Clase Administradora

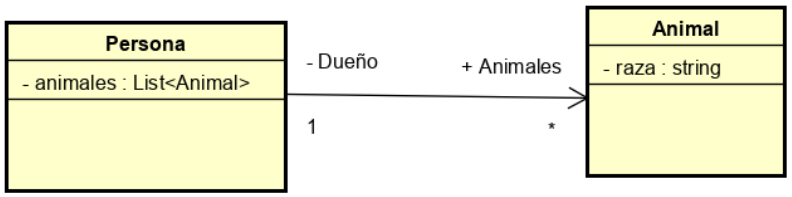
Dominio

Definición:

En programación orientada a objetos entendemos por dominio a la representación en clases y objetos para resolver los requisitos del sistema a implementar.

Estas clases resuelven la "lógica" del problema, sin importar como está definida la interfaz de usuario.

El punto de partida luego de detectar las clases es realizar un diagrama de clases en UML.



Dominio

Guía para definir un dominio:

- 1. Descubrir las clases candidatas relacionadas con los requisitos del sistema a implementar y representarlas en un diagrama UML.
- 2. Añadir las relaciones entre las clases con su navegación y multiplicidad.
- 3. Añadir los atributos necesarios.
- 4. Buscar la funcionalidad del sistema para poder añadir los métodos.

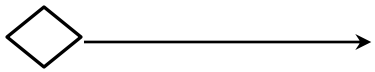
Clase administradora

La clase administradora es una clase encargada del mantenimiento de la mayoría de las clases del sistema, a excepción de algunas clases, tales como las clases de asociación

Agregación

Por lo general la relación usada es la agregación

- Describe una relación “tiene un … “ o “es parte de … "
- Se refiere al concepto de definir un objeto en terminos de sus componentes.



Clase administradora

Características:

- Conoce casi todo el dominio.
- Atiende las solicitudes que reciba el dominio.
- Permite tener un único punto de entrada al dominio.
- Oculta la complejidad del sistema y el conjunto de clases que forma el dominio.
- Se puede usar sin importar si es llamada desde la consola o MVC
- Es responsable de mantener su propia información en atributos tipo listas
- Conoce y puede informar de sus atributos

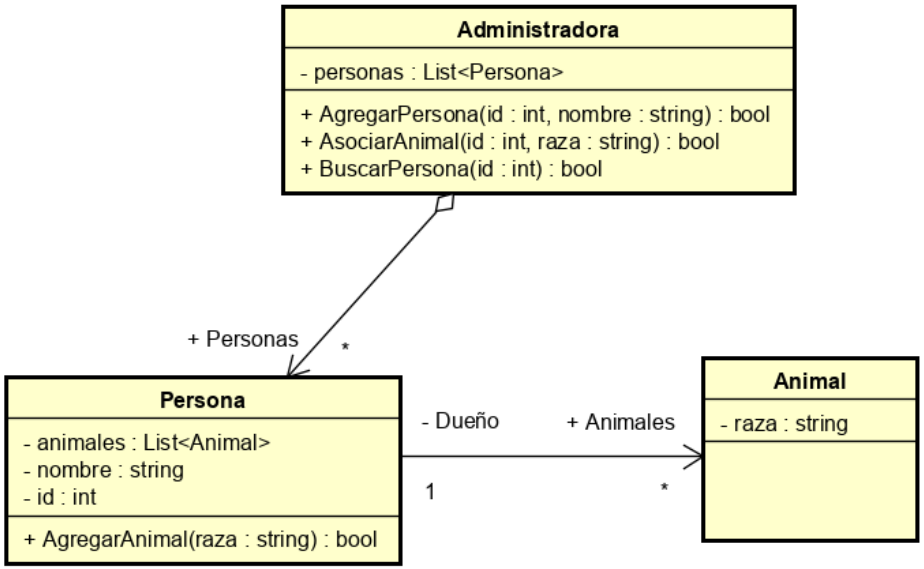
Clase administradora

Permite:

- Mejor acoplamiento.
- Desacopla la interfaz de usuario de la capa del dominio.
- Facilita el mantenimiento.
- Permite una mayor escalabilidad.
- Mejora la responsabilidad.

Clase administradora

Diagrama UML:



Clase administradora

Implementación:

Es importante que las listas siempre sean realizadas con atributos privados y para agregar un nuevo item siempre se realiza a través de un método y nunca con la propiedad set.

```
private List<Persona> _personas = new List<Persona>();
```

```
public void AgregarPersona(Persona nuevaPersona)
{
    // verifica si los datos de la persona son correctos
    try
    {
        nuevaPersona.Validar();
    }
    catch (Exception e)
    {
        throw e;
    }
    // verifica que no exista una persona con el mismo id
    if (ExistePersona(nuevaPersona.Id))
        throw new Exception("La persona ya existe");

    // se añade la persona a la lista
    _personas.Add(nuevaPersona);
}
```

Clase administradora

Implementación: Controles

```
public bool ExistePersona(int id)
{
    bool encontrada = false;
    int i = 0;
    while (!encontrada && i < _personas.Count)
    {
        if (_personas[i].Id == id)
            encontrada = true;
        else
            i++;
    }
    return encontrada;
}
```

```
// en Persona
public void Validar()
{
    if (_nombre.Length == 0)
        throw new Exception("Nombre no puede ser vacío");
    if (_id < 0)
        throw new Exception("Id debe ser mayor a 0");
}
```