Universidad de Valladolid

Departamento de Informática

Programación III. I.T. Informática de Sistemas

Ejercicios de aplicación de patrones de diseño

Curso 2008–09

Ejercicio 1

El Servicio de Emergencias Sanitarias de Castilla y León, antes conocido como 061, está organizado en una jerarquía de bases. Algunas de estas bases realmente disponen de personal y ambulancias (por ejemplo la de Medina del Campo), mientras que otras son «unidades administrativas» que agrupan un conjunto de bases (por ejemplo Valladolid, que agrupa la unidad del Hospital Clínico Universitario y la de Medina del Campo entre otras).

Para cada base, interesa modelar los siguientes datos:

- Nombre de la base.
- Número de ambulancias, que en el caso de las bases compuestas es la suma de las ambulancias disponibles en las bases que las componen en ese momento.
- Tiempo medio de asistencia, que en el caso de las bases compuestas es la media de los tiempos medios de asistencia registrados en las bases que las componen en ese momento.

Elaborar un diseño que permita modelar adecuadamente esta situación, implentando completamente en Eiffel las clases que modelan las bases, sean del tipo que sean.

Nota: Este ejercicio fue propuesto en el examen extraordinario del curso 2004/2005.

Ejercicio 2

Disponemos de una clase *BATERIA* que modela el funcionamiento de la batería de un ordenador portátil, cuya forma corta es la siguiente:

```
feature(s) from BATERIA
  -- Características que monitorizan el estado
  conectado: BOOLEAN
    -- ¿La batería está conectada?
  cargando: BOOLEAN
    -- ¿La batería está cargando?
  carga: INTEGER
    -- En tanto por ciento
  tiempo: INTEGER
    -- Minutos restantes estimados
  notifica
    -- Método sin código que es invocado por los métodos
    -- privados cada vez que éstos cambian el estado del objeto
end of BATERIA
```

Ahora se desea elaborar dos observadores capaces de informar sobre el estado de la batería, indicando el tiempo restante o el tanto por ciento de carga respectivamente.

Elaborar un diseño adecuado a estos propósitos (se recomienda encarecidamente que este diseño esté basado en el patrón observador) Implementar completamente la clase que juega el papel del sujeto concreto en ese diseño.

Nota: Este ejercicio fue propuesto en el examen ordinario del curso 2004/2005.

UVa **Universidad** de **Valladolid.** Dpto. de Informática. Edificio de Tecnologías de la Información y las Telecomunicaciones, Campus Miguel Delibes. 47011 Valladolid. Tfno. 983.423000 ext.5617/Fax 983.423671.
www.infor.uva.es/~felix e-mail:felix@infor.uva.es

Ejercicio 3

En una aplicación de venta y configuración de ordenadores personales se considera un ordenador como la suma de varios componentes (una unidad central y varios elementos periféricos). El mínimo imprescindible para que se considere un ordenador es la unidad central, un dispositivo de entrada y otro de salida, pero pueden añadirse todos los que deseemos ofertar o que nos pidan los clientes. Para crearlo habrá que dar esos componentes mínimos y se puede modificar la configuración en cualquier momento añadiendo, quitando o cambiando exclusivamente periféricos. Por último, el precio de venta del ordenador es la suma de sus componentes.

Todos componentes tienen información sobre el nombre del fabricante, el modelo y el precio de venta, que cambia con frecuencia.

Los dispositivos de entrada que manejamos actualmente son el teclado, el ratón y la tableta gráfica. En todos los casos necesitamos saber el tipo de conector que utiliza (será un STRING) y los puertos válidos(varios valores de tipo entero).

Los dispositivos de salida de que disponemos son las pantallas y las impresoras (de inyección y láser). También tenemos que saber los puertos válidos. Además para las impresoras necesitamos saber el tipo de cartucho o "tónerütilizado y el número de páginas impresas desde el último cambio de "tóner"(sólo para impresoras láser).

Por último tenemos un dispositivo especial, la pantalla táctil que sirve de dispositivo de entrada y de salida simultáneamente.

Diseñe las clases y relaciones que representen una solución adecuada para este problema. Escriba en Eiffel al menos las clases que representen el ordenador, los dos primeros niveles de la jerarquía de componentes y las impresoras láser.

Nota: Se dispone de una clase *mi_lista[g−−>*componente) con las siguientes características: *contador*, *precio_total_lista*, *poner(a:G)* y *quitar(a:G)*.

Ejercicio 4

En un sistema de alarma de un edificio se consideran detectores de humo, sensores de temperatura, sensores de presión, etc. Todos estos elementos tienen un estado conectado/desconectado y en consecuencia se puede pasar de un estado a otro (cuando se crean, están desconectados). Todos ellos son capaces de proporcionar una medida (un valor REAL) y tienen un valor umbral que se fija inicialmente al crear el elemento. El sistema recorre en un bucle continuo todos sus elementos conectados. Cuando la medida de uno de ellos supera su valor umbral el sistema dispara la alarma. Para evitar falsas alarmas, varios elementos se pueden unir formando arrays (y los arrays a su vez en otros arrays) y para este sensor complejo, la alarma sólo se dispara si el valor medio de los elementos del array supera el umbral definido para ese elemento compuesto. Diseñar una solución con las clases necesarias y sus características para este problema, implementando en Eiffel al menos todo lo relacionado con el disparo de la alarma.

Ejercicio 5

Una empresa se organiza en una jerarquía de unidades de negocio. Para este caso, de la empresa nos interesa el presidente, el NIF y la dirección postal, mientras que de cada unidad de negocio nos interesa el gerente, el número de empleados, los beneficios brutos del último trimestre, la inversión en edificios y el número medio de contratos realizados por semana.

Además una unidad de negocio puede estar formada por varias unidades de negocio. En este caso empleados, beneficios e inversiones se obtienen como la suma de los datos correspondientes a las unidades de negocio que la componen, mientras que el número medio de contratos es la media de los números medios de éstas unidades de negocio.

Diseñar un conjunto de clases que resuelvan el problema, utilizando el patrón compuesto e implementando completamente en Eiffel las clases necesarias.

Ejercicio 6

Diseñe e implemente en Eiffel una solución que simule el funcionamiento de un ascensor sin memoria. El ascensor debe tener como características `parado:BOOLEAN`,`piso_actual:INTEGER`, `mover_a(piso:INTEGER)`, además de procedimientos privados que permiten `mover_arriba` y `mover_abajo` un piso cada vez, de modo que, cuando el ascensor esté en el piso -1 y reciba el mensaje `mover_a(3)`, la implementación implique 4 llamadas a `mover_arriba`.

Se necesitan los siguientes tipos de paneles:

- El panel interno tendrá tantos botones como pisos y mostrará en cada momento el piso en que se encuentra el ascensor. El procedimiento que simule la pulsación de botones será `mover_a(piso:INTEGER)`.
- El panel externo, del que habrá tantos ejemplares como pisos. Tendrá un botón para llamar al ascensor, y un testigo luminoso que indique si el ascensor está en movimiento. El procedimiento `llamar_ascensor` simulara la pulsación del botón.
- El panel de control, situado fuera del ascensor, que dispone de botones de llamada para cada piso, testigo luminoso de movimiento y muestra el piso en que se encuentra el ascensor en cada momento.

Es importante que ambos tipos de paneles reflejen siempre la información actualizada. La clase `APLICACION` (que **NO** hay que implementar) creará el ascensor y los paneles necesarios, y simulará el funcionamiento del sistema mediante un bucle en que el usuario «pulsa» el botón que desea (mediante una opción de menú), se invoca la correspondiente operación en uno de los paneles y se muestra por pantalla el efecto en todos los paneles a medida que el ascensor va pasando de piso en piso.

Ejercicio 7

Diseñe un sistema que muestre la hora en distintas ciudades. Para ello se utilizará una clase `RELOJ` que almacena internamente la fecha y hora del sistema local (utilice `TIME` por ejemplo). Otra clase `HORA_EN` se encarga de mostrar la hora local en cierto lugar del mundo, determinado en el momento de creación. La información aparecerá en pantalla cada vez que se actualice la hora, con el siguiente formato:

```
Madrid, 20:10:15
```

La clase `APLICACION` creará una instancia de la clase `RELOJ` y tantas instancias de `HORA_EN` como se desee. Para ello habrá que indicar la ciudad (`STRING`) y la diferencia horaria (`INTEGER` entre -12 y 12). La clase `APLICACION` simulará el paso del tiempo enviando el mensaje `poner_en_hora` a la instancia de `RELOJ` de forma continua.

Implemente en Eiffel las clases `APLICACION`, `RELOJ` y `HORA_EN`.

Utilice el patrón observador. Se dispone de las clases ya implementadas:

- Clase diferida `SUJETO`, que tiene implementadas las características `observadores:ARRAY[OBSERVADOR]` `insertar(o:OBSERVADOR)` y `notificar`.
- Clase diferida `OBSERVADOR`, que tiene declarada la característica `actualizar`.

Cuestión adicional: Muestre el paso de mensajes entre los objetos creados cuando se actualiza el reloj (preferiblemente con un diagrama de secuencia).

Ejercicio 8

Una empresa de mobiliario modular realiza ventas de varios tipos de «elementos». Los elementos pueden ser piezas simples o grupos de «elementos» que forma un «kit», con lo que un «kit» es un grupo de «elementos» de cualquiera de los dos tipos.

Para cada tipo de elemento simple existe una clase definida que lo modela, heredera de la clase *ELEMENTO* que dispone de características diferidas *precio:REAL* y *codigo:INTEGER*, sin embargo no existe en el sistema ninguna clase que modele el «kit».

Elabore el diagrama de clases de un sistema que permita la gestión de los elementos que vende la empresa, considerando que cada «kit» tendrá un código numérico propio, y que el precio de un «kit» se calcula como la suma de los precios de sus elementos, pero con un descuento del 10%.

Implemente completamente la clase que permite modelar un «kit», prestando especial atención a su construcción y al cálculo de su precio, que debe realizarse en tiempo de ejecución.

Ejercicio 9

Deseamos modelar los ficheros del sistema operativo mendiante objetos Eiffel. Sobre los ficheros normales almacenaremos su nombre y tamaño (que consieraremos fijado en tiempo de creación, pero variable). Sobre los «links» almacenaremos el nombre, pero su tamaño será en todo momento el tamaño del fichero directorio o «link» a que están conectados. Sobre los directorios guardaremos su nombre, mientras que su tamaño será la suma de los tamañós de los ficheros que contiene. Los directorios se crean vacíos pero se les pueden añadir otros ficheros en cualquier momento.

Considerando que la estructura de directorios no tiene un nivel de profundidad máximo, elaborar el diagrama de clases adecuado para este problema e implementar todas las clases necesarias.
Nota: Ejercicio de examen del curso 2005/06

Ejercicio 10

Elaborar un iterador que permita recorrer un vector tanto en sentido ascendente como en sentido descendente. Es importante elaborar un diseño que permita extender la implementación a iteradores del mismo tipo útiles para recorrer listas ligadas u otro tipo de estructuras.
Nota: Ejercicio de examen del curso 2005/06

Ejercicio 11

La facturación de un taller mecánico requirere de un conjunto de clases que modelen las piezas que se utilizan en una reparación. Sobre cada *PIEZA* se debe reflejar su número de pieza, su nombre, su precio y el tiempo de mano de obra que se facturará por su instalación.

Para ciertas reparaciones se utilizan conjuntos de piezas denominados «kits» (aceite, tapa y filtro de aceite para un cambio de aceite). Los «kits» tienen su propio número, nombre y tiempo de mano de obra, pero su precio resulta de la suma de los precios de las piezas que lo componen menos un 10% de descuento.

Considerando que nombre y número de pieza son muy estables, el precio de las piezas se actualiza con frecuencia y que las piezas que componen un «kit» pueden a su vez set «kits», elaborar el diagrama de clases de la solución propuesta para esta situación e implementar totalmente las clases que modelan los tipos de piezas.
Nota: Ejercicio de examen del curso 2006/07

Ejercicio 12

En un sistema en funcionamiento disponemos de una clase *IMPRESORA* capaz de gestionar una impresora. Entre las características de esta clase, disponemos de *trabajos_en_cola:INTEGER*, *encendida:BOOLEAN*, *imprimiendo:BOOLEAN*, con los significados naturales, además de un método *update* encargado de actualizar la información contenida en el objeto con la situación real del aparato. Sabiendo que *update* es el único método capaz de cambiar el estado de *IMPRESORA*, elaborar una clase *SISTEMA_DE_IMPRESIÓN* capaz de informar sobre el estado de un conjunto de impresoras conectadas al sistema.

La solución del problema, que debe basarse en el patrón observador, debe constar de:

- Un diagrama de clases del diseño utilizado.
- Una implementación de *SISTEMA_DE_IMPRESORAS* que modele al menos el número de impresoras conectadas al sistema, el número de impresoras encendidas, el número de impresoras que están imprimiendo y el número total de trabajos pendientes.

Nota: Ejercicio de examen del curso 2006/07

Ejercicio 13

Una empresa que comercializa cosméticos utiliza un conjunto de clases herederas de la clase diferida *PRODUCTO* para modelar los productos de su catálogo. En la forma corta de la clase *PRODUCTO* encontramos, entre otras, las siguientes características.

```
deferred class interface PRODUCTO
feature(s) from PRODUCTO
  -- Secciones de la interfaz omitidas
  nombre:STRING
  precio:DOUBLE
  ref:INTEGER
    -- Identifica el producto
  farmacia:BOOLEAN
    -- ¿Es de venta exclusiva en farmacias?
end of deferred PRODUCTO
```

Ahora la empresa va a incorporar los lotes de productos como elemento promocional. Un lote, que tiene su propio nombre y referencia, está formado por un grupo de productos. Su precio se obtiene como la suma de los precios de los productos que lo componen con un descuento específico del lote, que puede cambiar con frecuencia. Dado el carácter promocional de los lotes, es frecuente que cambien de composición o que un lote esté formado por varios lotes y/o productos individuales. También es frecuente que los productos cambien de precio. Por restricciones legales, un lote no puede contener ningún producto marcado para venta en farmacias.

Diseñe e implemente en Eiffel las clases necesarias para incorporar lotes al catálogo de la empresa.

Nota: Ejercicio de examen del curso 2007/08

Ejercicio 14

Un sistema informático en uso dispone de una clase *CONTADOR* con la siguiente forma corta:

```
class interface CONTADOR
creation
  make
feature(s) from CONTADOR
  valor: INTEGER
  incremento
    ensure
      valor = old valor + 1
end of CONTADOR
```

El objetivo de la clase *CONTADOR* es que cada una de sus instancias refleje el número de segundos transcurridos desde su creación. Para ello, un elemento independiente del sistema llama al método *incremento* del contador cada segundo.

Ahora deseamos construir una clase *RELOJ* que refleje la hora, minuto y segundo actual, utilizando para ello a la clase anterior. Un reloj se inicializa con una hora, minuto y segundo determinado, y va actualizando estos valores conforme su correspondiente contador refleja el paso del tiempo.

Diseñar y construir las clases necesarias para resolver este nuevo problema prestando especial atención a la programación bajo contrato, utilizando adecuadamente los patrones de diseño y considerando que es razonable que un solo contador pueda ser utilizado por varios relojes.

Nota: Ejercicio de examen del curso 2007/08

Ejercicio 15

Con la entrada en vigor del EEES, el viejo profesor tiene que cambiar la forma de calificar sus asignaturas. Ahora conviven asignaturas con un solo examen final con asignaturas con una multitud de pruebas, tanto pruebas simples como pruebas formadas por varias pruebas de diversos tipos.

Para cada prueba, el viejo profesor desea mantener un registro por alumno, con la nota máxima posible y la nota del alumno (la nota mínima es siempre cero) además de una cadena de caracteres que identifica la prueba y una marca que indique si la prueba ha concluído o no.

La nota (tanto máxima como del alumno) de algunas de las pruebas formadas por varias pruebas se calcula como la suma de las pruebas que la componen, pero en otras es el resultado de calcular la media de esas pruebas. Evidentemente una prueba no ha concluído hasta que no lo hacen todas las que la componen, y sólo en ese caso se puede calcular la nota.

Diseñe y construya un conjunto de clases que permitan modelar esta situación, explicando al viejo profesor los motivos que le llevaron a construir esta solución.

Nota: Ejercicio de examen del curso 2008/09

Ejercicio 16

En un sistema informático se dispone de una clase *IBEX* que mantiene la información sobre la cotización de los valores del IBEX35 un extracto de cuya forma corta es el siguiente:

```
class interface IBEX
creation
  make
    -- Creación
feature(s) from IBEX
  -- Para simplificar el IBEX35 está compuesto sólo por tres empresas
  abengoa: DOUBLE
    -- Cotización de Abengoa
  acciona:DOUBLE
    -- Cotización de Acciona
  acerinox:DOUBLE
    -- Cotización de Acerinox
  actualiza
    -- Actualiza las cotizaciones de todos los valores
end of CONTADOR
```

Ahora necesitamos clases que mantengan información actualizada sobre el índice IBEX35. En concreto necesitamos:

- Una clase que mantenga actualizado el valor medio de las acciones del IBEX35
- Una clase que mantenga información actualizada sobre el valor actual de Acciona y sobre la variación en tanto por ciento sobre su última cotización.

Diseñar y constuir las clases Eiffel que satisfagan estas nuevas necesidades.

Nota: Ejercicio de examen del curso 2008/09