Programación Orientada a Objetos. Primer Parcial 2º semestre 2010.
Viernes 03 de setiembre de 2010

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

```
public interface IElectrodomestico{
    void Prender();
    String Color {get;set;}
}

public interface IHogar {
    String Lugar {get;set;}
}

public interface IMesa: IHogar {
    String Material {get;set;}
}

public interface ICocina{
}

public interface IEnfriador:IElectrodomestico{
    String Temperatura {get;set;}
}

public interface ISecado: IElectrodomestico {
    String Velocidad {get;set;}
}

public class Aspiradora: IHogar, IElectrodomestico {}
public class MesaRedonda: IMesa {}
public class MesaCuadrada: IMesa {}
public class Heladera: ICocina, IEnfriador {}
public class Balanza: IElectrodomestico {}

class Program
{
    static void Main(string[] args)
    {
        /1/ IHogar a = new Aspiradora();
        /2/ Aspiradora p = a;
        /3/ IMesa s = new IHogar();
        /4/ MesaRedonda mr = s;
        /5/ ICocina h = new Heladera();
        /6/ Heladera e = h;
        /7/ IEnfriador ie = new Heladera();
        /8/ IElectrodomestico el = ie;
    }
}
```

1.1 Indique las líneas de código incorrectas y justifique brevemente.

2.- Basándote exclusivamente en las líneas correctas del ejercicio anterior, responde las siguientes preguntas:

2.1 ¿Que tipos tienen los objetos referenciados en las variables definidas en las líneas correctas?
2.2 ¿Que mensajes puede recibir un objeto cómo el creado en la línea 1? ¿Cómo lo sabes?
2.3 ¿Que mensajes puede recibir un objeto bajo los términos de la variable declarada en la línea 7? ¿Cómo lo sabes?

3.- Sean las siguientes clases e interfaces:

```
interface IReposteria {
   String Nombre { get; set; }
}

class TortaDulce : IReposteria {
   private String Nombre;
   public String Nombre { get { return nombre; }
                          set { nombre = value;}}
}

class TortaSalada : IReposteria {
   private String Nombre;
   public String Nombre { get { return nombre; }
                          set { nombre = value;}}
}
```

Y el siguiente código:

```
IReposteria tortaAlfajor = new TortaDulce();
tortaAlfajor.Nombre = “Alfajor”;
IReposteria tortaZapallitos = new TortaSalada();
tortaZapallitos.Nombre = “Torta Zapallitos”;
IReposteria ramonNavarro = new TortaDulce();
ramonNavarro.Nombre = “Ramón Navarro”;
ramonNavarro = tortaAlfajor;
ramonNavarro.Nombre = “Diamanta”;
tortaZapallitos = “Diamanta”;

Console.WriteLine(“Ramon Navarro: ” + ramonNavarro.Nombre);
Console.WriteLine(“Alfajor: ” + tortaAlfajor.Nombre);
Console.WriteLine(“Zapallitos: ” + tortaZapallitos.Nombre);
```

3.1 ¿Qué imprime por consola?
3.2 ¿Existen objetos iguales al final del código? Indique cuáles y en qué líneas se crean. Si no encuentras objetos iguales realiza las modificaciones necesarias en el código para que los haya.
3.3 ¿Existen variables que referencien a objetos idénticos al final del código provisto? Indique cuáles son las variables. 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 TortaSalada inmutables? Justifica. Si no lo son, realiza los cambios necesarios para que sean.

4.- Observe el siguiente código:

```
public class CanalAnalogico{
    Encriptacion enc;
    public Encriptacion EngriptacionAnalogica()
    { return this.enc; }
    /* Otra implementación del canal Analógico */
}

public class CanalDigital{
    Encriptacion enc;
    public Encriptacion EngriptacionDigital()
    { return this.enc; }
    /* Otra implementación del canal Digital */
}
```

```
public class Canalera{
    ArrayList Canales;
    public void VerCanales()
    {
        Encriptacion enc;
        foreach (Object canal in Canales){
          if (canal is CanalAnalogico)  {
            enc = ((CanalAnalogico)canal).EngriptacionAnalogica();
          }
          else
          {
            enc = ((CanalDigital)canal).EngriptacionDigital();
          }
          Desencriptar(enc)
        }
    }
    public void Desencriptar(Encriptacion enc)
    { /* Algoritmo complejo de desencriptación */ }
}
```

Critique el código en base a:

4.1 OCP

4.2 LSP

4.3 Responde con **verdadero o falso** y **explica** brevemente:

a.- El SRP indica que las clases deben poseer una única operación.

b.- El patrón experto aumenta la dependencia entre clases.

c.- La solución a un problema aplicando el patrón experto, puede contradecir la solución del mismo problema al aplicar el SRP.

d.- Si el LSP no se cumple, entonces el OCP tampoco se cumple

5.- Modifica el código anterior para solucionar los problemas encontrados en 4.1 y 4.2.

6.- Sea el siguiente código:

```
public class Yen {
    Double Cambio() { /*compleja formula, propia del Yen*/}
}
public interface Moneda {
    Double ObtenerCambio();
}
public class CambiarMoneda {
    public void Cotizar(Moneda m) {
        /* Complejo código de inicialización del proceso.
        Console.WriteLine("El cambio del día de su moneda es:",
                          m.ObtenerCambio());
    }
}
```

6.1 Imprime el valor del cambio del YEN sin modificar el código anterior. Programa las líneas que precises para invocarlo desde el *program*.

7.- A esta altura del curso estarás maravillado de la potencia que provee la aplicación del polimorfismo de tipos. Para demostrarnos tu conocimiento te pedimos que respondas las siguientes preguntas:

7.1 ¿Qué entiendes por polimorfismo de tipos en Programación Orientada a Objetos?

7.2 Provee un breve fragmento de código en donde muestres la utilidad del mismo. Señala qué parte del código se comporta de manera polimórfica.

7.3 "Cuando una operación es polimórfica, el modo en que se ejecuta la misma se determina en tiempo de compilación." ¿Estás de acuerdo con esta afirmación? Explica brevemente tu respuesta.

7.4 Indica en base a que se determina el método a ejecutarse en una operación polimórfica.

8. Sea el siguiente código:

```
class Celular {
    private Chip3G chip3G;
    private Chip_GSM_UMTS chipGSM;
    private Chip_GSM_UMTS chipUMTS;
    private int id;
    private bool state;

    public void bloquear() { this.state=false; }
    public void desbloquear() { this.state = true; }
    public void addChip3G(Chip3G chip){
        if(chip!= null){
            if (chipGSM==null && chipUMTS==null){
                this.chip3G=chip;
            }
        }
    }
    public void addChipUMTS(Chip_GSM_UMTS chip){
        if(chip!= null){
            if (chip3G==null && chipGSM==null){
                this.chipUMTS=chip;
            }
        }
    }
    public void addChipGSM(Chip_GSM_UMTS chip){
        if(chip!= null){
            if (chipUMTS==null && chip3G==null){
                this.chipGSM=chip;
            }
        }
    }
}
```

```
class Chip_GSM_UMTS{
    private int identificador;
    private bool estado;
    private int numeroTelefono;

    public void activarChip(){
        this.estado = true;
    }
    public void CambiarNumero(int numero){
        this.numeroTelefono = numero;
    }
    public int ObtenerNumero(){
        return this.numeroTelefono;
    }
}
```

```
class Chip3G{
    private int id;
    private string estado;
    private int numeroTel;

    public void activarChip(){
       this.estado = "activado";
    }
    public void CambiarNumero(int numero){
       this.numeroTel = numero;
    }
    public int ObtenerNumero(){
       return this.numeroTel;
    }
}
```

El código presentado previamente es parte del sistema de gestión de celulares de una famosa empresa que brinda servicios de telefonía celular en nuestro país. La misma brinda la posibilidad que sus abonados tengan chips 3G, GSM y UMTS, no obstante cada celular puede **tener solo un chip asociado**. Se sabe que a corto y mediano plazo la empresa va a poner en plaza:

- Soporte a Chips con otras tecnologías: 4G, GSP, etc. A futuro se van a agregar soporte a nuevas tecnologías.
- Venta de celulares que soportan varios chips integrados, incluso varios del mismo tipo.

8.1 Realice modificaciones al código planteado desde el punto de vista de POO para que el sistema soporte los nuevos requerimientos.
8.2 Lista los principios que crees haber aplicado en tu solución.