Programación orientada a objetos. Primer parcial 2º semestre 2007.

Viernes 7 de setiembre de 2007.

Al igual que las demás instancias de evaluación este examen será calificado de acuerdo a la escala de letras vigente: D, R, B, MB, BMB, S o D, R, B, MB, S según la generación de cada alumno. Por eso las preguntas no tienen puntos.

Las respuestas deben aparecer en el orden en que están formuladas las preguntas.
Escriban sólo del anverso de la hoja.
Numeren las hojas.
Gracias y mucha suerte.

Sean las siguientes clases e interfaces y el código que las usa:

```
public interface Mortal {
        void Comer();
        String Nombre { get; }
}
interface Griego : Mortal {
}
interface Fenicio : Mortal {
   void Navegar();
}
class Filosofo : Griego {
   void Filosofar() { /*...*/ } /*...*/
}
class Ciudadano : Griego {
   void Votar() { /*...*/ } /*...*/
}
class Program {
   static void Main(string[] args) {
      /*1*/ Filosofo socrates = new Filosofo();
      /*2*/ Mortal s1 = socrates;
      /*3*/ Griego s2 = s1;
      /*4*/ Ciudadano aristofanes = new Fenicio();
      /*5*/ Fenicio dagon = new Mortal();
      /*6*/ Griego g2 = aristofanes;
      /*7*/ Griego g3 = dagon;
      /*8*/ Ciudadano g4 = new Filosofo();
   }
}
```

1.1 Indica que líneas no compilan y justifica por qué. Deben validar línea por línea. Si una asignación es incorrecta, deben asumir que la variable de todas formas fue definida.

1.2 ¿Qué mensajes puedo enviar a las variables definidas en las líneas correctas? ¿Cómo lo sabes?

Tomando en cuenta las interfaces y clases definidas en el ejercicio 1

2.1 ¿Qué tipos tiene un objeto instancia de Ciudadano?

2.2 Un objeto referenciado por una variable de tipo Griego ¿Puede poseer una implementación de void Filosofar()? En caso afirmativo, provee un ejemplo de ésto, en caso negativo, explica por qué.

Tomando en cuenta las interfaces y clases definidas en el ejercicio 1 y el siguiente código:

```
/*1*/ Filosofo socrates = new Filosofo();
/*2*/ socrates.Nombre = "Socrates";
/*3*/ Griego zenon = new Filosofo();
/*4*/ zenon = socrates;
/*5*/ Mortal heraclito = new Filosofo();
/*6*/ heraclito.Nombre = "Socrates";
```

Suponiendo que al crear un nuevo Filosofo su nombre se crea en nulo, al final de este código,

3.1 ¿Qué variables referencian al mismo objeto?

3.2 ¿Qué variables referencian a objetos iguales?

3.3 ¿Qué línea debes eliminar para que queden dos objetos iguales y uno distinto?

4. Desde el punto de vista de un programa orientado a objetos, ¿es correcto que un objeto de un programa no sea, en ningún caso, emisor de mensajes? ¿y que no sea en ningún caso receptor?. Justifica.

5. ¿Qué te permiten hacer las interfaces que no te permiten hacer las clases? ¿Que te permiten hacer las clases que no te permiten hacer las interfaces?.

Lee el siguiente código

```
class Vehiculo {
   public Int32 cantidadDeRuedas;
   public Vehiculo(Int32 cantidadDeRuedas) {
      this.cantidadDeRuedas = cantidadDeRuedas;
   }
}
class ImpresorDeVehiculo {
   public void ImprimirNombre(Vehiculo vehiculo) {
      if (vehiculo.cantidadDeRuedas == 2)
         Console.WriteLine("Moto");
      if (vehiculo.cantidadDeRuedas == 3)
         Console.WriteLine("Triciclo");
      else if (vehiculo.cantidadDeRuedas == 4)
         Console.WriteLine("Auto");
   }
}
```

Analiza muy brevemente el código anterior de acuerdo a:

6.1 Encapsulación

6.2 Patrón Experto

6.3 SRP

6.4 LSP

7. Vuelve a programar el programa anterior basandote en el patrón Experto y OCP. Aprovecha la flexibilidad que te provee el polimorfismo de tipos e indica donde lo utilizas.

Sea el siguiente programa:

```
class Impresora {
   public void Imprimir(Caracter caracter) {
      ImprimirPorPantalla(caracter.GetForma());
   }
   private void ImprimirPorPantalla(Forma forma) {
      // implementación altamente compleja!
   }
}
class Caracter {
   public Forma GetForma() {
      // implementación increiblemente compleja!
   }
}
class Circulo {
   private Forma forma;
   public Circulo() {
      this.forma = CrearNuevaForma();
   }
   private Forma CrearNuevaForma() {
      // implementación terriblemente compleja y costosa!
   }
}
class Forma {
   // código y más código
}
```

8. Programa lo necesario para que Impresora pueda imprimir instancias de Circulo guiándote por el OCP (Principio Abierto/Cerrado). Justifica por qué tu programa aplica correctamente el OCP.