Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesora:	Rocío Alejandra Aldeco Pérez
Asignatura:	Programación Orientada a Objetos
Grupo:	6
No de Práctica(s):	9
Integrante(s):	Ugalde Velasco Armando
No. de Equipo de cómputo empleado:	
No. de Lista o Brigada:	
Semestre:	2021-1
Fecha de entrega:	4 de diciembre de 2020
Observaciones:	

CALIFICACIÓN: _____

Práctica de Estudio 7 y 8: Herencia y polimorfismo
Programación Orientada a Objetos Grupo 6
Facultad de Ingeniería
Departamento de Computación

Objetivo de la práctica: Utilizar UML como herramienta para diseñar soluciones de software para un lenguaje de programación orientado a objetos.

Realiza las siguientes actividades después de leer y revisar en clase la **Práctica de Estudio 9: UML**.

Caso de estudio - Clínica Veterinaria

Imagina que un veterinario te solicita el desarrollo de un sistema que le permita administrar su clínica veterinaria. Para esto, un colega tuyo visitó al veterinario y recabo los siguientes requerimientos escritos en forma de prosa.

Un veterinario tiene como pacientes animales y como clientes personas. Un cliente es una persona que tiene un identificador único de cliente, un apellido paterno, un apellido materno y nombre(s), un número de cuenta bancaria, una dirección, un teléfono y un correo electrónico. Los clientes pueden tener varias mascotas, cada mascota tiene un identificador único de mascota, un nombre, una especie, una raza, color de pelo, fecha de nacimiento aproximada, peso medio del animal en las últimas 10 visitas y el peso actual del animal. Asimismo, se guardará un historial médico con cada enfermedad que tuvo y la fecha en la que enfermó. Adicionalmente cada mascota tiene un calendario de vacunación, en el que se registrará la fecha de cada vacuna, la enfermedad de la que se vacuna.

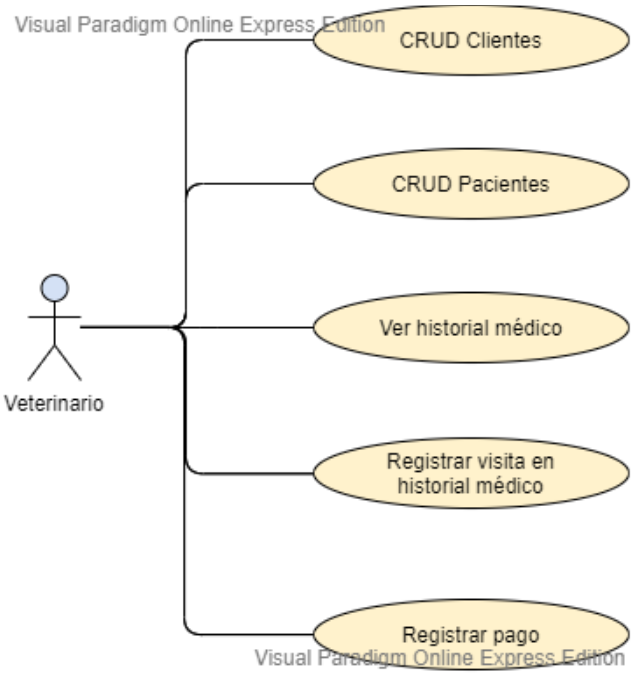
El veterinario desea realizar las siguientes acciones:

a) CRUD de clientes.
b) CRUD de pacientes.
c) Visualizar el historial médico de un paciente.
d) Registrar visitas en el historial médico de un paciente.
e) Registrar el pago de un cliente después de una visita.
f) Enviar de manera automática un correo electrónico al cliente una semana antes de la fecha de vacunación de un paciente.

Práctica de Estudio 7 y 8: Herencia y polimorfismo
Programación Orientada a Objetos Grupo 6
Facultad de Ingeniería
Departamento de Computación

Usando como referencia este caso de estudio, realiza los siguientes diagramas de UML que representen la estructura estática y dinámica de la solución de este caso.

1. Diagrama de casos de uso.



Casos de uso

En el diagrama se muestran las funcionalidades de la aplicación visibles al usuario. Nótese que no se incluye la **acción f**, debido a que se menciona que el envío de correo debe de realizarse de forma automática. Es decir, esta acción no se debe ejecutar explícitamente por el usuario.

2. **Diagrama de Clases** (uno sólo) que muestre la relación que existe entre todas las clases de programa.

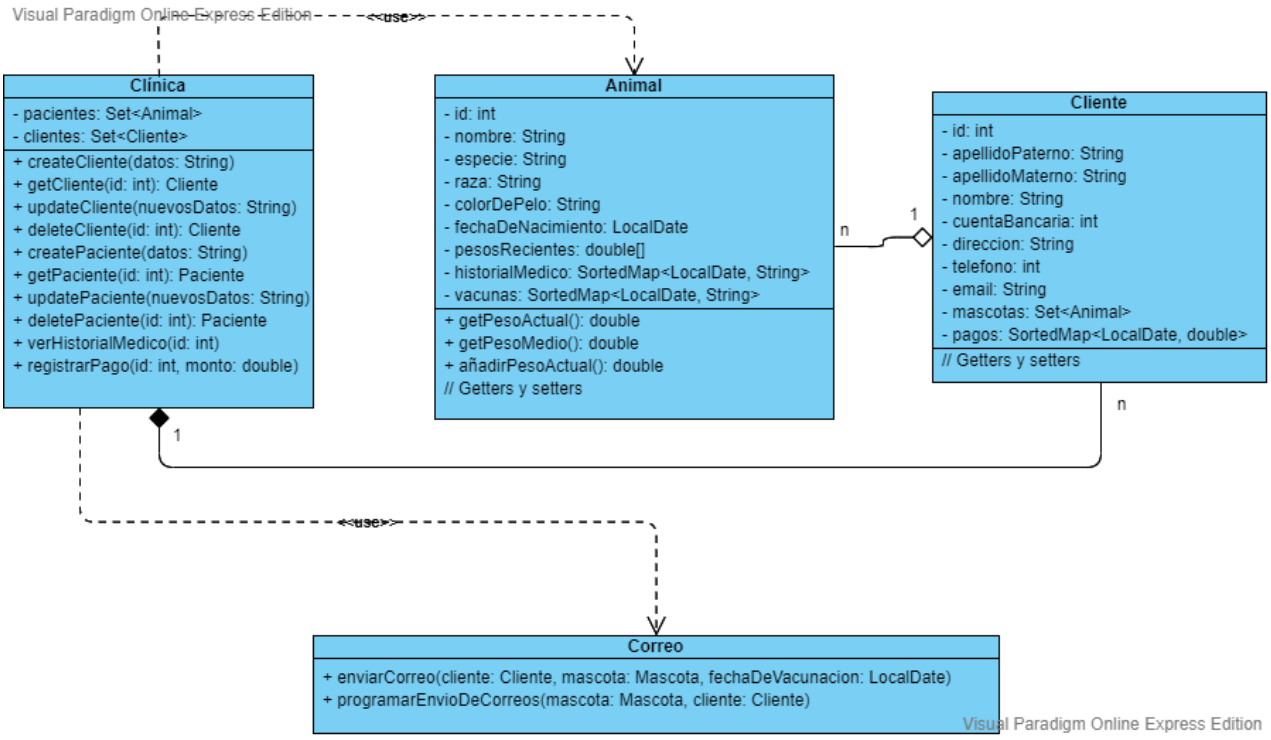


Diagrama de clases

Se implementaron cuatro clases para cumplir con la funcionalidad requerida: **Clínica**, que, como su nombre lo indica, representa la entidad de la veterinaria y contiene la funcionalidad visible al usuario; **Correo**, que contiene la funcionalidad necesaria para el envío de correos electrónicos de forma automática; **Animal**, que representa a los pacientes; y finalmente, **Cliente**, que representa a entidades con el mismo nombre.

Nótese que, si bien no se definieron explícitamente los **getters** y **setters** para cada clase, se añadió un comentario indicando que éstos se deben implementar.

Además, es importante identificar las relaciones que existen entre las clases: **Clínica utiliza** la clase **Correo** y **Animal**, una **Clínica** está **compuesta** de **Clientes**, y existe una relación de **agregación** entre **Cliente** y **Animal**, ya que, si bien ambas entidades pueden funcionar independientemente, los animales deben pertenecer a algún cliente.

3. **Diagrama de estados** del método que registrará el peso medio del paciente.

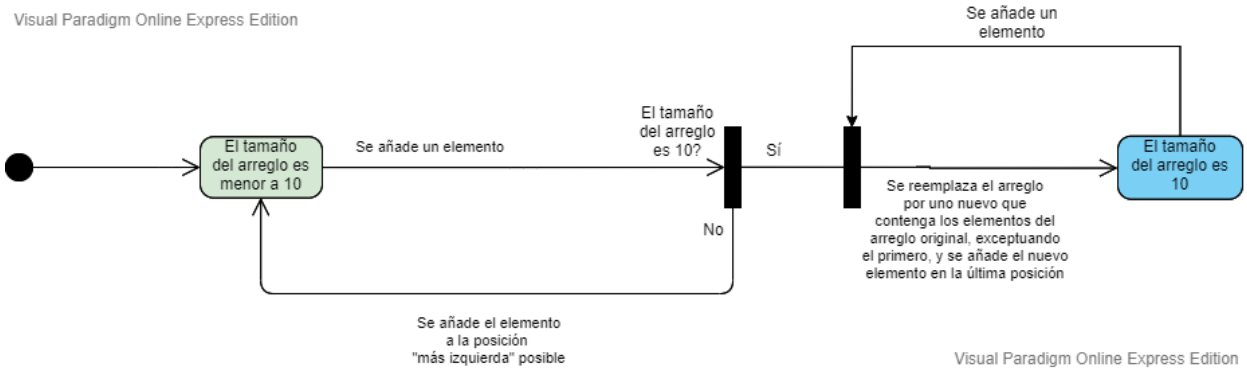


Diagrama de estado del proceso de adición de peso actual

Se decidió almacenar los pesos de las últimas **10** visitas en un arreglo de datos de tipo **double**, de tamaño **10**. El atributo que contiene los datos anteriores es **pesosRecientes**, presente en la clase **Animal**. Para obtener el peso actual de un animal, simplemente se retorna el último elemento añadido, es decir, el que se encuentre más cercano a la última posición del arreglo. En cambio, para obtener el promedio del peso en las últimas 10 visitas, se obtiene el promedio de los elementos presentes en el arreglo.

Dicho lo anterior, nos podemos percatar de que, al momento de registrar un peso nuevo para una mascota, se pueden presentar dos casos: el primero se presente cuando el tamaño del arreglo es **menor a 10**, por lo tanto, solamente se añade el elemento a éste en la posición más cercana posible al principio del arreglo; en cambio, si el arreglo se encuentra lleno *(tiene 10 elementos)*, se debe reemplazar por uno nuevo que contenga los elementos del arreglo original, exceptuando el primero y añadiendo el nuevo elemento en la última posición. Lo anterior se encuentra representado en el diagrama de estado.

Es importante mencionar que es posible implementar directamente la funcionalidad anterior con una estructura que soporte operaciones **FIFO,** como **ArrayDeque** o **LinkedList**, ya implementadas en el lenguaje.

4. **Diagrama de secuencia** de la acción F.

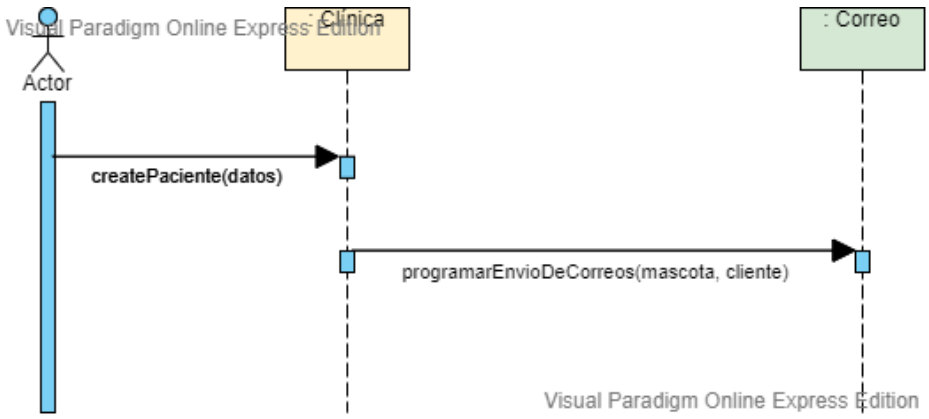


Diagrama de secuencia de la funcionalidad de envíos de correo automáticos

Antes de crear un paciente nuevo, el cliente responsable ya debe de estar registrado. Entonces, después de crear al nuevo paciente, éste se asocia con su cliente responsable, añadiendo la instancia de **Animal** al conjunto presente en el atributo **mascotas** del cliente, por medio de su setter.

Posteriormente, el método **createPaciente**, además de registrar la mascota correspondiente, programa el envío de los correos electrónicos utilizando el método **programarEnvioDeCorreos**, presente en la clase **Correo**. Este método recibe los datos necesarios para los envíos de correo, es decir, los datos de la mascota y el cliente, que incluyen el correo electrónico y el calendario de vacunación de la mascota.

5. Genera un documento **pdf** con la correspondiente carátula que presente y explique de manera concisa estos diagramas **UML**. Finalmente concluye. Recuerda subir este documento en PDF a **Schoology**.

CONCLUSIONES

UML es un lenguaje de modelado que nos permite representar el diseño de un sistema por medio de diagramas, de una forma visual y clara. Gracias a las especificaciones y propuestas de distintos tipos de diagramas, nos permite documentar y comprender un sistema desde diversas perspectivas de forma estandarizada, lo cual contribuye significativamente al proceso de desarrollo de software.

Es importante mencionar que, si bien no es necesario memorizar el estándar completo, es necesario estar familiarizados con los propósitos de cada diagrama y las reglas y significados de los elementos. De esta forma, al momento de realizar proyectos de una escala considerable, podremos tomar ventaja de éste y utilizarlo cuando sea pertinente.