

REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA EDUCACIÓN
UNIVERSIDAD POLITÉCNICA TERRITORIAL DE CARACAS “MARISCAL SUCRE”
PNF INFORMATICA
SECCION 7122 (701201)

INFORME DE PROYECTO P.O.O

Integrante

Alber Campos

Desarrollo de las Clases y el Manejo de Datos

Para empezar con el desarrollo de este sistema(siendo un completo y absurdo dolor de cabeza), lo primero que tuve que pensar fue en cómo organizar toda la información de los trabajadores, así que lo que hice fue crear una jerarquía de clases que pudiera modelar los roles que pedía el documento, la base de todo el programa es la clase Empleado, que viene siendo la clase padre o la clase principal de la que heredan las demás. En esta clase definí los atributos que todos comparten, como el nombre y el salario base.

Uno de los problemas que me encontré al principio fue el tema del ID, porque el PDF decía que tenía que ser único y **automático**, entonces, para no complicarme tanto y asegurarme de que funcione bien, usé una variable global fuera de la clase llamada **contador** en el cual, básicamente, cada vez que se crea un empleado nuevo, el constructor llama a esa variable, le suma uno y se lo asigna al empleado **asegurándome** de que nunca haya dos IDs iguales (otro dolor de cabeza pero finalmente logrado) también me aseguré de encapsular los datos de forma correcta con *nombre* y *salario* para que sean privados y nadie pueda cambiarlos desde fuera del código sin usar los métodos que creé para eso, que son los **getters**.

Después de tener lista la clase base, usé la herencia para crear las otras tres clases **Desarrollador**, **Diseñador** y **Gerente**, usando *super* o *Init* cada una de ellas para no tener que volver a escribir el código de asignar nombre y salario, al desarrollador le agregué una lista para guardar los lenguajes que sabe y una variable para su nivel de experiencia, al Diseñador le puse una lista para sus herramientas y otra variable para su especialidad, y el Gerente es un poco diferente porque no tiene habilidades técnicas como tal, sino que tiene una lista llamada *lista* subordinados donde voy guardando a los empleados que están a su cargo, lo definí así desde el principio porque luego la forma en que cada uno calcula su salario depende totalmente de estos atributos que les puse.

Cómo hice el Cálculo de Salarios y el uso del “Polimorfismo”

La parte del salario fue donde tuve que meterle lógica y razonamiento pero en cantidades industriales, porque cada tipo de empleado cobra de forma distinta, aunque el método se llame igual en todos según el pdf *calcular salario mensual*, esto es lo que entiendo por polimorfismo, que el mismo método actúe diferente según el objeto.

Para el Desarrollador, lo que hice fue usar varios *if* anidados, 1- tomo el salario base que viene de la clase padre y luego pregunto: ¿*Es Junior*? si es así, le sumo 200, si no, pregunto si es *SemiSenior* y le sumo 500, y si es *Senior* le sumo 1000, lógica secuencial bastante directa pero efectiva para lo que se pedía en el PDF

Con el Diseñador fue más largo el código porque tenía que revisar las herramientas una por una, realicé un ciclo, un bucle, que recorriera toda la lista de herramientas que tiene el diseñador y dentro de ese ciclo puse condiciones, por ejemplo, tenía que ver si usaba "Figma" si el programa encontraba la palabra "Figma" en la lista, activaba una variable booleana para saber que sí lo usa y luego al final sumar los 300\$ en el mismo proceso aproveché para contar cuántas herramientas de Adobe tenía (como Photoshop o Illustrator) y usé un contador simple que iba sumando uno cada vez que encontraba esas palabras y al final del método hago las sumas finales, es decir, si tiene solo una herramienta y es de Adobe, le sumo un bono, y si tiene 3 o más herramientas en total, le sumo otro bono, fué un poco más complicado que el desarrollador pero funciona.

Ahora bien, para el Gerente fue donde apliqué la recursividad que pedía el PDF, el salario del gerente depende de sumar los salarios de toda la gente que tiene a cargo, así que en lugar de usar un *for* normal, creé una función auxiliar dentro de la clase que se llama a sí misma, recibiendo la lista del equipo y un número que es la posición actual, es decir que si la posición es igual al tamaño de la lista, devuelve 0 (ese es el caso base para que no caiga en bucle) si no, calcula el salario del empleado en esa posición y le suma el resultado de llamar a la misma función pero con la posición siguiente.

Al final, esa suma recursiva me da el total de la nómina del equipo, le saco el 15% y eso se lo sumo al sueldo base del gerente.

Validaciones y Reglas para Asignar Proyectos

Otra parte importante fue controlar que no se hicieran asignaciones incorrectas, porque el documento tenía varias restricciones, en la clase Proyecto, que funciona por composición porque tiene una lista de empleados adentro, creé el método para inscribir gente, lo primero que hago ahí es verificar que el empleado no esté ya en la lista, hago un recorrido rápido comparando los IDs, y si el ID ya existe, hace un **print** con un mensaje de error, si el empleado no está repetido, entonces tengo que ver si el empleado "puede" aceptar el proyecto.

Para no llenar el código principal de *if* puse esta lógica dentro de la clase Empleado en un método que se llama *intentar asignar proyecto* aquí tuve que usar una forma de detectar qué tipo de clase es el objeto, usando type (self) name, si es un Gerente, de una vez le digo que no, devuelvo **false** y muestro un error porque los gerentes no trabajan en proyectos y si es Desarrollador, puse que el límite es 3, y si es Diseñador el límite es 2.

Luego cuento cuántos proyectos tiene ya asignados ese empleado revisando el tamaño de su lista **lista_proyectos_asignados** si el número que tiene es menor al límite, entonces lo agrego y devuelvo **True**, lo que le avisa a la clase Proyecto que la asignación fue exitosa y si ya está lleno, devuelvo **False**.

También validé la parte de armar el equipo del Gerente y como no tendría nada de sentido que un gerente mande a otro gerente, en el método **agregar_miembro_equipo** reviso el tipo de objeto que me están pasando permitiendo solo que se agregue a la lista si el objeto es instancia de Desarrollador o de Diseñador en el cual, si intentan meter otra cosa, el programa lo rechaza.

Cálculos Financieros y el Menú del Sistema

Para ver si los proyectos son viables o no, hice un método en la clase Proyecto que calcula el costo total, en pocas palabras recorre a todos los empleados inscritos en ese proyecto, llama a su método de calcular salario (que ya incluye todos los bonos que expliqué antes) y va sumando todo en una variable acumuladora, luego, ese total lo comparo con el presupuesto. Como el PDF da una regla que dice que el costo no puede pasar del 70% del presupuesto, hago esa multiplicación simple y si el costo es menor o igual, el método devuelve **Verdadero**, indicando que es viable.

Finalmente, para que todo esto se pudiera usar, creé un menú en la consola con un ciclo while infinito, lo hice lo más sencillo posible, así que puse opciones del 1 al 9, usando listas globales para ir guardando todos los empleados y proyectos que se van creando, porque si no los guardo en una lista global se borrarían y bueno, no tendría sentido entonces así que desde el menú puedo crear empleados pidiendo los datos con un **input**, crear proyectos y hacer las asignaciones buscando por ID y para buscar por ID, lo que hago es recorrer la lista de empleados uno por uno hasta que encuentro el que tiene el número que escribió el usuario.

También incluí la opción 9 que es la prueba automática que pedía el PDF, ahí puse el código "duro", o sea, escribí los datos directamente en el código para crear un gerente, dos desarrolladores y un diseñador, asignarlos entre ellos y luego forzar el error de asignar un cuarto proyecto para demostrar que mis validaciones de límites funcionan correctamente.

Y bueno, eso es todo.

Pd: Los compañeros que me ayudaron fueron Naomy y Gabriel, adicional a esa ayuda, correcciones y consejos de la IA (Gemini y Copilot) sin abusar de ellos ni pidiéndole directamente que me hicieran el código. De igual forma usted podrá comprobar lo mismo en GitHub. Gracias por leer profesor, espero no aplazar la materia.

Link del Repositorio: <https://github.com/Albert1869/Proyecto.git>