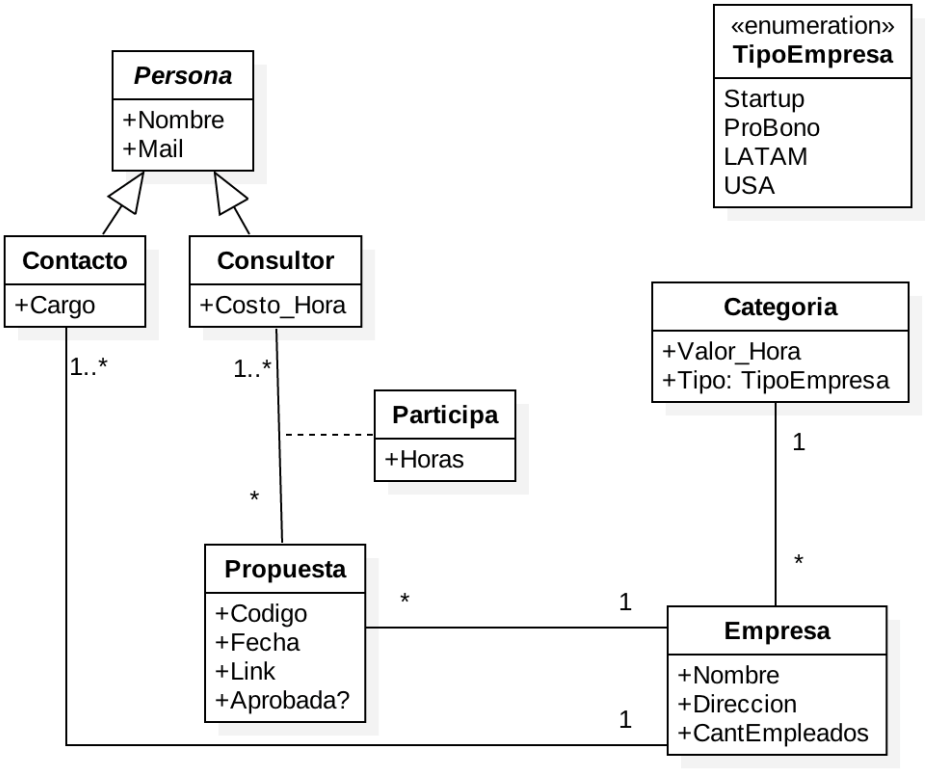
Programación 4

PARCIAL FINAL EDICIÓN 2019 - SOLUCIÓN

Problema 1

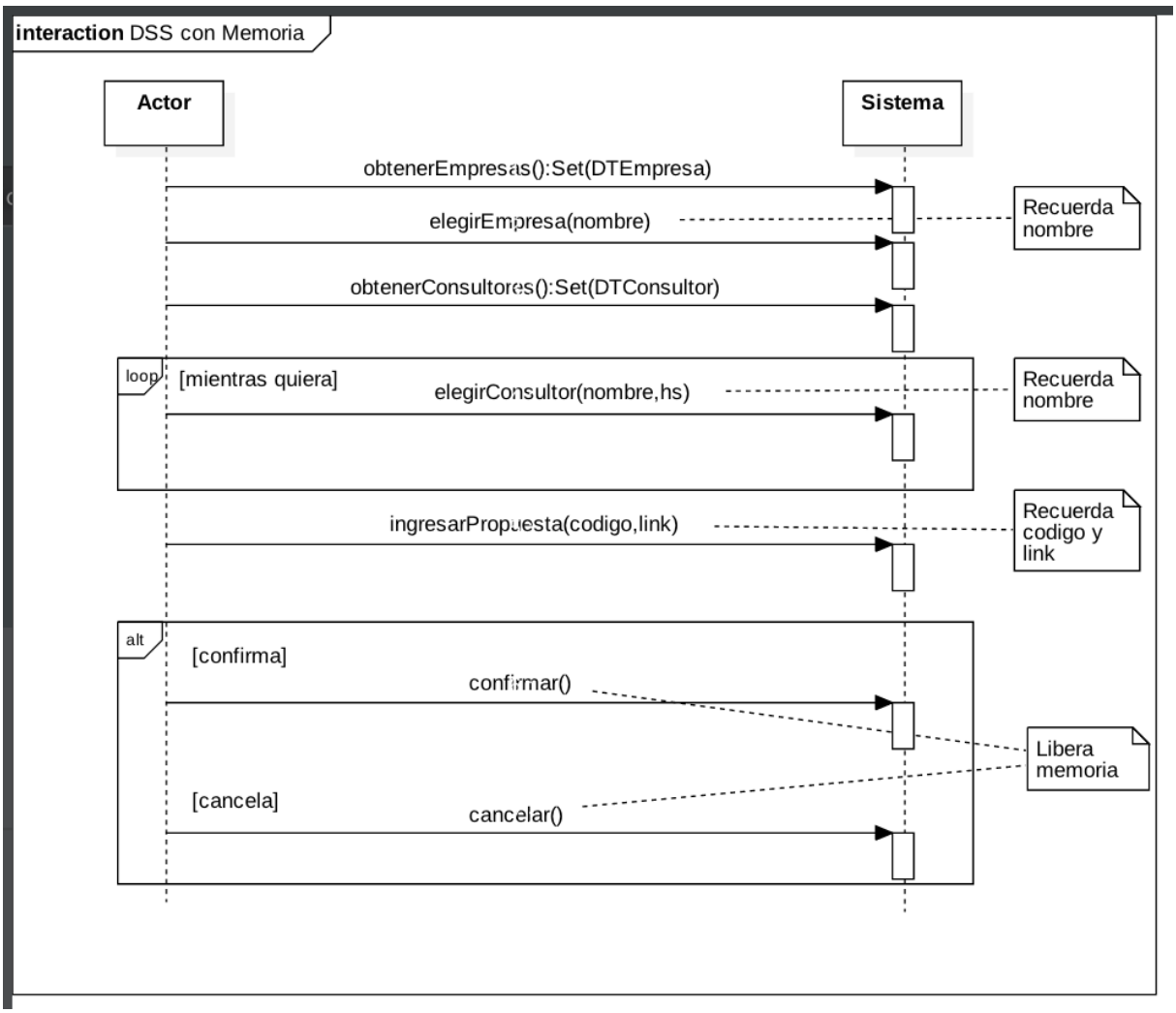
a)

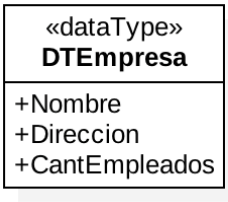


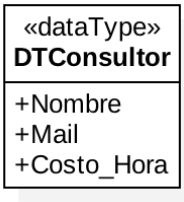
Restricciones:

- El nombre identifica a la Persona
- El código identifica a la Propuesta
- El nombre identifica a la Empresa
- El valor_hora de las empresas de tipo ProBono debe ser cero.
- CantEmpleados en Empresa es mayor o igual a la cantidad de instancias de Contacto asociadas a la misma.

b)

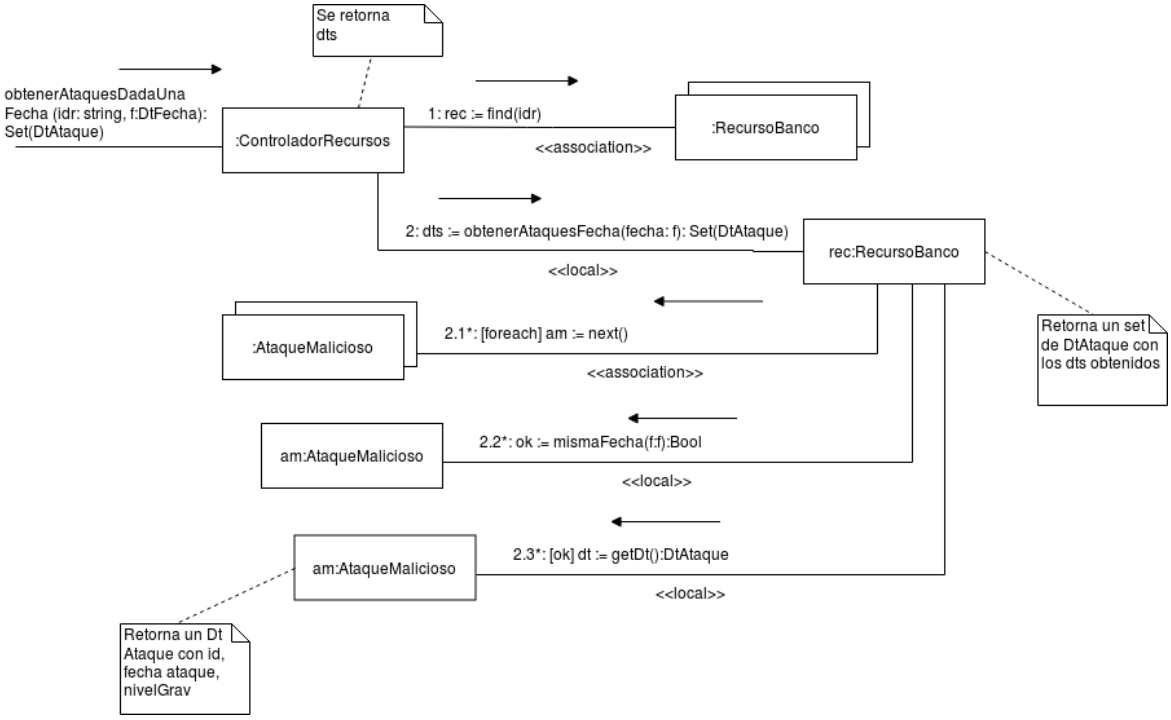

interaction DSS con Memoria
Actor
Sistema
obtenerEmpresas():Set(DTEmpresa)
elegirEmpresa(nombre)
Recuerda nombre
obtenerConsultores():Set(DTConsultor)
loop
[mientras quiera]
elegirConsultor(nombre,hs)
Recuerda nombre
ingresarPropuesta(codigo,link)
Recuerda codigo y link
alt
[confirma]
confirmar()
Libera memoria
[cancela]
cancelar()

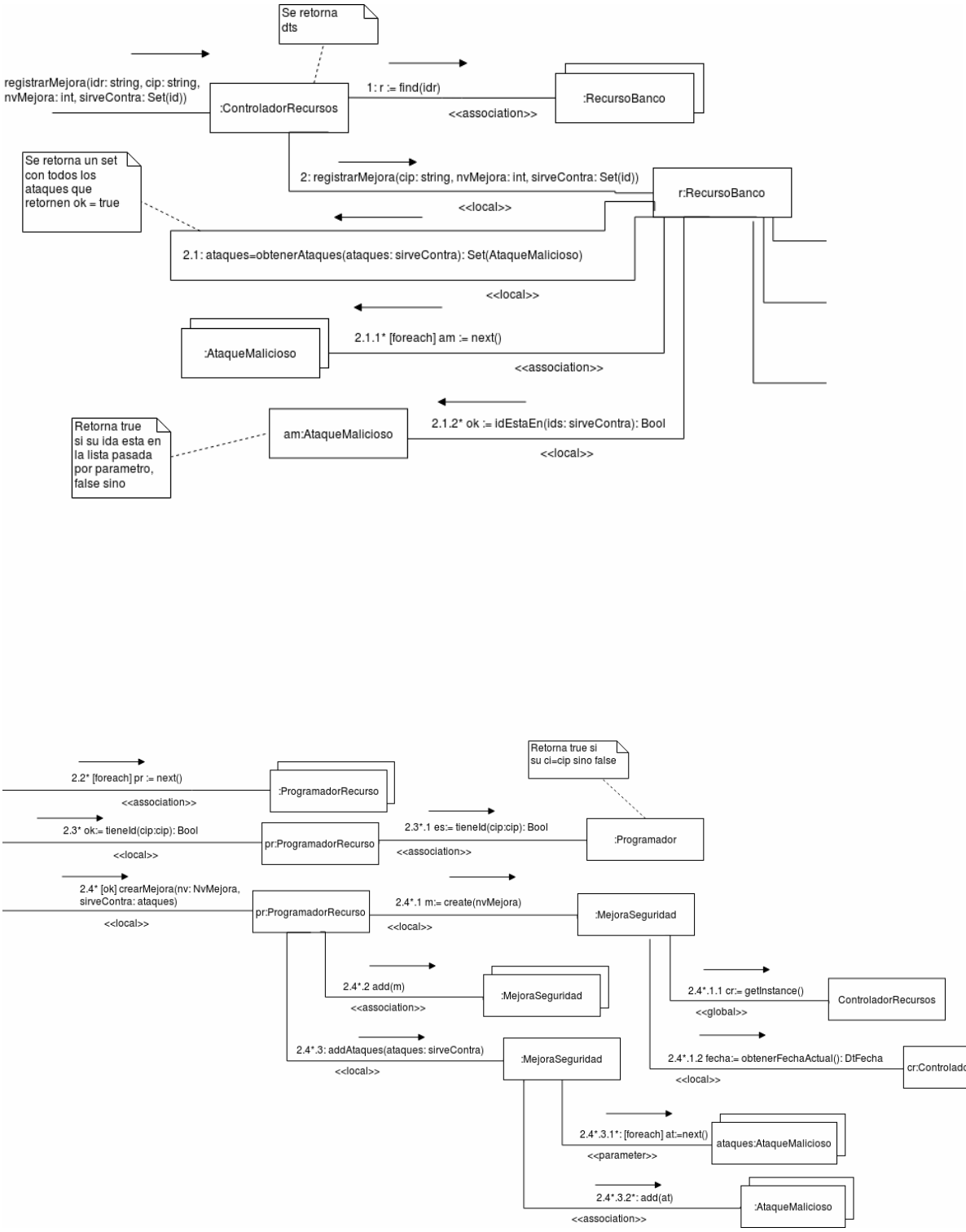

«dataType»
DTEmpresa
+Nombre
+Direccion
+CantEmpleados


«dataType»
DTConsultor
+Nombre
+Mail
+Costo_Hora

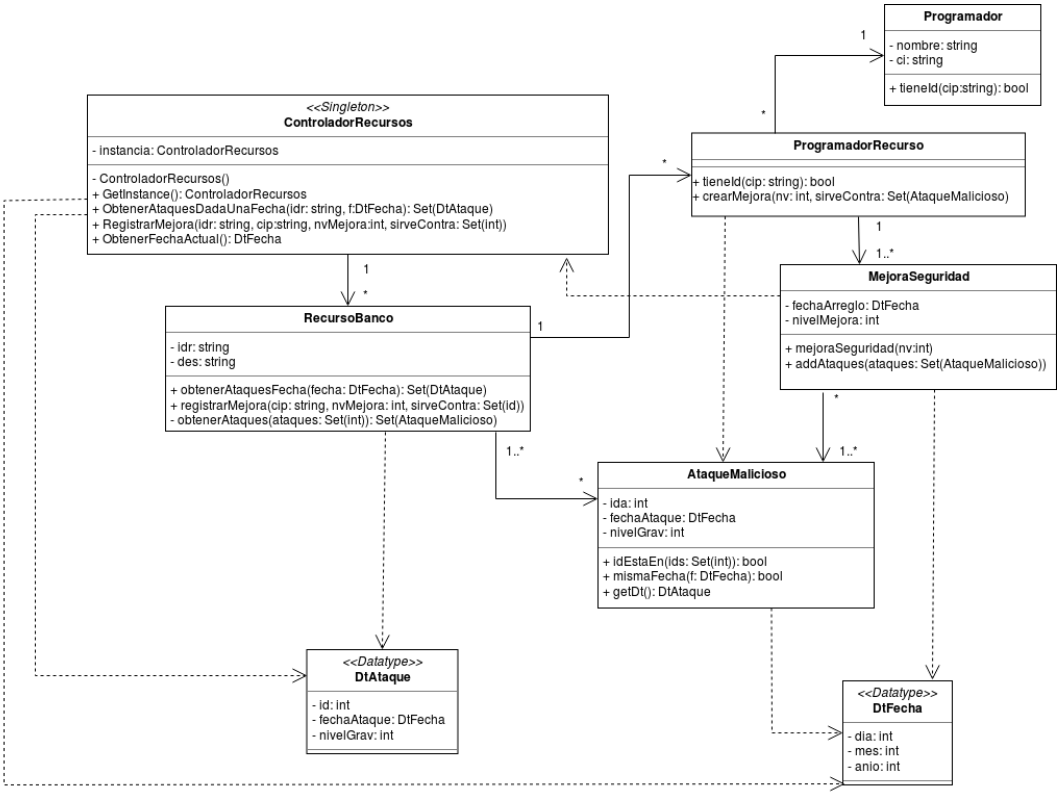
Problema 2

a)


Se retorna dts
obtenerAtaquesDadaUna Fecha (idr: string, f:DtFecha): Set(DtAtaque)
:ControladorRecursos
1: rec := find(idr)
<<association>>
:RecursoBanco
2: dts := obtenerAtaquesFecha(fecha: f): Set(DtAtaque)
<<local>>
rec:RecursoBanco
:AtaqueMalicioso
2.1*: [foreach] am := next()
<<association>>
Retorna un set de DtAtaque con los dts obtenidos
am:AtaqueMalicioso
2.2*: ok := mismaFecha(f:f):Bool
<<local>>
am:AtaqueMalicioso
2.3*: [ok] dt := getDt():DtAtaque
<<local>>
Retorna un Dt Ataque con id, fecha ataque, nivelGrav


Se retorna dts
registrarMejora(idr: string, cip: string, nvMejora: int, sirveContra: Set(id))
:ControladorRecursos
1: r := find(idr)
<<association>>
:RecursoBanco
Se retorna un set con todos los ataques que retornen ok = true
2: registrarMejora(cip: string, nvMejora: int, sirveContra: Set(id))
<<local>>
r:RecursoBanco
2.1: ataques=obtenerAtaques(ataques: sirveContra): Set(AtaqueMalicioso)
<<local>>
:AtaqueMalicioso
2.1.1* [foreach] am := next()
<<association>>
Retorna true si su ida esta en la lista pasada por parametro, false sino
am:AtaqueMalicioso
2.1.2* ok := idEstaEn(ids: sirveContra): Bool
<<local>>
Retorna true si su ci=cip sino false
2.2* [foreach] pr := next()
<<association>>
:ProgramadorRecurso
2.3* ok:= tieneId(cip:cip): Bool
<<local>>
pr:ProgramadorRecurso
2.3*.1 es:= tieneId(cip:cip): Bool
<<association>>
:Programador
2.4* [ok] crearMejora(nv: NvMejora, sirveContra: ataques)
<<local>>
pr:ProgramadorRecurso
2.4*.1 m:= create(nvMejora)
<<local>>
:MejoraSeguridad
2.4*.2 add(m)
<<association>>
:MejoraSeguridad
2.4*.1.1 cr:= getInstance()
<<global>>
ControladorRecursos
2.4*.3: addAtaques(ataques: sirveContra)
<<local>>
:MejoraSeguridad
2.4*.1.2 fecha:= obtenerFechaActual(): DtFecha
<<local>>
cr:ControladorRecursos
2.4*.3.1*: [foreach] at:=next()
<<parameter>>
ataques:AtaqueMalicioso
2.4*.3.2*: add(at)
<<association>>
:AtaqueMalicioso

b)


Programador
- nombre: string
- ci: string
+ tieneId(cip:string): bool
1
*
<<Singleton>>
ControladorRecursos
- instancia: ControladorRecursos
- ControladorRecursos()
+ GetInstance(): ControladorRecursos
+ ObtenerAtaquesDadaUnaFecha(idr: string, f:DtFecha): Set(DtAtaque)
+ RegistrarMejora(idr: string, cip:string, nvMejora:int, sirveContra: Set(int))
+ ObtenerFechaActual(): DtFecha
ProgramadorRecurso
+ tieneId(cip: string): bool
+ crearMejora(nv: int, sirveContra: Set(AtaqueMalicioso)
1
1..*
MejoraSeguridad
- fechaArreglo: DtFecha
- nivelMejora: int
+ mejoraSeguridad(nv:int)
+ addAtaques(ataques: Set(AtaqueMalicioso))
1
*
RecursoBanco
- idr: string
- des: string
+ obtenerAtaquesFecha(fecha: DtFecha): Set(DtAtaque)
+ registrarMejora(cip: string, nvMejora: int, sirveContra: Set(id))
- obtenerAtaques(ataques: Set(int)): Set(AtaqueMalicioso)
1..*
AtaqueMalicioso
- ida: int
- fechaAtaque: DtFecha
- nivelGrav: int
+ idEstaEn(ids: Set(int)): bool
+ mismaFecha(f: DtFecha): bool
+ getDt(): DtAtaque
<<Datatype>>
DtAtaque
- id: int
- fechaAtaque: DtFecha
- nivelGrav: int
<<Datatype>>
DtFecha
- dia: int
- mes: int
- anio: int

Problema 3

ContrDocs.hpp

```
class ContrDocs {
      public:
            static ContrDocs *getInstance();
            int nuevoTexto(set<DataPreamb *>);
            int nuevaPlanilla(set<DataPreamb *>);
            int copiarDoc(int);
            void eliminarDoc(int);
      private:
            ContrDocs();
            set<string> generarPreamb(set<DataPreamb *>);
            static ContrDocs *instance;
            static int darNuevoId();
            static int id;
            map<int, Documento *> docs;

}
```

ContrDocs.cpp

```
ContrDocs::instance = NULL;
ContrDocs::id= 0 ;

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

set<string> ContrDocs::generarPreamb(set<DataPreamb *cb>) {
      set<string> result;
      set<DataPreamb *>::iterator it;
      for (it=cb.begin(); it!=cb.end(); ++it)
            result.insert((*it)->toString());
}

int ContrDocs::darNuevoId() {
      ContrDocs::id++;
      return ContrDocs::id;
}

int ContrDocs::nuevoTexto(set<DataPreamb *cb>) {
      int nuevoId = ContrDocs::darNuevoId();
      docs[nuevoId] = new Texto(nuevoId, generarPreamb(cb));
}

int ContrDocs::nuevaPlanilla(set<DataPreamb *cb>) {
      int nuevoId = ContrDocs::darNuevoId();
      docs[nuevoId] = new Planilla(nuevoId, generarPreamb(cb));
}

int ContrDocs::copiarDoc(int idDoc) {
      int nuevoId = ContrDocs::darNuevoId();
      Documento *nuevoDoc = docs[idDoc]->copiar();
      nuevoDoc->setId(nuevoId);
      docs[nuevoId] = nuevoDoc;
}
```

```
void ContrDocs::eliminarDoc(int idDoc) {
	Documento *doc = docs[idDoc];
	docs.erase(idDoc);
	delete doc;
}

ContrDocs::ContrDocs(){
}
```

Documento.hpp

```
class Documento {
	private:
		int idDoc;
		set<string> contBase;
	public:
		set<string> getContBase();
		int getId();
		void setId(int);
		virtual ~Documento();
		virtual Documento *copiar() = 0;
	protected:
		Documento(int, set<string>);
}
```

Documento.cpp

```
set<string> Documento::getContBase() {
	return contBase;
}

int Documento::getId() {
	return idDoc;
}

void Documento::setId(int idDoc) {
	this->idDoc = idDoc;
}

Documento::~Documento(){
}

Documento::Documento(int idDoc, set<string> contBase){
	this->idDoc = idDoc;
	this->contBase = contBase;
}
```

Texto.hpp

```
class Texto: public Documento {
	private:
		set<Pagina*> pags;
	public:
		Texto(int, set<string>);
		virtual ~Texto();
		Texto *copiar();
}
```

```
Texto.cpp
---------

Texto::Texto(int    idDoc,    set<string>    contBase):Documento(idDoc,
contbase) {
      pags.insert(new Pagina);
}

Texto::~Texto(){
      for(set<Pagina*>::iterator it=pags.begin();it!=pags.end();++it)
            delete (*it);
}

Texto *Texto::copiar() {
      Texto *result = new Texto(this->getId(), this->getContBase());
      for(set<Pagina*>::iterator it=pags.begin();it!=pags.end(); ++it)
            result.insert(new Pagina(*it));
      return result;
}

Planilla.hpp
------------

class Planilla: public Documento {
      private:
            set<Hoja *> hojas;
            set<Macro *> macros;
      public:
            Planilla(int, set<string>);
            virtual ~Planilla();
}

Planilla.cpp
------------

Planilla::Planilla(int     idDoc,     set<string>     contBase)     :
Documento(idDoc,contbase) {
      hojas.insert(new Hoja);
}

Planilla::~Planilla(){
      for(set<Hoja *>::iterator it=hojas.begin();it!=hojas.end();++it)
            delete(*it);

      for(set<Macro*>::iteratorit=macros.begin();it!=macros.end();++it
)
            delete(*it);
}

Planilla *Planilla::copiar() {
      Planilla    *result    =    new    Planilla(this->getId(),    this-
>getContBase());
      for(set<Hoja *>::iterator it=hojas.begin();it!=hojas.end();++it)
            result.insert(new Hoja(*it));
      for(set<Macro   *>::iterator   it=macros.begin();it!=macros.end();
++it)
            result.insert(new Macro(*it));
      return result;
}
```