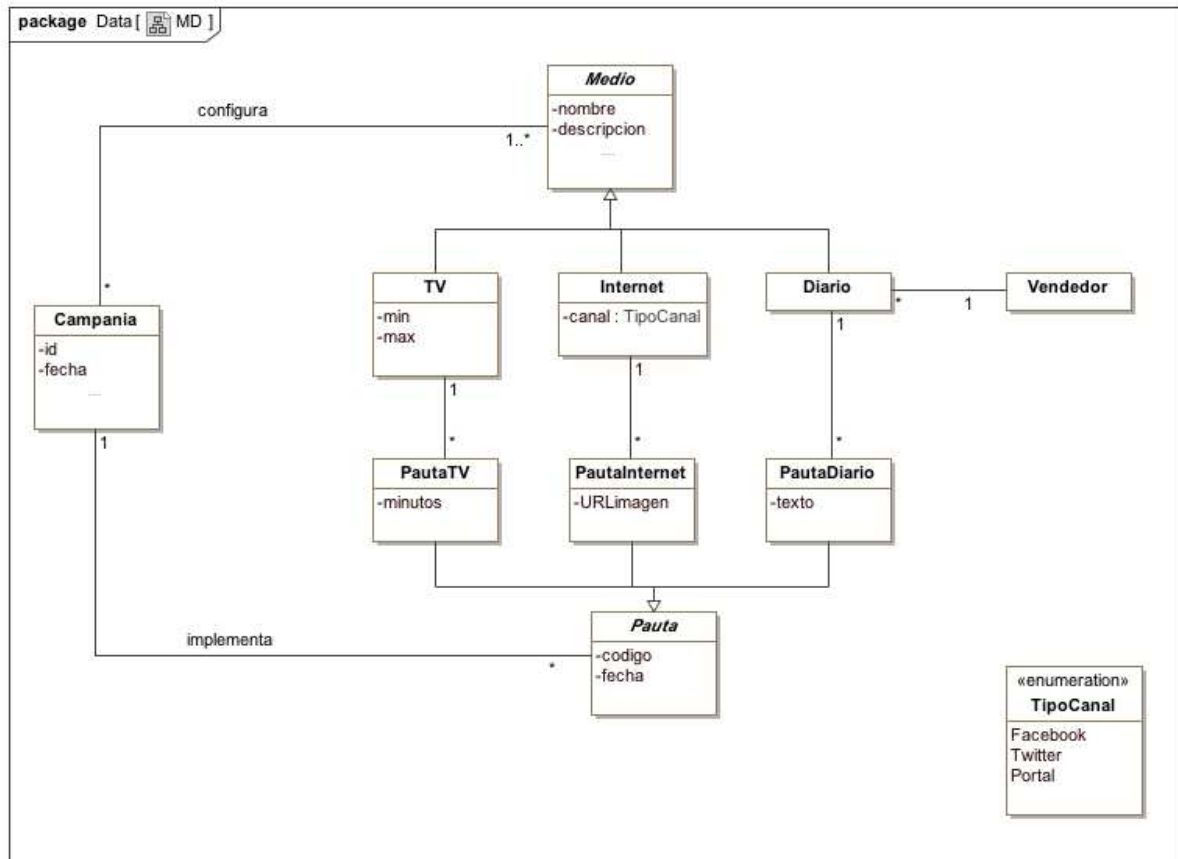


Programación 4

PARCIAL FINAL EDICIÓN 2014 - SOLUCIÓN

Problema 1

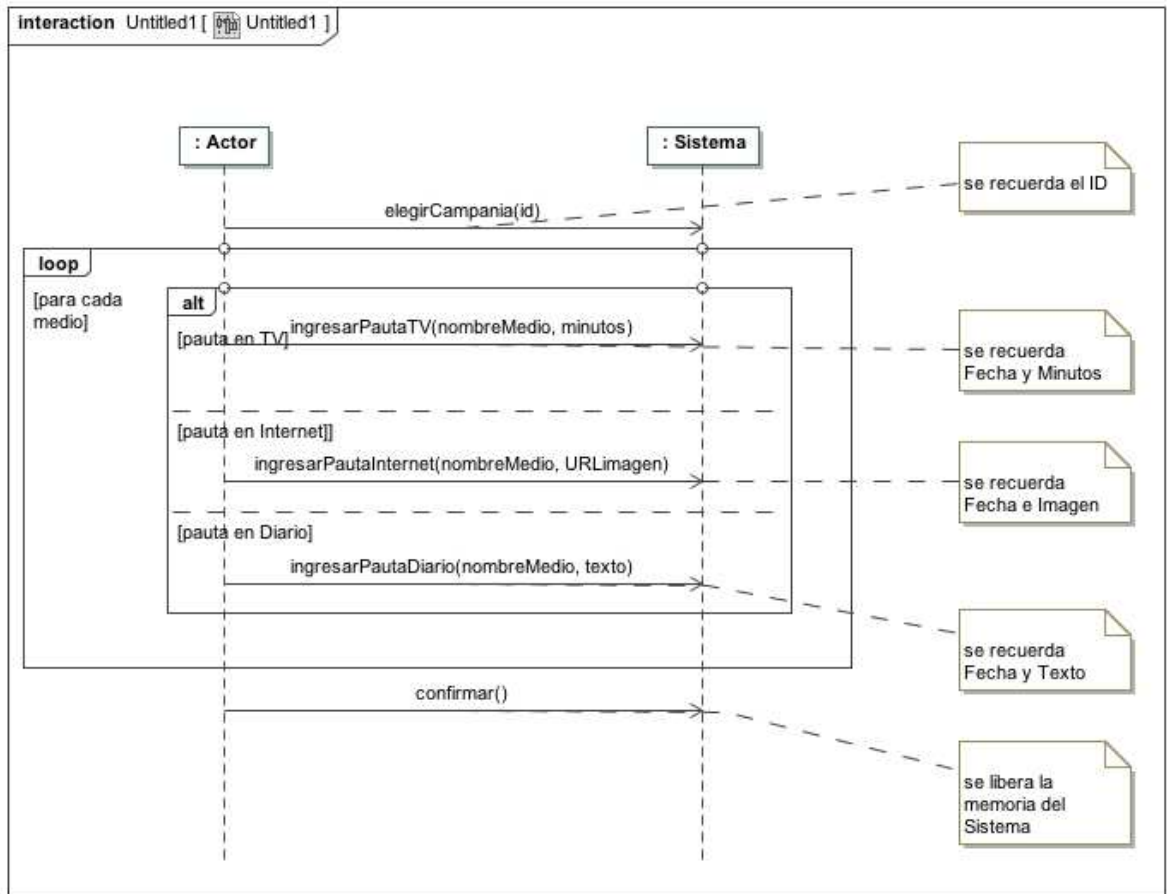
i.



Restricciones:

- [unicidad] El ID de la campaña la identifica.
- [unicidad] El nombre del medio lo identifica.
- [unicidad] El código de la pauta la identifica.
- [regla de negocio] Los minutos de una PautaTV estan entre el mínimo y máximo definido en su medio correspondiente.
- [circular] El medio referenciado por una pauta debe haber sido definido/configurado en la misma campaña a la que la pauta pertenece.

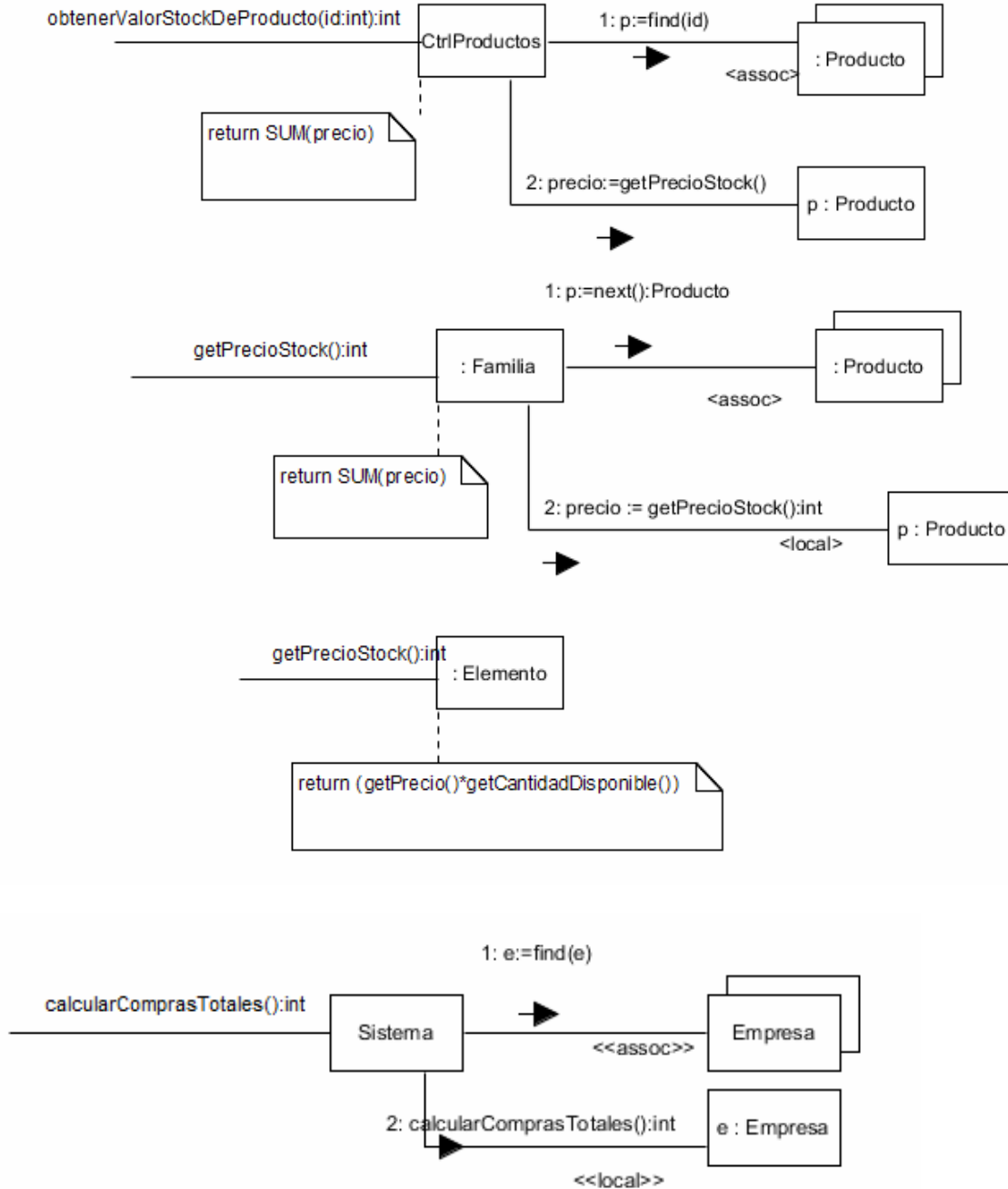
ii.

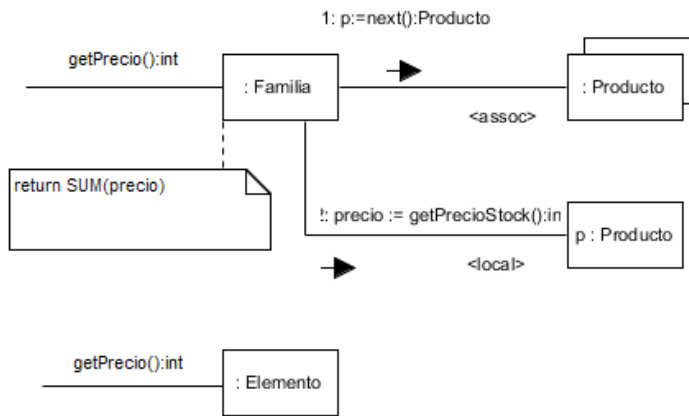
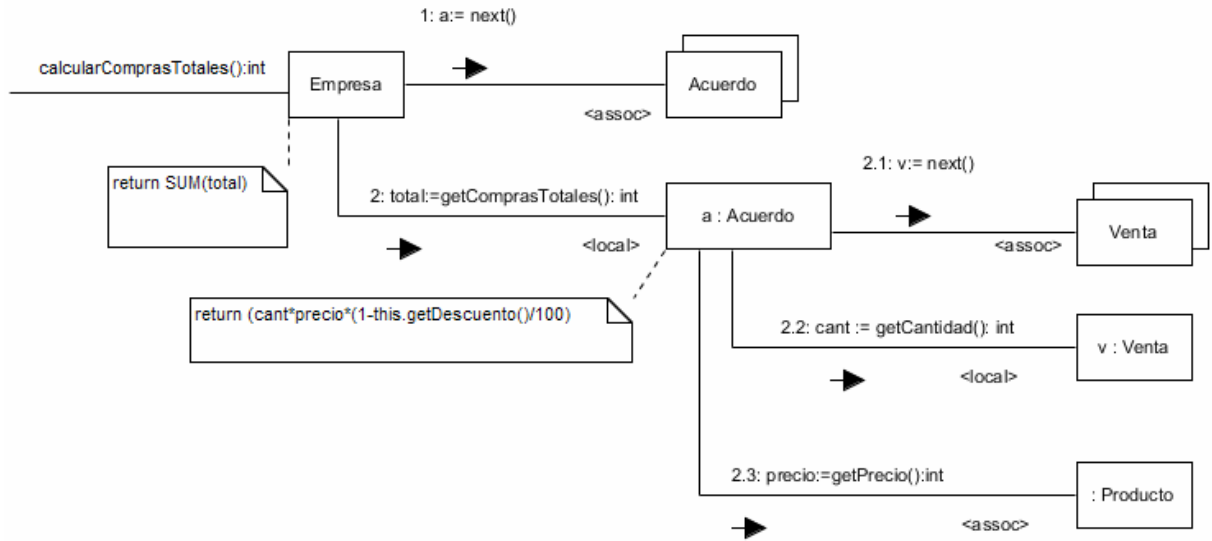


Problema 2

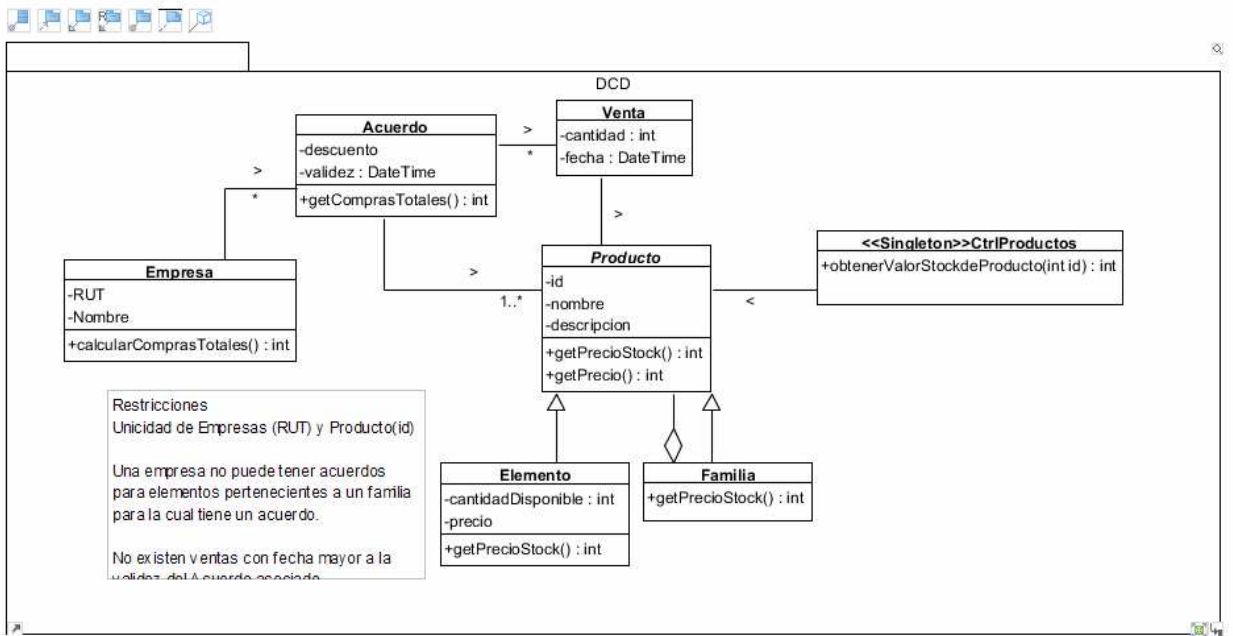
Parte A:

i.



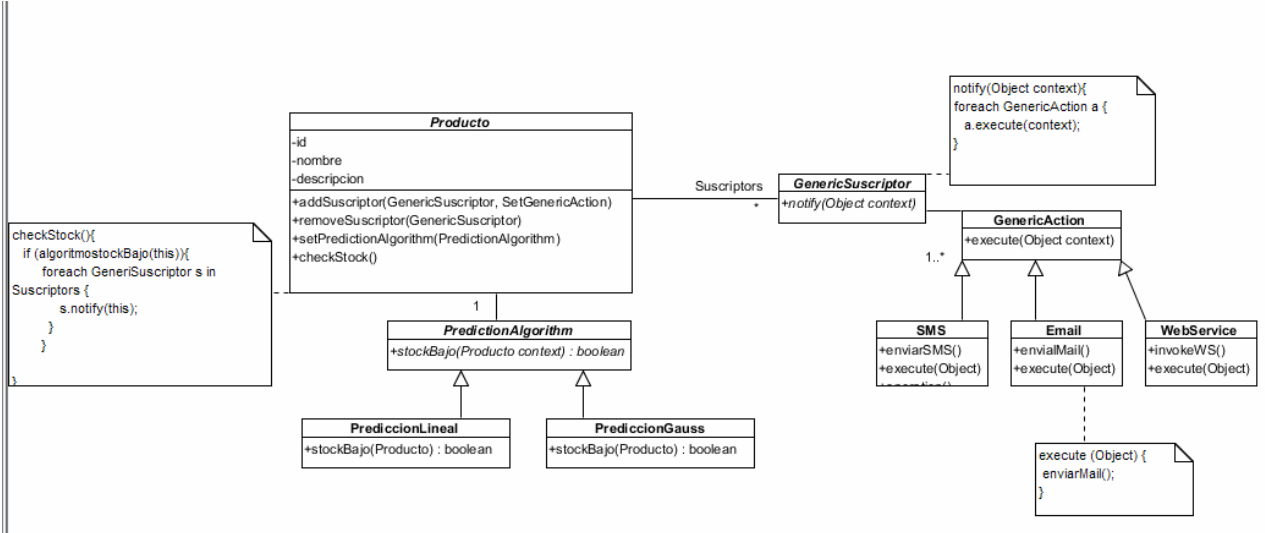


ii.



ParteB:

i.



Problema 3*Parte A:*

```

class A {
private:
    int dato;
    static A *instance;
    A();
public:
    static A *getInstance();
    int getDato();
    void setDato(int);
}

A *A::instance = NULL;

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

int A::getDato() {
    return dato;
}

void A::setDato(int d) {
    dato = d;
}

```

Parte B:

```

class Paquete : public ICollectible {
public:
    virtual double calcularPeso() = 0;
    virtual double calcularVolumen() = 0;
    virtual ~Paquete();
}

Paquete::~~Paquete() {
}

class Sencillo : public Paquete {
private:
    double peso, volumen;
public:
    Sencillo(double, double);
    double getPeso();
    void setPeso(double);
    double getVolumen();
    void setVolumen(double);
    double calcularPeso();
    double calcularVolumen();
}

Sencillo::Sencillo(double p, double v) {
    peso = p;
    volumen = v;
}

```

```

double Sencillo::getPeso() {
    return peso;
}

void Sencillo::setPeso(double p) {
    peso = p;
}

double Sencillo::getVolumen() {
    return volumen;
}

void Sencillo::setVolumen(double v) {
    volumen = v;
}

double Sencillo::calcularPeso() {
    return peso;
}

double Sencillo::calcularVolumen() {
    return volumen;
}

class Complejo : public Paquete
private:
    ICollection *componentes;
    OptimizadorVolumen *optimizador;
public:
{
    Complejo(ICollection *, OptimizadorVolumen *);
    ~Complejo();
    double calcularPeso();
    double calcularVolumen();
    void setOptVol(OptimizadorVolumen *);
}

Complejo::Complejo(ICollection *comps, OptimizadorVolumen *opt) {
    componentes = new List;
    IIterator *it = comps->getIterator();
    while (it->hasCurrent()) {
        componentes->add(it->getCurrent());
        it->next();
    }
    delete it;
    optimizador = opt;
}

Complejo::~~Complejo() {
    IIterator *it = componentes->getIterator();
    ICollectible *elem;
    while (it->hasCurrent()) {
        elem = it->getCurrent();
        it->next();
        componentes->remove(elem);
        delete elem;
    }
    delete it;
    delete componentes;
}

```

```
double Complejo::calcularPeso() {
    IIterator *it = componentes->getIterator();
    double result = 0;
    while (it->hasCurrent()) {
        result = result + ((Paquete *)it->getCurrent())->getPeso();
        it->next();
    }
    delete it;
    return result;
}

double Complejo::calcularVolumen() {
    return opt->volOptimo(componentes);
}

void Complejo::setOptVol(OptimizadorVolumen *opt) {
    optimizador = opt;
}

class OptimizadorVolumen {
public:
    virtual double valOptimo(ICollection *) = 0;
    virtual ~OptimizadorVolumen();
}

OptimizadorVolumen::~~OptimizadorVolumen() {
}
```