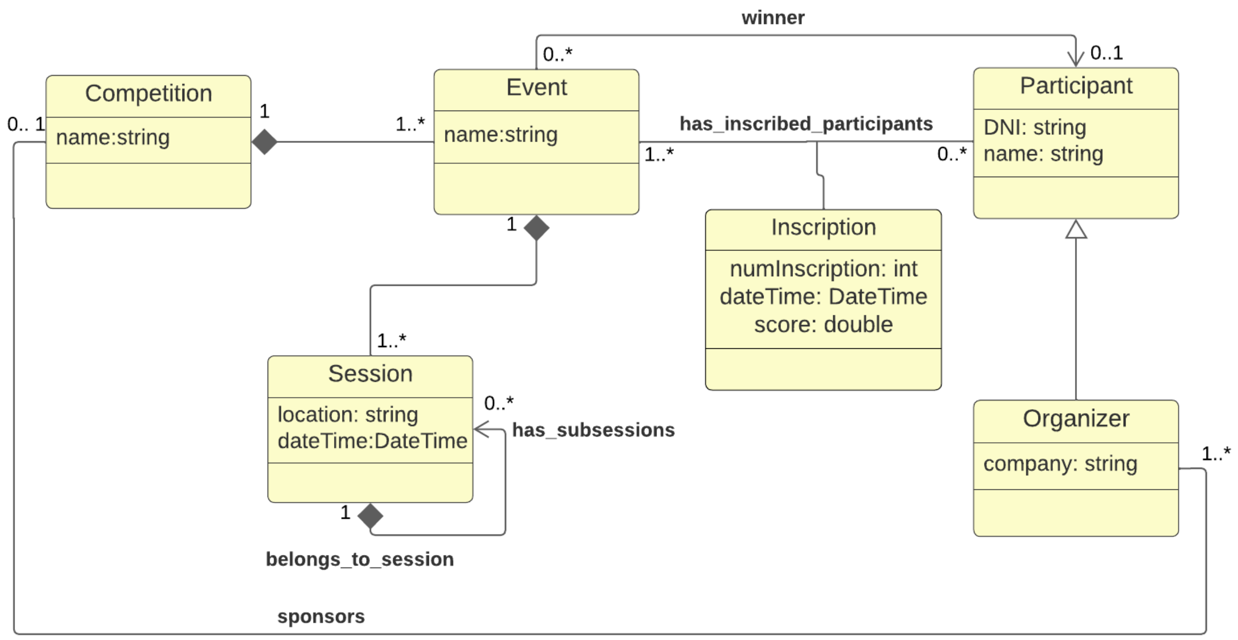
Ingeniería del Software (ISW)

Evaluación Teoría

19-01-2023

ETSInf-UPV

5. *(2 points)* Given the following UML class diagram:



Note: There is a navigation restriction between Event and Participant and between Session and Session.

a) *(1 point)* Obtain the C# design following the design patterns studied (do not add any class methods).

b) *(0.5 points)* Obtain the C# design of the constructor with parameters for each class (just the header, do not implement them. Do not define the constructor without parameters).

c) *(0.5 points)* Using the constructors from the previous question implement the necessary code to create a system consistent with the diagram for the following requirement: The “VLC Testing” competition is registered with just one event “Hakckaton JUnit” and two inscribed participants. One of them is both organizer and sponsor of the competition. Use any other arbitrary values for the remaining attributes.

Ingeniería del Software (ISW)

Evaluación Teoría
19-01-2023
ETSInf-UPV

a) Solution comes from the Spanish version of the model. Properties:

```
public class Competicion
{
    public string nombre {get; set;}

    //Relaciones
    public virtual ICollection<Evento> eventos {get; set;}
    public virtual ICollection<Promotor> promotores {get; set;}
}

public class Evento
{
    public string nombre {get; set;}

    //Relaciones
    public virtual Competicion competicion {get; set;}
    public virtual ICollection<Sesion> sesiones {get; set;}
    public virtual ICollection<Inscripcion> inscripciones {get; set;}
     public virtual Participante ganador {get; set;}
}

public class Inscripcion
{
    public int numIndcripcion {get; set;}
    public DateTime fechaHora {get; set;}
    public double puntuacion {get; set;}

    //Relaciones
    public virtual Evento evento {get; set;}
    public virtual Participante participante {get; set;}
}

public class Sesion
{
    public string lugar {get; set;}
    public DateTime fechaHora {get; set;}

    //Relaciones
    public virtual Evento evento {get; set;}
    public virtual ICollection<Sesion> tiene_subsesiones {get; set;}
}

public class Participante
{
    public string DNI {get; set;}
    public string nombre {get; set;}

    //Relaciones
    public virtual ICollection<Inscripcion> inscripciones {get; set;}
}

public class Promotor : Participante
{
    public string empresa {get; set;}

    //Relaciones
    public virtual Competicion competicion {get; set;}
}
```

b) *Constructors*

```
public Evento(string nombre)  //Se relaja la cardinalidad con Competición y con
                              // y con Sesión
{    }


public Competicion(string nombre, Promotor promotor, Evento evento)
{    }


public Sesion (string lugar, DateTime fechaHora, Evento evento)
{    }



public Inscripcion(int numIndcripcion, DateTime fechaHora, double puntuacion,
                   Evento evento, Participante participant)
{    }


public Participante(string DNI, string nombre)
{    }


public Promotor(string DNI, string nombre, string empresa):base(DNI, nombre)
{    }
```

c) *Initialization*

```
Evento HackJUnit = new Evento("Hackaton JUnit");

Promotor ObjPromotor1 = new Promotor("20123456", "Maria", "HP");

Competicion VCLTesting= new Competicion("VCL Testing", ObjPromotor1, HackJUnit);
HackJUnit.Competicion =VCLTesting;
ObjPromotor1.Competicion=VCLTesteng;

Sesion ObjSesion1 = new Sesion("UPV", DateTime.Now, HackJUnit);
HackJUnit.sesiones.Add(ObjSesion1);


Participante ObjPart1 = new Participante("19123456", "Álex");


Inscripcion ObjInscrip1 = new Inscripcion (1, DateTime.Now, 0, HackJUnit, ObjPromotor1);
HackJUnit.inscripciones.Add(ObjInscrip1);
ObjPromotor1.inscripciones.Add(ObjInscrip1);


Inscripcion ObjInscrip2 = new Inscripcion(2, DateTime.Now, 0, HackJUnit, ObjPart1);
HackJUnit.inscripciones.Add(ObjInscrip2);
ObjPart1.inscripciones.Add(ObjInscrip2);
```