Programación Avanzada

SEGUNDO PARCIAL – 4/12/2008

_____ _____

Nombre y Apellido c.i.

Por favor siga las siguientes indicaciones:

- Escriba con lápiz.
- 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.
- Escriba las hojas de un solo lado.
- Comience cada ejercicio en una hoja nueva.
- El total máximo de puntos del parcial es **100**.
- El parcial contiene un total de: **4** páginas.

Mucha Suerte y Felices Fiestas!!

Problema 1 (Total: 40 puntos) [Teórico]

Responda brevemente cada una de las siguientes preguntas:

a) Nombrar 3 (tres) criterios GRASP y explicar brevemente en qué consiste cada uno.
b) ¿Qué 2 (dos) enfoques se citaron en el curso para diseñar una colaboración?
c) ¿Qué tipos de de visibilidad pueden existir entre objetos, en el contexto de un diagrama de comunicación, y cuándo se dan estas visibilidades?
d) Describir las responsabilidades que pueden ser asignadas a una colección de objetos en un diagrama de comunicación.
e) ¿Qué es un controlador?
f) ¿Qué relación existe entre controladores e interfaces del sistema?
g) ¿Qué elementos están presentes en los diagramas de clases de diseño y no en los diagramas de modelo de dominio?
h) Defina brevemente como se mapean a C++ las relaciones de generalización, realización y dependencia.
i) Explique brevemente que es una colección genérica y una colección concreta (o tipada).

Problema 2 (Total: 40 puntos) [Práctico]

EL DCD de la *Figura 2a* muestra el diseño de una parte de un sistema de gestión de inmuebles, mientras que el DSS de la *Figura 2c* muestra el comportamiento del sistema para el caso de uso *Alta Inmueble*.

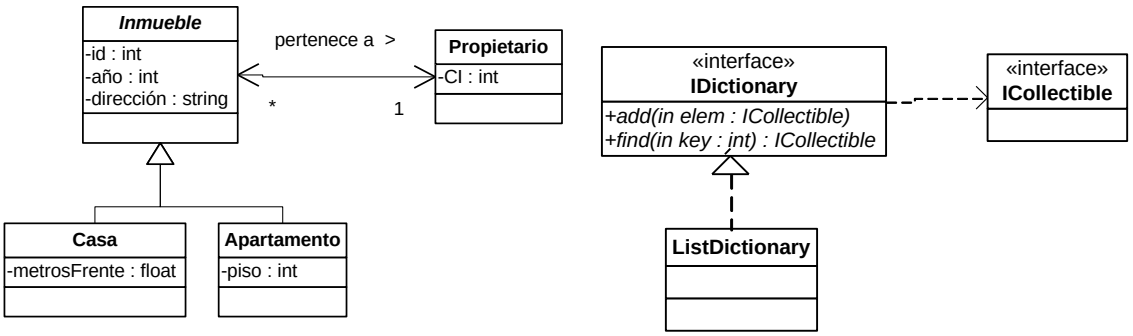

Figura 2a

Figura 2b

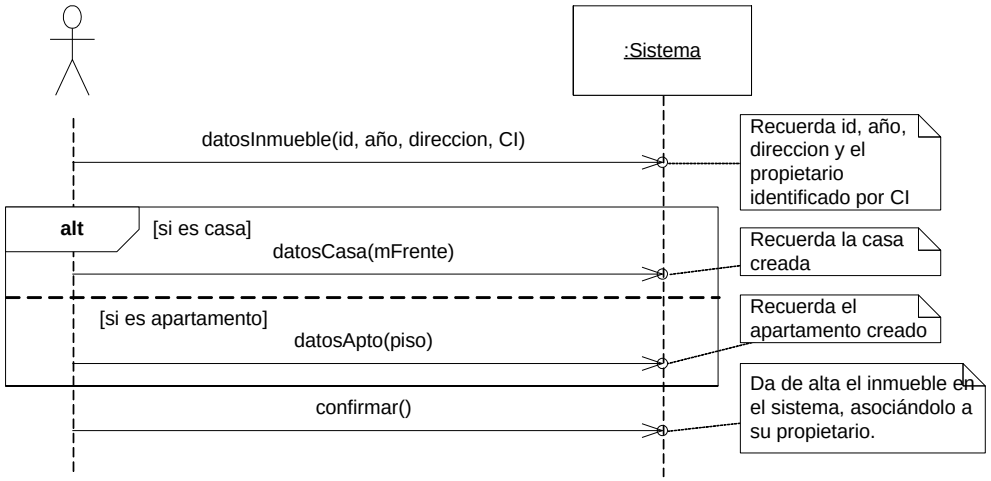

Figura 2c

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 2a* 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 se dispone de una implementación de las interfaces y clases indicadas en la *Figura 2b*. No es necesario incluir directivas al preprocesador en el código.

Problema 3 (Total: 20 puntos) [Práctico]

PARTE **A**:
Considerar la siguiente definición de clase en C++, suponiendo que todas las operaciones están correctamente implementadas.

```
class Clase {
   private:
      int entero;
      float real;
   public:
      Clase();
      Clase(int,float);
      Clase(Clase &);
      ~Clase();
      Clase operator= (Clase);
      Clase operator+ (Clase);
      void desplegar();
}
```

Considerar también el siguiente procedimiento `main()`.

```
void main(){
   Clase c1;
   Clase c2(3,2.54);
   Clase c3=c2;
   c1=c2+c3;
   c1.desplegar();
}
```

Suponer que se ejecuta el programa. Hacer una lista enumerando ordenadamente cada operación invocada a raíz de dicha ejecución, justificando.
Recordar que una línea de código puede causar que se invoque más de una operación.

PARTE **B**:

```
class claseA {
   public:
      void func1(int a);
      virtual void func2(short a);
      virtual void func3(int a);
};

class claseB : public claseA {
   public:
      virtual void func1(int a);
      virtual void func2(int a);
      void func3(int a);

};
```

```
class claseC : public claseB {
   public:
      virtual void func1(int a);
      virtual void func2(short a);
      void func3(int a);
};

class claseD : public claseA {
   public:
      void func1(int a);
      virtual void func2(short a);
};
```

Considerando la definicion de clases anterior, indicar a qué clase corresponde el metodo invocado en cada invocación:

```
claseA *a;
claseB *b;
claseC *c;
claseD *d;
claseA *ap;
claseB *bp;
int in = 0;
short sh = 4;

a = new claseA();
b = new claseB();
c = new claseC();
d = new claseD();

ap = b;
bp = c;

ap->func1(in);
```

```
ap->func3(in);
bp->func3(in);
bp->func1(in);
bp->func2(in);

ap = c;
ap->func2(in);
ap->func2(sh);

ap = b;
ap->func2(sh);

ap = d;
ap->func1(in);
a->func1(in);

bp = b;
bp->func2(sh);
```