Programación Orientada a Objetos.

Primer parcial 2do semestre 2011 - Miércoles 7 de setiembre de 2011

1.- Sean las siguientes clases e interfaces, y las siguientes asignaciones:

```
public interface IComida{
    void Preparar();
}
public interface ICarne:IComida {
    void ProcesarCarne();
}
public interface IDulce {
    void AgregarAzucar();
}
public interface ITarta:IDulce, IComida{
    void ColocarMasa();
}
public interface IBebidaDulce: IDulce {
    int Capacidad();
}
public class Caramelo: IDulce {
  public String Sabor(){ /* Código */ }
}
public class Churrasco: ICarne {
  String ObtenerPuntoCoccion(){ /* Código */ }
}
public class TartaManzana: ITarta {}
public class AguaDulce: IBebidaDulce {}
public class Pollo: IComida {}
public class TartaFrutilla: ITarta {}

class Program
{
    static void Main(string[] args)
    {
        /1/ IDulce d = new TartaManzana();
        /2/ ITarta t = d;
        /3/ ICarne ca = new IComida();
        /4/ Caramelo c = new Caramelo();
        /5/ TartaFrutilla tf = new ITarta();
        /6/ IComida p = new Pollo();
        /7/ Pollo po = p;
        /8/ TartaManzana tm = tf;
    }
}
```

a) Indique las líneas de código incorrectas y justifique brevemente. Si alguna línea es incorrecta, asume de todos modos que la variable fue definida.

2.- Responde las siguientes preguntas basándote en el ejercicio anterior:
a) ¿Qué tipos tienen los objetos creados en las líneas 1 y 6?
b) ¿Qué mensajes puede recibir un objeto cómo el creado en la línea 4? ¿Cómo lo sabes?
c) ¿Qué e mensajes puede recibir un objeto bajo los términos de la variable declarada en la línea 5? ¿Cómo lo sabes?
d) ¿Qué contrato firman las variables declaradas en las líneas 3 y 7?

3.- Sean las siguientes clases:

```
class A {
      public Int32 a;
      public String b;
      public A(Int32 a, Int32b)
      {
            this.a = a;
            this.b=b;
      }
}
class B {
      public Int32 a;
      public String b;
}
 class Program
    {
        static void Main(string[] args)
        {
            /1/ A a = new A(0,"Hola");
            /2/ B b = new B();
            /3/ B c = b;
            /4/ A d = a;
            /5/ d.a = 5;
            /6/ c.b = "Hola";
            /7/ b.a = 5;
            /8/ B e = new B();
            /9/ B f = e;
        }
  }
```

a) Indica si la clase A esta debidamente encapsulada, y si no lo está programa los cambios necesarios para que lo esté.
b) Una de las principales diferencias entre las clases A y B, es que la segunda no tiene un constructor definido. Explica cómo influye esto, y cuál es la utilidad del constructor de A.
c) Indica que variables apuntan a objetos iguales y que variables apuntan a objetos idénticos al final del código provisto en el *Main* del *Program*.
d) Si no encuentraste variables que referencien a objetos idénticos, o variables que referencien a objetos iguales en la parte anterior, agrega las líneas de código que precises para que ambos casos se den.

4.- Sea el siguiente código:

```
interface ILineaTelefonica
{
    int GetInterno();
}
class LineaIP : ILineaTelefonica
{
    public int GetInterno()
    { /* Devuelve el número de interno */ }
    public int GetIP()
    { /* Codigo complejo que devuelve la IP */ }
}
class LineaComun : ILineaTelefonica
{
    public int GetInterno()
    { /* Devuelve el número de interno */ }
    public int GetNumeroTelefono()
    { /* Codigo complejo que devuelve el número de télefono */ }
}
```

```
class CentralTelefonica
{
    ArrayList lineas;
    public void Llamar(ILineaTelefonica origen, int interno)
    {
        foreach (ILineaTelefonica t in lineas)
        {
            if (t.GetInterno() == interno)
            {
                if (t.GetType() == typeof(LineaComun))
                {
                    ConectarLlamada(origen,
                              ((LineaComun)t).GetNumeroTelefono());
                }
                else
                {
                    ConectarLlamada(origen, ((LineaIP)t).GetIP());
                }
            }
        }
    }
    public void ConectarLlamada(ILineaTelefonica origen,
                                int lineaDestino)
    {/* Código que conecta la llamada */}
}
```

a) Indica si el código anterior cumple o no con los siguientes principios y justifica tu respuesta. Debes indicar si los principios se cumplen o no se cumplen, y explicar tu afirmación.

- OCP
- SRP
- Experto

5.- Modifica el código del ejercicio anterior para que cumpla con los 3 principios planteados en la pregunta (si es que no se cumplían). Debes programar los cambios que sean necesarios en tu solución.

6.- Dadas las siguientes preguntas, marca la opción **MÁS** correcta (debes marcar una sola opción):

a) Una operación polimórfica se da si existe/n:
i- dos o más operaciones con el mismo selector pero diferente firma dentro de una clase.
ii- una operación implementada únicamente en una clase.
iii- una operación implementada en una o más clases.
iv- una operación implementada en dos o más clases.

b) Una clase es:
i- Un archivo separado de todos los demás.
ii- Una fábrica de objetos.
iii- Un conjunto de variables de instancia y sus valores.
iv- Un conjunto de variables de instancia y de métodos sin implementación.

c) Las interfaces implementan a las clases:
i- Verdadero
ii- Falso

d) El estado de un objeto es:
i- el conjunto de variables de instancia de una clase y sus respectivos tipos
ii- el conjunto de instancias de la clase
iii- el conjunto de valores que se asocian a las variables de un objeto.
iv- el conjunto de las responsabilidades de conocer que tiene una clase.

e) Los tipos se relacionan con todo lo que se lista a continuación, excepto:
i- Definen un conjunto de operaciones.
ii- Definen contratos.
iii- Permiten la re-utilización de código con sus subtipos.
iv- Definen relaciones jerárquicas

f) Un objeto es inmutable cuando:
i- Durante toda su vida su estado no puede cambiar.
ii- Existe un constructor que inicializa los valores de todas sus variables.
iii- Todas sus variables de instancia fueron definidas como privadas.
iv- No existen cambios en la definición de los tipos de sus variables.

g) Cuando nosotros programadores, no definimos un constructor en una clase en C#, entonces:
i- No se pueden crear instancias de esa clase.
ii- Solo podremos crear objetos invocando a un constructor por defecto sin parámetros.
iii- Solo podremos crear objetos inmutables
iv- Hay que crear un constructor sobrecargado para posibilitar la creación de instancias.

h) Encapsulación es sinónimo de:
i- ilusión de simplicidad
ii- ocultamiento de información
iii- KISS, keep it simple, stupid

7.- Sean las siguientes clases e interfaces:

```
public interface IDulce {
    void AgregarAzucar();
}
public interface IBebidaDulce: IDulce {
    int Capacidad();
}
public class Coca_Cola: IBebidaDulce {
}
public class Yoyo: IDulce {
}
```

a) Explica, basándote en el ejemplo anterior, lo que plantea el principio de sustitución de Liskov (LSP).
b) Programa un pequeño fragmento de código, también utilizando las clases e interfaces de este ejercicio, que invoque una operación que se comporte polimórficamente. Señala la operación.
c) Explica, para dicha operación polimórfica, en base a que se determina el método que se ejecuta.