Programación orientada a objetos. Examen febrero 2012.

28 de febrero de 2012.

1.- Sean las siguientes clases e interfaces:

```
interface IDibujoAnimado { }
interface IPerro { }
interface Personaje: IDibujoAnimado { }
abstract class PerroAnimado : IDibujoAnimado { }
class PerroSalchicha : IPerro { }
class Perro : PerroAnimado { }
class Ogro : Personaje { }
class Pitufo : Personaje { }
class Superheroe : IDibujoAnimado{ }
class Elfos { }
class Programa {
    void Principal()
    {
        /*1*/ IDibujoAnimado o = new Ogro();
        /*2*/ Personaje p = o;
        /*3*/ PerroAnimado pa = new PerroAnimado();
        /*4*/ Personaje pi = new Pitufo();
        /*5*/ Superheroe s = p;
        /*6*/ Elfos e = o;
        /*7*/ IDibujoAnimado d = pi;
    }
}
```

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 definida en la línea **3** y los tipos del objeto creado en la línea **4.**

1.3) Usando las clases e interfaces definidas anteriormente, programa un ejemplo de código donde existan:
a- *dos objetos de distinta clase,*
b- *dos variables de distinto tipo* que referencien a uno de los objetos (de la parte a),
c- y una variable que referencie al otro objeto (de la parte a).

2.- Sea el siguiente código:

```
interface IReproducible { }
class CD: IReproducible {
    public void GirarConLaser();
}
class DiscoVinilo : IReproducible {
    public void GirarConPua();
}
class EquipoAudio {
    private List<IReproducible> musica;
    public List<IReproducible> Musica
    {
        get { return musica; }
        set { musica = value; }
    }
    private Bandeja bandeja;
    void Reproducir(IReproducible reproducible) {
        bandeja.IntroducirDisco(reproducible);
           if (reproducible is CD) {
            ((CD)reproducible).GirarConLaser();
        }
        else {
            ((DiscoVinilo)reproducible).GirarConPua();
        }
    }
}
```

```
class Bandeja {
    public void IntroducirDisco(IReproducible disco) {
        //Método complejo que coloca un reproducible en la bandeja reproductora
    }
    public void MostrarTitulo(EquipoAudio equipo) {
        // Metodo que despliega el titulo del cd en el display
    }
}
```

2.1) Indica si los siguientes principios se cumplen:
- OCP
- Patrón Experto
- ISP
- SRP

2.2) Crea un constructor que reciba argumentos, para la clase Audio.

3.- Modifica el código del ejercicio anterior para que los principios que no se cumplían, ahora lo hagan. Puedes modificar y adicionar todo lo que consideres necesario.

4.- Acerca del concepto “Diseño por contratos”:
4.1) ¿Qué entiendes por “Diseño por contratos?
4.2) Define qué entiendes por precondición, postcondición e invariante.
4.3) Indica quién es el culpable del no cumplimiento de una postcondición y que significa esto para el contrato definido.
4.4) Dadas las siguientes clases, crea una precondición y una postcondición. Luego genera un fragmento de código, que provoque una violación de ambas.

```
class TV
{
    public void Apagar()
    { }
    public void AvanzarCanal()
    { }
}
class ControlRemoto
{
}
```

5.- Sea el siguiente código:

```
class Guitarra
{
    public void Tocar()
    { /* Complejo código que hace funcionar la guitarra */ }
}
class Bajo
{
    public void Tocar()
    { /* Complejo código que hace funcionar el bajo */ }
}
class Banda
{ }
class Artista
{
    public void HacerMusica(Banda banda)
    { }
}
class Program
{
    static void Main()
    {
        Artista a = new Artista();
    }
}
```

5.1) Completa todo lo que creas necesario en el código anterior para que el Artista pueda hacer música. Puedes modificar las clases actuales, pero no agregar nuevas clases o interfaces.

5.2) ¿Utilizaste algún mecanismo de reutilización de código para lograrlo? Si tu respuesta es afirmativa, indica cual.
5.3) Ahora sí, agregando todas las clases e interfaces que quieras, programa una solución que te permita agregar nuevos instrumentos al modelo, sin que esto repercuta en lo que ya tienes programado.

6.- Observa el siguiente programa, escrito por una persona sin conocimientos de programación orientada a objetos:

```
interface IVehiculo
{
    int Avanzar();
    bool tienecombustible();
    void CargarCombustible() { /* Implementacion común a los vehículos */ }
}
class Auto : IVehiculo
{
    public string nombre;
    public int velocidad;
    public int combustible;
    public void Avanzar()
    { /* Implementacion */ }
    public bool tienecombustible()
    {
        return this.combustible == 0;
    }
    public bool Gano
    {
        get { /* devuelve true si el auto ganó */}
    }
}
class Carrera
{
    ArrayList competidores;
    public int Distancia
    {
        get { return this.Distancia; }
        set { this.Distancia = value; }
    }
    public Carrera(int distancia)
    {
        this.Distancia = distancia;
    }
    public bool Correr()
    {
        foreach (IVehiculo v in this.competidores)
        {
            if (v.tienecombustible)
            {
                v.Avanzar();
                if (v.Gano())
                {
                   Console.WriteLine("El vehículo {0} ha ganado", v.ToString());
                }
            }
        }
    }
}
```

6.1) Encuentra 5 errores que presenta el código anterior (ej: buenas prácticas, lógica, sintaxis, construcción, o principios)
6.2) Soluciona 3 de ellos. Puedes hacerlo en la letra del examen para no reescribir código.

7.- Dadas las siguientes clases, debes agregar las clases necesarias para que el programa compile:

```
public class Base
{
    public void UnMetodo()
    {
        Console.WriteLine("Esto se imprimirá en la consola…");
    }
}
public class Programa
{
    static void Main()
    {
        Sucesora s = new Sucesora();
        s.UnMetodo();
        Compuesta c = new Compuesta();
        c.UnMetodo();
    }
}
```

7.1) Agrega las clases que hagan falta para que el programa compile usando Composición y Delegación y Herencia.

Consideraciones generales:

- Puedes contestar parte de las preguntas en las hojas del enunciado (**siempre que digas en las hojas de examen que parte de la respuesta está allí**)
- Puedes usar ambos lados de la hoja de examen para responder.
- Debes numerar las hojas.
- No es necesario que copies los fragmentos completos de código para responder una pregunta (simplemente puedes aclarar diciendo algo como "...código desde la línea 3 a la 6...", en ese caso numera las líneas.