HISTORIA DE USUARIO

Nombre de la HU:	Herencia y polimorfismo en el sistema de la clínica
Objetivo de la HU	Como desarrollador del sistema de la clínica veterinaria, quiero diseñar y construir el modelo de pacientes y mascotas utilizando Programación Orientada a Objetos (POO) y diagramas UML, para que el sistema sea modular, mantenible y refleje correctamente la realidad de la clínica.

TASK1	**Comprender y aplicar UML para el diseño inicial del sistema.**
	• Crear un **diagrama de clases** que represente a los pacientes, mascotas y sus relaciones. • Definir atributos principales (ejemplo: nombre, edad, teléfono en paciente; nombre, especie, raza en mascota). • Dibujar las asociaciones: un paciente puede tener una o varias mascotas. • Usar una herramienta digital (Lucidchart, Draw.io, StarUML) o hacerlo a mano y digitalizarlo.
TASK2	**Definir las clases en C# basadas en el diagrama UML.**
	• Implementar la clase Paciente con atributos (nombre, edad, dirección, teléfono) y métodos básicos (mostrar información). • Implementar la clase Mascota con atributos (nombre, especie, raza, edad) y métodos (mostrar información). • Asegurar que cada clase tenga constructores para inicializar sus datos.
TASK 3	**Instanciar objetos y establecer relaciones.**
	• Crear objetos Paciente y Mascota en el programa. • Relacionar un paciente con una o varias mascotas mediante listas (List<Mascota> dentro de la clase Paciente). • Probar la asociación mostrando todas las mascotas de un paciente en pantalla.

TASK 4	**Aplicar encapsulación con modificadores de acceso.** • Definir atributos como private y exponerlos mediante propiedades (get y set). • Asegurar que los datos sensibles (ejemplo: teléfono del paciente) estén protegidos. • Usar public, private y protected correctamente para evitar accesos indebidos.
TASK 5	**Implementar herencia y polimorfismo.** • Crear una clase base Animal con atributos generales (nombre, edad, especie). • Hacer que la clase Mascota herede de Animal y añada atributos propios (raza, dueño). • Implementar un método EmitirSonido() en Animal y sobrescribirlo en Mascota para diferentes especies (ejemplo: perro = "Guau", gato = "Miau"). • Probar el polimorfismo llamando al mismo método desde diferentes tipos de mascotas.
TASK 6	**Usar abstracción con clases abstractas e interfaces.** • Definir una clase abstracta ServicioVeterinario con un método abstracto Atender(). • Implementar subclases (ejemplo: ConsultaGeneral, Vacunacion) que sobrescriban el método. • Crear una interfaz IRegistrable con un método Registrar(), e implementarla en Paciente y Mascota.
Criterios de aceptación. • Existe un diagrama UML claro que muestra clases, atributos y relaciones entre paciente y mascota. • Las clases en C# reflejan correctamente el diseño definido en UML. • Se han instanciado objetos y las relaciones entre pacientes y mascotas funcionan en la práctica.	

- Los atributos están encapsulados mediante propiedades y modificadores de acceso.
- Se ha implementado herencia para reutilizar código y establecer jerarquías.
- El polimorfismo funciona mediante sobrecarga y sobrescritura de métodos.
- Se ha aplicado abstracción con clases abstractas e interfaces.
- El código es modular, mantenible y organizado.

History points: 20 puntos

Cierre de actividad:
Al finalizar esta tercera semana, serás capaz de diseñar y construir un modelo orientado a objetos para la clínica veterinaria, utilizando UML como guía y aplicando los pilares de la POO en C#. Habrás definido clases, creado relaciones entre pacientes y mascotas, encapsulado datos, aplicado herencia, implementado polimorfismo y usado abstracción con clases abstractas e interfaces. Esto te permitirá dar el salto hacia arquitecturas más flexibles y profesionales en las próximas semanas.