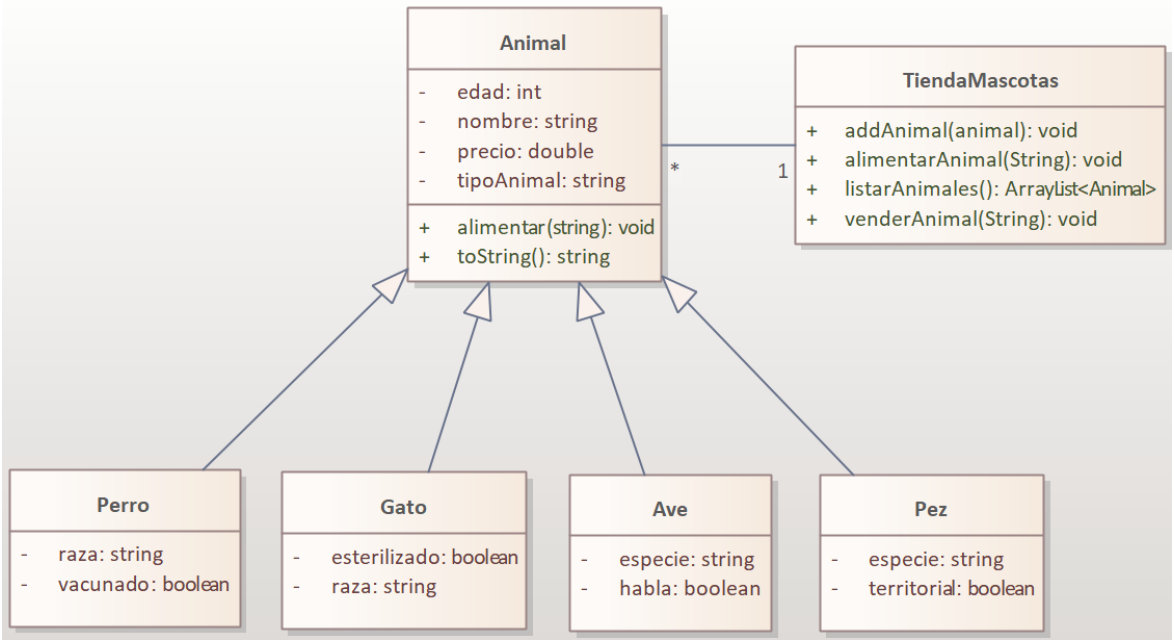
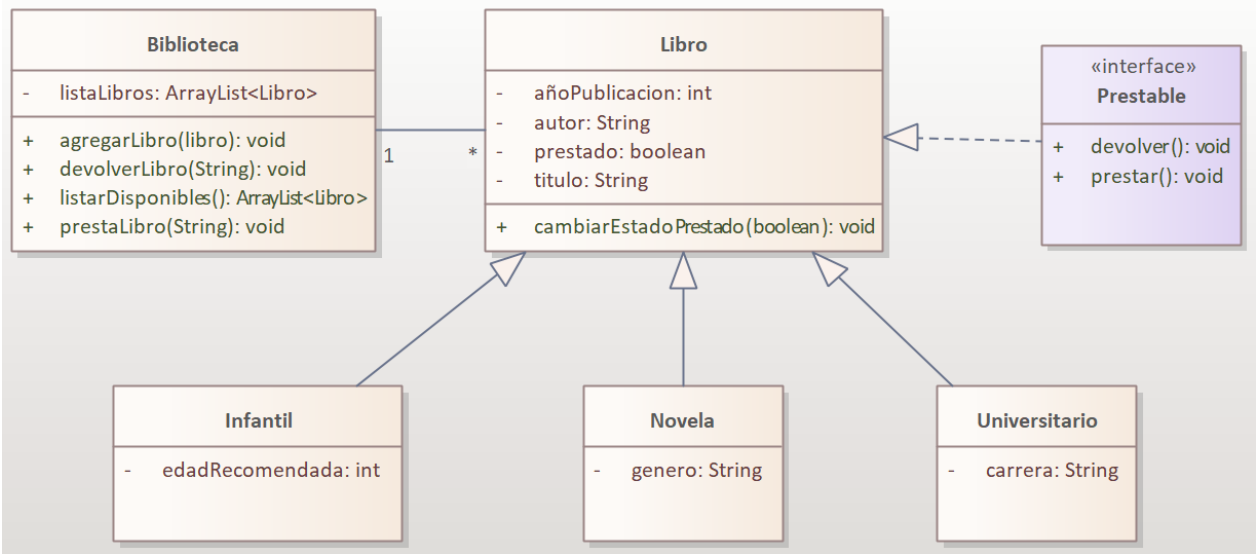
Trabajo practico N°2

1. Herencia en Java: La herencia en Java es un mecanismo que permite a una clase heredar atributos y métodos de otra clase. La clase que hereda se conoce como subclase o clase derivada, y la clase de la que hereda se conoce como superclase o clase base. Los beneficios de la herencia incluyen la reutilización de código, la organización jerárquica de clases y la facilitación de la extensibilidad y la mantenibilidad del código.
2. Representación en UML: En un diagrama UML, la herencia se representa mediante una flecha sólida que apunta desde la subclase hacia la superclase.
3. Se recomienda usar la herencia cuando una clase es una versión más específica de otra clase más general.
4. Interfaz en Java: Una interfaz en Java es una colección de métodos abstractos y constantes. Las interfaces permiten la definición de métodos que deben ser implementados por cualquier clase que las implemente. Las interfaces en Java permiten la implementación de la abstracción y el polimorfismo.
5. Representación en UML: En un diagrama UML, una interfaz se representa como una caja con el nombre de la interfaz en la parte superior, y los métodos que define dentro de la caja. La flecha punteada con un triángulo hueco se conecta desde la clase que implementa la interfaz hacia la interfaz.
6. Recomendaciones para usar una interfaz: Se recomienda usar una interfaz cuando se necesita proporcionar una especificación para un conjunto de métodos sin definir la implementación concreta. También se usa cuando se quiere lograr una mayor flexibilidad y modularidad en el diseño del software.
7. Diferencia entre interfaz y clase abstracta: En Java, una interfaz solo puede contener métodos abstractos y constantes, mientras que una clase abstracta puede contener métodos abstractos, métodos concretos y variables de instancia. Una clase puede implementar múltiples interfaces pero solo puede heredar de una clase abstracta.
8. Tipos de relaciones entre clases y objetos: Algunas relaciones comunes son la asociación, la agregación, la composición y la dependencia.
9. Representación en UML:
   - Asociación: Se representa con una línea sólida entre las clases.
   - Agregación: Se representa con una línea sólida con un rombo en el extremo de la clase que contiene al otro objeto.
   - Composición: Similar a la agregación, pero con un rombo relleno en el extremo de la clase que contiene al otro objeto.
   - Dependencia: Se representa con una línea punteada desde la clase que depende hacia la clase de la que depende.
10. Recomendaciones para cada tipo de relación:
    - Asociación: Se usa cuando dos clases están relacionadas de alguna manera pero no tienen una relación fuerte.
    - Agregación: Se utiliza cuando una clase es "dueña" de otra clase pero la otra clase puede existir de forma independiente.
    - Composición: Se usa cuando la relación entre dos clases es tal que la clase que contiene a la otra es responsable de la existencia y el ciclo de vida de la otra clase.
    - Dependencia: Se usa cuando una clase depende de otra temporalmente, pero no hay una relación estructural entre ellas.

11. UML


Animal
- edad: int
- nombre: string
- precio: double
- tipoAnimal: string
+ alimentar(string): void
+ toString(): string
TiendaMascotas
+ addAnimal(animal): void
+ alimentarAnimal(String): void
+ listarAnimales(): ArrayList<Animal>
+ venderAnimal(String): void
*
1
Perro
- raza: string
- vacunado: boolean
Gato
- esterilizado: boolean
- raza: string
Ave
- especie: string
- habla: boolean
Pez
- especie: string
- territorial: boolean

12. UML


Biblioteca
- listaLibros: ArrayList<Libro>
+ agregarLibro(libro): void
+ devolverLibro(String): void
+ listarDisponibles(): ArrayList<Libro>
+ prestaLibro(String): void
1
*
Libro
- añoPublicacion: int
- autor: String
- prestado: boolean
- titulo: String
+ cambiarEstadoPrestado(boolean): void
«interface»
Prestable
+ devolver(): void
+ prestar(): void
Infantil
- edadRecomendada: int
Novela
- genero: String
Universitario
- carrera: String