# Programación 4

EXAMEN FEBRERO 2018

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas.
- Escriba su nombre y número de documento en todas las hojas que entregue.
- Numere las hojas e indique el total de hojas en la primera de ellas.
- Recuerde entregar su número de examen junto al examen.
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial.

**Problema 1 (35 puntos)**

Cada verano, los estudiantes de la Fing vacacionan de forma masiva en las playas de la costa Rochense. Sin embargo, en los últimos años los precios de alquileres han subido mucho y cada vez se hace más difícil disfrutar de unas merecidas vacaciones. Motivados por esta problemática, un grupo de estudiantes tuvo la brillante idea de construir una aplicación que le permita a personas que viven todo el año en balnearios, alquilar cuartos, camas e incluso sofás dentro de sus propias casas. De esta forma los "lugareños" también pueden beneficiarse del turismo veraniego y los estudiantes acceden a opciones de alojamiento mucho más económicas.

Como "Surf fingero de Sofás" no era un muy buen nombre para una aplicación decidieron llamarla "CouchSurfFing" y le han encomendaron el desarrollo de la misma.

De cada usuario interesa conocer su email que lo identifica, nombre, fecha de nacimiento y un teléfono. A su vez, los usuarios pueden ser huéspedes, que son quienes buscan alojamiento, o anfitriones, que son quienes ofrecen alojamiento.

Para cada anfitrión se conoce información de su casa (que es única para el usuario), donde recibirá a los huéspedes. De una casa se conoce la dirección, una descripción y una lista de URLs de fotos. Adicionalmente se conoce las opciones de alojamiento que ofrece dentro de la misma. Una opción de alojamiento puede ser una habitación privada, una cama en una habitación compartida o un sofá. Para cada opción se conoce un código que la identifica, su precio por noche e interesa además saber las fechas en que NO se encuentra disponible.

Por otro lado, los huéspedes realizan reservas sobre una opción de alojamiento. Para cada reserva se conoce su código que la identifica, la fecha de cuando comienza y la fecha cuando termina. Adicionalmente, cada reserva tiene un estado que puede ser: VIGENTE, CERRADA, CANCELADA. Toda reserva comienza siendo VIGENTE y puede pasar a CERRADA cuando el usuario se hospeda en el alojamiento reservado, o CANCELADA si se el usuario la cancela antes.

Para "enganchar" más a los nuevos usuarios, los huéspedes pueden realizar una evaluación de un alojamiento luego de haberse hospedado en el mismo, es decir cuando su reserva cambia al estado CERRADA. De una evaluación se conoce un código que la identifica, un puntaje en una escala de [1..5], la fecha en que fue realizada y un comentario.

Se relevó además el siguiente caso de uso:

| | |
|---|---|
| Caso de Uso: | IngresarReservas |
| Actor: | Huésped |
| Descripción: | El caso de uso comienza cuando un huésped indica que desea hacer una o más reservas. Para eso el usuario ingresa la fecha de inicio y la fecha de finalización de la misma. A continuación el sistema lista todos los anfitriones registrados, que tienen al menos una opción de alojamiento disponible entre las fechas ingresadas y el huésped selecciona uno. Luego se listan todas las opciones de alojamiento disponibles para el anfitrión seleccionado, mostrando el tipo (habitación, cama, sofá) y su precio por noche. Finalmente el huésped tiene la opción de seleccionar una y el sistema realiza la reserva. El comportamiento se repite hasta que el usuario no quiera hacer más reservas. |

**Se pide:**

a. Modelo de Dominio de la realidad anterior con restricciones en lenguaje natural. Se deben especificar los tipos de los atributos, así como también los datatypes y enumerados utilizados.

b. Diagrama de Secuencia del Sistema (DSS) del Caso de Uso, incluyendo el uso de datatypes y de manejo de memoria del Sistema, en caso de ser necesario.

**Problema 2 (35 puntos)**

*Parte A:*

Describa el GRASP llamado "Bajo Acoplamiento", incluyendo las posibles desventajas de NO seguir este criterio.

*Parte B:*

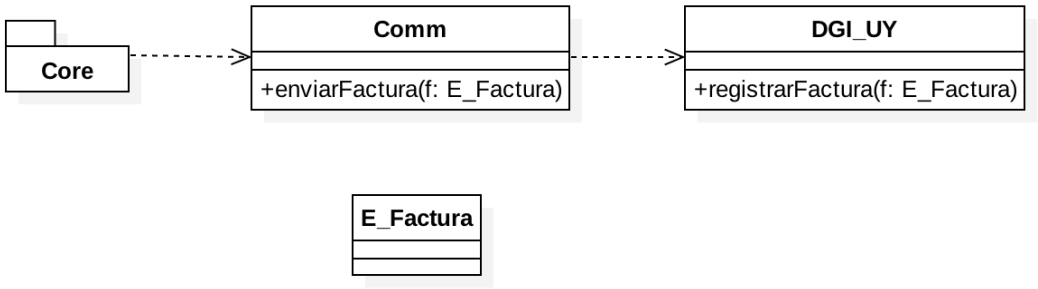
Una empresa de software local dedicada a la facturación electrónica cuenta con un producto que desde hace ya más de un año se comercializa exclusivamente en nuestro país. No obstante, la empresa se encuentra en proceso de expansión hacia otros mercados, lo cual hace que se enfrente a numerosos desafíos, tanto comerciales como técnicos.

En particular, hasta el momento el producto solo se comunica con el ente recaudador de nuestro país (la Dirección General Impositiva, o DGI).

Siendo uno de los pilares de la facturación electrónica el envío inmediato de los comprobantes de las ventas realizadas en los comercios a la DGI (es decir, las facturas), la empresa de software debe modificar su producto de forma que permita comunicarse con los diferentes entes recaudadores de los nuevos mercados a los cuales pretende entrar, adecuándose a cada uno de ellos.

A modo de ejemplo, la DGI de Uruguay recibe únicamente el objeto E_Factura (que contiene todos los datos de la compra realizada y es un estándar ISO) mientras que la DGI de Argentina recibe además un Time Stamp (es decir, la fecha y hora exacta hasta el milisegundo del momento en que se realiza el envío de la factura).

El diseño actual del producto se muestra en el siguiente diagrama, en donde el paquete Core contiene todas las clases que componen el núcleo del producto y no deben ser modificadas (la clase Comm es la encargada de las comunicaciones entre el Core y los entes, siendo parte de lo que se debe rediseñar):




**Se pide:**

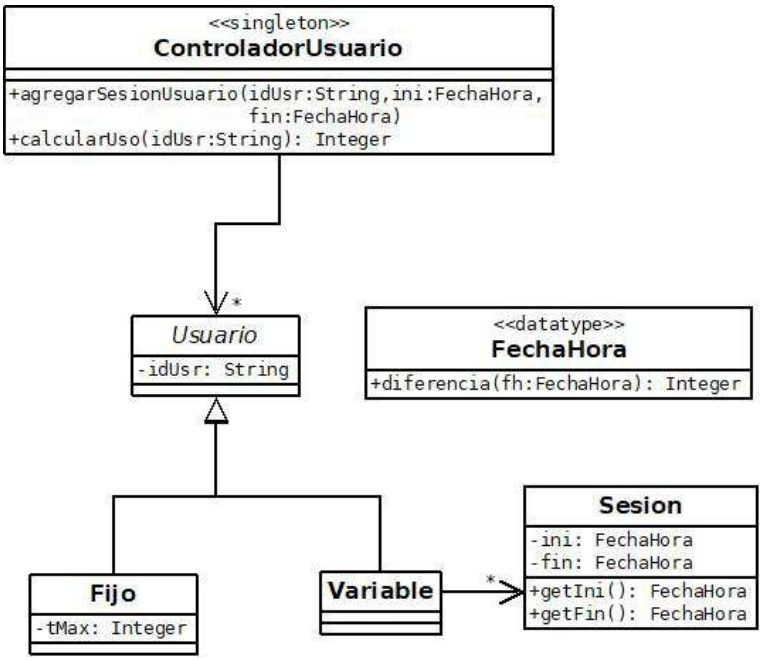
a. Indique qué patrón(es) de diseño utilizaría para permitir la comunicación con nuevos entes recaudadores (ej: con la DGI Argentina), minimizando el impacto de los cambios y sin modificar ninguna clase que se encuentre dentro del paquete Core.

b. Realice el nuevo diagrama de clases correspondiente a su solución.

c. Muestre, mediante un diagrama de comunicación, cómo sería la interacción bajo su nueva solución, cuando una clase del Core desea comunicarse con un ente recaudador (puede utilizar DGI_UY o DGI_AR como ejemplo).

**<u>Problema 3</u> (30 puntos)**

Un servicio de cómputo en la nube ofrece capacidades para cálculo remoto en dos modalidades diferentes: una donde el usuario paga una tarifa fija mediante la cual accede hasta un máximo de tiempo y otra donde el usuario no tiene restricción de tiempo y paga por el tiempo efectivamente utilizado. La figura muestra un Diagrama de Clases de Diseño (DCD) parcial para representar la información de los usuarios.



La operación `agregarSesionUsuario` agrega la información de una sesión correspondiente a un usuario de tipo variable. La operación `calcularUso` devuelve el tiempo total de un usuario cualquiera; para un usuario fijo corresponde al valor de `tMax` mientras que para uno variable corresponde a la suma de los tiempos de todas sus sesiones. La operación `FechaHora::diferencia` devuelve un valor en las mismas unidades que el atributo `tMax`, que son las requeridas por la operación `calcularUso`.

**Se pide:**

a. Agregar al DCD las operaciones necesarias para implementar `agregarSesionUsuario` y `calcularUso`.

b. Implementar en C++ los .h de todas las clases y datatypes del DCD resultante.

c. Implementar las operaciones `agregarSesionUsuario` y `calcularUso` completamente, es decir, implementando todas las operaciones auxiliares que sean necesarias.

**Aclaraciones:**

- Puede utilizar colecciones genéricas (realizaciones de ICollection e IDdictionary) o paramétricas (STL).
- Agregar al DCD, declarar e implementar estrictamente las operaciones necesarias para las implementaciones que se piden, con las siguientes excepciones: (i) incluir atributos y operaciones necesarias para el singleton `ControladorUsuario` y (ii) no implementar el datatype `FechaHora`.
- No incluir directivas al precompilador.