Programación avanzada

EXAMEN JULIO 2011

Solución

Problema 1

a) Indique brevemente la relación entre los siguientes elementos: Caso de Uso, Escenario, Operación del Sistema, Contrato y Diagrama de Secuencia del Sistema.

b) En una ciudad extranjera se está a punto de desarrollar un nuevo sistema de transporte colectivo similar al Sistema de Transporte Metropolitano (STM) de la ciudad de Montevideo. El nuevo sistema a desarrollar considerará líneas de transporte (compuestas por tramos) y registrará los viajes realizados sobre esos tramos y la forma en que éstos fueron abonados. Cada tramo se origina en una parada y termina en otra parada (diferente a la original) totalizando cierta cantidad de kilómetros entre ambas. Así por ejemplo la línea número 808 tendrá diez tramos: el primero de 0,8 kilómetros entre las paradas "Plaza JC" (ubicada en la esquina de las calles X y Y) y "Monumento a DC" (ubicado en la esquina de las calles A y B); el segundo… y así sucesivamente. Existirán dos medios de pago en este nuevo sistema de transporte: tickets y tarjetas inteligentes. Ambos permitirán a los pasajeros viajar durante cierto período de tiempo, pero existen restricciones diferentes para cada medio de pago. Mientras que un ticket permite utilizar su tiempo únicamente para 1 o 2 viajes, la tarjeta permite utilizar su tiempo para cualquier cantidad de viajes. En ambos casos no se podrán tomar nuevos viajes si se ha agotado el tiempo del medio de pago. Cada viaje incluirá su duración de forma de poder controlar la validez de los medios de pago.

Se pide:

i) Diagrama de dominio de la realidad planteada con restricciones en lenguaje natural.

ii) Especificar en OCL <u>únicamente</u> las dos siguientes restricciones:
- unicidad del número de línea
- la suma de las duraciones de los viajes sea menor o igual a la duración del medio de pago

iii) Diagrama de Secuencia del Sistema basándose únicamente en el siguiente caso de uso (asumiendo que existe un único escenario para el caso de uso):

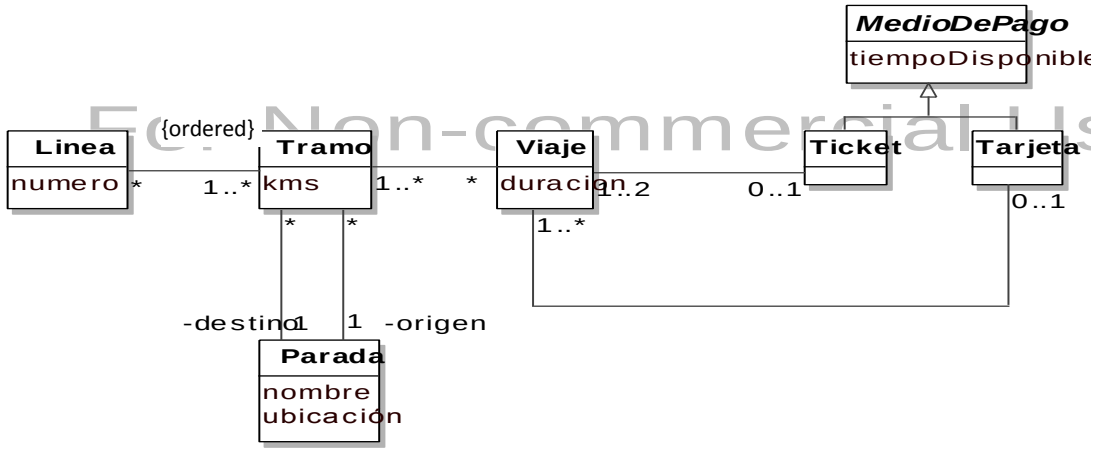
Nombre	Alta de Línea	Actores	Administrador del Sistema
Descripción	El caso de uso comienza cuando el administrador del sistema selecciona la opción de Alta de Línea. El administrador deberá primero ingresar el número de la línea, a lo cual el sistema le devolverá una lista con toda la información de las paradas de las cuales el administrador deberá elegir (mediante su nombre) una de ellas como la próxima parada. Así continuará eligiendo las paradas que conformarán los tramos de la línea hasta que decida terminar el ingreso. Por tanto cada vez que el administrador selecciona una parada, ésta será la parada de destino de un tramo y a su vez la parada de origen del tramo siguiente (si es que éste existe). La distancia (en kilómetros) entre cada par de paradas será calculada automáticamente por el sistema.		

Solución

a) Un caso de uso tiene uno o varios escenarios. Para cada escenario se realiza un DSS el cual contiene operaciones del sistema. Para cada operación del sistema se realiza un contrato.

b)

i)



Restricciones en lenguaje natural:

- El número de línea identifica a cada línea.
- Las paradas de origen y destino de un tramo son diferentes.
- La parada de destino de un tramo es la de origen del tramo siguiente.
- Todos los tramos de un viaje se corresponden con la misma línea.
- Un viaje es pagado por medio de un ticket o por medio de una tarjeta, no por ambos.
- La suma de las duraciones de los viajes realizados con un medio de pago es menor o igual al tiempo disponible de ese medio de pago.

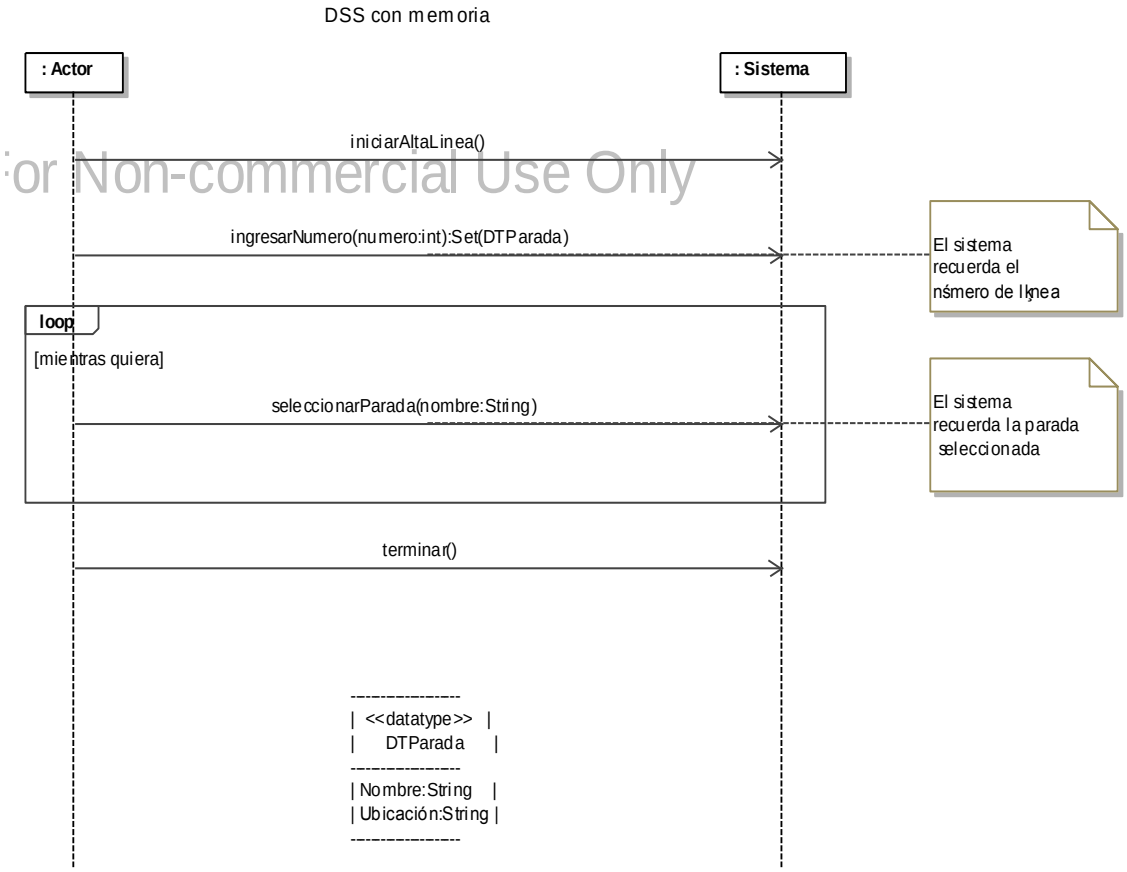
ii)

Restricciones en OCL:

- context Linea inv:
  Linea.allInstances->isUnique(numero)

- context MedioDePago inv:
  self.viaje.duracion->sum() <= self.tiempoDisponible

iii)



Problema 2

El DCD de la Figura 3a muestra el diseño de una parte de un sistema de gestión de inmuebles, mientras que el DSS de la Figura 3c muestra el comportamiento del sistema para el caso de uso Alta Inmueble.

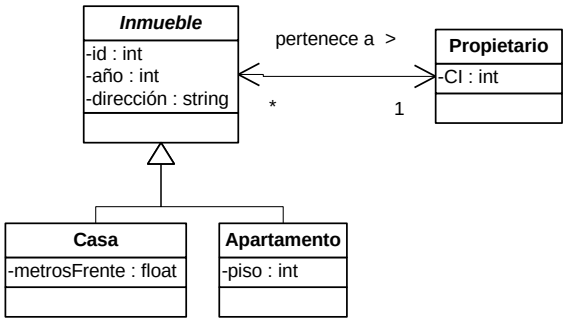


Figura 3a

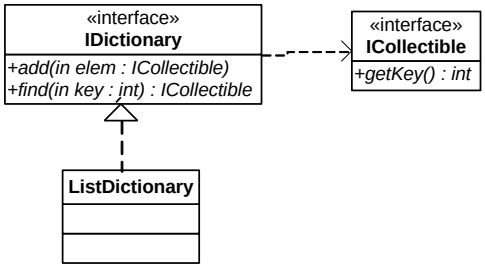


Figura 3b

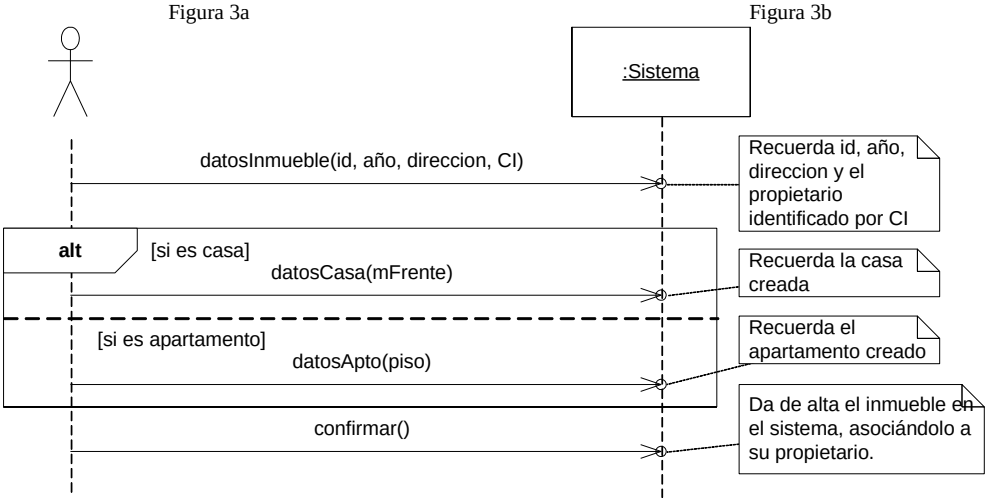


Figura 3c

PARTE A:
Se pide:

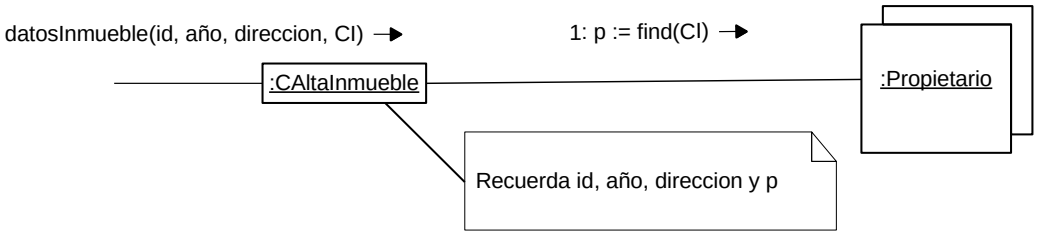
i. Realizar los diagramas de comunicación de las operaciones del sistema involucradas en el caso de uso. Asumir que se decidió crear un controlador para el caso de uso, el cual mantiene las colecciones de inmuebles y propietarios.
ii. Completar el DCD de la Figura 3a con las nuevas clases (incluido el controlador y su interfaz) y operaciones que hayan surgido en su solución a la parte Ai.

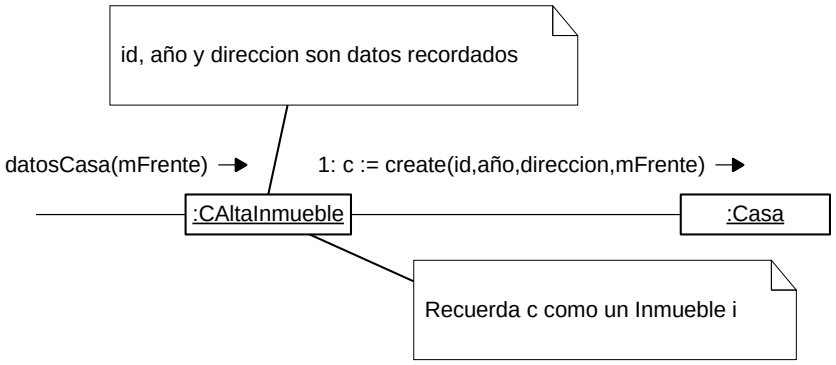
PARTE B:
Implementar en C++ los archivos de cabecera de todas las clases del DCD resultante de su solución a la parte Aii. Implementar completamente la clase que representa el controlador del caso de uso Alta Inmueble. Incluir constructor y destructor en las clases que considere necesario. Asuma que dispone de implementaciones de las interfaces y clases indicadas en la Figura 3b. No es necesario incluir directivas al preprocesador en el código.

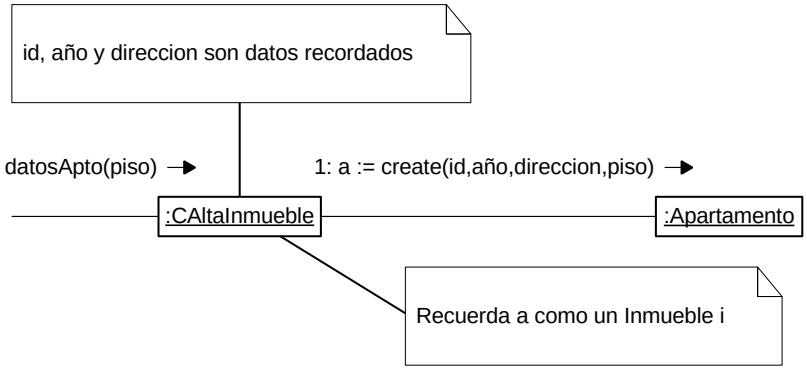
Solución

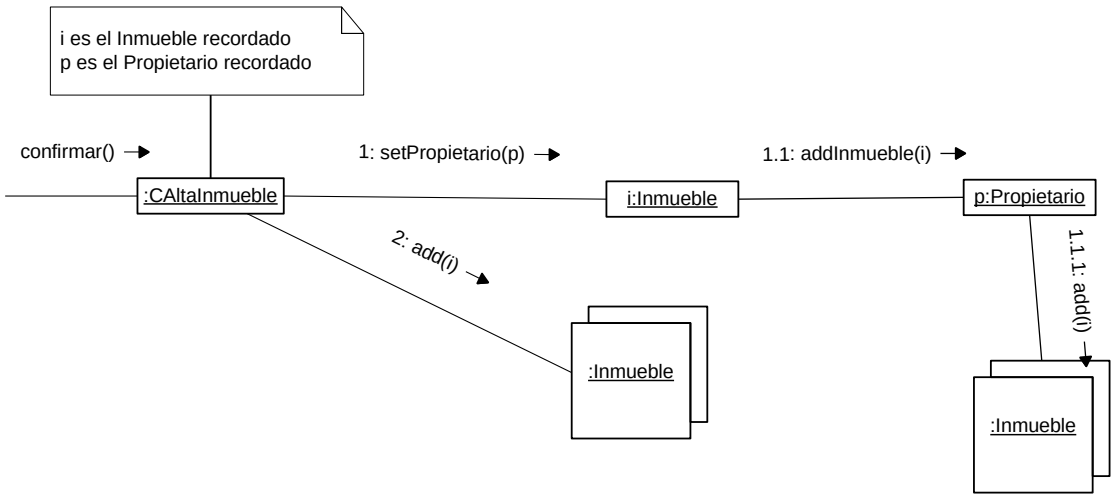
PARTE A:

i.

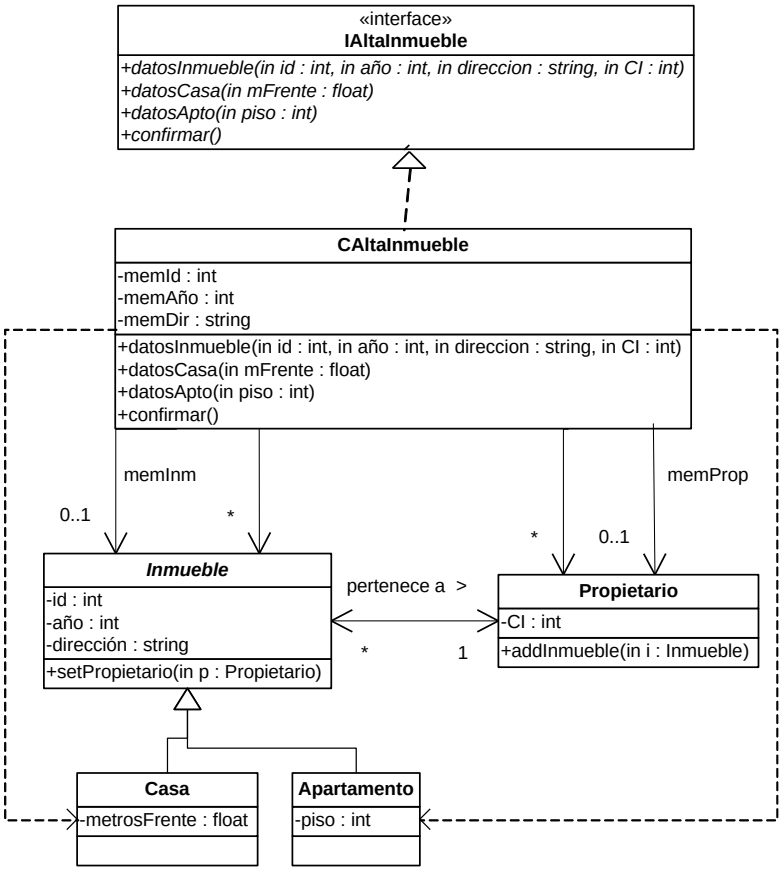








ii.



PARTE B:

```
class IAltaInmueble {
   public:
        virtual void datosInmueble(int id, int anio, String direccion, int CI) = 0;
        virtual void datosCasa(float mFrente) = 0;
        virtual void datosApto(int piso) = 0;
        virtual void confirmar() = 0;
        virtual ~IAltaInmueble();
}
```

```
class CAltaInmueble : public IAltaInmueble {
   private:
        IDictionary *inmuebles, *propietarios;

        int memId, memAnio;
        String memDir;
        Inmueble *memInm;
        Propietario *memProp;

        CAltaInmueble();
        static CAltaInmueble *instancia;
   public:
        void datosInmueble(int id, int anio, String direccion, int CI);
        void datosCasa(float mFrente);
        void datosApto(int piso);
        void confirmar();
        static CAltaInmueble *getInstancia();
}

CAltaInmueble *CAltaInmueble::instancia = NULL;

CAltaInmueble *CAltaInmueble::getInstancia() {
```

```
   if (instancia == NULL)
        instancia = new CAltaInmueble;
   return instancia;
}

CAltaInmueble::CAltaInmueble() {
   inmuebles = new ListDictionary();
   propietarios = new ListDictionary();
}

CAltaInmueble::datosInmueble(int id, int anio, String direccion, int CI) {
   memId = id;
   memAnio = anio;
   memDir = direccion;
   memProp = propietarios->find(CI);
}

CAltaInmueble::datosCasa(float mFrente) {
   memInm = new Casa(memId, memAnio, memDir, mFrente);
}

CAltaInmueble::datosApto(int piso) {
   memInm = new Apartamento(memId, memAnio, memDir, piso);
}

CAltaInmueble::confirmar() {
   memInm->setProp(memProp);
   inmuebles->add(memInm);
}

class Inmueble : public ICollectible {
   private:
        int id, anio;
        String direccion;
        Propietario *prop;
   public:
        Inmueble(int id, int anio, String direccion);
        void setPropietario(Propietario *prop);
        int getKey();
        virtual ~Inmueble();
}

class Casa : public Inmueble {
   private:
        float mFrente;
   public:
        Casa(int id, int anio, String direccion, float mFrente);
}

class Apartamento : public Inmueble {
   private:
        int piso;
   public:
        Apartamento(int id, int anio, String direccion, int piso);
}

class Propietario : public ICollectible {
   private:
        int CI;
        IDictionary *inmuebles;
   public:
        Propietario(int CI);
        int getKey();
        void addInmueble(Inmueble *i);
}
```