Programación Avanzada

EXAMEN JULIO 2009

30/07/2009

_____ _____

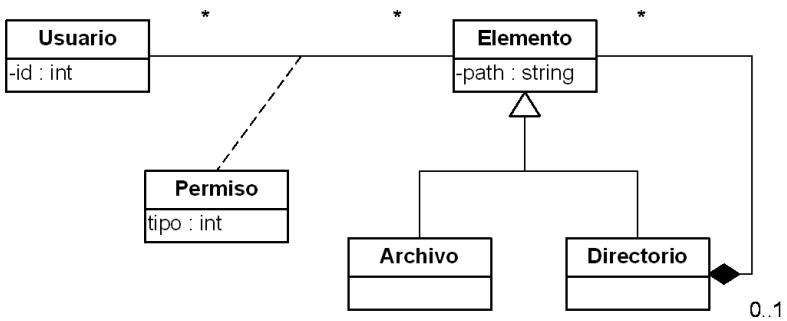
Nombre y Apellido C.I.

Por favor siga las siguientes indicaciones:

- Este examen contiene un total de 5 páginas.
- Escriba con lápiz.
- Escriba las hojas de un solo lado.
- 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.
- Comience cada ejercicio en una hoja nueva.
- El total máximo de puntos del examen es **100 puntos**.
- El examen se aprueba con **60 puntos**.

Problema 1 (30 puntos)

Se analizó un sistema de archivos que permite asignar permisos a usuarios del sistema operativo. El modelo de dominio realizado se puede ver en la figura siguiente. Los elementos del sistema de archivos pueden ser archivos o directorios, quien a su vez puede estar formado por otros elementos. Los elementos se identifican por su dirección absoluta (path). Sobre los elementos se definen permisos de acceso de cierto tipo que se asocian independientemente a cada usuario del sistema.



Además, se definieron dos operaciones que permiten asignar permisos y consultarlos posteriormente. Estas operaciones están definidas por los siguientes contratos reducidos.

Operación	`altaDePermiso(idUsr:Integer, pathEl:String, tipoP:Integer)`
Pre- y poscondiciones	

`pre: Existe una instancia de Usuario con id igual a idUsr.`
`pre: Existe una instancia de Elemento con path igual a pathEl.`
`pre: No existe una instancia del tipo asociativo Permiso con links a las instancias de Usuario y Elemento identificadas anteriormente. En caso de que la instancia de Elemento sea un Directorio, ninguno de sus Elementos están recursivamente linkeados con instancias de Permiso para ese mismo Usuario.`
`post: Se creó una instancia de Permiso con tipo=tipoP y se asoció a las instancias de Usuario y Elemento identificadas en las precondiciones. Si el elemento era un Directorio entonces para todas las instancias de Elemento que lo componen se creó una instancia de Permiso como se definió antes, y así recursivamente.`	

Operación	`consultarPermisos(idUsr:Integer):Set(DataPermiso)`
Pre- y poscondiciones	
`pre: Existe una instancia de Usuario con id igual a idUsr.` `post: Para cada instancia de Elemento con el cual esté linkeado la instancia de Usuario, por medio del tipo asociativo Permiso, se define un datavalue de tipo DataPermiso que contiene el tipo de Permiso y path del Elemento relacionado con el Usuario. Se devuelve el conjunto de datavalues en cuestión.`	

Se pide:

a. Realice los Diagramas de Comunicación para las operaciones especificadas en los contratos. No es necesario indicar las visibilidades.

b. Realice el Diagrama de Clases de Diseño completo de la solución.

Problema 2 (40 puntos)

Se desea desarrollar un prototipo de un sistema de reserva y compra de boletos por Internet para los vuelos de una aerolínea. Los analistas del proyecto generaron la siguiente descripción de la realidad:

> *Todo vuelo parte de un aeropuerto y finaliza en un aeropuerto. De los aeropuertos se conoce el nombre (que lo identifica), la ciudad y el país al que pertenecen. De los vuelos se conoce el código de vuelo (que lo identifica), la fecha de salida, la hora de salida, la hora de llegada. Los vuelos pueden ser simples, cuando van de un aeropuerto a otro sin escalas, o compuestos, cuando para llegar al destino final se debe hacer escalas o conexiones. Cada vuelo simple tiene una cantidad fija de asientos que pueden ser reservados o vendidos. En el caso de vuelos compuestos, las reservas y ventas implican la reserva o venta de asientos en cada uno de los tramos (vuelos simples) que forman parte del vuelo compuesto. De cada asiento se conoce su número, la clase a la que pertenece, su costo, si se encuentra en una salida de emergencia o no y si está reservado o no. Los clientes tienen itinerarios. Los itinerarios agrupan todos los asientos de vuelos simples que un cliente reserva o compra para llegar a un destino. Los itinerarios se crean cuando un cliente hace una reserva para un vuelo. Luego, el cliente puede confirmar el itinerario pagando el importe total correspondiente utilizando una tarjeta de crédito. De un cliente se conoce, el nombre, el número de pasaporte (que lo identifica) y la nacionalidad. De los itinerarios se conoce el vuelo al que corresponde y el costo que se calcula a partir del costo de los asientos.*

i. Realice el Modelo de Dominio de la realidad planteada (exclusivamente el diagrama de modelo de dominio UML y las restricciones en lenguaje natural). No contemple las restricciones que hagan referencia a las fechas y horas. Contemple completa y exclusivamente la descripción de la realidad expuesta anteriormente.

Considere la siguiente descripción del caso de uso *Compra de Ticket a Partir de un Itinerario*

> *El cliente ingresa su número de pasaporte, el sistema muestra todos los itinerarios del cliente. El cliente va seleccionando, de a uno, los itinerarios que desea comprar. El sistema calcula el costo total del itinerario y muestra la información al cliente. El cliente confirma que desea seguir con el proceso y el sistema solicita la información de la tarjeta de crédito (número y fecha de vencimiento). El sistema autoriza el pago y marca los asientos como vendidos.*

ii. Realice el Diagrama de Secuencia del Sistema del caso de uso *Compra de Ticket a Partir de un Itinerario*. Especifique los tipos de parámetro y retorno de las operaciones.

Problema 3 (30 puntos)

a) Considerar la siguiente definición de clase en C++, suponiendo que todas las operaciones están correctamente implementadas. Considerar también el siguiente procedimiento `main()`.

```
class Clase {
    private:
        int entero;
        float real;
    public:
        Clase();
        Clase(int, float);
        Clase(Clase &);
        ~Clase();
        Clase operator=(Clase &);
        Clase & operator+(Clase);
        void desplegar();
};
```

```
1: void main() {
2:    Clase c1;
3:    Clase c2(3,2.54);
4:    Clase c3=c2;
5:    c1=c2+c3;
6:    c1.desplegar();
7: }
```

Suponer que se ejecuta el programa. Indicar, para cada línea y en orden, cada operación invocada a raíz de dicha ejecución, justificando. Recordar que una línea de código puede causar que se invoque más de una operación.

b) En cuanto al patrón Singleton
   i. Explicar en no más de **3 líneas** cuál es su objetivo.
   ii. De un ejemplo en C++ (archivos de cabecera y de implementación) de una clase en la cual se aplique dicho patrón. No es necesario incluir directivas al preprocesador.