ZoocialNetwork

> *Examen Extraordinario de Programación - Curso 2022-2023*

NOTA: *Antes de comenzar asegúrese de descompactar el archivo `ZoocialNetwork.zip`. Asegúrese también de que su código compila, y la aplicación de consola ejecuta (debe lanzar una excepción). Recuerde que todo el código a evaluar debe ir en el archivo `Exam.cs`*

En el Zoológico Nacional se han propuesto atraer un mayor número de clientes para el próximo verano. Para ello, entre otras iniciativas, quieren crear una red **zoocial** para la interacción entre los usuarios que deseen visitar el Zoológico. Esto permitiría programar visitas en grupos, realizar encuestas, etc. Para resolver este problema solicitaron la ayuda de los estudiantes de `MATCOM`.

La tarea consiste en hacer una primera implementación de algunas de las funcionalidades más importantes de la **ZoocialNetwork**.

Descripción de la red zoocial ZoocialNetwork

Como es de suponer, el elemento fundamental de **ZoocialNetwork** son los usuarios. Por lo que debe existir una representación para los mimsos, que contenga su nombre, edad, género y mascota favorita (esto último a petición de los encargados del zoológico).

En su implementación, usted usará la definición de usuario que se encuentra en el proyecto `ZoocialNetwork`:

```
public interface IUser
{
    string Name {get;}
    int Age {get;}
    string Gender {get;}
    string FavouritePet {get;}
}
```

Para implementar las funcionalidades asociadas a la red zoocial **ZoocialNetwork**, usted debe utilizar la interfaz `INetwork` del proyecto `ZoocialNetwork`.

```
public interface INetwork
{
    // ... métodos que veremos a continuación
}
```

Una primera funcionalidad básica en la red es que se permita agregar usuarios. Para ello se utilizará el método `AddUser`, que recibe las propiedades del usuario, nombre, edad, género y mascota favorita. Este método retornará una instancia de `IUser`, correspondiente al usuario agregado a la red.

```
public interface INetwork
{
    IUser AddUser(string name, int age, string gender, string favouritePet);
}
```

Relación de amistad entre los usuarios.

La red zoocial propuesta tiene una noción de amistad transitiva. Esto implica que si un usuario **a** es amigo de los usuarios **b** y **c**, entonces **b** y **c** también son amigos. Por tanto, si dos usuarios se hacen amigos, entonces lo harán también todos los que eran amigos de ellos por separado.

Luego, usted deberá implementar la funcionalidad `ConnectUsers` que recibe el nombre de dos usuarios de la red que desean hacerse amigos. La implementación realizada debe ser consistente con la definición de amistad dada. Si alguno de los usuarios recibidos no existe, debe lanzar una excepción.

```
public interface INetwork
{
    void ConnectUsers(string username1, string username2);
}
```

Para los usuarios de **ZoocialNetwork** y los trabajadores del Zoológico es fundamental poder consultar la información de un usuario específico. Para esto se implementan un grupo de métodos, el primero de ellos debe ser `GetUserInfo`, que recibe el nombre del usuario y devuelve la instancia de `IUser` correspondiente.

Otra funcionalidad importante es conocer el grupo de amigos de un usuario determinado, para lo que se implementa el método `GetUserNetwork`. Este método recibe el nombre del usuario y retorna un `IEnumerable<IUser>` con todos los amigos del mismo, incluyendo al propio usuario.

En determinados contextos, puede ser importante filtrar, dentro de la red de amigos de un usuario, aquellos que cumplan con determinadas características, por ejemplo todos los que tengan como máscota favorita un perro. Esto ayudaría a la directiva del Zoológico a programar actividades temáticas y entre otras opciones. Para esto se implementa una sobrecarga del método `GetUserNetwork`. Esta sobrecarga recibe además del nombre del usuario, un predicado `filter`, correspondiente a las restricciones deseadas.

En caso de que en alguno de los métodos anteriores se reciba un nombre de usuario que no se corresponda con ningún usuario de la red, se debe lanzar una excepción. A continuación se muestran las definiciones de estos métodos.

```
public interface INetwork
{
    IUser GetUserInfo(string username);
    IEnumerable<IUser> GetUserNetwork(string username);
    IEnumerable<IUser> GetUserNetwork(string username, Func<IUser,bool> filter);
}
```

Otra de las iniciativas del Zoológico es regalarle una camiseta a cada visitante como recuerdo de su estancia en el centro. Las visitas se harán por grupos, donde cada grupo estará compuesto por usuarios que son amigos entre ellos. Cuando se recibe a un grupo, debe tenerse la cantidad de camisetas suficientes como para darle una distinta a cada miembro. Por tanto, los trabajadores quieren saber la cantidad mínima de camisetas tal que, no importa el grupo de amigos que vaya, van a ser suficientes. Para ayudar a los trabajadores se implementa el método `MinTshirts` que devuelve el valor esperado.

```
public interface INetwork
{
    int MinTshirts();
}
```

Finalmente, se implementa el método `NumberOfGroups`, que devuelve un número entero que indica la cantidad de grupos de amigos distintos que existen en la red **ZoocialNetwork**.

```
public interface INetwork
{
    int NumberOfGroups();
}
```

Luego, la interfaz `INetwork` queda de la siguiente forma

```
public interface INetwork
{
    IUser AddUser(string name, int age, string gender, string favouritePet);
    void ConnectUsers(string username1, string username2);

    IUser GetUserInfo(string username);
    IEnumerable<IUser> GetUserNetwork(string username);
    IEnumerable<IUser> GetUserNetwork(string username, Func<IUser,bool> filter);

    int MinTshirts();
    int NumberOfGroups();
}
```

Implementando ZoocialNetwork

Usted debe dar una implementación de `IUser` e `INetwork`. Las clases que implementan estas interfaces, y todo el código adicional que haga falta para su funcionamiento, deben estar en el archivo `Exam/Exam.cs`, que será **el único archivo evaluado**.

Para evaluar su código, se ejecutará el método `CreateZoocialNetwork` en la clase `Exam`. En este método usted debe simplemente devolver una instancia de su implementación de la interfaz `INetwork`.

En el proyecto `Exam`, archivo `Program.cs`, que es una aplicación de consola, usted puede adicionar todo el código que considere necesario para probar su implementación. Ese código no será evaluado.

Para todos los llamados a todos los métodos que no sean correctos (por ejemplo que no exista el nombre de usuario especificado), usted debe lanzar una excepción correspondiente.

Ejemplo

A continuación mostramos un ejemplo sencillo. El código de este ejemplo está en el método `Main` de la clase `Program`.

Primero se crea la instancia de la red zoocial utilizando el método `CreateZoocialNetwork` y se agregan 4 usuarios a la misma.

```
// rear Red Zoocial
var net = Exam.CreateZoocialNetwork();

// Agregar usuarios a la red
var user1 = net.AddUser("Pedro", 15, "Hombre", "Perro");
var user2 = net.AddUser("Laura", 15, "Mujer", "Gato");
var user3 = net.AddUser("Pablo", 13, "Hombre", "León");
var user4 = net.AddUser("Lola", 18, "Mujer", "Perro");
```

Luego se consulta la información del primero de los usuarios añadidos y se valida según sus datos.

```
// Consultar información de un usuario
var user = net.GetUserInfo("Pedro");
Debug.Assert(user.Name == "Pedro" && user.Age == user1.Age && user.Gender == user1.Gender && user.FavouritePet == user1.Favourit
```

A continuación se agregan 16 usuarios a la red, cuyas instancias se guardarán en la lista `newUsers`:

```
List<IUser> newUsers = new List<IUser>();
// Agregar 16 usuarios nuevos
string[] pets = {"Perro", "Gato", "León", "Mono"};
string[] genders = {"Mujer", "Hombre"};
for(int i = 0; i < 16; i++)
{
    var nuser = net.AddUser($"Usuario{i+1}", 14, genders[i%2], pets[i%4]);
    newUsers.Add(nuser);
}
```

A continuación se hacen amigos cada uno de los 4 primeros usuarios de 4 de los 16 agregados anteriormente. Finalmente se hacen amigos **Pedro** y **Lola**.

```
// Conectando Usuarios
for(int i = 0; i < 4; i++)
{
    net.ConnectUsers("Pedro", newUsers[i].Name);
}

for(int i = 4; i < 8; i++)
{
    net.ConnectUsers("Laura", newUsers[i].Name);
}

for(int i = 8; i < 12; i++)
{
    net.ConnectUsers("Pablo", newUsers[i].Name);
}

for(int i = 12; i < 16; i++)
{
    net.ConnectUsers("Lola", newUsers[i].Name);
}

net.ConnectUsers("Pedro", "Lola");
```

Una vez creados los enlaces anteriores se chequea el estado de la red de amigos del usuario **Pedro**, utilizando el método `GetUserNetwork`. Debe cumplirse que la cantidad de usuarios devueltos por el método sea igual a 10.

```
// Consultar la red de un usuario
var pedroNet = net.GetUserNetwork("Pedro");
Debug.Assert(pedroNet.Count() == 10);
```

A continuación se filtrarán los amigos de **Pedro** que cumplen la condición de que su mascota favorita sea **Perro**.

```
var pedroNetDog = net.GetUserNetwork("Pedro", x => x.FavouritePet == "Perro");
Debug.Assert(pedroNetDog.Count() == 4);
```

Luego se comprueban los valores de los métodos `MinTshirts` y `NumberOfGroups`:

```
// Consultar MinTshirts
int minTshirts = net.MinTshirts();
Debug.Assert(minTshirts == 10);

// Consultar Número de grupos
int nGroups = net.NumberOfGroups();
Debug.Assert(nGroups == 3);
```