Programación Orientada a Objetos / Programación II - UNL-FICH - Parcial 1 - 24/09/24 - Tema A

Ej1(30pts) Se quiere modelar la venta de entradas para un show:
a) Implemente una clase *Entrada* para representar una entrada a un show. La clase debe guardar el nro de asiento, el tipo de entrada (string), el precio y si está vendida o no. Todos estos datos deben inicializarse en un constructor. La clase debe tener además métodos para consultar los datos, y registrar la venta de esa entrada cuando se realice.
b) Reutilizando la clase *Entrada*, cree una clase *Show* para gestionar la venta de entradas a un show. La clase debe recibir en su constructor el nombre del show, y tener métodos para: (1) consultar el nombre; (2) habilitar entradas recibiendo un tipo, un precio, y un rango de nros (desde, hasta)*; (3) vender una entrada, indicando su número (debe retornar *true* si la venta es posible, *false* si el número no existe o no está disponible) (4) consultar la lista de entradas y (5) calcular la recaudación total.
*Por ejemplo: si se le pasa *"platea"*, 10000, 100, 299, debe agregar al show 200 entradas de tipo platea disponibles a $10000 cada una, numeradas del 100 al 299 inclusive.
c) Escriba un programa de prueba que cree un *Show* llamado *"Susy Plus Plus y los Compiladores en vivo"*, registre entradas tipo *"platea"* a $15000 para los nrs 100 a 199; y entradas tipo *"palco"* a $8000 para los nros 500 a 650. Luego el programa debe permitir vender *N* entradas: para cada venta primero se muestra la lista de entradas disponibles (nro, tipo, precio de cada una), se le pide al usuario el nro de asiento que quiere comprar, y se registra la venta, o se muestra un mensaje de error si el nro es incorrecto. Al final del programa debe informar la recaudación del show.

Ej2(25pts) Sin utilizar la clase std::vector: **a)** Escriba una función *copia_sin_x* que reciba un arreglo de enteros *a* y un entero *x*. La función debe generar y retornar un nuevo arreglo que contenga todos los elementos de *a* menos *x*. Ejemplo: si *a*={ 1,3,4,1,6,8,1,4,3 } y *x*=1, entonces debe retornar { 3,4,6,8,4,3 }.
b) Escriba un programa cliente que permita ingresar al usuario un arreglo de *n* enteros (*n* también es dato), un entero *x*, y muestre el arreglo que genera la función.

Ej3(30pts) En un juego de plataformas el personaje puede juntar estrellas para conseguir cosas. Hay distintos colores de estrellas que le dan distintos beneficios. **a)** Implemente una clase *Jugador* que guarde los puntos y la cantidad de vidas de un jugador, y que tenga métodos para consultar su estado, sumar o restar una vida, y sumar puntos. **b)** Implemente una clase *Estrella* para modelar una estrella cualquiera, que tenga como atributo su color (que deberá inicializar en el constructor) y un método *Aplicar* que reciba un *Jugador* y le aplique el beneficio de juntar esa estrella. **c)** Implemente dos clases para dos tipos de estrellas en particular: una roja cuyo beneficio es ganar una vida, y una azul cuyo beneficio será sumar una cierta cantidad de puntos (que se indique en su constructor). **d)** Implemente una función de prueba que reciba una estrella cualquiera (la roja y la azul son solo ejemplos, podría haber más) y un jugador, y aplique el beneficio, informando el estado del jugador antes y después de hacerlo. Desde un programa cliente cree un jugador e invoque a la función una vez con cada tipo de estrella del punto b.

Ej4(15pts) Explique/justifique: a) ¿Qué entiende por sobrecarga de operadores? **b)** ¿Cuándo es necesario implementar el operador de copia? **c)** Dada la clase *Carta* del recuadro, implemente las sobrecargas necesarias para comparar dos cartas (solo para saber si son iguales, y si son distintas).

```
class Carta {
   string palo;  int nro;
public:
   Carta(string p, int n);
   string VerPalo() const;
   int VerNro() const;
};
```

Solución: *https://youtu.be/6Vt5g8-9vxA*

Programación Orientada a Objetos / Programación II - UNL-FICH - Parcial 1 - 24/09/24 - Tema B

Ej1(30pts) "Programadores Anónimos" es una página de preguntas y respuestas de programación, donde los usuarios van ganando insignias y subiendo de nivel a medida que cumplen ciertos hitos (publicar su primer pregunta, o su primer respuesta, recibir el primer "me gusta", que su pregunta llega a las 1000 visitas, etc).
a) Implemente una clase *Usuario* para guardar los datos de un usuario: nombre y lista de insignias (*string*s). La clase debe tener (1) un constructor para inicializar el nombre, (2) un método para consultarlo, (3) un método para agregarle una insignia, (4) otro para consultar si el usuario tiene determinada insignia, y (5) uno para calcular el nivel (nivel 0 si no tiene insignias, 1 si tiene menos de 5, 3 si tiene más de 10).
b) Implemente una clases *ListaUsuarios* para representar la lista de usuarios del sistema. La clase debe tener métodos para: (1) agregar un nuevo usuario al sistema indicando su nombre; (2) métodos que permitan consultar la lista completa; (3) agregarle una insignia a un usuario; y (4) calcular el porcentaje de usuarios que tienen una determinada insignia que se pase como parámetro.
c) Escriba un programa de prueba que permita primero intentar registrar 10 usuarios (debe verificar que el nombre no exista, y mostrar un mensaje de error si ya existe), luego registrar 50 insignias (ingresando por cada una dos *string*s, nombre de usuario e insignia), y muestre la lista de usuarios completa indicando por cada uno nombre y nivel. Finalmente permita ingresar una insignia e informe el porcentaje de usuarios que la han conseguido.

Ej2(25pts) Sin utilizar la clase std::vector: **a)** Escriba una función *filtrar_rango* que reciba un arreglo de enteros *a* y dos valores *min* y *max*. La función debe generar y retornar un nuevo arreglo que contenga todos los elementos de *a* que no estén en ese rango de valores [*min*;*max*]. Ejemplo: si *a*={ 1,3,4,2,6,8,1,4,3 }, min=3, y *max*=5, entonces debe retornar { 1,2,6,8,1 }. **b)** Escriba un programa cliente que permita ingresar al usuario un arreglo de *n* enteros (*n* también es dato), dos enteros *min* y *max*, y muestre el arreglo que genera la función.

Ej3(30pts) a) Defina una clase para representar a un jugador del juego piedra-papel-tijera. La clase debe guardar el nombre del jugador (que se inicializará en su constructor) y tener un método para jugar que retorne "piedra", "papel" o "tijera". **b)** Defina dos clases para representar a dos formas de jugar en particular: una cuyo método jugar elija su jugada al azar; y otra que siempre juegue "tijera". **c)** Implemente una función *comparar(...)* que pueda recibir dos jugadores de cualquier tipo (los del punto b son solo ejemplos, podría haber más), simule 1000 partidas entre ellos e informe el porcentaje de ganados de cada uno. **d)** Escriba un programa cliente que genere dos jugadores, uno de cada tipo (del punto b), e invoque a la función de prueba.

Ej4(15pts) Explique/justifique: a) ¿Cuáles son las dos formas de sobrecargar un operador? **b)** ¿Siempre se puede elegir entre ambas? **c)** Dada la clase *Recorrido* del recuadro, implemente las sobrecargas necesarias para sumar y restar dos recorridos.

```
class Recorrido {
   float tiempo, distancia;
public:
   Recorrido(float t, float d);
   float VerTiempo() const;
   float VerDistancia() const;
};
```

Solución: ***https://youtu.be/mYVO7ln1duc***