Pauta / Auxiliar I

METODOLOGÍAS DE DISEÑO Y PROGRAMACIÓN

CC3002 @ 2009

Ejercicio 1

Las definiciones dadas en el enunciado fueron tomadas de respuestas de examen de otras ediciones de un curso similar. Todas las definiciones están incompletas y solo mencionan algunos aspectos de la definición de clase. El objetivo de este ejercicio es contrarrestar estas respuestas con la definición dada en la cátedra.

Ejercicio 2

a) No. El concepto de datatype es a nivel de clase. Un objeto es una instancia de una clase, por lo que no tiene sentido compararlo con un datatype. Un datavalue es una instancia de un datatype, carece de identidad, mientras que un objeto es una instancia de una clase, y tiene la propiedad de identidad.

b) Como se ve en el curso la respuesta es sí. Los datavalues son una instancia de un datatype y son usados generalmente como valores de atributos, pero un atributo puede ser también la representación de uno o varios links a objetos.

Ejercicio 3

Tanto el pasaje de parámetros por valor como el operador de asignación realizan copias. Las modificaciones realizadas al parámetro efectivo (o al objeto del lado izquierdo de una asignación), no se verán reflejadas en el objeto original. Si se trata de un datavalue no se tendrá problemas de consistencia dado que el mismo carece de identidad, pero no se aplica lo mismo para los objetos. Lo dicho anteriormente se aclara pensando que dos copias d1 y d2 de un mismo datavalue son lo mismo, pero dos copias o1 y o2 de un objeto, son objetos diferentes. Como regla general es recomendable evitar copias de objetos.

Ejercicio 4

a) Si. Existe una asociación entre objetos de A y B.

b) Si. Existe una asociación entre objetos de B y C.

c) No. No existe ninguna asociación entre objetos de A y C.

d) No. No puede existir más de un link perteneciente a la misma asociación entre el mismo par de objetos.

Ejercicio 5

a)

i) A

ii) B

iii) B, C, D y E

iv) E

v) A y B

b) Ver cátedra.

```
A a;
B b;
...
a = b;
```

c)

FD_A	FD_B	FD_C	FD_D	FD_E
ATT_{at1}	ATT_{at1}	ATT_{at1}	ATT_{at1}	ATT_{at1}
OP_{op1}	OP_{op1}	OP_{op1}	ATT_{at2}	OP_{op1}
OP_{op2}	OP_{op2}	OP_{op2}	OP_{op1}	OP_{op2}
$MET_{A::op2}$	OP_{op3}	$MET_{C::op1}$	OP_{op2}	OP_{op3}
	$MET_{A::op2}$	$MET_{A::op2}$	OP_{op4}	$MET_{E::op1}$
	$MET_{B::op3}$		$MET_{D::op1}$	$MET_{E::op2}$
			$MET_{A::op2}$	$MET_{B::op3}$
			$MET_{D::op4}$	

d) Es instancia directa de E e indirecta de A y B.

Ejercicio 6

a) No es válida, d no es instancia de clase E.

b) Es válida, e es instancia directa de clase E.

c) Es válida, por la propiedad de subsumption, como f es instancia de clase F y F es subclase de E, f también es instancia de clase E, en particular, f es instancia indirecta de clase E.

Ejercicio 7

a) No. Una operación es una especificación de una transformación o consulta que un objeto puede ser llamado a ejecutar, éstas tienen un nombre y una lista de parámetros, mientras que un método es la implementación de las mismas para una determinada clase. Una misma operación puede estar asociada a diferentes métodos (polimorfismo).

b)

Clases:	Empresa, Empleado, Fijo, Jornalero
Atributos:	Empleado : nombre Fijo : nombre (heredado de Empleado) y sueldo Jornalero : nombre (heredado de Empleado), cantHoras y valorHora Empresa : nombre
Operaciones:	Empleado : getNombre() y getCobro() Fijo : las mismas que Empleado (heredadas) Jornalero : las mismas que Empleado (heredadas) Empresa : getTotal()
Métodos:	Los métodos son las implementaciones de las operaciones de las clases. En este caso se asocian a getCobro() métodos distintos en Comun y Jornalero, que calculan la liquidación de acuerdo al tipo de empleado.

c) Una clase es abstracta cuando cada miembro de esa clase debe ser un miembro de una subclase. Esto se aplica cuando el concepto que queremos representar no existe en sí mismo (es abstracto) sino que engloba características comunes de otros conceptos. Esto sucede en el ejemplo ya que un Empleado es un Empleado Fijo o un Jornalero y no existen (en el dominio planteado por el problema) Empleados que no pertenezcan a alguna de estas categorías.

d)

```
float Empresa::getTotal()
{
   float total = 0;
   Empleado e;
   IteratorEmpleado it = empleados.getIterator();
   while (it.hasNext())
   {
       e = it.next();
       total = total + e.getCobro();
   }
}
```

Notar que la operación getCobro() es polimórfica, por lo que se invoca indistintamente para todos los tipos de empleados.

e) Polimorfismo es la capacidad de asociar diferentes métodos a la misma operación. En este ejemplo se aplica polimorfismo al asociar diferentes métodos a la operación getCobro() en las clases Fijo y Jornalero como vimos en la respuesta de b).

Ejercicio 8

a) Ver cátedra.

b)

i) Interfaz: C
Realización: P1 y P2 realizan C

ii) Interfaz: Salida de audio del DVD
Realización: televisión y home theater; ambas realizan la salida de audio del DVD

iii) Interfaz: F1
Realización: Parlantes P1 y P2; realizan la interfaz F1

iv) Interfaz: Conjunto de operaciones para manejo de colecciones
Realización: Estructuras de lista, árbol y tablas de dispersión; realizan la interfaz de colección

c) En todos los casos, el separar la interfaz de la realización permite al usuario de la primera independizarse de los detalles internos de la segunda. De esta forma el usuario del lavarropas utilizará indistintamente las unidades fabricadas con P1 y P2, el usuario del reproductor de DVD simplemente conecta su salida de audio sin tener que preocuparse de los detalles del dispositivo de salida (ídem para los distintos parlantes) y el usuario de la colección utiliza las operaciones sin preocuparse por los detalles internos de la implementación de las estructuras de datos de las mismas.

Ejercicio 9

a) El objeto b es estáticamente declarado como de clase A por lo tanto el compilador utiliza únicamente esta información para decidir. Para que la invocación sea válida, la clase A debe contener la operación op() ya sea abstracta o concreta.

b) Asumiendo que el código compiló, la clase A contiene la operación op(). El runtime consulta el tipo dinámico del objeto b (que es B) y busca un método para op() en B (notar que por lo antedicho es seguro que B hereda la operación op()). Necesariamente deberá encontrar un método para op() en B, ya sea introducido por B o heredado de A. La situación alternativa (que no se encuentre un método para op() en B) no puede ocurrir. Si así fuese, op() sería abstracta en B, lo que causaría que la clase B fuese abstracta, y en consecuencia el new en la línea 4 habría producido un error de compilación (las clases abstractas no se pueden instanciar) evitando la ejecución que se encuentra en discusión.