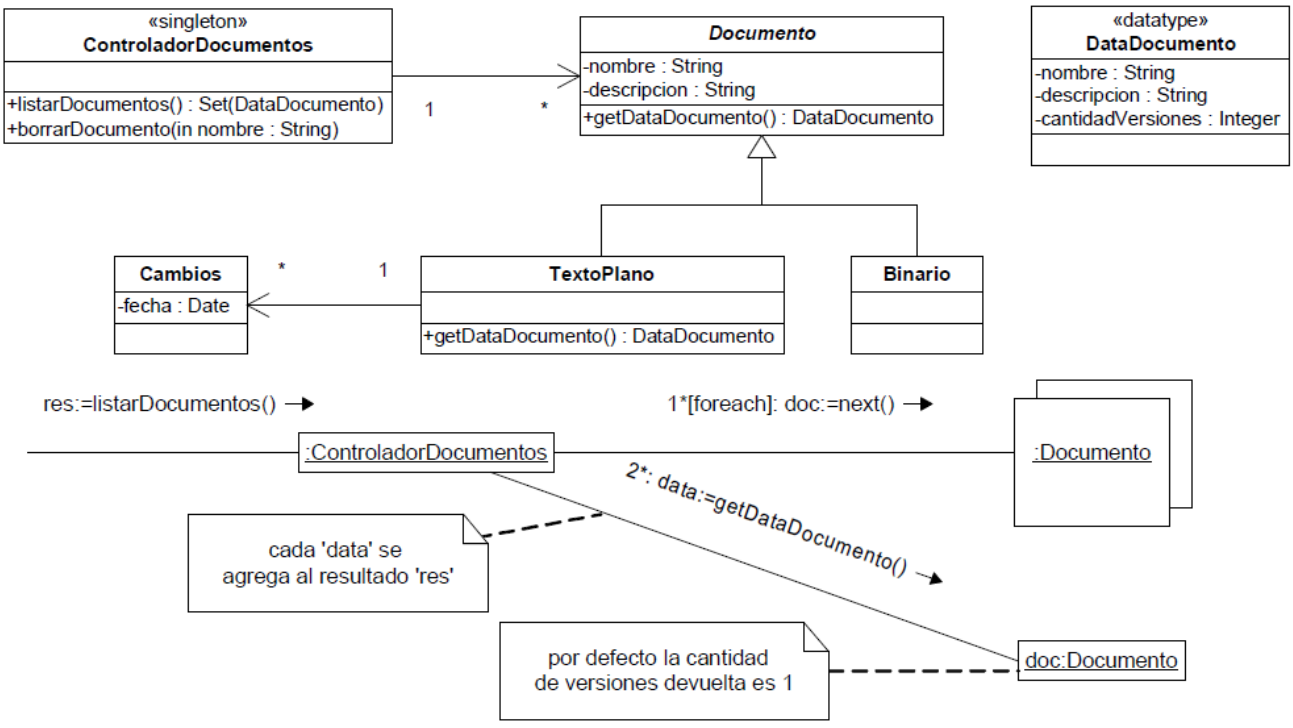
Segundo Parcial 2012

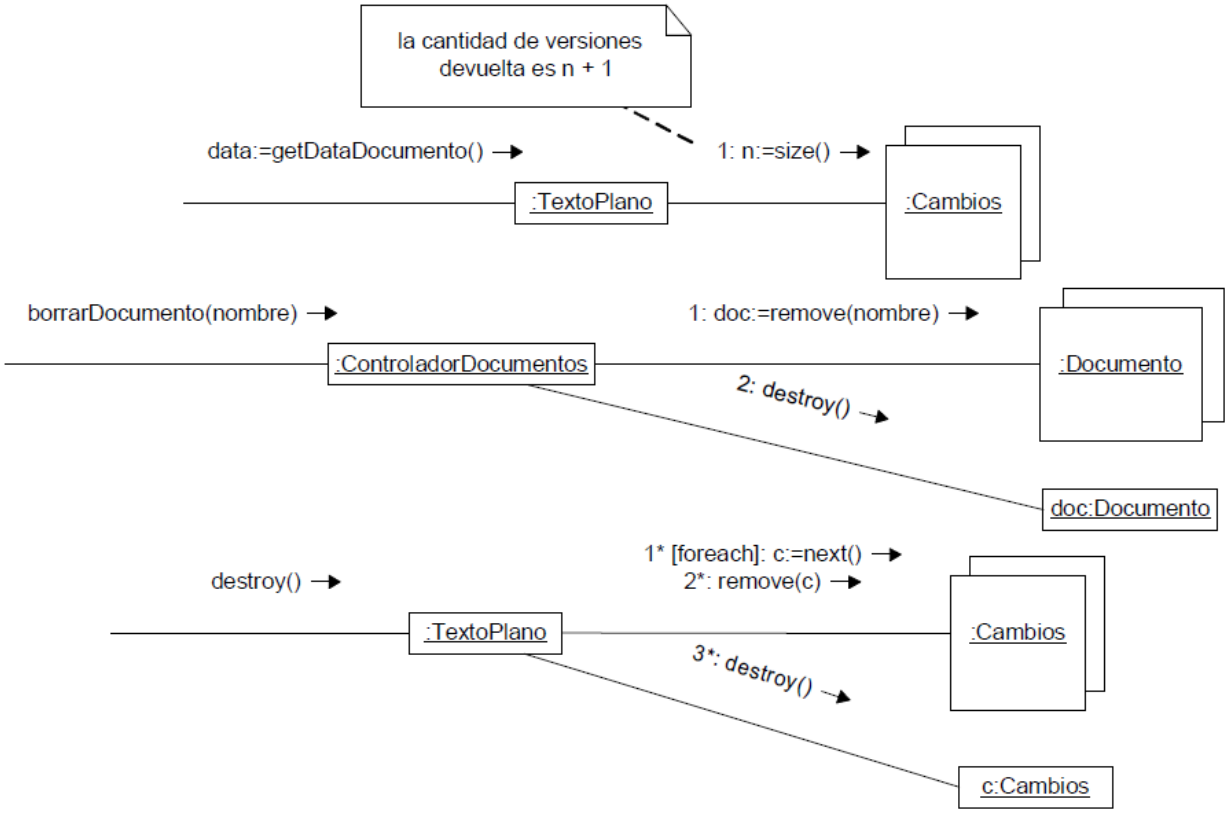
Presentar en la resolución del parcial:

· Indique su nombre completo y número de cédula en cada hoja.
· Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.
· Escriba las hojas de un solo lado.
· **Comience cada ejercicio en una hoja nueva.**
· El parcial es individual y sin material. **APAGUE SU CELULAR.**
· **Escriba con lápiz y de forma prolija.**
· Duración: 2:30 horas.

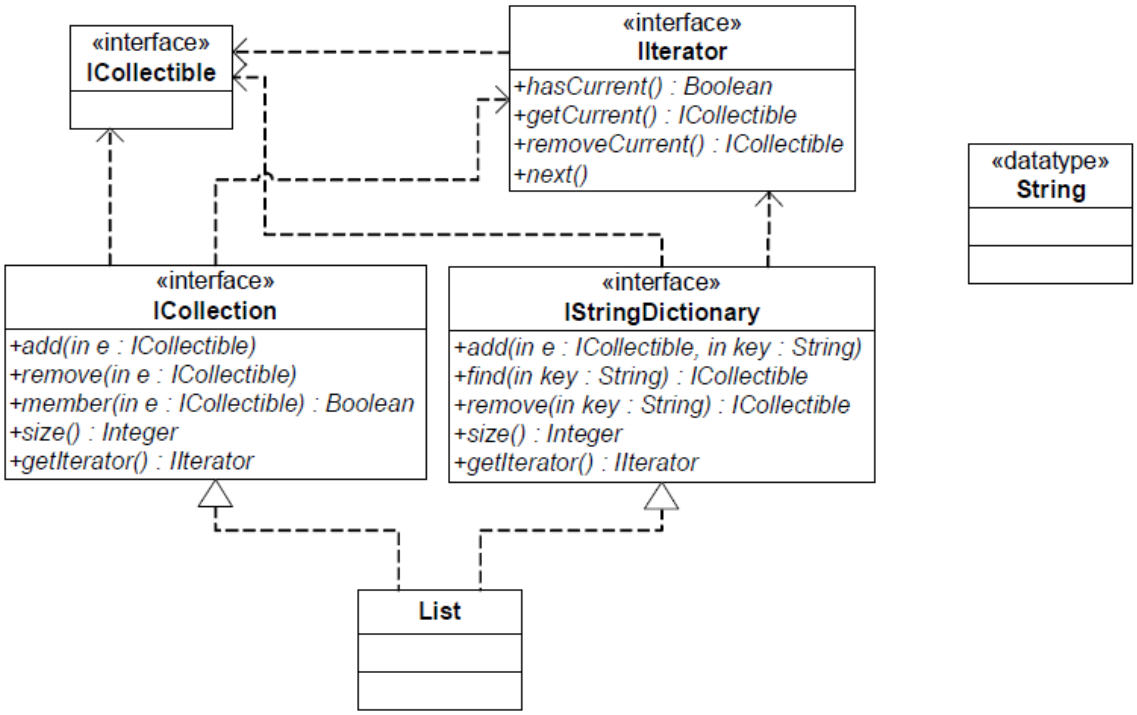
Problema 1 (30 puntos)

Se desea construir un repositorio de documentos. El mismo debe ser capaz de manejar tanto documentos en formato binario (por ejemplo PDF's) como documentos de texto plano. De los últimos es necesario mantener registrados todos los cambios que hayan sufrido desde su creación. Como parte de la etapa de diseño se construyó la siguiente colaboración:





Se cuenta con las siguientes clases ya implementadas como parte de la infraestructura:



SE PIDE:
Implementar en C++ completamente la colaboración salvo la clase Cambios.
Incluir el código relativo al patrón Singleton en la clase ControladorDocumentos.

OBSERVACIONES:
i. No es necesario definir colecciones concretas.
ii. No es necesario incluir directivas al preprocesador en el código.
iii. Le operación removeCurrent() de la interfaz IIterator remueve el elemento actual de la colección y lo devuelve. A la vez posiciona el iterador en el elemento siguiente al removido.

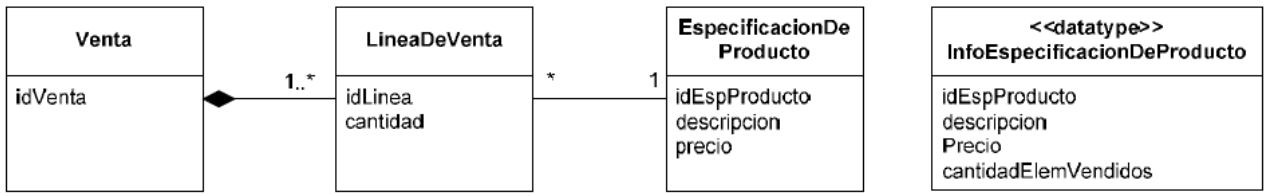
```
// Documento.hh
class Documento : public ICollectible{
	private:
		String nombre;
		String descripcion;

	public:
		virtual DataDocumento getDataDocumento();
		virtual ~Documento();
};

// Documento.cc
DataDocumento Documento::getDataDocumento(){
	return DataDocumento(this->nombre, this->descripcion, 1);
}

Documento::~Documento(){}

// Binario.hh
class Binario : public Documento{
};

// TextoPlano.hh
class TextoPlano : public Documento{
	private:
		ICollection * cambios;

	public:
		DataDocumento getDataDocumento();
		~TextoPlano();
};

// TextoPlano.cc
DataDocumento TextoPlano::getDataDocumento(){
	DataDocumento aux = Documento::getDataDocumento();
	aux.setCantidadVersiones(this->cambios->size() + 1);
	return aux;
}

TextoPlano::~TextoPlano(){
	IIterator * i = this->cambios->getIterator();
	while (i->hasCurrent())
	{
		delete i->removeCurrent();
	}
	delete i;
	delete this->cambios;
}
```

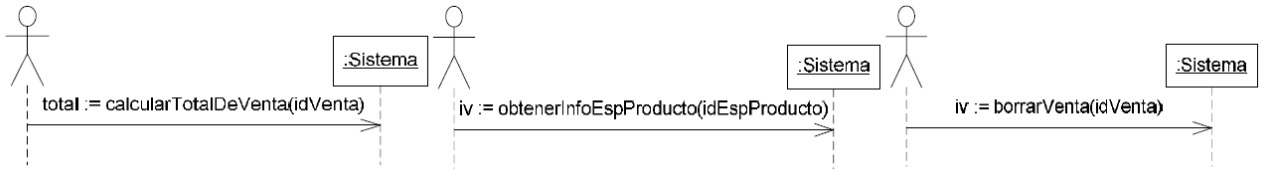
```
// ControladorDocumentos.hh
class ControladorDocumentos{
	private:
		static ControladorDocumentos * instance;
		IStringDictionary * documentos;
		ControladorDocumentos();

	public:
		static ControladorDocumentos * getInstance();
		ICollection * listarDocumentos();
		void borrarDocumento(String nombre);
};

// ControladorDocumentos.cc
ControladorDocumentos * ControladorDocumentos::instance = NULL;

ControladorDocumentos::ControladorDocumentos(){
	this->documentos = new List();
}

ControladorDocumentos * ControladorDocumentos::getInstance(){
	if (ControladorDocumentos::instance == NULL){
		ControladorDocumentos::instance = new ControladorDocumentos();
	}
	return ControladorDocumentos::instance;
}

ICollection * ControladorDocumentos::listarDocumentos(){
	IIterator * i = this->documentos->getIterator();
	List * resultado = new List();
	while (i->hasCurrent())
	{
		// Se crea una copia para devolver
		resultado->add(new DataDocumento(
		((Documento*)i->getCurrent())->getDataDocumento()));
		i->next();
	}
	delete i;
	return resultado;
}

void ControladorDocumentos::borrarDocumento(String nombre){
	delete this->documentos->remove(nombre);
}
```

Problema 2 (30 puntos)

a) Describa brevemente 3 criterios GRASP.

b) La empresa para la cual usted trabaja se ha embarcado en el desarrollo de un sistema para el registro de sus ventas. Como resultado de la fase de análisis, además de generarse los artefactos habituales, se detectó que las clases fuertes del modelo de dominio son Venta y EspecificaciónDeProducto. Además, se ha decido utilizar un controlador de Fachada que será responsable de manejar las colecciones de las clases fuertes (en otras palabras, no habrán manejadores).
A usted se le han asignado los casos de uso calcular total de venta, obtener información sobre un producto y borrar una venta. El modelo de dominio, los DSS y los contratos para estos casos de uso se muestran a continuación.





Nombre	**Calcular total de una venta**
Operación	`calcularTotalDeVenta(idVenta:Integer):Integer`
Entrada	idVenta – Identificador de la venta
Salida	El monto total de la venta
Descripción	Calcula el total de una venta.
Pre- y poscondiciones	

`pre 1: Existe una instancia de Venta cuyo atributo idVenta se corresponde con el valor del parámetro de entrada idVenta.`
 	
`post: La salida se calcula como la suma de los subtotales de las líneas de venta para las cuales existe un link con la venta identificada por idVenta. El subtotal de cada línea de venta se calcula como la multiplicación entre el valor de su atributo cantidad y el valor del atributo precio de la instancia de EspecificaciónDeProducto para la cual existe un link con la línea de venta.`	

Nombre	**Obtener información sobre un producto**
Operación	`obtenerInfoEspProducto(idEspProd:Integer):InfoEspecificaciónDelProducto`
Entrada	idEspProd – Identificador de la especificación del producto vendido.
Salida	La información del producto indicado.
Descripción	Obtiene la información del producto indicado.
Pre- y poscondiciones	

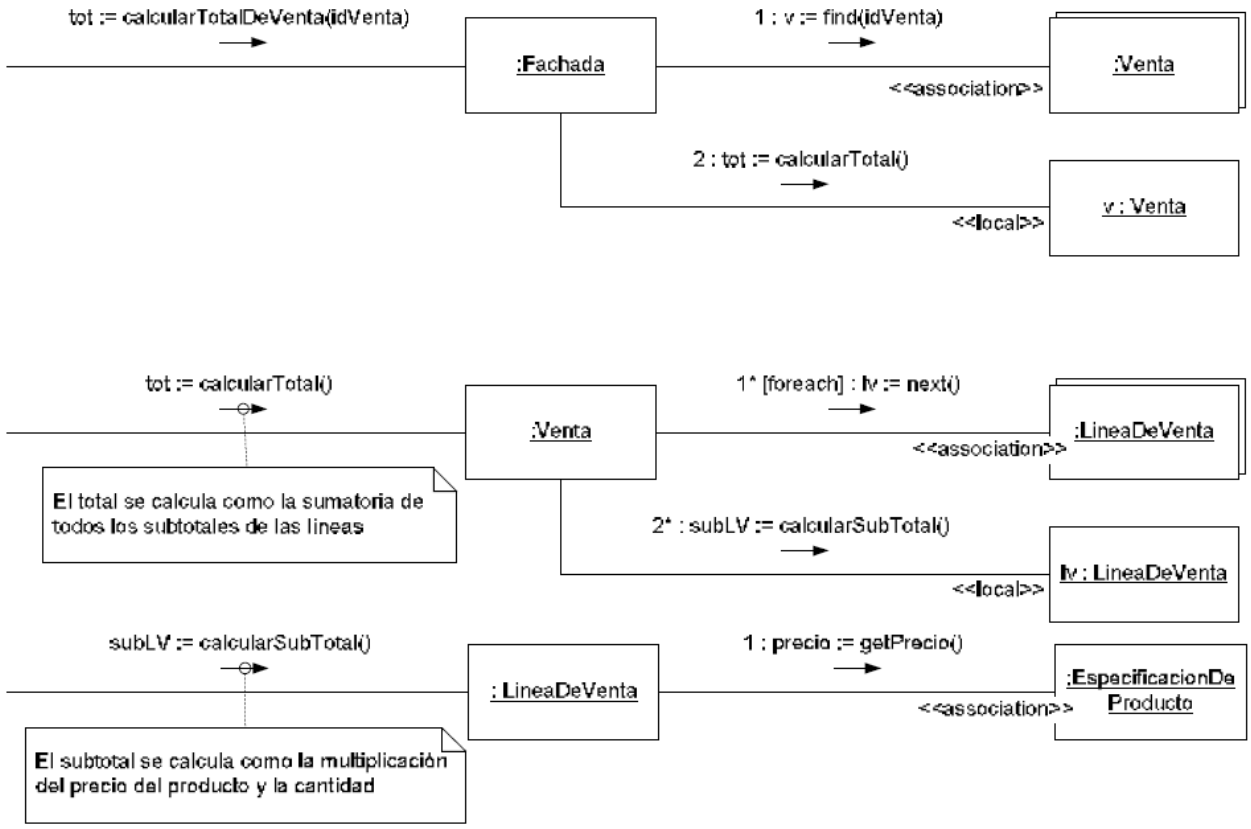
`pre 1: Existe una instancia de EspecificacionDeProducto para la cual el valor de su atributo IdEspProducto se corresponde con el valor indicado en el parámetro idEspProd.`
 	
`post 1: La salida es un datavalue de tipo InfoEspecificacionDeProducto. El valor del atributo idEspProducto se corresponde con el valor del parámetro idEspProd. Los valores de los atributos descripción y precio se corresponden con los valores de los atributos de igual nombre de la instancia de EspecificaciónDeProducto. El valor del atributo cantidadElemVendidos se calcula como la suma de las cantidades de todas las líneas de Venta para las cuales existe un link con la instancia de EspecificaciónDeProducto.`	

Nombre	**Borrar una venta**
Operación	`borrarVenta(idVenta:Integer)`
Entrada	idVenta – Identificador de la venta a borrar.
Salida	No tiene.
Descripción	Obtiene la información del producto indicado.
Pre- y poscondiciones	

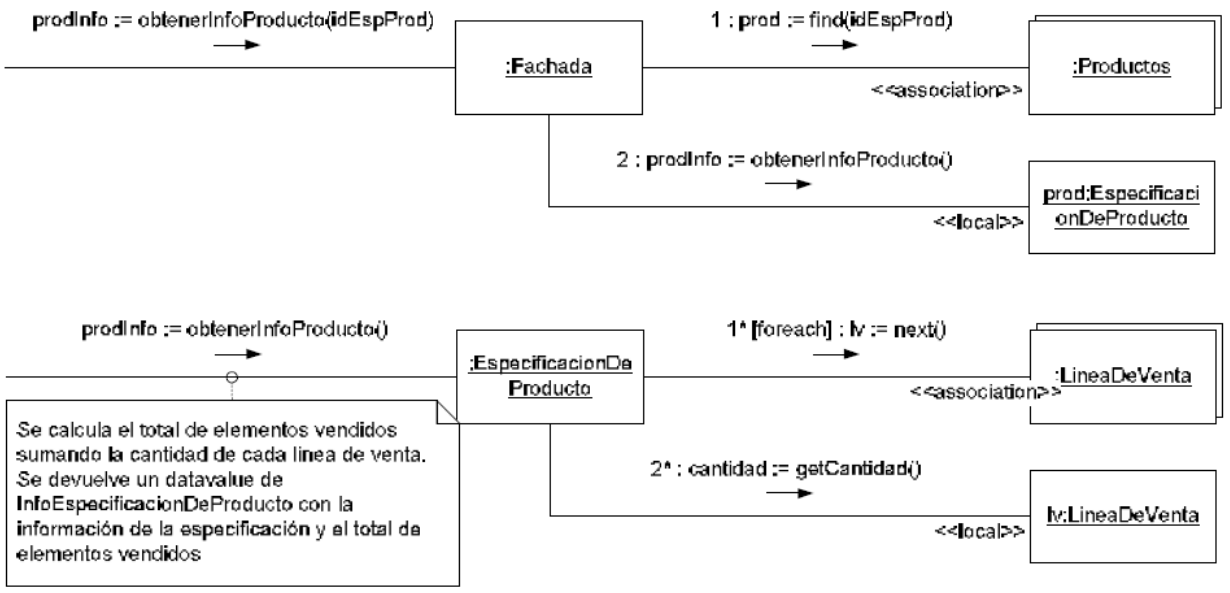
`pre 1: Existe una instancia de Venta para la cual el valor del atributo idVenta se corresponde con el valor del parámetro de entrada idVenta.`
 	
`post 1: La instancia de Venta cuyo atributo idVenta se correspondía con el indicado por el parámetro idVenta no existe más.`
 	
`post 2: Las instancias de LineaDeVenta que tenían un link con la instancia de venta especificada en el punto anterior no existen más.`
 	
`post3: Los links que existían entre las instancias de EspecificaciónDeProducto y las líneas de Venta especificadas en el punto anterior no existen más.`	

Se pide: Realice los diagramas de comunicación para las operaciones que se especifican en los diagramas de secuencia. Los diagramas deben mostrar las visibilidades utilizadas, cumplir con los contratos y con las decisiones de diseño que ya han sido tomadas.

i) Calcular total de venta.



ii) Obtener Información de un Producto



iii) Borrar una venta