RETO

Imagínate el siguiente contexto:

"Para una entrevista de trabajo te piden que diseñes un diagrama de clases UML, te reúnes con tus antiguos compañeros y compañeras de clase para practicar, cambiar impresiones y que te ayuden en el diseño del diagrama"

El reto consiste en que practiques y aprendas a realizar un diagrama de clases **UML** en el que intervienen **más de dos entidade**s de forma que hay algún tipo de relación entre ellas.

Utiliza **algún software de modelado**, por ejemplo, los citados en la unidad, o cualquier otro que sepas o quieras aprender a manejar, para crear un **sistema sencillo** de registro y gestión de empleados utilizando los principios de la programación orientada a objetos en Java y notación **UML** para el diseño de clases.

El sistema permitirá a los usuarios registrar nuevos empleados, actualizar información existente, consultar datos de empleados y generar informes principalmente.

Se deben definir las **clases, atributos y métodos que consideres** relevantes, así como las **relaciones** de asociación, agregación, herencia y visibilidad de cada elemento del sistema.

NOTA: **NO SE PIDE CÓDIGO JAVA,** simplemente ahora mismo estamos **diseñando**, más abajo tienes **ejemplos de lo que se pide** en este reto.

Las **funcionalidades** del sistema son las siguientes: (Proporciona el método apropiado en la clase correspondiente(**solo se pide la cabecera de cada método**, fíjate de los **ejemplos** que hay más abajo)

- Registro de **empleados** con sus datos personales y laborales.
- Actualización de información de **empleados** (datos personales y laborales).
- Consulta de datos de **empleado**(todos).
- Los empleados tendrán una **categoría** profesional.
- Consulta de **categorías** profesionales existentes.
- Registros de **departamentos** a los que puede pertenecer el empleado.

Tienes que pensar en.....cuáles serán las **relaciones mas apropiadas** entre las entidades que has detectado, los atributos y los métodos necesarios según las funcionalidades indicadas.

Una vez tengas el diagrama **UML,** en un documento de **texto**, redacta en forma de informe la **explicación** de los tipos de relaciones entre cada una de las entidades (herencia, agregación o composición, etc.) y el porqué , así como la funcionalidad de cada método que has incluido.

Sube a tu cuenta en GitHub:

- el **archivo en el que has generado el UML**
- el **documento de texto con la explicación**
- **y una captura de imagen** de tu sistema de clases **UML**. (Utiliza la herramienta **Recortes** de Windows para la **captura** (mira este enlace si nunca la has usado..

*******Según el **sistema que diseñes,** debes establecer **unas restricciones y especificaciones** que **justificarán** cada una de las relaciones entre las clases, **puede haber diferentes sistemas UML, para este mismo reto**, pero dependerá de la filosofía que siga el sistema, el relacionar de una forma u otra las entidades que intervienen**.*****

Las **dudas se pueden plantear en el foro de dudas de la unidad 3, en el hilo "Aquí solo las DUDAS del Reto propuesto en el Foro llamado DESAFIA TU LÓGICA_UD3"**

****Aprender a diseñar diagramas UML, te servirá para tu proyecto fin de ciclo**

Clases:

-Empleado
-Categoría
-Departamento

Atributos:

Empleado: nombre, id, edad, telf, numSS
Categoría: nombre, posiciónJerárquica, sueldo
Departamento: nombre, número, ubicación

Métodos:

Empleado: crearEmpleado(), borrarEmpleado(), consultarEmpleado(), editarEmpleado()
Categoría: añadirCategoria(), borrarCategoria(), consultarCategoria(), editarCategoria()
Departamento: crearDpto(), borrarDpto(), editarDpto(), consultarDpto()

Relaciones:

Empleado — Categoría: Una categoría puede tener cero empleados o muchos empleados y un empleado solo puede pertenecer a una categoría.

Empleado — Departamento: Un empleado debe pertenecer como mínimo a un departamento y cómo máximo a todos los posibles, mientras que un departamento puede estar compuesto por cero o por tantos empleados como haya.