Teoría ISW

Evaluación Teoría

19-01-2023

3h 30 min

Normas a seguir:

- No abrir el enunciado hasta que lo indique el/la profesor/a del aula.
- Se debe contestar a las preguntas en las hojas proporcionadas (no se darán hojas extra). Solo se podrá arrancar la hoja del enunciado.
- Entrega:
  - El/la alumno/a se debe identificar ante el profesor en el momento de la entrega.
  - El enunciado y las respuestas se depositarán en una caja habilitada para ello.
  - El procedimiento de entrega se realizará de forma ordenada. En cada momento solo podrá estar un/a alumno/a entregando.

Ingeniería del Software

ETS Ingeniería Informática

DSIC – UPV

Curso 2022-2023

Ingeniería del Software (ISW)

Evaluación Teoría
19-01-2023
ETSInf-UPV

NOMBRE:
GRUPO:

Tiempo: 3 horas 30 minutos

Cuestiones *(2 puntos)*

1. (0.5 *puntos*) Formas parte de un equipo de desarrollo de software que debe iniciar un proyecto software. ¿Qué aspectos deberías tener en cuenta para elegir la metodología de desarrollo a seguir? Razone al menos dos aspectos.

Respuesta: El equipo de desarrollo podría elegir entre una metodología ágil o una metodología tradicional. Entre los aspectos a tener en cuenta estarían:

// aquí valdría bien justificado cualquier aspecto que se menciona en la comparativa entre metodologías, y que se justifique adecuadamente, por ejemplo

- Si el producto a desarrollar se prevé que tiene requisitos cambiantes o muy cambiantes, mejor una metodología ágil que es más tolerante a los cambios.
- Si hay que hacer entregas frecuentes al cliente y en periodos cortos de tiempo, por ejemplo cada 3-4 semanas, mejor una metodología ágil que establece esta organización en sprints, frente a iteraciones más extensas en el tiempo de las metodologías tradicionales…
- Si el cliente quiere participar activamente en el proyecto y formar parte del equipo, mejor una metodología ágil, puesto que en la tradicional sólo habría reuniones puntuales en ciertos hitos.
- Negociación / Contrato …… etc….. (habría muchas posibles respuestas válidas…)

2. (*0.5 puntos*) ¿Es *Entity Framework* un claro ejemplo de patrón de acceso a datos (DAO)? Razone la respuesta.

Respuesta: No, puesto que se corresponde con el patrón de repositorio + unidad de trabajo (+ razonamiento)

3. (*1 punto*) Un programador poco experimentado ha implementado una funcionalidad para mostrar en un cuadro de lista (combo box) los DNIs de los clientes que están sancionados con una multa. Para ello ha implementado el siguiente código (asuma que el código compila y se ejecuta correctamente). ¿Qué problema de diseño tiene su solución y qué desventajas presenta? ¿Cómo debería hacerlo, no es necesario que reescriba el código sólo explique la solución alternativa?

```
public void LoadData(){

SqlConnection conn = new SqlConnection("Data Source=computer_name;" +
"Initial Catalog=simplesql;" + "User ID=sa;" + "Password=pass;");
```

```
SqlDataReader dr;

SqlCommand cmd = new SqlCommand();
cmd.CommandTimeout = 60;
cmd.Connection = conn;
cmd.CommandType = CommandType.Text;
cmd.CommandText = "SELECT Dni FROM Customers WHERE fined=1";

conn.Open();

customersComboBox.Items.Clear();
if (conn.State == ConnectionState.Open) {
      dr = cmd.ExecuteReader();
      while (dr.Read())
       customersComboBox.Items.Add(dr[0]);
}
customersComboBox.SelectedIndex = -1;
customersComboBox.ResetText();
dr.Close();

conn.Close();
}
```

Respuesta: El código implementado presenta un serio problema de diseño. No hay una separación entre capas dado que la persistencia, la lógica de negocio y la presentación están mezcladas. Esto provoca que si cambia el modelo (p.ej. cambia la forma en la que se multa a los clientes) o si se quiere cambiar la tecnología de persistencia (p.ej. cambiar de una base de datos relacional a una persistencia como un servicio en la nube) habría que modificar este código por lo que la mantenibilidad del software empeora.

Para solucionar este problema deberíamos tener una clara separación entre capas mediante una arquitectura multicapa cerrada: Presentación, Lógica de Negocio y Persistencia con interfaces bien definidas. La capa de negocio tendría un servicio que nos proporcione la información necesaria y este servicio de negocio haría uso de un DAL para el acceso a la persistencia.

Ingeniería del Software (ISW)

Evaluación Teoría
19-01-2023
ETSInf-UPV

Problemas *(8 puntos)*

4. *(4.5 puntos)* A partir del siguiente enunciado se pide obtener:

    a) (2.5 puntos) El diagrama de clases en UML que modela el sistema propuesto. En el diagrama de clases no hay que incluir los métodos ni los tipos de los atributos.

    b) (2 puntos) El diagrama estructurado de casos de uso en UML que modela el sistema propuesto.

ISWSoft ha sido contratada por el Hospital Princeton-Plainsboro para desarrollar MyApp, un sistema software para la gestión de enfermedades raras que afectan principalmente a niños en edad temprana. Aunque la sintomatología puede afectar al desarrollo físico esto se puede evitar mediante un riguroso control de los alimentos ingeridos. Para ello, cada equipo compuesto por médicos de diferentes especialidades hará el seguimiento de la evolución de los pacientes que tengan asignados a través del nuevo software. De cada médico estará disponible, su nombre, la especialidad a la que pertenece y si es el director del equipo. Un médico puede pertenecer a varios equipos y ser director de varios equipos. MyApp también les permitirá a los médicos definir objetivos nutricionales, los cuales delimitarán el porcentaje de macronutrientes (carbohidratos, proteínas y lípidos) que cada paciente debe ingerir por día. Estos objetivos, junto con el peso, altura y edad de los pacientes, servirán para que MyApp pueda calcular menús personalizados diarios para cada uno. Los menús de cada paciente para toda la semana se calculan automáticamente al inicio de cada semana. De cada paciente infantil habrá que indicar su nombre, asimismo tendrá un responsable que será el encargado de utilizar MyApp y del cual también habrá que indicar su nombre.

Cada menú personalizado incluirá la fecha de cuándo se espera que el paciente lo consuma. El menú puede ser de un desayuno, una comida o una cena y tendrá asociada una única receta. En la receta se especificará el número de comensales para los que está calculada, así como un texto con las instrucciones de preparación y la lista de ingredientes y cantidades de la que está compuesta. De los ingredientes se dispondrá del nombre, y la composición nutricional de cada uno (carbohidratos, proteínas y lípidos) por cada 100 gr. Una vez el menú esté disponible, los responsables del paciente infantil pueden consultarlo y reportar el porcentaje de comida que finalmente han consumido sus hijos. En el caso de que un menú no fuera del agrado del niño, también podrá ser modificado por un médico del equipo sustituyendo la receta por alguna de las ya existentes o creando una nueva en MyApp a partir de los ingredientes disponibles. De igual manera, los responsables del paciente infantil reportarán diariamente los síntomas de sus hijos, indicando el tipo de síntoma, intensidad y fecha.

Los especialistas médicos dispondrán de un usuario y contraseña para acceder al sistema. En la primera visita, cualquiera de los componentes del equipo médico podrá registrar al paciente, proporcionando a los responsables del paciente infantil las credenciales necesarias (login y contraseña) para entrar al sistema. Cualquier médico podrá consultar la lista de pacientes ya registrados, y si el paciente no existiera, entonces procedería a introducirlo en el sistema. Si, por el contrario, el paciente ya existiera, el médico en cuestión podría consultar la ficha del paciente, que mostrará la ingesta de alimentos y síntomas reportados por los responsables del paciente infantil y desde donde también se podrán definir los objetivos nutricionales para ese paciente.

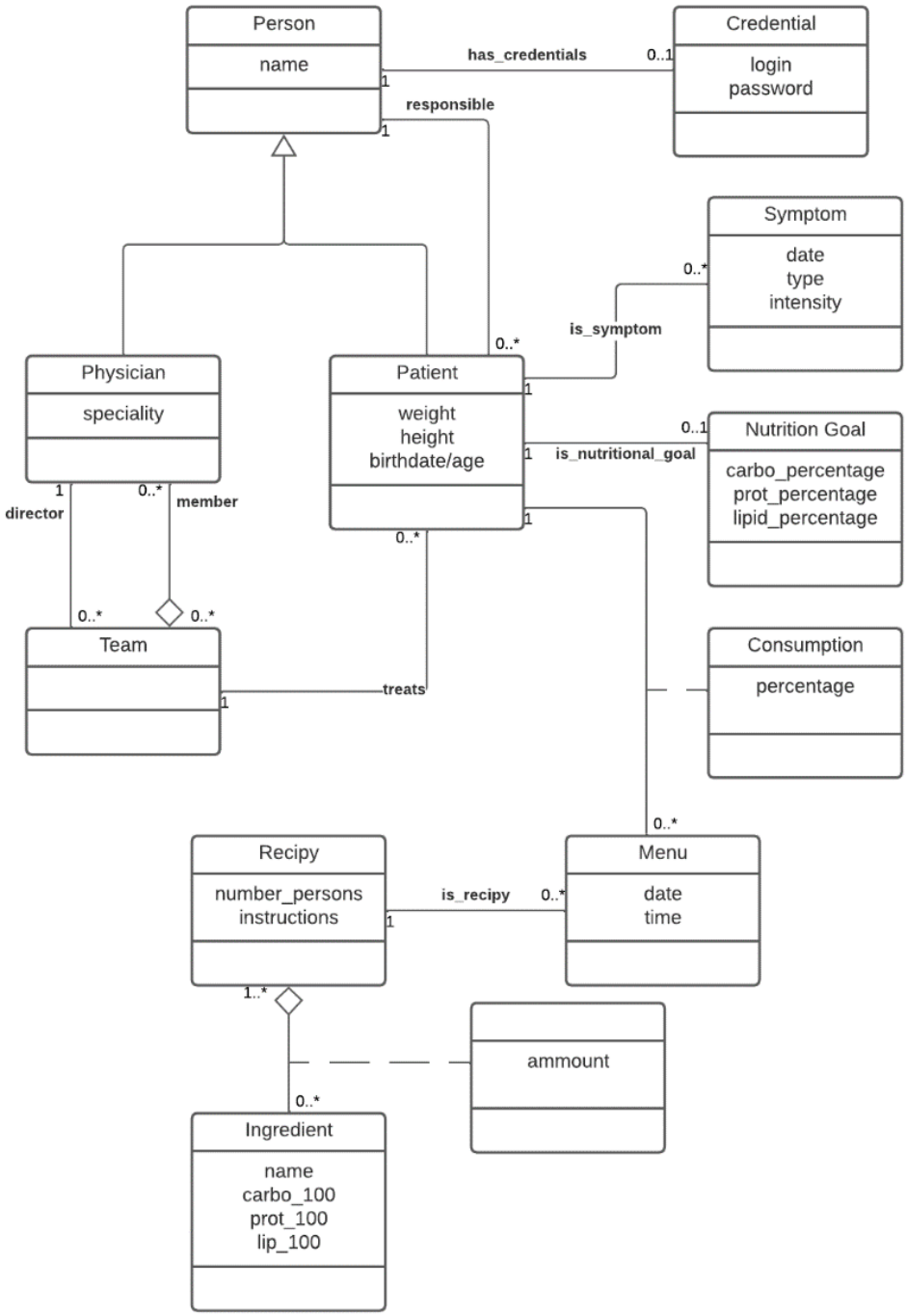
Ingeniería del Software (ISW)

Evaluación Teoría

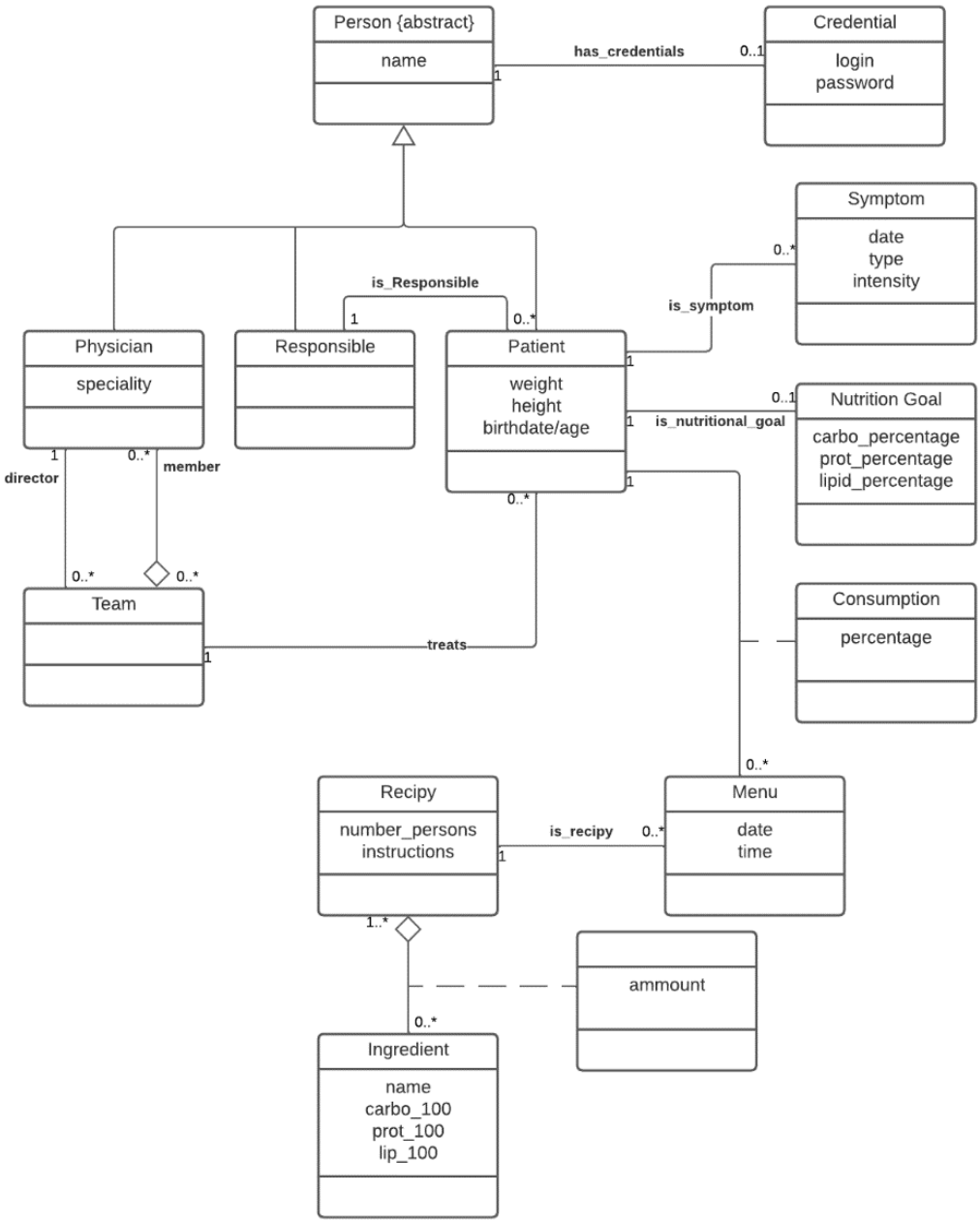
19-01-2023

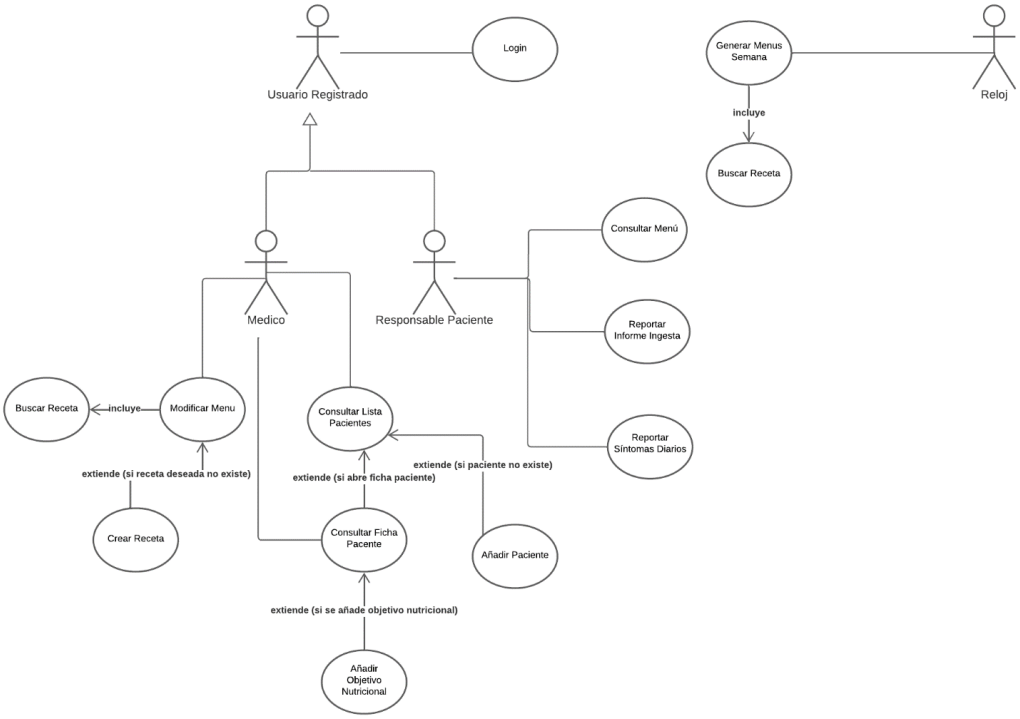
ETSInf-UPV

SOLUCION DC: Esta solución requiere de ciertas restricciones de integridad para evitar que un Paciente o un Médico sean responsables de otro paciente



SOLUCION ALTERNATIVA DC: Alternativamente sería válido sin necesidad de tener restricciones de integridad el siguiente modelo

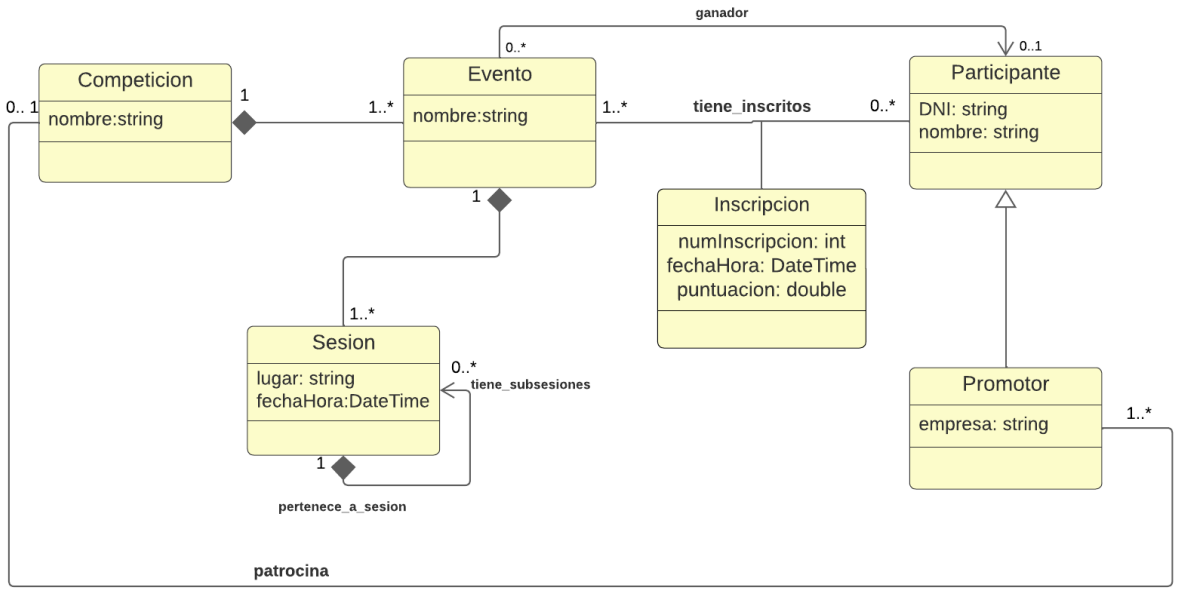



Usuario Registrado
Login
Generar Menus Semana
Reloj
incluye
Buscar Receta
Consultar Menú
Medico
Responsable Paciente
Reportar Informe Ingesta
Buscar Receta
incluye
Modificar Menu
Consultar Lista Pacientes
Reportar Síntomas Diarios
extiende (si receta deseada no existe)
extiende (si abre ficha paciente)
extiende (si paciente no existe)
Crear Receta
Consultar Ficha Pacente
Añadir Paciente
extiende (si se añade objetivo nutricional)
Añadir Objetivo Nutricional

Ingeniería del Software (ISW)

Evaluación Teoría
19-01-2023
ETSInf-UPV

5. *(2 puntos)* A partir del siguiente diagrama de clases en UML:



Nota: Hay restricción de navegabilidad entre Evento y Participante, y entre Sesión y Sesión.

a) *(1 punto)* Realice el diseño de objetos en C#.

```
public class Competicion
{
    private string nombre;

    //Relaciones
    private ICollection<Evento> eventos;
    private ICollection<Promotor> promotores;
}

public class Evento
{
    private string nombre;

    //Relaciones
    private Competicion competicion;
    private ICollection<Sesion> sesiones;
    private ICollection<Inscripcion> inscripciones;
     private Participante ganador;
}

public class Inscripcion
{
    private int numIndcripcion;
    private DateTime fechaHora;
    private double puntuacion;

    //Relaciones
    private Evento evento;
    private Participante participante;
}

public class Sesion
{
    private string lugar;
    private DateTime fechaHora;

    //Relaciones
    public Evento evento;
    private ICollection<Sesion> tiene_subsesiones;
}

public class Participante
{
    private string DNI;
    private string nombre;

    //Relaciones
    private ICollection<Inscripcion> inscripciones;
}

public class Promotor : Participante
{
    private string empresa;

    //Relaciones
    private Competicion competicion;
}
```

b) *(0.5 puntos)* Realice el diseño en C# del constructor con parámetros de cada clase (sólo la cabecera, no los implemente. No defina el constructor sin parámetros).

```
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)
{    }
```

si se relaja bien el conflicto 1..1. Esta es una solución, pero puede haber otras

```
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) *(0.5 puntos)* Usando los constructores del apartado anterior, implemente el código necesario para crear una instancia del sistema consistente con el diagrama de clases, para el siguiente supuesto: Se registra la competición "VCL Testing" con un único evento "Hackaton JUnit" y dos participantes inscritos, uno de ellos promotor (que también será patrocinador de la competición). Use los valores que desee para el resto de los atributos.

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

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

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

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


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


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

HackJUnit.AddInscripcion(ObjInscrip1);
ObjPromotor1.AddInscripcion(ObjInscrip1);


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

6. (*1.5 puntos*) Diseñe los casos de prueba para el siguiente fragmento de código siguiendo la técnica del camino básico de caja blanca, dibuje el grafo de flujo, calcule la complejidad ciclomática, especifique los caminos independientes y los casos de prueba asociados a cada camino.

La clase Child tiene tres propiedades públicas de tipo entero: HomeBehaviour, AcademicGrades y Gifts. HomeBehaviour y AcademicGrades pueden contener valores entre 0 y 10. SantaCalculator es un método que devuelve el número de casas que tiene que visitar Santa Claus y modifica la lista children, actualizando la propiedad Gifts de sus objetos Child. **Considere que solo la línea indicada puede dar excepción y que children NO contendrá elementos null.**

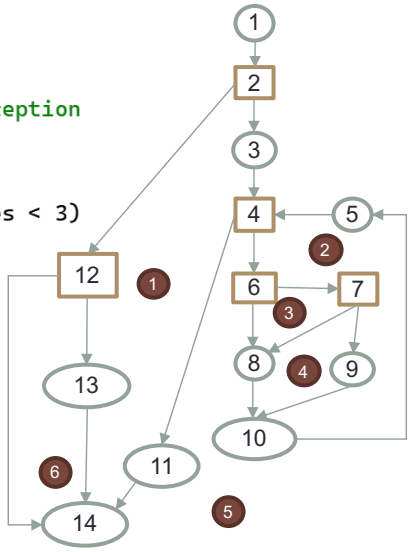
```
public static int SantaCalculator(List<Child> children)
{
  int numHomes = 0;
  try
  {
   int count = children.Count();       /*Puede producir ArgumentNullException si children no
inicializada*/

for (int i=0;i<count; i++){
 if (children[i].HomeBehaviour < 2 || children[i].AcademicGrades < 3)
        children[i].Gifts = 0;
    else
        children[i].Gifts = (children[i].AcademicGrades +
                                              children[i].HomeBehaviour) / 4;
    numHomes++;
   }
   return numHomes;
  }
  catch (ArgumentNullException)
  {
    return numHomes;
  }
}
```

Solución:

V(G)= 6
Áreas = 6
Nodos predicado= 5 → 5+1=6
Nodos = 14
Aristas = 18 → 16-14+2= 6

Nombre:
Grupo:

```
public static int SantaCalculator(List<Child> children)
{
  int numHomes = 0;
  try
  {
   int count = children.Count(); /* Puede producir ArgumentNullException
                                  * si children no inicializada */
   for (int i=0;i<count; i++)
   {
     if (children[i].HomeBehaviour < 2 || children[i].AcademicGrades < 3)
         children[i].Gifts = 0;
     else
         children[i].Gifts = (children[i].AcademicGrades +
                              children[i].HomeBehaviour) / 4;
     numHomes++;
   }
   return numHomes;
  }
  catch (ArgumentNullException)
  {
     return numHomes;
  }
}
```



Camino	Entrada	Salida	
	children	**return**	**children**
1-2-12-14 Lista no inicializada, excepción distinta de ArgumentNullException	No puede probarse		
1-2-**12**-**13**-14. Lista no inicializada	null	0	null
1-**2**-**3**-4-11-14 Lista sin elementos	[]	0	[]
1-2-3-**4**-**6**-8-10-5-4-11-14 Un elemento. Mal comportamiento	[{HomeBehavior=1;AcademicGrades=4;Gifts=0}]	1	[{HomeBehavior=1;AcademicGrades=4;Gifts=0}]
1-2-3-4-**6**-**7**-8-10-5-4-11-14 Un elemento. Buen comportamiento y malas notas	[{HomeBehavior=7;AcademicGrades=1;Gifts=0}]	1	[{HomeBehavior=7;AcademicGrades=1;Gifts=0}]
1-2-3-4-6-**7**-**9**-10-5-4-11-14 Un elemento. Buen comportamiento y notas	[{HomeBehavior=8;AcademicGrades=8;Gifts=0}]	1	[{HomeBehavior=8;AcademicGrades=8;Gifts=4}]

Nombre:
Grupo:

Nombre:
Grupo:

Nombre:
Grupo:

Nombre:
Grupo:

Nombre:
Grupo:

Nombre:
Grupo:

Nombre:
Grupo: