Programación Avanzada
SOLUCIÓN
EXAMEN FEBRERO 2011

Por favor siga las siguientes indicaciones:
- Escriba con lápiz y de forma prolija.
- 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 examen es individual y sin material. APAGUE SU CELULAR.
- La duración del examen es de 3 horas.

Problema 1 (20 puntos)

a)
i. Describa brevemente las técnicas de identificación de conceptos vistas en el curso.
ii. Muestre, mediante un ejemplo sencillo, como se define utilizando UML un tipo no primitivo y un atributo cuyo tipo es el tipo no primitivo definido anteriormente.

b) Una empresa consultora en gestión de proyectos desea desarrollar un sistema para la planificación y control de avance de sus proyectos. Se le ha encargado a usted que realice el análisis de requerimientos.

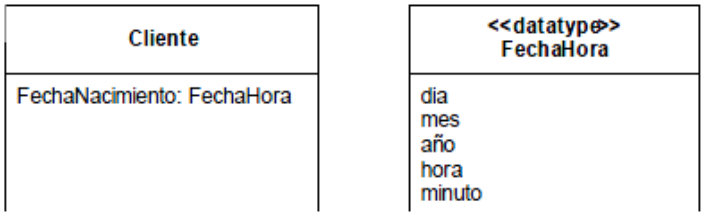
De cada proyecto interesa registrar su nombre (que lo identifica) y una breve descripción.
Un proyecto puede tener asociados planes para su concreción. De cada plan se conoce su nombre (que lo identifica) y una breve descripción de sus objetivos. Un plan consta de un conjunto no vacío de tareas planificadas para lograr cumplir sus objetivos. De cada tarea, el sistema debe registrar su nombre (que la identifica) y su descripción. Una tarea debe pertenecer al menos a un plan. Las tareas de un plan pueden estar relacionadas con otras tareas del mismo plan de las que depende para poder ser realizada. Esta dependencia varía de plan en plan, es decir, para una misma tarea planificada, sus dependencias son distintas en los distintos planes a los que pertenece. Finalmente, el sistema debe verificar que una tarea no dependa de sí misma. Sin embargo, no es necesario que el sistema detecte dependencias circulares entre las tareas.

Se pide: Realice el modelo de dominio correspondiente y escriba en lenguaje natural sus restricciones.

SOLUCION:

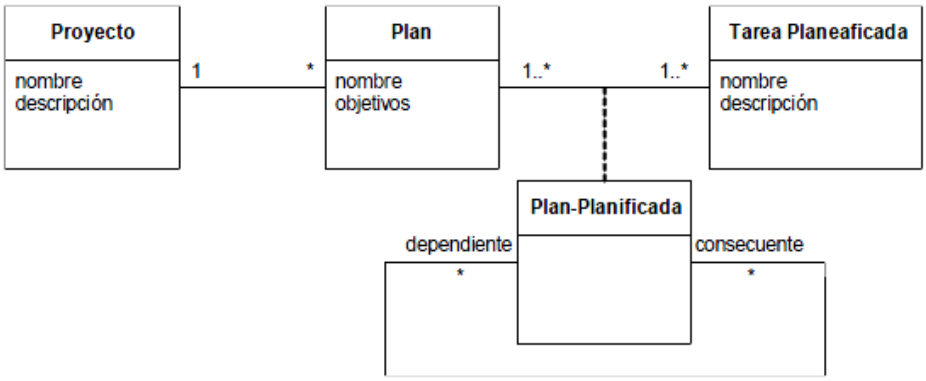
a)

ii.



b)

i.



Restricciones:

context Proyecto
-- El nombre identifica el proyecto.

context Plan
-- El nombre identifica al plan.

context Tarea
-- El nombre identifica la tarea.

context Plan-Planificada
-- Las tareas de un plan sólo pueden estar relacionadas con otras tareas del mismo plan.
-- Análogo para los dependientes (no requerida, se pone a modo de aclaración)
-- Una tarea no puede depender de sí misma
-- Análogo para los dependientes (no requerida, se pone a modo de aclaración)

Problema 2 (20 puntos)

a) Conteste brevemente qué es un contrato de software y cómo se relaciona con los DSS.

b) Actualmente la ciudad de Montevideo (y en particular la IMM) está implementando el nuevo Sistema de Transporte Metropolitano (STM). Este cambio está orientado a mejorar la movilidad de los ciudadanos (usuarios del nuevo sistema de transporte) en todo el departamento y tiene previstas ampliaciones para toda el área metropolitana.
Este nuevo sistema incorpora la utilización de nueva tecnología: la tarjeta inteligente. Esta tarjeta inteligente será utilizada por los usuarios del sistema para realizar viajes a través de la metrópolis.

A continuación se presenta el Caso de Uso que el equipo de trabajo actual ha generado para el Uso de la Tarjeta:

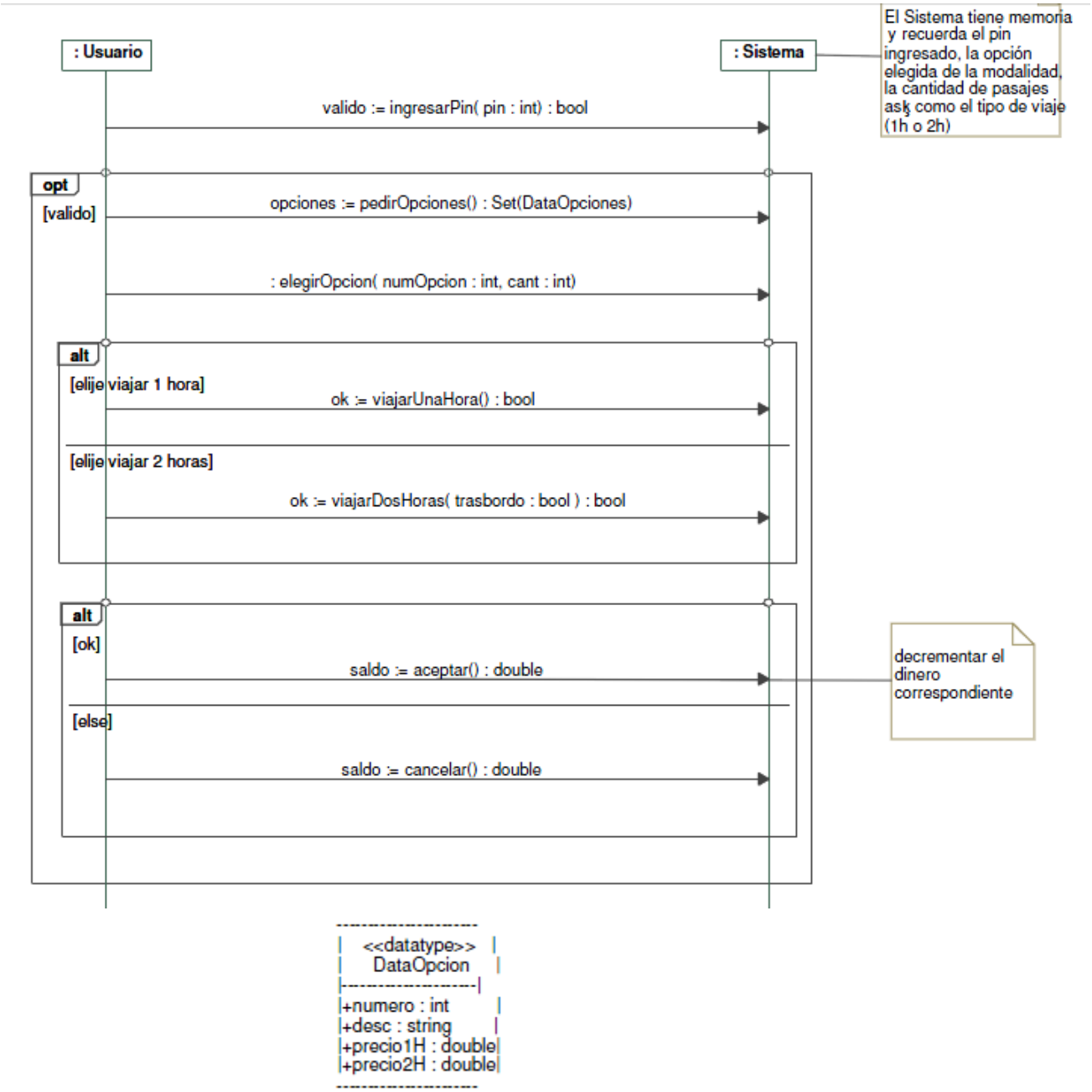
Nombre:	Uso de la Tarjeta	**Actor:**	Usuario del STM
Sinopsis:	Este caso de uso comienza cuando el usuario ingresa el PIN de su tarjeta inteligente. Si éste es correcto, el usuario deberá pedir un listado con las modalidades de viajes disponibles en ese momento. El listado de opciones de modalidades está compuesto por un número no conocido a priori de opciones conteniendo: número de opción (por ejemplo: 1), descripción de la opción (por ejemplo: viaje metropolitano), el precio de esa modalidad para un viaje de 1 hora (por ejemplo: 10 pesos) y el precio de esa modalidad para un viaje de 2 horas (por ejemplo: 15 pesos). Una vez que el usuario ingresa el número de opción de la modalidad deseada de viaje así como la cantidad de boletos a comprar, deberá elegir por cuánto tiempo desea viajar, existiendo siempre dos tipos de duración: viaje por 1 hora y viaje por 2 horas. Sólo para el caso de viajar por 2 horas se debe indicar si se realizará trasbordo o no, ya que esto si bien no incide en el precio, debe constar en el boleto a imprimir. Luego de seleccionar el tiempo del viaje el Sistema controla que el saldo de la tarjeta sea suficiente según lo seleccionado, y en caso de serlo el usuario confirma la compra y de lo contrario cancela la compra. Ya sea que se acepte o cancele el viaje, el Sistema devolverá el saldo final de la tarjeta.		

Se pide: Realice un Diagrama de Secuencia del Sistema para el Caso de Uso. Incluya los tipos de los parámetros y valores de retorno de las operaciones.

SOLUCION:

a) Un contrato de software especifica (declarativamente) el comportamiento o efecto de una operación. Existe un contrato de software por cada operación del DSS.

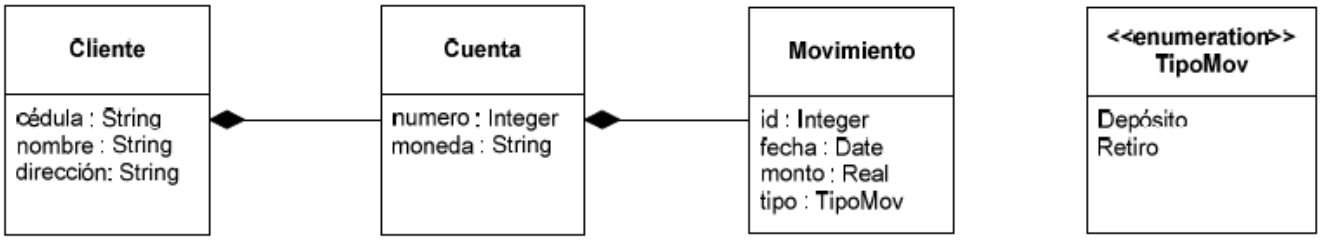
b)



Problema 3 (30 puntos)

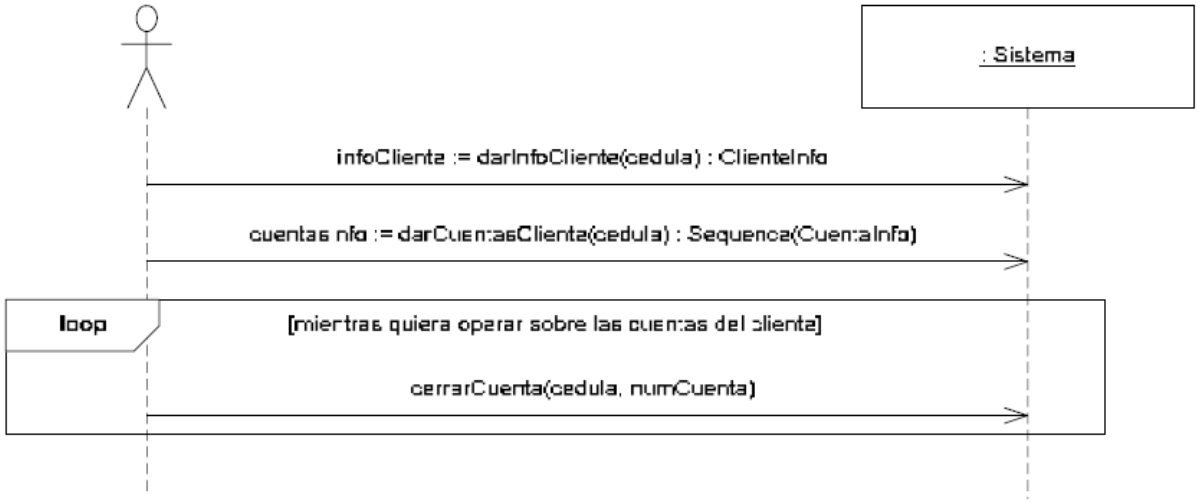
a) Defina brevemente los distintos tipos de visibilidad vistos en el curso.

b) Un banco está desarrollando un nuevo programa para gestionar los clientes y sus cuentas.
Ya se cuenta con una primera versión del modelo de dominio y se necesita desarrollar el Caso de Uso correspondiente a cerrar una cuenta de un cliente (Cerrar Cuenta).
A continuación se presenta el modelo de dominio y la descripción del caso en cuestión.



Notas:
· El atributo id identifica los movimientos y es auto-generado.
· El atributo número identifica las cuentas y es auto-generado.
· El atributo cédula identifica a los clientes.

Nombre:	Cerrar Cuenta	**Actor:**	Usuario
Sinopsis:	Una vez que el usuario ha iniciado la sesión y se encuentra en la pantalla principal, selecciona la opción de ver información de un cliente. El sistema solicita entonces la cédula del cliente y luego muestra sus datos (cédula, nombre y dirección) y una lista de sus cuentas (mostrando para cada cuenta el número y la moneda). El usuario indica al sistema la cuenta a borrar. El sistema borra la cuenta y actualiza la pantalla permitiendo al usuario seguir borrando cuentas. Para finalizar el usuario selecciona la opción de salir		



Se pide:
i. Especificar el contrato de la operación Cerrar Cuenta.
ii. Realizar el diagrama de comunicación de la operación Cerrar Cuenta.

SOLUCION:

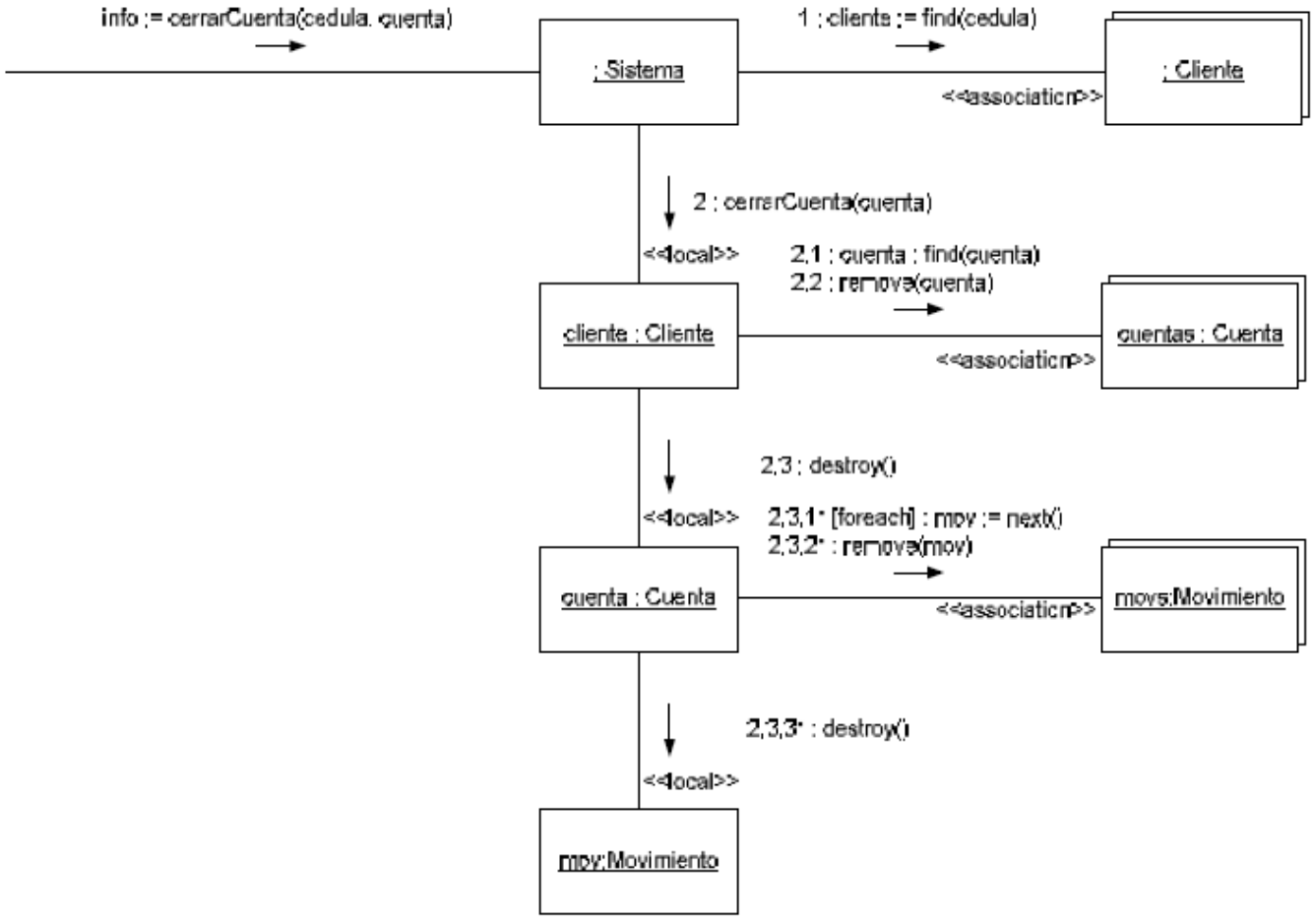
b) i)

void cerrarCuenta(cedula: int, nroCuenta: int)

pre: existe una instancia de Cliente identificada por la cédula = cedula.
pre: existe una instancia de Cuenta identificada por el número = nroCuenta.
pre: existe link entre la instancia Cuenta identificada por el número = nroCuenta y la instancia Cliente identificada por la cédula = cedula.

post: no existe instancia de Cuenta identificada por el número = nroCuenta.
post: no existen instancias de Movimiento que estaban asociadas con la instancia de Cuenta identificada por el número = nroCuenta.
post: se elimina el link entre la instancia de Cuenta identificada por el número = nroCuenta y la instancia de Cliente identificada por cédula = cedula.

ii)

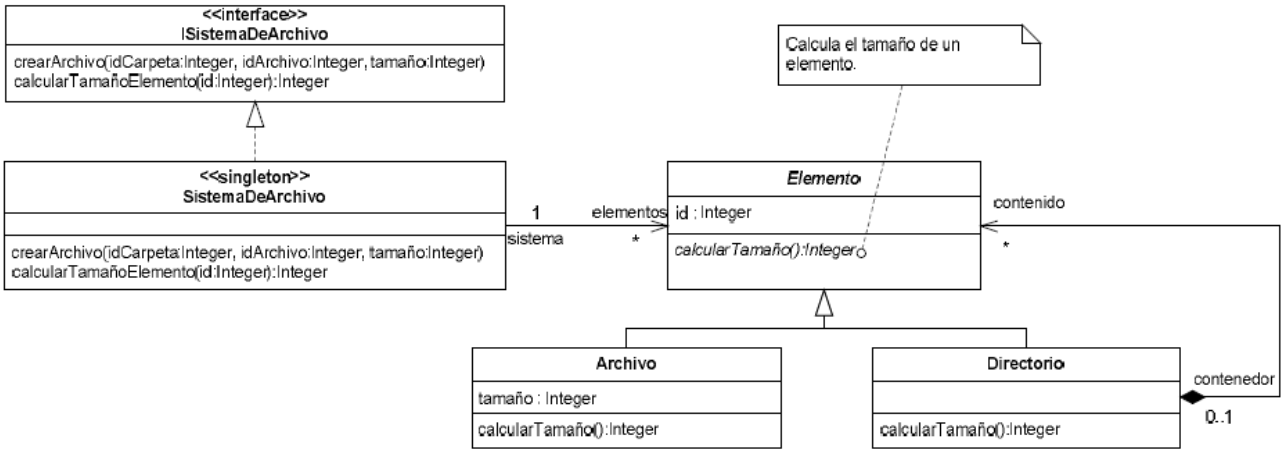


Problema 4 (30 puntos)

a) Enumere los diferentes tipos de relaciones entre elementos que pueden aparecer en un diagrama de clases de diseño e indique como se implementan en C++.

b) Se le ha encomendado la tarea de implementar un prototipo de sistema de archivos. El sistema de archivos contendrá elementos, los cuales tendrán un identificador y podrán ser de dos tipos: directorios, que a su vez podrán contener a otros elementos; o archivos que registrarán el tamaño ocupado. En particular, se le ha pedido implementar la operación para calcular el espacio total ocupado por un elemento del sistema.

Se le ha entregado el siguiente diseño parcial que usted deberá respetar:



Se le ha entregado también la siguiente especificación para la operación pedida:

Operación	SistemaDeArchivo::calcularTamañoElemento(idElemento)
Descripción	Calcula el tamaño del elemento cuyo atributo id = idElemento. Si el elemento es un archivo, el resultado es su tamaño; si es un directorio es la suma de los tamaños de los elementos que contiene.

Se pide:
i) Implementar los .h de la interfaz ISistemaDeArchivo y de las clases SistemaDeArchivo, Elemento, Archivo y Directorio.

ii) Implementar en C++ la operación especificada anteriormente y todas aquellas operaciones necesarias para su implementación que pertenezcan a las clases SistemaDeArchivo, Elemento, Archivo y Directorio.

Observaciones:
· Asuma que existe una implementación estándar de las interfaces ICollectible, ICollection, IIterator, IDictionary e IKey.
· Asuma que existe una clase Lista que realiza las interfaces ICollection e IDictionary y una clase KeyInteger que realiza la interfaz IKey.
· No defina colecciones concretas.
· No incluya directivas de preprocesador.
· Se debe respetar el diseño parcial dado.

SOLUCION:

a)
· Dependencia.
Se implementa utilizando la directiva de preprocesador #include:
#include"ClaseRelacionada.h"

· Asociación con multiplicidad mayor que 1.
Se implementa mediante la utilización de colecciones o diccionarios:
ICollection* nombreDeRol;

· Asociación con multiplicidad 1 o 0..1.
Se implementa mediante atributos de tipo puntero a la clase relacionada:
ClaseRelacionada* nombreDeRol;

· Generalización.
Se implementa mediante la utilización de la herencia pública de C++:
class A : public B

· Realización.
Se implementa mediante la utilización de la herencia pública de C++:
class A : public IA

b)

```
class ISistemaDeArchivo {
        public:
                virtual void crearArchivo(int idDir, int idArch, int tam) = 0;
                virtual int calcularTamanoElemento(int id) = 0;
                virtual ~ISistemaDeArchivo();
};

class SistemaDeArchivo : public ISistemaDeArchivo {
        public:
                ~SistemaDeArchivo();
                void crearArchivo(int idDir, int idArch, int tam);
                int calcularTamanoElemento(int id);
        private:
                IDictionary* elementos;
                SistemaDeArchivo();
};

class Elemento: public ICollectible {
        public:
                Elemento(int id);
                ~Elemento();
                virtual int calcularTamano() = 0;
        private:
                int id;
};
```

```
class Archivo: public Elemento {
	public:
		Archivo(int id, int tam);
		int calcularTamano();
	private:
		int tamano;
};

class Directorio : public Elemento {
	public:
		Directorio(int id);
		~Directorio();
		int calcularTamano();
		Archivo* crearArchivo(int idArch, int tam);
	private:
		IDictionary* contenido;
};

int SistemaDeArchivo::calcularTamanoElemento(int id) {
	IKey* ek = new KeyInteger(id);
	Elemento* elem = (Elemento*) elementos->find(ek);
	delete ek;
	if (elem != NULL)
		return elem->calcularTamano();
	else
		throw invalid_argument("No existe elemento con el id pasado");
}

int Directorio::calcularTamano() {
	IIterator* it = this->contenido->getIterator ();
	Elemento* elemento;
	int tam = 0;
	while(it->hasCurrent ()) {
		elemento = (Elemento*) it->getCurrent();
		tam += elemento->calcularTamano();
		it->next();
	}
	delete it;
	return tam;
}

int Archivo::calcularTamano() {
	return this->tamano;
}
```