Programación orientada a objetos. Exámen Diciembre 2010.

15 de diciembre de 2010.

1. Sean las siguientes clases e interfaces:

```
interface IAlmacenable { }
interface IBotella : IAlmacenable { }
interface IBolsa : IAlmacenable { }
interface ICaja : IAlmacenable { }
abstract class Refresco : IBotella { }
class Agua: Refresco { }
class Zapato : ICaja { }
class Jugo : IBolsa, IBotella { }
class Yogurt : IBolsa, IBotella { }

class Programa {
    void Inicial()
    {
        /*1*/ IAlmacenable a = new Agua();
        /*2*/ Refresco r = a;
        /*3*/ IBotella b = new Refresco();
        /*4*/ IBolsa j = new Jugo();
        /*5*/ IBolsa y = new Yogurt();
        /*6*/ IBotella d = j;
        /*7*/ Yogurt t = y;
        /*8*/ ICaja c = new ICaja ();
    }
}
```

1.1 Indica las líneas de código incorrectas. Si una asignación es incorrecta, de todas formas asume que la variable fue definida.

1.2 Indica los tipos de la variable **t** de la **línea 7** y los tipos del objeto creado en la **línea 3**.

1.3 Realiza los cambios necesarios en el código anterior para que:
a) un **objeto** como el creado en la **línea 4** pueda recibir los siguientes mensajes: gusto(), color()
b) una **variable** como la definida en la **línea 5** pueda recibir los siguientes mensajes: capacidad() y color()

2. Sea el siguiente código

```
class Arbol {
    Int32 globos {get; set;}
    void ArmarArbol (){ }
    void DesarmarArbol (){ }
}
class Navidad {
    void DejarRegalos(){ }
    void AbrirRegalos(){ }
    void Brindar(){ }

}
```

Y las siguientes condiciones:
a) Sólo se brinda después de las 12.
b) Para dejar los regalos, tiene que estar armado el arbolito.
c) El arbolito queda armado cuando se colocan los 10 globos.
d) El arbolito se desarma después del 8 de enero y sólo si fue previamente armado.
e) Cuando se desarma el arbolito, se sacan los 10 globos
f) En la Navidad siempre se dispone de al menos 3 y a los sumo 20 fuegos artificiales.
g) Los regalos se abren después de brindar.

2.1 Indica qué condiciones corresponden a precondiciones, postcondiciones e invariantes.
2.2 Modifica las clases necesarias para cumplir todas las condiciones mencionadas. Además, declara las precondiciones, postcondiciones e invariantes usando Debug.Assert(bool Condition).

3. Sea el siguiente código:

```
interface IFecha {
      /* Devueleve Verdadero si hoy es un día festivo */
      bool EsDiaFestivo();
}
class RedactorTarjetas {
     void CrearTarjeta (IFecha fecha)
     {
         if (fecha.EsDiaFestivo())
         {
             EscribirYGraficar("Felices Fiestas");
         }
         else
         {
            EscribirYGraficar("Gracias por preferirnos");
         }
     }
     void EscribirYGraficar(String mensaje)
      {/*Código que escribe y grafica mensajes en diferentes formatos
       según el destinatario */}
 }
```

El código anterior provee un redactor de tarjetas que según la fecha actual imprime en ellas diferentes mensajes. La manera de escribir esos mensajes y de graficarlos, es diferente según el destinatario que recibirá las tarjetas. Se necesita entonces modelar una solución que permita reutilizar el código referente a la determinación del texto a incluir en los mensajes, pero que permita tener diferentes maneras de implementar lo referente a la impresión propiamente dicha, cosa resuelta actualmente por el método EscribirYGraficar().

3.1 Resuelve este problema utilizando herencia o usando composición y delegación. Utiliza el mecanismo que creas más conveniente para el problema y justifica brevemente. Debes programar la solución.
3.2 Describe cómo lo resolverías utilizando el mecanismo de reutilización que NO utilizaste en la parte 3.1. Ej: Si utilizaste herencia describe cómo lo resolverías usando composición y delegación y viceversa.
3.3 Se dice que la herencia permite reutilización estática mientras que composición y delegación permite reutilización dinámica. Indica por qué es posible hacer esta afirmación.
3.4 Además de herencia y composición y delegación, ¿qué otro mecanismo de reutilización, de los dados en clase, conoces?

4. Basándote exclusivamente en el código provisto originalmente por el ejercicio 3:
4.1 Describe brevemente qué entiendes por el principio de sustitución de Liskov (LSP). Critica el código en base al LSP.
4.2 Describe brevemente qué entiendes por el principio abierto/cerrado (OCP). Critica el código en base al OCP.
4.3 Describe los cambios que harías en el código para hacer cumplir dichos principios, en caso que alguno no se cumpliera.

5. Sea el siguiente código:

```
class Bailarin
{
    private String nombreCompleto;
    public String NombreCompleto { get; }
    private String nivel;
    public String Nivel { get; }
    private String genero;
    public String Genero { get; }
}

class PoolBailarines
{
    private IList<Bailarin> bailarines;
    public IList<Bailarin> Bailarines { get; }
    public void AgregarBailarin(Bailarin bailarin)
    {/* código que agrega un bailarín a la lista*/}
    public void RemoverBailarin(Bailarin bailarin)
    {/*código que saca un bailarín de la lista*/}
    public IList<Bailarin> BuscarBailarinPorGenero(String genero)
    {/* código que devuelve una lista de bailarines de un género*/}
}
class AutorizadorBailarin
{
    private PoolBailarines pool;
    public AutorizadorBailarin(PoolBailarines pool)
    {
        this.pool = pool;
    }
    public void AutorizarBailarin(Bailarin bailarin) {
        pool.AgregarBailarin(bailarin);
    }
    public void DesautorizarBailarin(Bailarin bailarin) {
        //Bailarin se jubila
        pool.RemoverBailarin(bailarin);
    }
}
```

Contesta las siguientes preguntas, sobre el código anterior:
5.1 ¿Es la clase Bailarin inmutable? Si no lo es, indica qué cambios realizarías para que lo fuera.
5.2 Critica el código en base al ISP (Principio de segregación de interfaces), ¿lo cumple?
5.3 Critica el código en base al SRP (Principio de responsabilidad única), ¿lo cumple?

6. Programa las modificaciones necesarias del código del ejercicio 5 para que éste:
6.1 Cumpla ISP, en caso que no lo cumpla (por más que sea una violación menor).
6.2 Cumpla SRP, en caso que no lo cumpla (por más que sea una violación menor).

7. Sea el siguiente código:

```
public class CyborgOS
{
    public virtual void Boot()
    { /* Bootea el sistema operativo */ }
}
public class EOS
{
    public virtual void Iniciar()
    { /* Inicia el sistema operativo */ }
}
public class BlueBerryOS
{
    public virtual void Levantar()
    { /* Levanta el sistema operativo */ }
}
public class EPhone
{
    private EOS sistemaOperativo;
    public void Iniciar()
    {
        this.sistemaOperativo.Iniciar();
    }
}
public class BlueBerry
{
    private BlueBerryOS sistemaOperativo;
    public void Levantar()
    {
        this.sistemaOperativo.Levantar();
    }
}
```

Imagina que un amigo desea utilizar un teléfono BlueBerry con el nuevo sistema operativo CyborgOS y otro amigo desea hacer lo mismo con su EPhone. Cómo ya eres un programador experto te piden ayuda.

7.1 ¿Cómo harías para que el BlueBerry y el EPhone puedan utilizar el SO CyborgOS? Intenta no modificar nada del código. Puedes agregar todo el código que quieras.
7.2 Como hiciste tan buen trabajo con el EPhone y el BlueBerry, te invitan a la convención internacional de GSM para que propongas un cambio, de forma que cualquier celular pueda utilizar cualquier sistema operativo. ¿Qué cambio propondrías? ¿Utilizaste algún patrón de los vistos durante el curso en tu solución?