Informe de Deberes

Curso: Programación Orientada a Objetos-1323

Fecha: 08/12/2024

I. Portada

1. **Título del Proyecto:**
Sistema de Gestión de Proyectos de TIC
2. **Integrantes:**
    - Jácome San Lucas Alexander Sebastian
    - Nathaly Stefanía Cusapaz Quiña
    - Henry Bolivar Borja Milan
    - Alan Nicolás Neira Guevara
    - Sheyla Daniela Bernal Correa
3. **Fecha de Entrega:**
    - 08/12/2024

II. Índice

III. Introducción

- **Descripción del Proyecto:**
    - El presente proyecto describe un diseño e implementación de un sistema de gestión de proyectos utilizando Programación Orientada a Objetos (POO). El objetivo es desarrollar una solución que incluya las funcionalidades de registrar proyectos, asignar tareas y mostrar su estado. Este documento detalla el análisis, diseño e implementación básica del sistema en Java.

- **Antecedentes:**
  - UML (o Lenguaje Unificado de Modelado, por sus siglas en inglés) es un software de lenguaje de modelado que se usa para representar el diseño de un sistema específico. UML tiene varios tipos de diagramas, que muestran diferentes aspectos de las entidades a representar.
  - Diagrama de clases

    Los diagramas de clase son los más utilizados en UML y permiten representar un conjunto de clases, interfaces y su relación entre ellos.

    Los objetos son entidades que tienen características que los diferencian de otros y realizan diferentes acciones como borrador, lápiz, mesa. En POO, las características son variables a las que se conoce como atributos y las acciones que realiza un objeto se le conoce como métodos.
  - La serialización de Objetos es el proceso mediante el cual un objeto en Java se convierte en una secuencia de bytes y ser almacenado en un archivo, transmitido por una red o guardado en una base de datos (Oregoom.com, 2024).
  - La encapsulación es la capacidad de ocultar los detalles internos de un objeto y exponer sólo lo necesario para su funcionamiento. Esto asegura la protección de datos mediante métodos controlados. El encapsulamiento consiste en ocultar los atributos de una clase (haciéndolos privados) y proporcionar acceso a ellos mediante métodos públicos específicos llamados getters (obtener) y setters (establecer) (Jaramillo, 2024).

IV. Objetivos del Proyecto

- **Objetivo General:**

Desarrollar un sistema de gestión de proyectos utilizando Programación Orientada a Objetos (POO), que permita registrar proyectos, asignar tareas y mostrar su estado de manera eficiente y organizada.

- **Objetivos Específicos:**
  - Definir las clases principales y atributos como métodos a utilizar en el desarrollo del sistema.
  - Establecer relaciones entre las clases definidas.
  - Aplicar principios de las buenas prácticas de programación como la reutilización de código.

V. Metodología

- **Herramientas y Tecnologías Utilizadas:**

Una vez finalizado el proceso de abstracción y se tiene el diseño de clase deseada, el siguiente paso es pasar a un lenguaje de programación. Las transformaciones y las funciones de desarrollo visual permiten a los desarrolladores y a los arquitectos diseñar y desarrollar un modelo de forma concurrente. Los arquitectos crean elementos de modelo conceptuales y, al mismo tiempo, los desarrolladores crean elementos de código o modifican los elementos de código que ha generado una transformación.

Las funciones de desarrollo visual permiten a los arquitectos actualizar el modelo UML con elementos de código nuevos. Los arquitectos pueden volver a ejecutar la transformación para generar código para elementos de modelo nuevos.

Encapsulamiento

Las clases Proyecto y Tarea mantienen sus atributos como privados, lo que garantiza que solo se puedan acceder y modificar a través de métodos públicos (getters y setters). Esto sigue el principio de encapsulamiento de la POO, asegurando la integridad de los datos.

- **Procedimiento:**

Crear un diagrama para visualizar las relaciones entre clases, atributos y métodos.

Identificar cómo interactúan los usuarios con el sistema para realizar tareas específicas.

Determinar cómo la información se moverá de un lado a otro.

Diseñar una interfaz sencilla que permite a los usuarios interactuar con el sistema mediante menú de opciones facilitando acciones como registrar proyectos, asignar tareas y consultar el estado de los proyectos.

Verificar que los proyectos se registren correctamente, almacenando toda la información necesaria.

Añadir comentarios de códigos claros y concisos para describir el propósito de cada clase y método, facilitando la compresión y mantenimiento futuro del sistema.

Verificar que las clases están bien encapsuladas y que las responsabilidades están claramente definidas.

VI. Desarrollo del Proyecto

- **Diseño del Algoritmo:**

Descripción General

Este programa se presenta como un sistema de gestión de proyectos que permite registrar proyectos y sus tareas asociadas, mostrar información detallada de cada proyecto y gestionar el estado de las tareas. Se estructura en torno a las siguientes clases:

Proyecto: Representa un proyecto con un nombre, una descripción y una lista de tareas.

Tarea: Define una actividad específica dentro de un proyecto, incluyendo un nombre, una descripción y un estado.

GestorDeProyectos: Administra una lista de proyectos, permitiendo su registro y visualización.

Descripción Detallada de las Clases

Clase Tarea

Atributos:

nombre: Nombre de la tarea.

descripcion: Descripción de la tarea.

estado: Estado de la tarea (inicialmente "Pendiente").

Métodos:

getNombre(), getDescripcion(), getEstado(): Obtienen información de la tarea.

cambiarEstado(String nuevoEstado): Cambia el estado de la tarea.

toString(): Representa la tarea como una cadena de texto.

Clase Proyecto

Atributos:

nombre: Nombre del proyecto.

descripcion: Descripción del proyecto.

tareas: Lista de tareas asociadas al proyecto.

Métodos:

agregarTarea(Tarea tarea): Agrega una tarea al proyecto.

mostrarEstado(): Muestra el nombre, descripción y lista de tareas del proyecto.

Clase GestorDeProyectos

Atributos:

proyectos: Lista de proyectos registrados.

Métodos:

registrarProyecto(Proyecto proyecto): Agrega un proyecto al gestor.

mostrarProyectos(): Muestra todos los proyectos registrados con sus tareas.

Clase Main

Crea instancias de GestorDeProyectos, Proyecto y Tarea.

Registra proyectos y muestra el estado actual de todos ellos.

- **Código Fuente:**

Proyecto

```
import java.util.ArrayList;
import java.util.List;

public class Proyecto {
    private String nombre;
    private String descripcion;
    private List<Tarea> tareas;

    public Proyecto(String nombre, String descripcion) {
        this.nombre = nombre;
        this.descripcion = descripcion;
        this.tareas = new ArrayList<>();
    }

    public void agregarTarea(Tarea  tarea) {
        tareas.add(tarea);
    }

    public void mostrarEstado() {
        System.out.println("Proyecto: " + nombre);
        System.out.println("Descripción: " + descripcion);
        System.out.println("Tareas:");
        for (Tarea tarea : tareas) {
            System.out.println(tarea);
        }
```

```
    }
}
```

Registrar

```
import java.util.ArrayList;
import java.util.List;

public class GestorDeProyectos {
    private List<Proyecto> proyectos;

    public GestorDeProyectos() {
        proyectos = new ArrayList<>();
    }

    public void registrarProyecto(Proyecto proyecto) {
        proyectos.add(proyecto);
    }

    public void mostrarProyectos() {
        for (Proyecto proyecto : proyectos) {
            proyecto.mostrarEstado();
            System.out.println();
        }
    }
}
```

Tareas

```
public class Tarea {
    private String nombre;
    private String descripcion;
    private String estado;

    public Tarea(String nombre, String descripcion) {
        this.nombre = nombre;
        this.descripcion = descripcion;
        this.estado = "Pendiente";
    }

    public String getNombre() {
        return nombre;
    }

    public String getDescripcion() {
        return descripcion;
    }
```

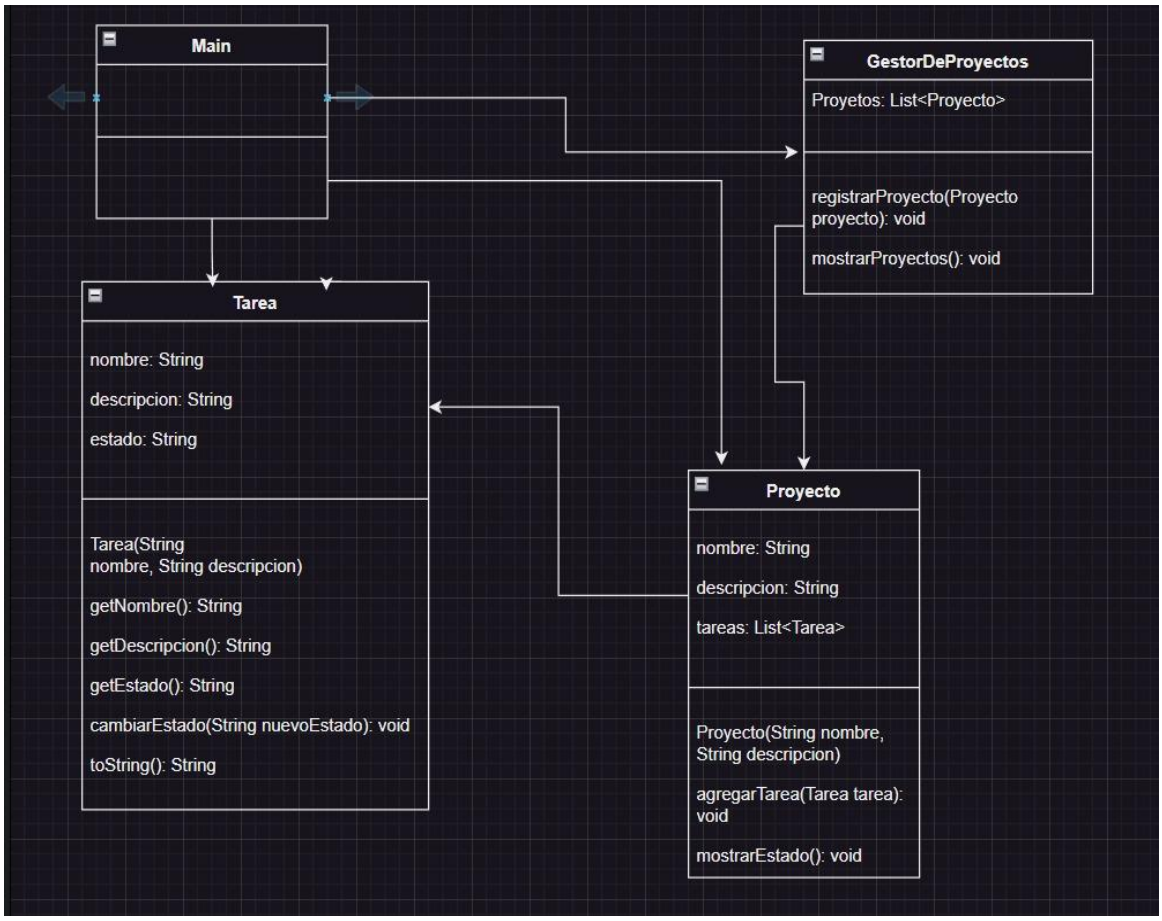
```
    public String getEstado() {
        return estado;
    }

    public void cambiarEstado(String nuevoEstado) {
        this.estado = nuevoEstado;
    }

    @Override
    public String toString() {
        return "Tarea:  " + nombre + ", Descripción: " + descripcion + ", Estado: " + estado;
    }
}
```

Deberes

```
public class Main {

    public static void main(String[] args) {

        GestorDeProyectos gestor = new GestorDeProyectos();



       Proyecto proyecto1 = new Proyecto("Desarrollo de Software", "Proyecto para
desarrollar una aplicación de gestión.");

        proyecto1.agregarTarea(new Tarea("Análisis de requisitos", "Reunir y analizar los
requisitos del cliente."));

        proyecto1.agregarTarea(new Tarea("Diseño de la arquitectura", "Diseñar la
arquitectura del sistema."));

        gestor.registrarProyecto(proyecto1);

        Proyecto proyecto2 = new Proyecto("Marketing Digital", "Proyecto para implementar
una estrategia de marketing digital.");

        proyecto2.agregarTarea(new Tarea("Investigación de mercado", "Investigar el
mercado objetivo."));
```

```
        proyecto2.agregarTarea(new Tarea("Crear contenido", "Desarrollar contenido para
redes sociales."));

            gestor.registrarProyecto(proyecto2);

        gestor.mostrarProyectos();
    }
}
```

- **Diagrama UML**



Este diagrama UML representa el sistema de gestión de proyectos orientado a objetos. Las clases principales son GestorDeProyectos, Proyecto y Tarea, mientras que Main sirve como punto de entrada para interactuar con el sistema.

VII. Resultados

- **Análisis de Resultados:**

Pruebas Realizadas:

Entrada:

```
Proyecto: Desarrollo de Software
Descripción: Proyecto para desarrollar una aplicación de gestión.
Tareas:
Tarea: Análisis de requisitos, Descripción: Reunir y analizar los requisitos del cliente., Estado: Pendiente
Tarea: Diseño de la arquitectura, Descripción: Diseñar la arquitectura del sistema., Estado: Pendiente

Proyecto: Marketing Digital
Descripción: Proyecto para implementar una estrategia de marketing digital.
Tareas:
Tarea: Investigación de mercado, Descripción: Investigar el mercado objetivo., Estado: Pendiente
Tarea: Crear contenido, Descripción: Desarrollar contenido para redes sociales., Estado: Pendiente
```

El diseño del sistema sigue los principios de Programación Orientada a Objetos (POO), utilizando encapsulación para proteger los datos y proporcionar acceso controlado a través de métodos públicos. Las clases se organizan de forma que cumplen con cada una de sus funcionalidades.

VIII. Conclusiones

- **Conclusiones Generales:**
  - Se logró diseñar una estructura sólida y lógica para las clases, lo que facilitó la encapsulación de datos y la claridad en las responsabilidades de cada entidad. Esto permitió una mejor comprensión del sistema y su propósito, además de brindar flexibilidad para futuras modificaciones o expansiones.
  - La implementación de métodos bien definidos para gestionar los datos permitió una manipulación eficiente y precisa de la información. Tanto la recuperación como la actualización de los datos resultaron rápidas y organizadas, lo que mejora la experiencia del usuario y reduce la probabilidad de errores.

- Las validaciones implementadas fueron efectivas para evitar inconsistencias en los datos. Esto contribuyó a mantener la integridad del sistema, asegurando que solo se registraran entradas válidas y acorde con los parámetros establecidos.

En conclusión, este proyecto logró cumplir con el objetivo de organizar y administrar de forma eficiente las entidades y funciones necesarias para un torneo. La definición y estructura de las clases facilitaron un manejo claro y ordenado de los datos, y los métodos implementados hicieron sencilla la actualización y recuperación de información

- **Recomendaciones:**
  - Se debe realizar las pruebas necesarias hasta poder conseguir nuestro código limpio y que no nos salte ninguna falla

  - Considero importante seguir mejorando la estructura del sistema para que sea más flexible y pueda adaptarse a ocasiones más complejas o con reglas específicas. También sería útil implementar reportes automatizados que muestren estadísticas o resultados en tiempo real, lo que facilita la toma de decisiones. Por otro lado, sería bueno evaluar y mejorar las validaciones periódicamente para garantizar que el sistema siga siendo confiable ante posibles escenarios no contemplados.

Referencias

Codigofacilito. (2010). UML y diagrama de clases.

https://codigofacilito.com/articulos/uml_diagramas_de_clase

Oregon.com. (2024). Serialización de Objetos en Java.

https://oregoom.com/java/serializacion-de-objetos/#google_vignette

Jaramillo, L. (11 agosto 2024).Programación Orientada a Objetos.

https://luisjaramillom.github.io/POO.io/

Oracle. (2023). *The Java™ Tutorials*. https://docs.oracle.com/javase/tutorial/

Visual Pradigm. (9 de febrero de 2022). *¿Cuáles Son Los Seis Tipos De Relaciones En Los Diagramas De Clases UML?* https://blog.visual-paradigm.com/es/what-are-the-six-types-of-relationships-in-uml-class-diagrams/