PROGRAMACIÓN ORIENTADA A OBJETOS **06/09/2006**

NOMBRE:_____ **Titulación:**_____

1. (**0'75 ptos**) ¿Qué significan los conceptos de ***alta cohesión*** y ***bajo acoplamiento*** y qué tienen que ver con el modelo de objetos?

2. Dado el siguiente código Java:

```
public class A {
      int at;
      …
}
```

```
public class B{
     A at;
     …
      public void met(int a){
              at.at = a;
      }
}
```

a) (**0'5 ptos**) Indique el nivel de visibilidad que habría que asignar a los atributos de la clase A para que el código anterior fuese correcto. Indica todas las posibilidades.

b) (**0'5 ptos**) ¿Y si escribiéramos el código C++ equivalente? Indica todas las posibilidades.

c) (**0'5 ptos**) ¿Y si escribiéramos el código Eiffel equivalente? Indica todas las posibilidades.

3. Sea la clase `Fraccion` la encargada de representar los números racionales (por ejemplo, $\frac{3}{4}$ ó $\frac{2}{5}$). La implementación de esta clase en Eiffel sería:

```
class Fraccion creation make
feature
   numerador: INTEGER
   denominador: INTEGER

   make (n:INTEGER, d:INTEGER) is do
      numerador:= n;
      denominador := d;
   end

   inversa is
     require  numerador != 0
     local aux: INTEGER
     do
       aux := numerador
       numerador := denominador
       denominador := aux
     ensure numerador = old denominador ;
            denominador = old numerador
   end

   invariant
      denominador > 0
end
```

a) (**0'5 ptos**) ¿Qué relación existe entre las rutinas de creación y el invariante de la clase?

b) (**0'5 ptos**) Utiliza el método `inversa` para explicar la técnica del *Diseño por Contrato* propuesta por Bertrand Meyer.

c) (**1 pto**) Escribe el código equivalente de la clase `Fraccion` en Java conservando toda la semántica.

4. Un sensor va tomando medidas y cada vez que cambia de valor debe informar a distintos objetos que lo están observando, entre otros un objeto gráfico que se encarga de visualizar las medidas en una interfaz gráfica, un objeto alarma, que se dispara si la medida del sensor alcanza un determinado valor. De manera que el método `informar` de la clase `Sensor` es el encargado de recorrer la colección de los objetos que le están observando y de informarles del nuevo valor mediante el método `cambioEstado(double newValue)`.

a) (**1 pto**) Implementar parcialmente la clase `Sensor` en Java, incluyendo la manera de almacenar los objetos que están pendientes del cambio de estado y el método `informar` utilizando los iteradores que proporciona el lenguaje.

b) (**0'25 ptos**) ¿Los iteradores de Java son iteradores internos o externos? Justifica la respuesta.

5. Un carro de la compra está formado por una colección de líneas de compra. Cada línea de compra representa el producto que se ha añadido al carro y el número de unidades que se ha pedido. Sea la clase `CarroCompra` una clase genérica Eiffel que queremos que sirva para representar compras de cualquier producto de empresas diferentes: multimedia, coches, papelería, … con sólo que sean inventariables, es decir, que tengan un número de referencia y un precio. La implementación propuesta para dicha clase es la siguiente:

```
class CarroCompra[G] feature
  ticket: LIST[LineaCompra[G]]
  …
  getTotal: REAL is do
   from ticket.start
   until ticket.after
   loop
     Result:=Result + ticket.item.getSubtotal
     ticket.forth
   end
  end
end
```

```
class LineaCompra[G] feature
    producto: G
    cantidad: INTEGER
    …
    getSubtotal: REAL is do
      Result:= producto.precio * cantidad
    end
End
```

a) (**0'5 ptos**) ¿Sería correcto el código propuesto? En el caso de que no lo sea explica la manera de resolverlo.

b) (**1 pto**) A partir de la implementación de los iteradores vistos en clase para el lenguaje Eiffel (clase `Linear_Iterator`), implementa un iterador que calcule el valor total de una compra (suma del importe de cada línea de compra). Implementa de nuevo el método `getTotal` de la clase `CarroCompra` haciendo uso del nuevo iterador.

6. **(1pto**) Suponga la implementación de las clases `CarroCompra` y `LineaCompra` en Java. Una vez sumado el importe total de la compra se pueden aplicar diferentes algoritmos de descuento. Por ejemplo, un descuento por cantidad (si la compra supera los 100 euros se hace una rebaja de 5 euros) o aplicar un porcentaje (aplicar un 10% de descuento sobre el total de la compra). Por tanto, cada algoritmo de descuento recibe como entrada el importe sobre el que tiene que aplicar el descuento y devuelve la cantidad a descontar. Explica e implementa la manera de pasar como parámetro al método `getTotal` el algoritmo de descuento que se le debe aplicar a la compra que se está haciendo, de manera que `getTotal` devuelva el importe que debe pagar el cliente (suma de las líneas de compra menos el descuento).

7. Supuesta la implementación del carro de la compra en C++, se ha decidido utilizar la herencia de implementación en lugar de una relación de clientela del siguiente modo:

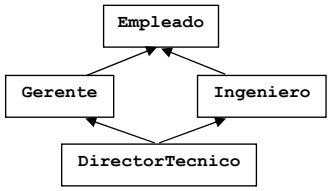
```
class CarroCompra: private ListaLineas { … };
```

a) (**0'5 ptos**) ¿Cuál sería el efecto del siguiente código?

```
ListaLineas *lista;
CarroCompra *carrito = new CarroCompra();
…
lista = carrito;
lista->add(linea);  //siendo add un método de la clase ListaLineas
```

b) (**0'5 ptos**) ¿Podemos implementar la misma relación de herencia y con el mismo efecto en Eiffel? Justifica la respuesta.

8. Dada la siguiente jerarquía de clases:



a) (**0'5 ptos**) Supuesta la implementación de dicha jerarquía en Eiffel. ¿Cuál es la política de compartición y replicación de las características (atributos y métodos) en este lenguaje?

b) (**0'5 ptos**) ¿Cómo debería implementarse dicha jerarquía con herencia repetida en el caso de C++?