Programación 4

LABORATORIO 1

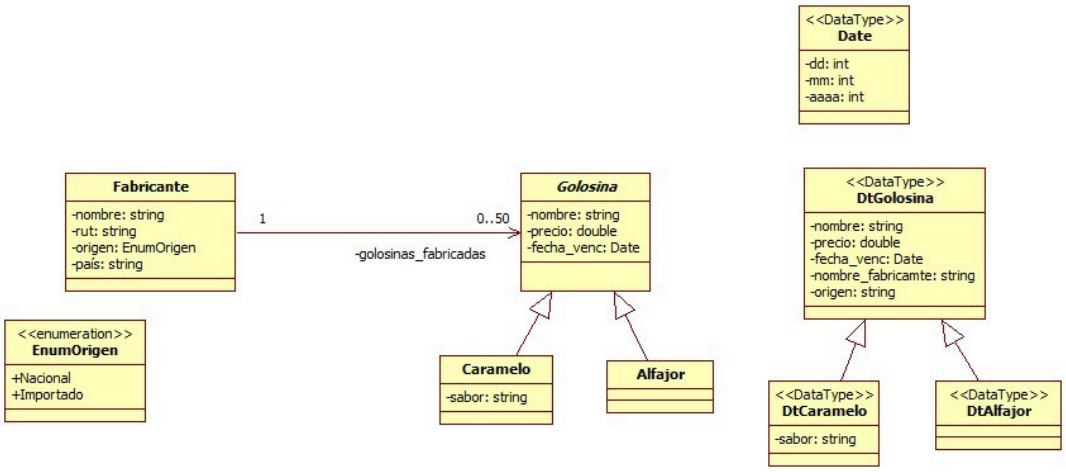
Consideraciones generales:

- La entrega podrá realizarse hasta el **lunes 30 de marzo hasta las 15:00hs.**
- El código fuente y el archivo Makefile deberán ser entregados mediante el EVA del curso dentro de un archivo con nombre <número de grupo>_lab1.zip (o tar.gz). Dentro del mismo archivo comprimido, se deberá agregar un archivo denominado resp_lab1.txt, con las respuestas a las preguntas planteadas, y un archivo integrantes.txt, con los nombres y correos electrónicos institucionales (@fing.edu.uy) de cada uno de los miembros del grupo.
- El archivo Makefile entregado debe ser independiente de cualquier entorno de desarrollo integrado (*IDE*, por sus siglas en inglés).
- Las entregas que no cumplan estos requerimientos no serán consideradas. El hecho de no realizar una entrega implica la insuficiencia del laboratorio completo.

Con este laboratorio se espera que el estudiante adquiera competencias en la implementación de operaciones básicas, el uso básico del lenguaje c++ (que se usará en el laboratorio) y entorno de programación en linux, así como reafirmar conceptos presentados en el curso [1, 2, 3, 4]. También se espera que el estudiante recurra al material disponible en el EVA del curso y que recurra a Internet con espíritu critico, identificando y corroborando fuentes confiables de información.

Problema

Se desea implementar un sistema de bajo porte para el registro de stock de un kiosco. En esta primera versión del sistema sólo se desean registrar golosinas, de las cuales interesa el nombre, precio, fecha de vencimiento y fabricante, en particular las dos clases de golosinas que manejaremos en esta etapa que son caramelos, de los cuales interesa conocer además su sabor, y alfajores. Del fabricante interesa conocer el nombre (que lo identifica), su rut y su origen. Una golosina solamente puede ser fabricada por un fabricante, y no pueden existir dos golosinas con el mismo nombre fabricadas por el mismo fabricante. En caso del fabricante de origen nacional, el país debe ser Uruguay; de no cumplirse esta restricción se debe lanzar una excepción del tipo std::invalid_argument. Ninguno de los atributos de las entidades de la presente realidad pueden ser vacíos.El diagrama que representa a esta realidad se muestra a continuación.



Ejercicio 1

Parte A - Implementar en C++ lo siguiente.

1. Todas las clases (incluyendo sus atributos, pseudoatributos, *getters*, *setters*, constructores y destructores), enumerados y datatypes que aparecen en el diagrama. Para las fechas en caso de recibir dd>31 o dd<1, mm>12 o mm<1 o aaaa<1900, se debe lanzar la excepción *std::invalid_argument*. No se deben hacer más controles que estos en la fecha (ej. La fecha 31/2/2000 es válida para esta realidad).
2. Una función main que implemente las siguientes operaciones:
    a) void agregarFabricante(string nombre,string pais,string rut,EnumOrigen origen)
    b) string* listarFabricantes(int & cant_fabricantes), retorna un arreglo de string con todos los nombres de fabricantes del sistema.
    c) void agregarGolosina(string nombre_fabricante, DtGolosina& golosina), que crea un nuevo tipo de golosina en el sistema, con la información incluida en el parámetro golosina agregándola al fabricante dado como parámetro.En caso de existir dos nombres de golosinas iguales para el mismo fabricante lanzar una excepción *std::invalid_argument*.
    d) void eliminarGolosina(string nombre_fabricante, string nombre_golosina),que elimina la golosina dada por el parámetro nombre_golosina del fabricante identificado por nombre_fabricante.
    e) DtGolosina** obtenerInfoGolosinasPorFabricante(string nombre_fabricante, int& cant_golosinas), retorna una lista con golosinas fabricadas por el fabricante nombre_fabricante. El largo del arreglo golosinas está dado por el parámetro cant_golosinas.

Nota: La función main, para lo que se refiere a este laboratorio mantendrá una colección de fabricantes, implementada como un arreglo de tamaño MAX_FABRICANTES.

3. Agregar la sobrecarga del operador de inserción de flujo (ej. <<) en un objeto de tipo std::ostream. El mismo debe imprimir las distintas clases de golosina(DtCaramelo, DtAlfajor) con el siguiente formato:

   Nombre Golosina:

   Precio: ...

   Nombre Fabricante: ....

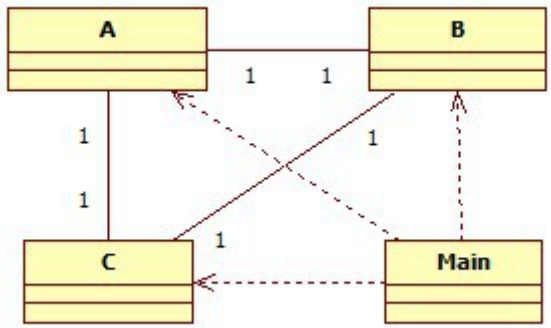
   Origen: ...

   Tipo: ...

Notas: El main debe manejar las excepciones lanzadas por las operaciones de los objetos. Por màs información consulte las referencias [2, 3]

Ejercicio 2

Se busca familiarizarse con la problemática de dependencias circulares en C++ [3]. Para esto se debe implementar y compilar la siguiente estructura dada por el diagrama de abajo.



Preguntas

1. ¿Cuáles son las dependencias circulares que fueron necesarias solucionar para que el programa compile?
2. ¿Qué es *forward declaration*?

Referencias

[1] Eva Programación 4

[2] Eva Programación 4, Material > Bibliografia y material complementario

[3] Eva Programación 4, Material > Material adicional

[4] Eva Programación 4, Unidad de Recursos Informáticos de la Facultad (Configuraciones y Salas estudiantes Linux)