UNIVERSIDAD DE GRANADA
Departamento de Ciencias de la Computación e Inteligencia Artificial

Metodología de la Programación (2019/20)
1º GII / GII-M / GII-ADE
Conv. Ordinaria - 15 de Junio de 2020

Duración: 2.5 horas

Una empresa de distribución desea implementar un sistema de gestión de ventas. Para ello se considerarán las siguientes entidades:

```
class Cliente {
  private
    int id_cliente;
    double descuento;
    string nombre;
    string NIF;
  ......
};
```

```
class  Venta {
  private:
    int id_cliente;
    int id_producto;
    double total;
    ......
};
```

```
class Clientes {
  private:
    int num_clientes;
    Cliente *los_clientes;
    ......
};
```

```
class Ventas {
  private:
    int num_ventas;
    Venta * las_ventas;
    ......
};
```

Los objetos de las clases `Clientes` y `Ventas` son **vectores dinámicos** de objetos de clase `Cliente` y `Venta`, respectivamente.

Paralelamente a los datos que residen en memoria durante la ejecución del programa existen ficheros de texto que contienen datos de clientes y ventas. En la figura 1 mostramos un ejemplo de fichero de clientes (A) y ventas (B). Observe que los ficheros contienen una *palabra mágica* al principio y después los datos se organizan en líneas, separando los distintos valores con espacios:

- Ficheros de clientes: cada cliente aparece en una línea separada, conteniendo los valores de los campos `id_cliente`, `descuento`, `nombre` y `NIF` (separados por espacios).
- Ficheros de ventas: cada venta aparece en una línea separada, conteniendo los valores de los campos `id_cliente`, `id_producto` y `total` (separados por espacios).

Figure 1: (A) Fichero de clientes y (B) fichero de ventas

Los datos de los clientes y ventas se guardan en ficheros de manera que se pueden crear objetos de la clase `Clientes` y `Ventas` a partir de su contenido, y después se puede guardar el contenido de los objetos en ficheros.

Suponga que dispone de los siguientes métodos públicos:

- Para la clase `Cliente` y `Clientes`: ver `cliente.h` y `clientes.h`, respectivamente.
- Para las clases `Venta` y `Ventas` : ver `venta.h` y `ventas.h`, respectivamente.

1. **Deberá implementar cualquier otra función/método que pudiera necesitar, aunque no se le pida explícitamente**.
   Complete los ficheros .h añadiendo los prototipos de los métodos públicos y privados que implemente en el examen, tanto los que se le pidan de manera explícita como los métodos auxiliares adicionales que tuviera que realizar.
2. **Si no se especificara la clase a la que pertenece un método deberá determinar cuál es la adecuada**.
3. **Especifique las precondiciones adecuadas a los datos y los métodos**.
4. **Escriba cuantos comentarios estime oportunos para clarificar el código. Hágalo usando la sintaxis de C++**

◁ Ejercicio 1 ▷ Métodos básicos [2 puntos]

Defina los siguientes métodos:

1. *(0.5 puntos)* **Constructor sin argumentos** para la clase `Clientes` para crear una colección vacía. Indique si es necesario implementar el constructor sin argumentos de la clase `Cliente`. Si lo fuera, impleméntelo y justifique porqué es necesario. Si no lo fuera, explique la razón.
2. *(0.5 puntos)* Implemente el **destructor** de las clases `Cliente` y `Clientes` si fueran necesarios. En caso contrario justifique porqué no es/son necesario(s).
3. *(1 puntos)* Implemente el **constructor de copia** y el **operador de asignación** para las clases `Cliente` y `Clientes` si fueran necesarios. En caso contrario justifique porqué no es/son necesario(s).

◁ Ejercicio 2 ▷ Operadores aritméticos y combinados [3 puntos]

1. *(1.5 puntos)* Sobrecargue el **operador +=** para la clase `Clientes`. El objetivo es incorporar un nuevo cliente a la colección de clientes. El operador recibe un dato `Cliente`. El nuevo cliente se añade **al final**. No puede haber en la colección dos clientes con el mismo `NIF`. Si el cliente ya está en la colección no se hace nada.
   Antes de añadir un dato `Cliente` a la colección se genera un nuevo `id_cliente` (proporcionado por `GeneraIdCliente`) y se modifica el objeto `Cliente` actualizando su `id_cliente` con el valor generado.
2. *(1.5 puntos)* Sobrecargue el **operador binario +** para combinar dos objetos de la clase `Clientes` (podrían ser, por ejemplo, los clientes de dos empresas del grupo). El resultado debe ser un nuevo objeto de la clase `Clientes`.
   Tenga en cuenta que los objetos `Clientes` que se combinan podrían tener valores *repetidos* (p.e. un cliente que está en los dos objetos `Clientes` que se combinan aparece en ambos con el mismo `NIF` -aunque puede tener diferentes `id_cliente`-). Si se detecta un cliente repetido (el mismo `NIF`) se mantienen los datos del cliente de la primera colección (objeto implícito). Los clientes de la segunda colección (la explícita) que **no** están en la primera cambiarán su `id_cliente` por uno nuevo, proporcionado por el método `GeneraIdCliente` (ya implementado).

◁ Ejercicio 3 ▷ Operadores de flujo [1 punto]

1. *(0.5 puntos)* Sobrecargue el operador << para la clase `Clientes`. Cada cliente deberá aparecer en una línea separada, conteniendo los valores de los campos `id_cliente`, `descuento`, `nombre` y `NIF` (separados por espacios). En la figura 1.A puede ver un ejemplo (no tenga en cuenta la primera línea).
2. *(0.5 puntos)* Sobrecargue el operador >> para la clase `Clientes`. Los datos de cada cliente aparecen en el flujo dispuestos en una línea con el formato indicado anteriormente.

◁ Ejercicio 4 ▷ Métodos y funciones para E/S [1.5 puntos]

1. *(0.75 puntos)* Implemente un constructor de la clase `Clientes` con un argumento que indica el nombre de un fichero de texto que contiene una serie de líneas de texto con los datos de los clientes. La primera línea del archivo contiene la cadena **FICHEROCLIENTES**. A continuación vendría la información de los clientes (figura 1.A).
2. *(0.75 puntos)* Implementar un método de la clase `Clientes` para guardar en el fichero `nombre` el contenido del objeto implícito. El contenido de la colección no se modifica.

   ```
   void EscribirClientes (const string nombre) const;
   ```

◁ Ejercicio 5 ▷ Métodos de cálculo Sobrecarga de los operadores relacionales > y == [1 punto]

Implemente los operadores > y == para la clase `Ventas`. El criterio se basa en el valor de la *venta media*. La venta media de un objeto `Ventas` se calcula sumando el importe de todas las ventas y dividiendo por el número de ventas. Es decir, si `ventas_Enero` y `ventas_Febrero` son dos objetos de la clase `Ventas`, entonces `ventas_Enero > ventas_Febrero` si el valor de la venta media de `ventas_Enero` es mayor que el de `ventas_Febrero`.

El operador == se construye también de acuerdo al criterio del valor de la venta media.

◁ Ejercicio 6 ▷ Aplicación [1.5 puntos]

Escriba un programa *completo* que reciba un número indeterminado de nombres de ficheros de ventas e informe de cual es el *mejor*. La comparación se efectuará en base al valor de la *venta media* (pregunta 5).

El programa mostrará también el valor de la venta media y en el caso en el que hubiera más de un fichero con el mismo valor de venta media deberá mostrar los nombres de todos ellos.