El código de andamiaje es una parte fundamental en el diseño orientado a objetos, ya que proporciona la estructura básica que permite implementar y validar el modelo conceptual en un lenguaje de programación como Java. En el contexto de tu práctica sobre el sistema de gestión de un refugio de animales, el código de andamiaje debe incluir los siguientes elementos:

1. Clases y Estructura del Modelo

Definición de Clases: Implementa todas las clases que representan las entidades principales del modelo, como Socio, Voluntario, Donante, Adoptante, Animal, Refugio, Adopción, Donación, etc.

Atributos: Declara los atributos esenciales de cada clase, reflejando los datos que se manejan en el modelo (por ejemplo, nombre, fechaAdopcion, cantidadDonada).

Constructores: Proporciona constructores que permitan crear instancias de las clases de manera consistente, asegurando que los objetos se inicialicen correctamente.

2. Relaciones y Asociaciones

Implementación de Asociaciones: Representa las relaciones entre clases, como asociaciones 1:N o N:M, utilizando estructuras adecuadas (listas, conjuntos, mapas).

Por ejemplo, un Refugio puede tener una lista de Animales refugiados.

Código para Mantener la Consistencia: Incluye métodos que aseguren la consistencia bidireccional de las relaciones, si es necesario.

Por ejemplo, al agregar un Animal a un Refugio, también se debe establecer el Refugio en el Animal.

3. Métodos y Operaciones

Métodos de Acceso (Getters y Setters): Proporciona métodos para acceder y modificar los atributos privados de las clases.

Operaciones Específicas: Implementa los métodos que representan las operaciones clave del sistema, como:

adoptar(Animal animal, Voluntario voluntario) en la clase Adoptante.

tramitarAdopcion(Adoptante adoptante, Animal animal) en la clase Voluntario.

registrarAnimal(Animal animal) en la clase Voluntario.

donar(double monto) en la clase Donante.

4. Gestión de Colecciones y Relaciones

Métodos para Manipular Colecciones: Añade métodos como add, remove, y get para gestionar colecciones de objetos asociados.

Por ejemplo, Refugio puede tener addAnimal(Animal animal) y removeAnimal(Animal animal).

Validaciones y Restricciones: Implementa lógica para validar condiciones al manipular las colecciones.

Por ejemplo, verificar que un Animal no esté ya adoptado antes de permitir su adopción.

5. Mantenimiento de Invariantes y Restricciones del Modelo

Validaciones en Métodos: Dentro de los métodos, incluye validaciones que aseguren que se cumplan las restricciones del modelo.

Por ejemplo, asegurar que un Socio no supere los límites de donaciones establecidos.

Manejo de Excepciones: Utiliza excepciones para manejar situaciones en las que no se puedan cumplir las operaciones solicitadas.

6. Clases Asociativas para Relaciones Complejas

Reificación de Asociaciones: Para relaciones con atributos adicionales o complejas, crea clases intermedias.

Por ejemplo, una clase Adopcion que asocie Adoptante, Animal y Voluntario, y que incluya la fecha de adopción.

7. Implementación de Herencia y Polimorfismo

Clases Base y Subclases: Si es apropiado, utiliza herencia para modelar relaciones de especialización.

Por ejemplo, Voluntario, Donante y Adoptante podrían ser subclases de Socio.

Interfaces o Clases Abstractas: Si un Socio puede tener múltiples roles, considera el uso de interfaces o composición en lugar de herencia múltiple (que Java no soporta directamente).

8. Comentarios y Documentación

Comentarios en el Código: Añade comentarios que expliquen la funcionalidad de métodos y clases.

Javadoc: Utiliza Javadoc para generar documentación del código.

9. Ejemplos de Uso y Pruebas Básicas

Método Main o Clase de Pruebas: Incluye una clase con un método main que permita instanciar las clases y probar las operaciones principales.

Esto es útil para verificar que el código de andamiaje funciona según lo esperado.

10. Manejo de Visibilidad y Encapsulamiento

Encapsulación: Asegura que los atributos sean privados y que el acceso se realice a través de métodos públicos o protegidos según corresponda.

Visibilidad Apropiada: Define correctamente la visibilidad de clases, métodos y atributos (public, private, protected).