Programación Orientada a Objetos

Segundo Parcial 2do semestre 2011 - Miércoles 26 de octubre de 2011

(a)

1.- Sea el siguiente código:

```
class Libro {
    private String nombre;
    private String precio;
}
class Librería {
    private IList libros;
    private IList empleados;
    private IList usuarios;
    private Double facturacion;
    public void RegistrarLibro(Libro libro)
    {    /* Se registra el libro en la Librería. */}
    public void RegistrarUsuario(Usuario usuario)
    {    /* Se registra el usuario en la Librería. */ }
    public void RegistrarEmpleado(Empleado empleado)
    {    /* Se registra el empleado en la Librería. */}
    public void PagarEmpleado(Empleado empleado, Double sueldo)
    {    /* Se realiza el pago del sueldo al Empleado. */}
    public void ReservarLibro(Libro libro, Usuario usuario)
    {    /* Realiza la reserva de un libro. */ }
    public void DevolverReserva(Libro libro, Usuario usuario)
    {    /* Devuelve la reserva del libro. */}
}
class Persona {
    private String nombre;
    private String apellido;
    private String nroDocumento;
    private String tipoDocumento;
}
class Empleado : Persona {
    private Double sueldo;
    private Double descuento;
}
class Usuario : Persona {/*imp*/}
```

Y las siguientes condiciones:

a) Un usuario se puede registrar una sola vez en la librería con el mismo documento (igual número documento e igual tipo documento).

b) En la biblioteca hay varios libros con el mismo nombre, por lo que al momento de reservar un libro tiene que haber stock disponible.

c) Un usuario devuelve un libro que tenía reservado.

d) La reserva de un libro implica que el stock de ese libro se disminuye en uno.

e) El sueldo del empleado es menor o igual al 10% de la facturación.

f) En la librería hay más de 33 libros.

g) Cuando se registra un libro en la librería, aumenta la cantidad de libros existentes y el stock correspondiente a ese libro.

1.1) Indique qué condiciones son precondiciones, postcondiciones o invariantes.

1.2) ¿Existe alguna relación donde la definición de contratos interfiera con el principio de sustitución de Liskov (LSP)? Justifica.

1.3) Programa nuevamente las clases del ejercicio anterior para que cumplan estas condiciones, modificándolas pero respetando su estructura actual utilizando Debug.Assert().

2.- Tomando como base el código provisto en el ejercicio 1, realiza los cambios necesarios para cumplir con lo que se pide a continuación.

Imagina que la librería tiene la responsabilidad de desplegar todos los usuarios y empleados de una manera uniforme. Para ello, mediante un único método *Desplegar*, se querrá mostrar para los usuarios todos sus datos personales (nombre, apellido, nro doc, tipo doc) y exactamente lo mismo para los empleados, pero agregando su sueldo y descuento. Implemente dicho método, y provee un pequeño fragmento de código que ejemplifique su uso.

3.- Sea el siguiente código:

```
abstract class Animal {
    public void Comer() {/*imp*/}
    abstract bool Mover();
}
interface IMamifero {
    void Amamantar();
}
class Gato : Animal, IMamifero {
    public void Amamantar() { /*imp*/}
    public override bool Mover() { return true; }
    public virtual void Trepar() {/*imp*/}
}
class GatoMontes : Gato {
    public void Pelear() { /*imp*/}
}
class Program {
    static void Main(string[] args){
        Animal a = new GatoMontes();
        GatoMontes b = new GatoMontes();
        Gato c = b;
        IMamifero d = new Gato();
        IMamifero e = (Gato) c;
    /1/ a.Comer();
    /2/ d.Amamantar();
    /3/ c.Trepar();
    /4/ b.Pelear();
    /5/ e.Amamantar();
    }
}
```

3.1) Indica el tipo de encadenamiento de las líneas /1/ a /5/.

4.- Para cada una de las siguientes preguntas, marca la opción más correcta:

4.1) Las siguientes sentencias son válidas tanto para clases abstractas como interfaces. Excepto por una de ellas:

a) No se pueden instanciar

b) Pueden poseer un constructor

c) Definen relaciones de generalización-especialización

d) Deben ser públicas

4.2) El mecanismo de reutilización de herencia, define relaciones más fuertes que el mecanismo de re-utilización de composición y delegación:

a) Verdadero

b) Falso

4.3) Sea el siguiente código:

i)	ii)

abstract class Poligono{
void M1() {}
void M2() {}
}
	class Poligono{
	abstract void M1();
	void M2() {}
	}

a) i e ii son correctas

b) i es correcta e ii es incorrecta

c) ii es correcta e i es incorrecta

d) ni i ni ii son correctas

4.4) La relación entre la cohesión y el acoplamiento se espera que sea:

a) Baja cohesión y alto acoplamiento

b) Alta cohesión y bajo acoplamiento

c) Baja cohesión y bajo acoplamiento
d) Alta cohesión y alto acoplamiento

4.5) Un alto grado de dependencias entre módulos es:
a) deseable
b) indeseable
c) necesario
d) inevitable

4.6) La composición y delegación es un mecanismo de re-utilización selectivo de código porque:
a) igual que la herencia se re-utiliza todo o nada
b) encapsula selectivamente lo que va a re-utilizar
c) lo que se re-utiliza se define en tiempo de ejecución
d) se debe elegir qué componente formará parte del objeto compuesto

4.7) "El encadenamiento dinámico:" es todo lo listado a continuación, excepto:
a) Se da en tiempo de ejecución
b) Habilita el polimorfismo
c) Se determina por el tipo de la variable
d) Es propio de los lenguajes sin tipos

4.8) De los siguientes casos, el diseño por contratos es más aplicable a:
a) Darle una especificación a un programa
a) Manejar excepciones del sistema (Ej.: Disco lleno)
b) Manejar validaciones de datos para el usuario
c) Para realizar casos de testeo con aserciones.

4.9) El culpable de que se viole una pre-condición, es:
a) El emisor y receptor del mensaje en su conjunto.
b) El receptor del mensaje.
c) Ni el emisor ni el receptor del mensaje, se debe a una falla externa
d) ninguno de los anteriores

4.10) La siguiente definición dentro de una clase abstracta: *sealed abstract void M1();* ¿tiene sentido?:
a) Verdadero
b) Falso

5.- Sea el siguiente código:

```
class Reproductor {
    VHSCodec codec;
    public void ReproducirVHS(VHS vhs)
    {
        Reproducir(this.codec.Decode(vhs));
    }
    public void Reproducir(VideoStream video)
    {
        /* Complejo código que reproduce un video */
    }
}
class VHSCodec {
    public VideoStream Decode(VHS video);
}
class VideoStream {}
class VHS {}
```

Este reproductor solamente reproduce VHS (una antigua tecnología utilizada en películas clásicas como Rocky I y II).
Como ya es muy difícil encontrar películas en VHS, sería bueno que el reproductor también pudiese decodificar otros medios como ser DVD, BluRay, Laser Disc y otros futuros medios que puedan surgir.
Haz todos los cambios que consideres necesarios para que este reproductor cumpla con esto, y pueda reproducir estos nuevos tipos.

6.- Todas las personas tienen un conjunto de características que se comparten. Además tienen por lo menos un rol, pudiendo tener varios. Por ejemplo hay estudiantes, hay doctores y hay profesores. A la vez cada persona podría tener varios roles ya que por ejemplo una persona podría ser un estudiante y un profesor al mismo tiempo. Se proveen las siguientes clases e interfaces como posibles elementos de la solución, pero no tienes por qué utilizarlos:

```
class Persona {}
class Estudiante {}
class Profesor {}
interface IRol {}
class Rol {}
```

6.1) Brinda una solución a este problema por medio de la composición y delegación
6.2) Brinda una solución a este problema utilizando herencia
6.3) Indica cuál de las soluciones te parece más apropiada y explica brevemente las razones.

7.- Sea el código del ejercicio 1, con algunos cambios:

```
class Libro
{
    private String nombre;
    private String precio;
    private DateTime fecha;
    public bool EsVigente(){/*imp*/}
    public bool EsDonado(){/*imp*/}
}
class Clasificador
{
    private Libro libro;
    public bool LibroRegistrable()
    {
       if (libro.EsVigente()) {
           return true;
       }
       else {
            return false;
       }
    }
    public bool LibroPrestable() {
       if (libro.EsDonado())
       {
           Console.WriteLine("Lectura solo en sala");
           return false;
       }
       else {
            return true;
       }
    }
    public Double VenderLibro(Empleado empleado){
         // Devuelve el precio del libro menos el descuento del empleado
    }
}
```

7.1) Indica si el código del Clasificador cumple con el ISP
7.2) Indica si el código del Clasificador cumple con el SRP
* Recuerda que solo puedes evaluar lo que está escrito
7.3) Describe los cambios necesarios para que ambos principios se cumplan. Si te resulta más claro, programa la solución a los problemas que hayas encontrado.