Algoritmos y Programación II

Trabajo práctico Nº 2:

“Lista polimórfica”

Objetivo

Desarrollar un programa orientado a objetos en C++ que maneje una lista polimórfica.

Datos de entrada

Habrá un archivo "trabajadores.txt" en donde cada línea tendrá el detalle de los servicios que realizó una persona para cierta empresa.

Requisitos

La aplicación debe estar completamente orientada a objetos o puede ser un híbrido, admitiendo solo el menú y algunas funciones menores como desarrollo estructurado.

La empresa cuenta con tres tipos de trabajadores que prestan sus servicios:

- Empleado: son los trabajadores mayoritarios. El sueldo es un valor mensual al que se le debe sumar un premio del 10% por presentismo si no faltó nunca y si no tiene más de tres llegadas tarde. Con más de tres llegadas tarde pierde el presentismo y con una o más ausencias, además de perder el presentismo, pierde el proporcional al día que no trabajó (se toma como base 30 días).
- Jornalero: trabajan por día. El monto a cobrar a fin de mes se calcula como el valor que cobra por día por la cantidad de días trabajados.
- Consultor: son los que capacitan y/o dan apoyo a los miembros de la empresa. Esta gente tiene un sueldo estipulado en función de horas cátedra que son horas semanales. Por ejemplo 4, significa 4 horas semanales, es decir, 16 horas mensuales (se toma como base 4 semanas). Entonces, si el monto es de $1800 por hora cátedra, cobrará 4 x 1800 = $ 7200 en el mes. Pero si se debe descontar alguna hora, es la hora real. Es decir, habría que hacer el siguiente cálculo:
  1 hora = $7200 / 16hs = $450 por hora efectiva.

Cada línea del archivo comenzará con una letra que puede ser: E (empleado), J (jornalero) o C (consultor). Luego irán los siguientes campos:

E	*legajo*	*apellido_nombre*	*sueldo_mensual*	*llegadas_tarde*	*ausencias*
J	*legajo*	*apellido_nombre*	*valor_diario*	*cantidad_dias*	
C	*legajo*	*apellido_nombre*	*valor_hora_catedra*	*horas_catedra*	*horas_a_descontar*

Ejemplo:

E	321	Perez_Rosa	42000	2	0
E	457	Sosa_Juan	40000	0	1
J	125	Rossi_Rocío	850	7	
C	82	Kachanovsky_Judith	1500	5	2
J	390	Ruiz_Pedro	900	11	

En la primera línea tenemos una empleada que cobra $42000 mensuales, con dos llegadas tarde y ninguna ausencia, por lo que debe cobrar un 10% más por presentismo.
En la segunda línea, al empleado debemos descontarle 1 día sobre la base de 30 y no pagarle el presentismo.
En la tercera y quinta línea simplemente hay que multiplicar el monto diario por los días trabajados.
En la cuarta línea tenemos una persona que cobra $1500 la hora cátedra y da 5 horas semanales, por lo que debería cobrar 5 x 1500 = $7500. Sin embargo hay que descontarle dos horas reales, es decir (7500 / 20) x 2 = $750.

Los objetos (trabajadores) se irán creando en forma dinámica e irán ingresando en una Lista polimórfica, la que estará ordenada por número de legajo.
Luego, la aplicación debe ofrecer un menú para:

- Consultar si un número de legajo existe o no. En caso afirmativo debe mostrar todo el objeto.
- Dar de baja cierto número de legajo.
- Dar de alta a un trabajador.
- Listar todos los trabajadores con sus respectivas liquidaciones de sueldo.
- Indicar sueldo máximo y a quién pertenece.
- Indicar sueldo mínimo y a quién pertenece.
- Indicar la sumatoria de todos los sueldos.
- Finalizar la aplicación.

Hay que programar un método a_cadena o to_string (si está en inglés) que muestre todos los datos de un trabajador, indicando:

- Tipo de trabajador
- Legajo
- Apellido y nombre
- Sueldo a cobrar detallado

Consideraciones

- El archivo está bien formado.
- Los objetos deben pertenecer a una clase en común.
- La clase de la cual heredan debe tener métodos virtuales y debe ser abstracta.
- Las clases deben implementar por lo menos los métodos *liquidar_sueldo* y *a_cadena*.

Qué se evalúa

- Buenas prácticas: código, modularización, comentarios, claridad
- POO
- Funcionalidad
- Polimorfismo
- Memoria dinámica

- Interfaz de usuario
- Pre y pos condiciones

Normas de entrega

El ejercicio debe ser desarrollado en equipos de cuatro o cinco personas.
Deben entregar:

- Documentación
    - Diagrama de clases UML
    - Diagrama de relación de clases.
    - Descripción de cada TDA, indicando las pre y poscondiciones de cada una de las operaciones.
- Código fuente
    - En los archivos .h también van las pre y poscondiciones.
- Archivo trabajadores.txt para testeo.

La fecha de entrega será entre el viernes 17/5 a las 23.55hs