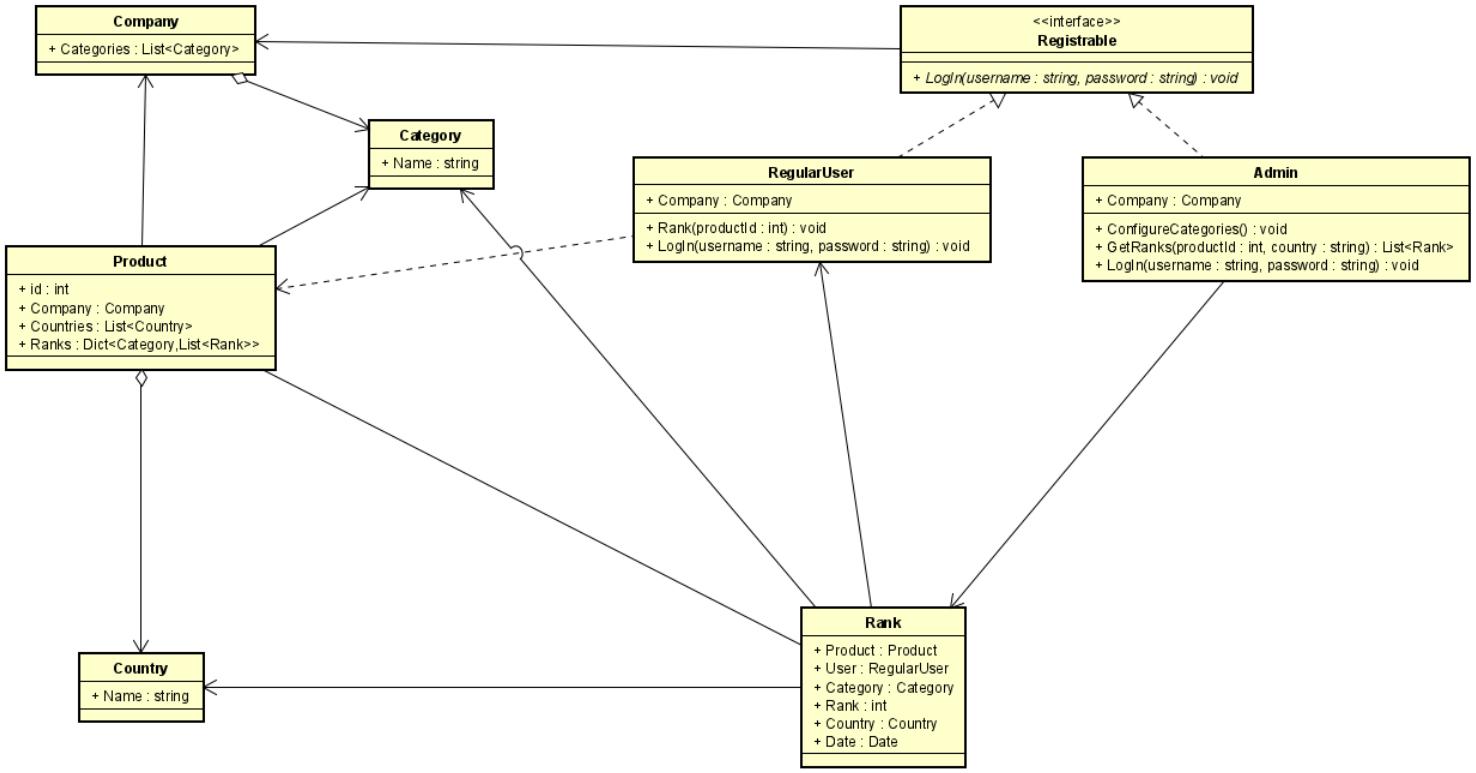
Analytical Challenge

Diagrama UML:



Supuestos:

- Agrego el método LogIn como ejemplo de cualquier comportamiento compartido entre todos los tipos de usuario, con el objetivo de justificar la utilización de polimorfismo.
- Las clases Country y Category tienen asociados más atributos y/o funciones que justifican su existencia como clase.
- Dado que una empresa puede tener múltiples categorías, con el objetivo de organizar y acceder a los rankings históricos de manera eficiente, se trabaja con un diccionario con la categoría como clave y una lista de clasificaciones como valor.

Programación en C#:

```
using System;
using System.Collections.Generic;

public interface IRegistrable
{
    void LogIn(string username, string password);
}

public class RegularUser : IRegistrable
{
    public Company Company { get; set; }

    public void LogIn(string username, string password)
    {
        // Lógica
    }

    public void Rank(int productId)
    {
        // Lógica
    }
}
```

```
public class Admin : IRegistrable
{
    public Company Company { get; set; }

    public void LogIn(string username, string password)
    {
        // Lógica
    }

    public void ConfigureCategories()
    {
        // Lógica
    }

    public List<Rank> GetRanks(int productId, string country)
    {
        // Lógica
        return new List<Rank>();
    }
}
```

Lo mostrado es una aproximación a cómo se programaría una interface y su respectiva implementación, así como también una clase en el lenguaje C#, con el objetivo de ampliar lo realizado en el diagrama UML.

Consulta:

Para realizar una consulta al modelo y obtener las clasificaciones para un momento específico X de un producto P en un país determinado C, debería:

1. Obtener el objeto de producto P basado en su identificador.
2. Acceder al diccionario de clasificaciones dentro de P.
3. Asignar la categoría deseada.
4. Filtrar por fecha y país,

Aproximación en código:

```
// Step 1:
Product productP = GetProductById(productId);

// Step 2
Dictionary<Category, List<Rank>> rankings = productP.Ranks;

// Step 3
Category desiredCategory = GetCategory(moment);

// Step 4:
List<Rank> allRankings = rankings[desiredCategory].FindAll(rank => rank.Country.Name == countryC && rank.Date == momentX);
```

Aclaración:

- En el proceso se incluye *GetCategory* por lo justificado en el tercer supuesto del modelo.