**Examen Final Programación 1**

1. **¿Qué clases principales crees que son necesarias para modelar este sistema? Justifica tus elecciones.**

Clases Principales:

* Empleado: Esta clase representa a los empleados de la empresa. Es fundamental para gestionar la información personal y la clasificación de los empleados.
* Proyecto: Esta clase se encarga de gestionar los proyectos a los que los empleados pueden ser asignados. Es esencial para organizar y administrar los distintos proyectos de la empresa.
* Sistema de Gestión: Esta clase es la responsable de coordinar y manejar las operaciones principales del sistema, como la contratación de empleados, creación de proyectos y asignación de empleados a proyectos. Actúa como el controlador del sistema.

1. **¿Qué atributos y métodos deben incluirse en cada clase para cumplir con los requisitos planteados?**

Clase Empleado:

* Atributos:
  + nombre: Representa el nombre del empleado.
  + categoría: Representa la categoría del empleado (Administrador, Operario, Peón).
  + proyectos: Una lista o vector que almacena los nombres de los proyectos asignados al empleado.
* Métodos:
  + asignarProyecto(): Asigna un proyecto al empleado.
  + mostrarInfo(): Muestra la información del empleado, incluyendo los proyectos asignados.

Clase Proyecto:

* Atributos:
  + nombre: Representa el nombre del proyecto.
  + listaEmpleados: Una lista o vector de punteros a los empleados asignados al proyecto.
* Métodos:
  + agregarEmpleado(): Agrega un empleado al proyecto.
  + mostrarInfo(): Muestra la información del proyecto y los empleados asignados.

Clase Sistema de Gestion:

* Atributos:
  + listaEmpleados: Un vector que almacena todos los empleados de la empresa.
  + listaProyectos: Un vector que almacena todos los proyectos de la empresa.
  + historial: Una pila que almacena el historial de acciones realizadas para poder deshacerlas.
* Métodos:
  + contratarEmpleado(): Contrata un nuevo empleado.
  + crearProyecto(): Crea un nuevo proyecto.
  + asignarEmpleadoAProyecto(): Asigna un empleado a un proyecto específico.
  + deshacerUltimaAccion(): Deshace la última acción realizada, permitiendo revertir cambios si se cometió un error.

**3. Propón un diagrama de relación entre clases (puedes hacerlo en texto si no tienes herramientas**

**gráficas).**

Empleado: Tiene atributos nombre, categoria, y proyectos.

Métodos: asignarProyecto(), mostrarInfo().

Proyecto: Tiene atributos nombre y listaEmpleados.

Métodos: agregarEmpleado(), mostrarInfo().

SistemaGestion: Administra listaEmpleados y listaProyectos, y mantiene un historial de acciones.

Métodos: contratarEmpleado(), crearProyecto(), asignarEmpleadoAProyecto(), deshacerUltimaAccion().

**Relaciones:**

* SistemaGestion tiene múltiples Empleado y Proyecto.
* Empleado puede estar en múltiples Proyecto.
* Proyecto contiene múltiples Empleado.

**1. ¿Qué decisiones de diseño tomaste durante el desarrollo y por qué?**

Durante el desarrollo del sistema, tomé varias decisiones clave de diseño:

* **Uso de clases y objetos**: Decidí estructurar el sistema utilizando clases, ya que las entidades (empleados, proyectos, asignaciones) tienen atributos y comportamientos bien definidos. Esto permite una mayor modularidad, reutilización de código y facilita el mantenimiento del sistema.
* **Validaciones dentro de los constructores**: Implementé validaciones directamente en los constructores para garantizar que los objetos solo se crearan si los datos son válidos (por ejemplo, validación de rango para el salario, unicidad de correos y nombres de proyectos). Esto asegura la integridad de los datos desde el inicio y evita errores posteriores.
* **Contenedores set para unicidad**: Usé set para almacenar correos electrónicos y nombres de proyectos. Esto garantiza que no se repitan y facilita la validación, ya que los set tienen una complejidad O(log n) para operaciones de búsqueda, inserción y eliminación.
* **Uso de vector para almacenar asignaciones**: Para el historial de asignaciones de proyectos a empleados, utilicé un vector porque permite almacenar múltiples elementos (proyectos) de manera dinámica y realizar iteraciones sobre ellos fácilmente.

**2. ¿Qué aspectos del problema fueron más difíciles de modelar y cómo los resolviste?**

El aspecto más desafiante fue **la gestión de la asignación de empleados a proyectos**. Modelar esta relación fue complejo porque involucra una conexión entre varias entidades y requería mantener un registro claro de qué empleados están trabajando en qué proyectos, así como la fecha de asignación.

Para resolver esto, decidí crear una clase adicional llamada **Asignacion** que almacena la relación entre el empleado y el proyecto, junto con la fecha de asignación. De esta forma, pude manejar fácilmente las asignaciones sin complicar la lógica de las clases Empleado o Proyecto directamente.

**3. Si tuvieras más tiempo, ¿qué mejorarías o agregarías al sistema?**

Si tuviera más tiempo, consideraría mejorar los siguientes aspectos:

* **Interfaz gráfica o interfaz de línea de comandos más amigable**: Añadir una interfaz más amigable para el usuario, como una interfaz gráfica o un menú más estructurado en la línea de comandos, que permita navegar entre opciones de manera más clara.
* **Persistencia de datos**: Agregar la capacidad de almacenar y recuperar los datos de empleados y proyectos desde archivos (por ejemplo, archivos de texto o bases de datos) para que el sistema no pierda la información al cerrarse.
* **Más funcionalidades**: Implementar más funcionalidades, como la posibilidad de eliminar asignaciones de proyectos, eliminar empleados o proyectos, y mejorar la gestión de permisos y roles (por ejemplo, solo administradores podrían agregar o eliminar proyectos).
* **Optimización de código**: Para mejorar el rendimiento, podría optimizar el uso de contenedores según la complejidad de las operaciones que se realicen con más datos