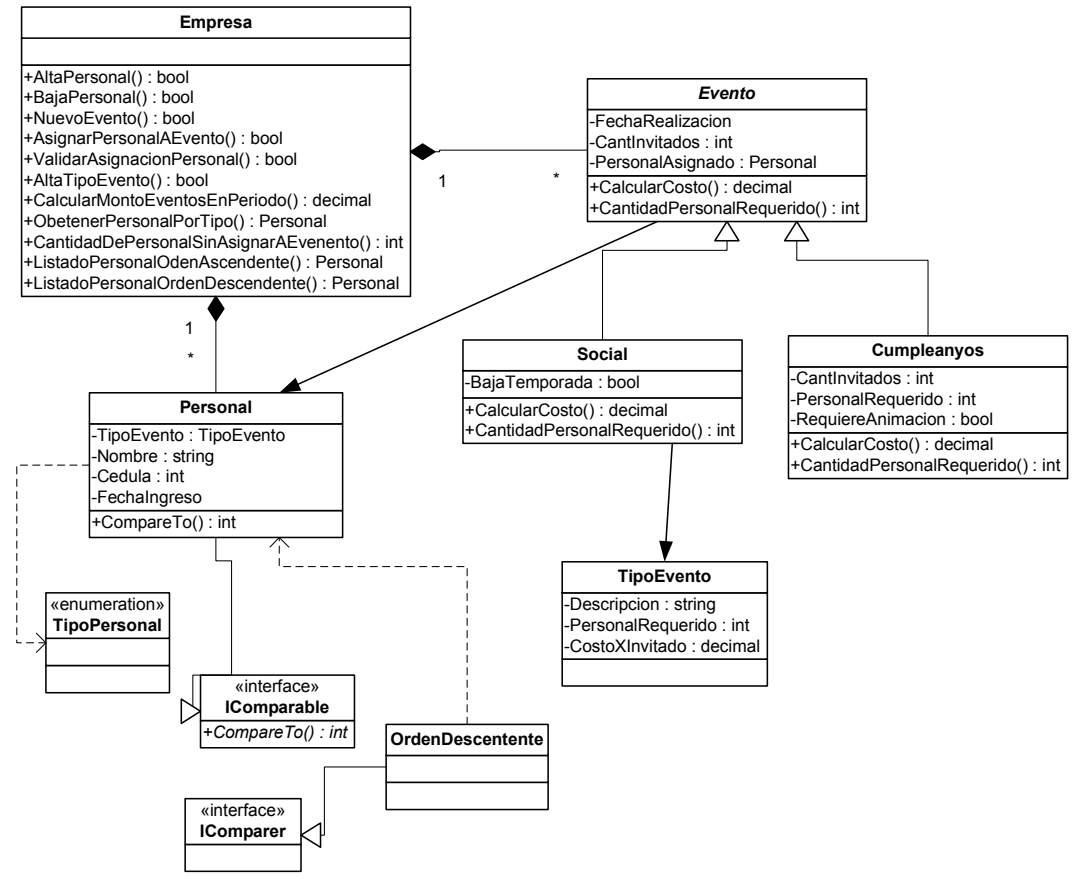
EVALUACIÓN	SOLUCIÓN	GRUPO	TODOS	FECHA	07/08/12
MATERIA	Programación 2 – Plan 2011				
CARRERA	AP – ATI - APW				
CONDICIONES	- Puntos: 100 - Duración 3 hs – incluyendo lectura de la letra. - Sin material - Indicar el nombre del docente en la primera hoja. - Dudas exclusivamente de la letra o sintaxis no trivial de VB.NET. - No escribir en la hoja de la letra. - No entregar la hoja de la letra. - Numerar las hojas				

NOTA: Se presenta una idea de solución no detallada

1 – Diagrama UML.



Adicionalmente el costo unitario del personal se almacena en la Clase Evento como atributo de clase (CostoUnitarioPersonal: int)

2 – e

```
//En clase evento
Public Abstract Class Evento
{
        ...
       Public Abstract Decimal CalcularCosto();
}

//En clase social
Public Class Social: Evento
{      …
       Public overrides Decimal CalcularCosto()
       {
              Decimal costo = 0;
              costo = this.CostoDelSalon + (this.CantInvitados * this.tipoEvento.CostoXInvitado)
       +
              (this.PersonalAsignado.Count * Evento.CostoUnitarioPersonal);
              If(this.BajaTemporada)
              {
                     Costo -= costo*0.1;
              }
              return costo;
       }
}

//En clase cumpleanios
Public Class Cumpleanyos: Evento
{      …
       Public overrides Decimal CalcularCosto()
       {
              Decimal costo = 0;
              costo = this.CostoDelSalon + (this.CantInvitados * this.CostoXInvitado) +
              (this.PersonalAsignado.Count * Evento.CostoUnitarioPersonal);
              If(this.RequiereAnimacion)
              {
                     Costo+= costo*0.15;
              }
              return costo;
       }
}
```

```
//En clase Empresa
Public class Empresa
{
	…
	Public Decimal CalcularMontoEventosPeriodoTiempo (
			DateTime pFechaIni, DateTime pFechaFin)
	{
		Decimal total = 0;
		For each(Evento evento in this.ListaEventos)
		{
			If(evento.FechaRealizacion >= pFechaIni &&
			evento.FechaRealizacion <= pFechaFin)
			{
				Total+= evento.CalcularCosto();
			}

		}
		return total;
	}
}
```

2 – g.

```
//En clase Personal
Public class Personal
{
	…
	Public overrides bool Equals(object obj)
	{
		If (obj is Personal)
		{
			return (this.Cedula == ((Personal) obj).Cedula);
		}
		return false;
	}
}
```

```
//En clase Empresa
Public Class Empresa
{
	…
	Public int CalcularCantidadDePersonalSinAsginar(
			DateTime pFechaIni, DateTime pFechaFin)
	{
		int cantidad = 0;

		List<Evento> listaAuxEventos = new List<Eventos>();

		For each(Evento evento in this.ListaEventos)
		{
			If(evento.FechaRealizacion >= pFechaIni &&
			evento.FechaRealizacion <= pFechaFin)
			{
				listaAuxEventos.Add(evento);
			}

		}

		For each(Personal personal in this.ListaPersonal)
		{
			Bool encontre = false;

			For each(Evento evento in listaAuxEventos)
			{
				If(evento.PersonalAsignado.Contains(personal))
					encontre = true;
			}

			If(!encontre)
				cantidad ++;
		}

		return cantidad;

	}
}
```

2 – h

```
//para ordenar en Personal

public Class Personal: IComparable<Personal>
{

        public int CompareTo(Personal other)
        {
                return this.Nombre.CompareTo(other.Nombre);
        }



}

public class OrdenarPersonalPorNombreDesc: IComparer<Personal>
        {
                public int Compare( Personal x, Personal y)
                {
                        return (-1) * ( x.CompareTo(y) );
                }
        }



//Clase Empresa
Public class Empresa
{
        …


        public List<Personal> OrdenarListaPersonalAlfabetica ()
        {
                List <Personal> resultado = this.ListaPersonal;

                return resultado.Sort();
        }

        public List<Personal> OrdenarListaPersonalAlfabeticaDescendente()
        {
                List <Personal> resultado = this.ListaPersonal;

                return resultado.Sort( new OrdenarPersonalPorNombreDesc() );
        }
}
```