trabajador completo y una empresa completa**. Podéis **añadirme el texto de la ejecución también al pdf**. Lo entregamos en un fichero pdf de nombre TareaTema8NombreApellidos.pdf.