ACTIVIDAD II - UNIDAD II



Estudiante: Mario Felipe Goenaga Lozano

AF17202: Programación Orientada a Objetos

Tutor: Dr. John Carlos Arrieta Arrieta

Programa de Ingeniería de Software

Facultad de Ciencias Económicas e Ingeniería

Universidad de Cartagena

07 de Abril de 2024

**Introducción.**

El presente trabajo escrito da cuenta del proceso de conversión de un diagrama de clases, creado en la fase previa de diseño, en clases implementadas mediante código fuente en Java. Es decir, se traducirá el diseño conceptual en una representación ejecutable del sistema. Así, el diagrama de clases construido en la etapa anterior del proyecto ofrece una vista organizada de las entidades y sus relaciones en el sistema: cada “clase” representa un elemento del sistema con atributos y métodos asociados, y las relaciones entre las clases describen la interacción y dependencia entre ellas.

Esta actividad es parte crucial en el llamado ciclo de vida del desarrollo de software, que en este caso, hace referencia a una aplicación para la gestión de proyectos en una empresa.

Luego, el objetivo principal de esta actividad es realizar la implementación concreta de las clases definidas con anterioridad en el diagrama, utilizando el lenguaje de programación Java. Esto implica traducir los conceptos y relaciones del diseño a código funcional, siguiendo las prácticas y convenciones recomendadas en el desarrollo de software, condensadas en la bibliografía ofrecida en la unidad correspondiente de la asignatura de Programación Orientada a Objetos.

**Justificación.**

La traducción del diagrama de clases UML, que visualiza la concepción de un sistema, a clases en código Java implica la materialización del diseño inicial en una aplicación funcional a partir de una consistencia entre la idea original y su ejecución práctica, garantizando que el producto final satisfaga los requerimientos del cliente. El uso de Java como lenguaje de programación permite acceder a ventajas: la capacidad de transportar el software a diferentes plataformas, su compatibilidad con otros sistemas y el acceso a una amplia gama de herramientas que simplifican el proceso de desarrollo y administración del sistema. Asimismo, Java facilita explorar los fundamentos de la POO, incluyendo conceptos como la encapsulación, la herencia y el polimorfismo.

**Objetivos.**

1. Comprender y aplicar los conceptos de la programación orientada a objetos en un contexto práctico.
2. Aprender a traducir los elementos y relaciones de un diagrama de clases UML a clases implementadas en código Java, desarrollando habilidades de codificación y diseño de software.
3. Practicar el uso de las herramientas y técnicas adecuadas para la implementación de clases en Java.

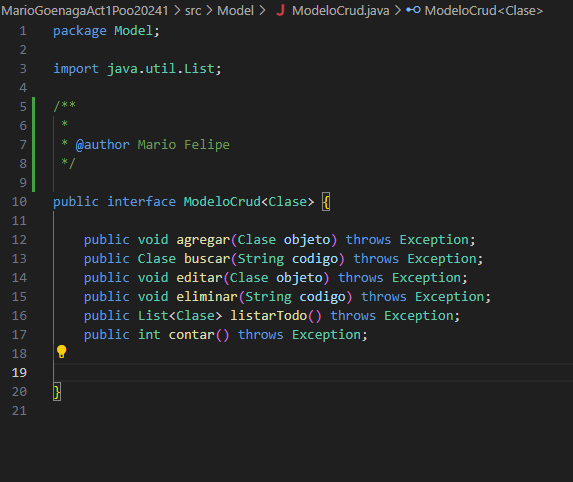
**Lista de requerimientos funcionales.**

* Debe mantener un registro de los proyectos de la empresa, que contenga de ellos, los siguientes datos: nombre en clave, denominación comercial, fecha de inicio, fecha de finalización, estado actual.
* Debe gestionar los recursos humanos de la empresa, permitiendo visualizar una ficha de información básica por cada empleado.
* Debe permitir la asignación de promotor a proyecto teniendo en cuenta un área de especialización.
* Debe permitir dividir el proyecto en diferentes tareas, cada una de ellas debe tener una descripción, tipo, fecha de inicio estimada, fecha de inicio real, duración estimada, duración real.
* Debe permitir a los empleados inscribirse o apuntarse en las diferentes tareas.
* Debe permitir generar y almacenar documentos asociados a las tareas emergentes del proyecto, a su vez estos documentos generarán una serie de versiones, cada una mas actualizada que la anterior los cuales también serán almacenados por la aplicación.
* Debe garantizar que los empleados asignados pertenezcan a la empresa y su estado actual sea activo.
* Deber hacer una discriminación de roles para definir los permisos y accesos para modificación de información de proyectos, empleados y documentos.
* Debe registrar el tiempo que cada empleado le dedica a una tarea.
* Debe identificar cuales son los empleados que tienen permitido actualizar o revertir versiones de documentos y debe dejar un registro de la fecha y responsable de la modificación.
* Deber permitir generar listas para cada una de las entidades.

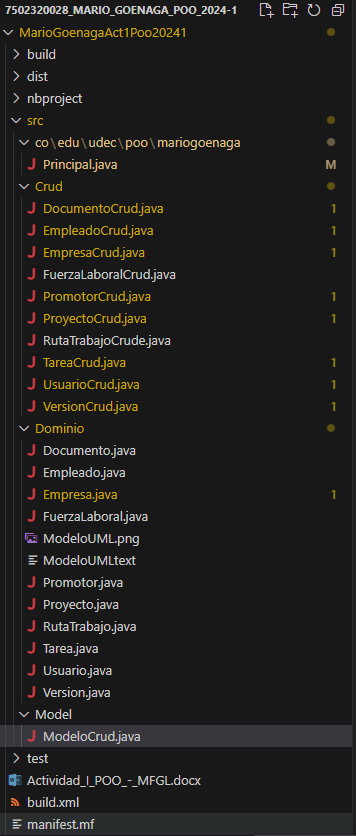
**Desarrollo.**

Nos enfocamos en crear e implementar el método CRUD (de sus siglas en inglés: Create, Read, Update y Delete). Esto es de vital importancia en base de datos puesto que nos permite tal como se menciona crear(agregar), leer(buscar), actualizar(editar) o borrar(eliminar) en o de una lista los atributos de la clase a la que se aplique dicho método.  
En este caso usaremos los métodos ListarTodo y Contar para complementar la función de mostrar listas y tamaño.

El modelo CRUD se declara en una interfaz única la cual cuenta con los métodos anteriormente mencionados.



Creamos una clase CRUD para cada una de las clases presentes en el folder Dominio y en cada una de ellas se implementa la dicha interfaz del modelo CRUD.



**Creación de métodos.**

* Agregar.

Para el desarrollo del método agregar se define como parámetro a un objeto el cual a su vez hace referencia a los atributos pertenecientes a la clase a la cual se le aplicará el modelo CRUD, se verifica la existencia del objeto a agregar usando HashMap usando su clave, si el objeto existe o no se le envía un mensaje correspondiente indicando de la existencia no existencia de este objeto.

* Editar.

Para el desarrollo del método editar primero usamos un set sobre el atributo deseado para asignar los nuevos valores y posteriormente dentro del try se define como parámetro a un objeto el cual a su vez hace referencia a los atributos pertenecientes a la clase a la cual se le aplicará el modelo CRUD, se verifica la existencia del objeto a agregar usando HashMap usando su clave, si el objeto existe o no se le envía un mensaje correspondiente indicando de la existencia no existencia de este objeto.

* Buscar y Eliminar.

Para el desarrollo de los métodos eliminar y buscar se definió como parámetro una cadena de texto llamada a la cual nombramos, se verifica la existencia del objeto en con un hash Map que usa como clave el parámetro string relacionado, definida su existencia envía el mensaje correspondiente.

* Contar Y Listar Todo.

Para desarrollar los métodos de Contar y Listar utilizamos ArrayList para guardar los elementos del HashMap y usar un for each para mostrar la lista de elementos y contar el número de elementos existentes en dicha lista.

Conceptos Aprendidos.

* ArrayList.

Lista de elementos de la misma clase, cuenta con número de posiciones indefinida, a diferencia de un Array que su número de posiciones es definida.

* HashMap.

Al igual que el ArrayList, el HashMap es un conjunto de elementos de la misma clase, pero éste ultimo asigna como clave uno de los atributos de la clase para usarlo como identificador del objeto.

* Interfaz.

Espacio en el que se plasman métodos que debe cumplir cada clase a la que éste le sea aplicado.

* Implements.

Por medio de éste implementamos los métodos de la interfaz a la clase.

* CRUD.

Modelo de vital importancia en base de datos, para agregar, buscar, editar o eliminar atributos de una lista.

Link GitHub.

[GitHub - Mario-Felipe/7502320028\_Mario\_Goenaga\_POO\_2024-1: Diseño de aplicación para optimizar los procesos de gestión para nuevos proyectos a nivel empresarial.](https://github.com/Mario-Felipe/7502320028_Mario_Goenaga_POO_2024-1)

Bibliografía.

<https://www.youtube.com/watch?v=q7qF5o5Og9I&list=PLfB6iylhfbYtmfgYtxlRuGGzNR6fT-MUV&index=8>

<https://www.youtube.com/watch?v=UunKnKoYiTg>