Homework 2 - OOP

I102 – Paradigmas de Programación

Fecha límite de entrega: 04/04/2025 – 22hs

La entrega se realizará mediante un archivo de texto que contenga el enlace a un repositorio privado de GitHub con la solución de cada ejercicio, organizados en carpetas separadas. Además, el archivo debe incluir un texto que indique cómo compilar el código de cada ejercicio. Cada carpeta de los ejercicios debe contener únicamente archivos sin compilar, como .h, .hpp, .cpp, entre otros (también se puede incluir un archivo .gitignore).

1. Programe una clase que permita expresar un determinado momento en el tiempo en el siguiente formato: HHh, MMm, SSs p.m./a.m. Por ejemplo: “03h, 14m, 42s p.m.”. Para este ejercicio, puede utilizar la librería *iomanip* con la funcionalidad provista por *setw* y/o *setfill*.
Deberá ser capaz de:
   a. Poder ser inicializada sin parámetros (en cuyo caso el tiempo es 0h, 0m, 0s a.m.)
   b. Poder ser inicializada pasándole sólo la hora (en cuyo caso el tiempo es “HHh, 0m, 0s a.m.”).
   c. Poder ser inicializada pasándole sólo la hora y los minutos (en cuyo caso el tiempo es “HHh, MMm, 0s a.m.”)
   d. Poder ser inicializada pasándole sólo la hora, los minutos y los segundos (en cuyo caso el tiempo es “HHh, MMm, SSs a.m.”)
   e. Poder ser inicializada pasándole la hora, los minutos, los segundos y “p.m.”/”a.m.” (en cuyo caso el tiempo es “HHh, MMm, SSs p.m./a.m.”).
   f. Agregar funcionalidad a los puntos 1.b-1.e de manera tal que no se introduzcan valores fuera de los rangos (por ejemplo, es incorrecto introducir MM = 74).
   g. Permitir cambiar y leer en forma individual: la hora, los minutos, los segundos y si se trata de a.m. o p.m. También leer todo junto en el formato pedido.
   h. Escribir un método que escriba por pantalla la hora en formato de hora 0 a 24, donde no se utiliza “p.m.”/”a.m.”, por ejemplo, 13hs p.m. no es válido.
   i. Escriba un código que permita verificar cada una de las funcionalidades pedidas en 1.a-1.g. Para la verificación se deberá pedir que se interactúe con el programa ingresando los datos por teclado. Por ejemplo, en el caso de ingresar un valor erróneo, indicar esto por pantalla y permitir ingresar un nuevo valor o bien permitir salir del programa.

2. Escribir una clase denominada Curso que contiene un vector (std::vector) con punteros a objetos estudiantes.

a. El objeto estudiante simplemente cuenta con el nombre completo, su legajo (que es único por alumno), una lista de cursos con su nota final y los métodos que crea necesarios para obtener los datos del alumno: nombre completo, legajo y su promedio general. En función de esto, califique todos los atributos correctamente.
b. Como se mencionó anteriormente, un objeto de la clase Curso contendrá la lista de estudiantes del curso (el vector conteniendo objetos tipo estudiante). La clase Curso permite:
   i. Inscribir y desinscribir estudiantes al curso.
   ii. Ver si un estudiante se encuentra inscripto o no en el curso buscándolo por su legajo.
   iii. Indicar si el curso está completo o no, teniendo en cuenta que el curso tiene una capacidad de 20 alumnos.
   iv. Imprimir la lista de estudiantes en orden alfabético. Para ello, utilice el algoritmo std::sort() en <algorithm>, el cual requerirá sobreescribir el operador "<", y sobreescriba el operador "<<" (del método y clase que correspondan) para presentar los datos por pantalla.
   v. Dado que algunos cursos comparten la mayor parte de los estudiantes, se desea poder hacer una copia del objeto curso. Justifique su respuesta con un comentario en el código (esta puede llevar varias líneas), indicando de que tipo de copia se trata y como la hizo.
c. ¿Qué tipo de relación existe entre los objetos curso y estudiante?
d. Proporcione un menú que permita evaluar lo pedido en este ejercicio.

3. Escriba una interfaz llamada "Numero" que indique la implementación de las operaciones suma, resta y multiplicación de números (si desea, puede también agregar la división), así como también un método denominado "toString" que muestra el número en formato string.

a. De esta interfaz, se deben implementar las siguientes tres clases:
   - Clase "Entero",
   - Clase "Real",
   - Clase "Complejo"
b. Verifique el funcionamiento de cada uno de estas clases mediante la implementación de cada una de las operaciones con los números que desee e imprima el resultado utilizando el método "toString".

4. Se tiene una clase abstracta que representa una cuenta de banco genérica. Esta clase tiene los siguiente atributos y métodos:

i. balance: el total de la cuenta (tipo double).
ii. titularCuenta: nombre del dueño de la cuenta (string).
iii. depositar(): deposita una cantidad de dinero en la cuenta.
iv. retirar(): retira una cantidad de dinero de la cuenta (método virtual puro).

v. mostrarInfo(): muestra la información de la cuenta (método virtual puro) que consta del balance, el tipo de cuenta y el nombre del titular.

De esta clase se derivan dos subclases:

CajaDeAhorro:

- El método retirar() impide retirar más dinero del que existe en el balance.
- Se realizará un descuento de $20 del balance luego de mostrar más de 2 veces la información de la cuenta.

CuentaCorriente:

- El método retirar() permite retirar dinero de la caja de ahorro si no existen suficientes fondos en este tipo de cuenta.
- En caso de que la caja de ahorro tampoco tenga dinero, se imprimirá un aviso de que la cuenta no posee dinero. Utilizar *friend* para esto.

Se pide:

a. Establecer los especificadores de acceso (public, private y private) en los atributos y métodos de todas las clases, explicando su elección en función de la seguridad y la la flexibilidad para futuros requerimientos de cambio.
b. Escriba el código para implementar todas clases.
c. Escriba el código que permita probar los puntos mencionados para CajaDeAhorro y CuentaCorriente.