Programación Orientada a Objetos. Primer Parcial 1er semestre 2010.
Viernes 16 de abril de 2010

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

```
public interface IViaLactea{
    void Mover();
    String Color {get;set;}

}
public interface ISistemaSolar: IViaLactea {
    String Composicion {get;set;}
    Long Masa {get;set;}
}
public interface IEstrella: IViaLactea {
    String Clasificacion {get;set;}
}
public interface IPlantea:ISistemaSolar{
     String TipoElongacion {get;set;}
}
public interface ICometa:ISistemaSolar{
     Int32 PeriodoOrbital {get;set;}
}
public interface IAsteroide:ISistemaSolar{
     String Tipo {get;set;}
}

public class Sol: IEstrella, ISistemaSolar {}
public class Tierra: IPlantea {}
public class Marte: IPlaneta {}
public class AsteroideTroyano: IAsteroide {}
public class Halley: ICometa {}

class Program
{
    static void Main(string[] args)
    {
        /1/ IViaLactea v = new Sol();
        /2/ IPlaneta p = new Tierra();
        /3/ ISistemaSolar s = new ICometa();
        /4/ Tierra t = p;
        /5/ IAsteroide b = new AsteroideTroyano();
        /6/ IViaLaceta u = b;
        /7/ IEstrella t = new Sol();
        /8/ ISistemaSolar i = t;
    }
  }
```

1.1 Indica las líneas de código incorrectas y justifica brevemente por qué son incorrectas.

2. Basándote **exclusivamente en las líneas correctas** del ejercicio anterior, responde las siguientes preguntas:
2.1 ¿Qué tipos tienen los objetos referenciados en estas variables?
2.2 ¿Qué mensajes puede recibir un objeto cómo el creado en la línea 2? ¿Cómo lo sabes?
2.3 ¿Qué mensajes puede recibir un objeto bajo los términos de la variable declarada en la línea 8? ¿Cómo lo sabes?

3. Sean las siguientes clases e interfaces:

```
interface Campeon {
   int Goles { get; set; }
}
class Uruguay : Campeon {
   private int goles;
   public int Goles { get { return goles; } set { goles = value; } }
}
class Wanderers : Campeon {
   private int goles;
   public int Goles { get { return goles; } set { goles = value; } }
}
```

Y es siguiente código:

```
Uruguay celeste = new Uruguay();
celeste.Goles = 3;
Campeon charrua = new Uruguay();
charrua = celeste;
charrua.Goles = 5;
Campeon bohemio = new Wanderers();
bohemio.Goles = 7;
Console.WriteLine(charrua.Goles + celeste.Goles + bohemio.Goles);
```

3.1 ¿Qué imprime por consola?
3.2 ¿Cuántos objetos iguales hay al final del código y en que líneas se crean? Si no encuentras objetos iguales realiza las modificaciones necesarias en el código para que los haya.
3.3 ¿Cuántas variables referencian a objetos idénticos al final del código? Si no encuentras variables que referencien a objetos idénticos realiza las modificaciones necesarias en el código para que las haya.
3.4 ¿Son los objetos instancia de Wanderers inmutables? Justifica. Realiza los cambios necesarios para que los objetos instancia de Wanderers sean inmutables, si no lo son, o mutables, si ya son inmutables.

4. Observa el siguiente código:

```
class Equipo {
   private int ganados;
   public int Ganados {
         get { return ganados; }
         set { ganados = value; }
   }
   private int empatados;
   public int Empatados {
         get { return empatados; }
         set { empatados = value; }
   }
}
class CalculadorPuntos {
   public int Calcular(Equipo t) { return t.Ganados * t.Empatados / 2;
}
```

4.1 Critica el código anterior en base al patrón Experto y, de ser necesario, vuelve a programar el código para que lo cumpla.
4.2 *"El patrón experto hace que mi programa sea menos frágil, ya que los cambios en una clase afectan menos que si no lo siguiera"*. Justifica esa afirmación y utiliza el código anterior para justificarla.

5. Sea el siguiente código:

```
class TraductorEspañol {
    string TraducirEspañol(string texto) { /* texto en español*/; }
}
class TraductorIngles {
    string TraducirIngles(string texto) { /* texto en inglés*/; }
}
class Parrafo {
    private string texto;
    string Traducir(string lenguaje) {
        if (lenguaje == "es") {
            return new TraductorEspañol().TraducirEspañol(texto);
        } else if (lenguaje == "en") {
            return new TraductorIngles().TraducirIngles(texto);
        } else {
            return "";
        }
    }
    void CorregirOrtografia() {
        /* complejo codigo que corrigue la ortografia del texto */
    }
}
```

Justifica si cumple los siguientes principios:

5.1 SRP

5.2 OCP

5.3 LSP

6. Programa el código nuevamente para que los principios se cumplan.

7. Sea el siguiente código:

```
class LavaVajilla {
   void Lavar() { /* lava lava */ }
}
class Horno {
    void Cocinar() { /* cocina cocina */ }
}
interface ICocinaMultifuncion {
   void CocinarComida();
   void LavarVajilla();
}
class Cocinero {
   void PrepararComida(ICocinaMultifuncion cocina) {
        cocina.CocinarComida();
        cocina.LavarVajilla();
    }
}
class Program {
    static void Main() {
        Horno horno = new Horno();
        LavaVajilla lavaVajilla = new LavaVajilla();
        Cocinero cocinero = new Cocinero();
    }
}
```

7.1 Haz que este cómodo `Cocinero` use un `Horno` y una `LavaVajilla` para poder cocinar. Intenta modificar la menor cantidad de clases posibles, pero puedes agregar tanto código como desees.