# LABORATORIO 4

GONZALO DE VARONA

A00358687

ALGORTIMOS Y PROGRAMACIÓN 1

UNIVERSIDAD ICESI

MAYO 5 DE 2019

**Laboratorio Unidad 3**

En la unidad 3 de nuestro curso hemos trabajado sobre estructuras contenedoras de tamaño fijo y variable, de manera que aprendimos cómo modelar dichas estructuras en un diagrama de clases, e implementar soluciones haciendo uso de ellas. Además, aprendimos como utilizar instrucciones iterativas para manipular estructuras contenedoras o para resolver problemas que requieren la repetición de un conjunto de instrucciones. El presente laboratorio les presenta una actividad en la cual se requiere aplicar todos los conocimientos adquiridos en esta unidad y verificar de esta manera el cumplimiento de los objetivos que han sido planteados para la unidad 3 descritos en el programa del curso. Para llevar a cabo este ejercicio es necesario realizar las actividades listadas a continuación:

**Actividades**

Lleve a cabo las siguientes actividades de cada una de las etapas de desarrollo de software:

1. Análisis del problema (Definición del problema, identificación de entidades, sus características y relaciones) y especificación de Requerimientos Funcionales
2. Diagrama de Clases Completo (incluye el Modelo y el Main en la interfaz). El modelo debe ser elaborado digitalmente, pero NO generado automáticamente (por ejemplo, no es válido entregar modelos generados por Object Aid o ninguna otra herramienta).
3. Diagrama de Objetos de la situación inicial de su software.
4. Trazabilidad del Análisis al Diseño. Una tabla a dos columnas en la que se relaciona cada requerimiento con el método o métodos que permiten satisfacer dicho requerimiento.
5. Implementación en Java. Incluya en la implementación, los comentarios descriptivos sobre los atributos y métodos de cada clase. Recuerde que todos los artefactos generados de fase de diseño e implementación deben ser en inglés.
6. Usar GitHub como repositorio de código fuente utilizando la estructura de carpetas aprendida en clase. Recuerde que se debe evidenciar su avance a través de los días en el laboratorio.

Recuerde que puede encontrar la Rúbrica laboratorio en el siguiente enlace.

Se espera que usted haya realizado la actividad 1 para el día viernes 8 de marzo, se realizará la revisión de esta actividad.

Para llevar a cabo la actividad 1 recuerde que:

1. Los sustantivos describen posibles candidatos a entidades.
2. Los verbos asociados a posibles entidades sugiere los requerimientos funcionales.
3. Las relaciones usualmente están dadas cuando encuentre frases como: “se relaciona con” o “tiene”.

Usted debe subrayar con diferentes colores las entidades, sus características, sus relaciones y los requerimientos funcionales identificados.

**Nota:** Usted debe entregar un archivo en formato pdf con toda la documentación (análisis, diseño y tabla de trazabilidad) y la URL de su repositorio git Hub donde se deben encontrar los archivos de codificación en sus respectivos paquetes.

Tenga en cuenta que su repositorio gitHub debe presentar una estructura base como por ejemplo:

Veterinary/
src/ bin/ docs/
model/ ui/

En la carpeta src (source code) contendrá las carpetas model y ui, donde usted deberá almacenar sus clases .Java

Su código debería compilar de acuerdo con lo explicado en la diapositiva 13 de esta presentación: http://tinyurl.com/y3bd9bg2

**Enunciado**
ENTIDADES Y SUS CARACTERÍSTICAS : _____
RELACIONES: _____
RF: _____

En la veterinaria ***"Mi pequeña mascota"*** han identificado la necesidad de una aplicación de software para ayudar con el funcionamiento de su negocio. La veterinaria ha trabajado en la ciudad de Cali durante más de 10 años y presta diversos productos y servicios a sus clientes. Debido a ser un negocio pequeño han decidido partir el proyecto del desarrollo de la aplicación en dos fases y le han solicitado a usted ayuda con la fase 1 del proyecto.

La primera fase del proyecto consta en modelar el funcionamiento de los procesos básicos de la veterinaria. La veterinaria desea poder registrar a sus clientes humanos y sus mascotas. De cada persona se desea conocer el nombre, la identificación, la dirección, y el teléfono de contacto. Una persona puede tener 1 o más mascotas, y de cada mascota se desea conocer el nombre, el tipo de animal (Por ahora se especifican perro, gato, ave y otros), la edad y el peso.

Entre los servicios que ofrece la veterinaria se encuentra la hospitalización, para lo cual tiene habilitados unos mini cuartos para las mascotas, específicamente tiene 8. Para hospitalizar una mascota se requiere saber si hay disponibilidad de un mini cuarto, en caso de haber disponibilidad, se debe crear la historia clínica de la mascota, con los datos iniciales correspondientes a la mascota (nombre de la mascota, el peso del animal, el tipo de animal, la edad del animal y el peso del animal), su dueño (nombre, la identificación, la dirección, y el teléfono de contacto), un estado (ABIERTA o CERRADA, las historias clínicas se crean con

un estado ABIERTA y se cambian a estado CERRADA cuando se da de alta al animalito), la fecha de ingreso, los síntomas presentados, el posible diagnóstico y los medicamentos recetados. De cada medicamento recetado se desea conocer el nombre, la dosis, el costo por dosis y la frecuencia con la cual se debe administrar al animalito.

Como parte de las funciones requeridas en la primera fase del proyecto se requiere poder realizar un informe de las historias clínicas de los pacientes hospitalizados en el momento de la consulta del reporte. Consultar los datos de contacto del dueño de un animalito hospitalizado a partir del nombre de su dueño o del nombre del animalito. Además, se desea calcular el costo de una hospitalización, teniendo en cuenta que cada día de hospitalización tiene un costo, el cual se calcula de acuerdo a la siguiente tabla.

| Peso (kg) | Tipo de Animal | Valor por cada día |
|---|---|---|
| 1 - 3 | Gato | $10.000 |
| 3.1 - 10 | | $12.000 |
| 10.1 - 20 | | $15.000 |
| Más | | $20.000 |
| 1 - 3 | Perro | $15.000 |
| 3.1 - 10 | | $17.000 |
| 10.1 - 20 | | $20.000 |
| 1 - 3 | | $25.000 |
| 1 - 3 | Ave | $10.000 |
| 3.1 - 10 | | $12.000 |
| 10.1 - 20 | | $20.000 |
| 1 - 3 | | $25.000 |
| 1 - 3 | Otros | $10.000 |
| 3.1 - 10 | | $17.000 |

| | | |
|---|---|---|
| 10.1 - 20 | | $30.000 |
| 1 - 3 | | $30.000 |

Al costo de la hospitalización diaria se le debe sumar el costo de los medicamentos, de acuerdo a la cantidad de dosis del medicamento que se le han aplicado al animalito.

La aplicación debe permitir dar de alta a un animalito que haya estado hospitalizado, una vez se da de alta a un paciente se elimina la relación de su historia clínica con el mini cuarto y se debe guardar en un historial de historias clínicas, por lo cual al dar de alta un paciente la veterinaria desea entregar un informe de los datos de la hospitalización del paciente y poner disponible el mini cuarto. Para la veterinaria también es importante conocer cuánto han sido sus ingresos por concepto de hospitalizaciones y permitir saber el número del mini cuarto que ocupa una mascota basado en su nombre.

Finalmente, la veterinaria le ha solicitado poder consultar en el historial de historias clínicas si una mascota ya ha tenido una hospitalización anterior cada vez que se va a crear una nueva historia clínica, en caso de que exista la historia clínica anterior debe anexarse en la nueva historia clínica.

**Laboratorio Unidad 4**

En la unidad 4 de nuestro curso hemos trabajado sobre el uso de la definición de contratos para la construcción de métodos y su invocación. Además, usamos las técnicas del experto y descomposición para realizar la asignación de responsabilidades de las clases. Dadas estas herramientas es posible escribir una clase completa del modelo de la solución siguiendo la especificación de un conjunto de contratos. Finalmente, es posible documentar dichos contratos utilizando la sintaxis definida por la herramienta Javadoc y la generación de un API a partir de Javadoc. El presente laboratorio les presenta una actividad en la cual se requiere aplicar todos los conocimientos adquiridos en esta unidad y verificar de esta manera el cumplimiento de los objetivos que han sido planteados para la unidad 4 descritos en el programa del curso. Para llevar a cabo este ejercicio es necesario realizar las actividades listadas a continuación:

**Actividades**

Lleve a cabo las siguientes actividades de cada una de las etapas de desarrollo de software:

1. Análisis del problema (Definición del problema, identificación de entidades, sus características y relaciones) y especificación de Requerimientos Funcionales
2. Diagrama de Clases Completo (incluye el Modelo y el Main en la interfaz). El modelo debe ser elaborado digitalmente, pero NO generado automáticamente (por ejemplo, no es válido entregar modelos generados por Object Aid o ninguna otra herramienta).
3. Diagrama de Objetos de la situación inicial de su software.

4. Trazabilidad del Análisis al Diseño. Una tabla a dos columnas en la que se relaciona cada requerimiento con el método o métodos que permiten satisfacer dicho requerimiento.
5. Implementación en Java. Incluya en la implementación, los contratos respectivos. Recuerde que todos los artefactos generados de fase de diseño e implementación deben ser en inglés.
6. API generado usando Javadoc.
7. Usar GitHub como repositorio de código fuente utilizando la estructura de carpetas aprendida en clase. Recuerde que se debe evidenciar su avance a través de los días en el laboratorio.

Recuerde que puede encontrar la Rúbrica laboratorio en el siguiente enlace.

Para llevar a cabo la actividad 1 recuerde que:

1. Los sustantivos describen posibles candidatos a entidades.
2. Los verbos asociados a posibles entidades sugiere los requerimientos funcionales.
3. Las relaciones usualmente están dadas cuando encuentre frases como: “se relaciona con” o “tiene”.

Usted debe subrayar con diferentes colores las entidades, sus características, sus relaciones y los requerimientos funcionales identificados.

**Nota:** Usted debe entregar un archivo en formato pdf con toda la documentación (análisis, diseño y tabla de trazabilidad) y la URL de su repositorio git Hub donde se deben encontrar los archivos de codificación en sus respectivos paquetes.

Tenga en cuenta que su repositorio gitHub debe presentar una estructura base como por ejemplo:

Veterinary/
src/ bin/ docs/

Dentro de los directorios src/ y bin/ estarán presentes estos directorios, representando cada uno de sus paquetes:
model/ ui/

El directorio src (source code) contiene sus clases .java dentro de cada uno de los directorios model y ui. Por otro lado el directorio bin (binary files) contiene los archivos .class en los directorios model y ui.

Su código debería compilar de acuerdo con lo explicado en la diapositiva 13 de esta presentación: http://tinyurl.com/y3bd9bg2

Recuerde el listado de etiquetas disponibles para documentar su código haciendo uso de Javadoc.

**Etiquetas Javadoc**

| Tag | Descripción | Uso | Versión |
|---|---|---|---|
| @author | Nombre del desarrollador. | nombre_autor | 1.0 |
| @version | Versión del método o clase. | versión | 1.0 |
| @param | Definición de un parámetro de un método, es requerido para todos los parámetros del método. | nombre_parametro descripción | 1.0 |
| @return | Informa de lo que devuelve el método, no se puede usar en constructores o métodos "void". | descripción | 1.0 |
| @throws | Excepción lanzada por el método, posse un sinonimo de nombre @exception | nombre_clase descripción | 1.2 |
| @see | Asocia con otro método o clase. | referencia (#método(); clase#método(); paquete.clase; paquete.clase#método()). | 1.0 |
| @since | Especifica la versión del producto | indicativo numerico | 1.2 |
| @serial | Describe el significado del campo y sus valores aceptables. Otras formas validas son @serialField y @serialData | campo_descripcion | 1.2 |
| @deprecated | Indica que el método o clase es antigua y que no se recomienda su uso porque posiblemente desaparecerá en versiones posteriores. | descripción | 1.0 |

**Tomado de:** https://es.wikipedia.org/wiki/Javadoc

## Enunciado

La primera entrega de avances del desarrollo del software de la veterinaria dejó a las directivas de esta prestigiosa entidad muy impresionadas. Por lo que ahora quieren que se les entregue un programa funcional que resuelva las necesidades anteriormente planteadas (ver Primer Enunciado del Laboratorio de la Veterinaria).

Junto a esas funcionalidades la veterinaria quiere que los usuarios que van a operar el software también puedan realizar las acciones definidas en los siguientes contratos:

```
/**
*Description This method allows to calculate the body mass index for a pet.
*pre: The pet was created before and its attributes height and weight are not null neither height must be zero.
*post: The BMI is calculated.
*@return The pet body mass index.
*@throws  If the height is zero, so an exception is thrown due to the division on zero does not exist.
*/

/**
*Description This method allows to update the basic data of a veterinary client, these data include, address and phone number.
*pre: The client was created before.
*post: The address and /or phone number of the client is updated.
*@param The new address of the client. This param could be empty.
*@param The new phone number of the client. This param could be empty.
*/

/**
*Description This method allows to add new medicines that were prescription during the hospitalization at the patient stories.
*pre: The patient clinic story must be not null.
*post: New medicines were added to the patient clinic story.
*@param The medicine name. This param must be not null.
*@param The medicine dose, this param refers to the amount of medicine supplied to the pet each time according the frequence assigned. This param must be not null.
*@param The medicine cost by each dose. This param could be empty.
*@param The frequency of medicine application. This param could be empty.
*@return A message that indiques if medicine was added to the patient clinic story
*/

/**
*Description This method allows to add new notes to the possible diagnostic during the hospitalization at the patient stories.
*pre: The patient clinic story must be not null.
*post: New notes were added to the possible diagnostic in the patient clinic story.
*@param The notes of possible diagnostic. This param must be not null.
*/

/**
*Description This method allows to add a new symptom presented during the hospitalization at the patient stories.
*pre: The patient clinic story must be not null.
```

*post: A new symptom were added to the patient clinic story.
*@param The new symptom presented. This param must be not null.
*/

Además, la veterinaria espera que se puedan manejar los otros servicios que ofrece. La veterinaria ofrece los servicios de baño de mascotas en la veterinaria, baño de mascotas a domicilio, corte de uñas, profilaxis dental y aplicación de vacunas. De cada servicio se desea conocer el tipo de servicio, el costo, la fecha de realización del servicio, el identificador de la mascota a la que se le realizó el servicio y el identificador del dueño de la mascota a la cual se le realizó el servicio.

Los costos de los servicios de la veterinaria se especifican a continuación.

| Tipo de Servicio | Costo del servicio |
|---|---|
| Baño de mascotas en la veterinaria | $20.000 |
| Baño de mascota a domicilio | $30.000 |
| Corte de Uñas | $8.000 |
| Profilaxis Dental | $12.000 |
| Aplicación de Vacunas | $45.000 |

Dados los nuevos servicios que presta la veterinaria se requiere implementar las nuevas funcionalidades que se espera el software también permitirá ejecutar. Estas funcionalidades deben ser asignadas a la clase correspondiente, recuerde que puede hacer uso de las técnicas de descomposición y del experto para realizar la asignación. A continuación se listan las nuevas funcionalidades requeridas.

- Calcular los ingresos por servicios
- Calcular los ingresos totales de la veterinaria
- Agregar al sistema nuevos servicios prestados por la veterinaria, es decir los servicios vendidos no nuevos tipos de servicios.
- Promedio de ingresos de la veterinaria en una semana.
- Reporte de servicios prestados dada una fecha inicial y una fecha final.

Su programa debe cumplir con:

- La creación de algunos valores iniciales para que el programa funcione. Para esto se requiere que:
    - Existan al menos 2 clientes, cada uno con al menos 2 mascotas como valores iniciales existentes en el programa.
    - Exista al menos un animal hospitalizado actualmente y una historia clínica en el historial. Exista al menos 3 servicios prestados.
    - Se calculen de manera correcta todos los RF identificados en la entrega pasada y esta.
- Se despliega un menú que permita al usuario elegir la opción que desea utilizar del programa. Al usuario elegir la opción se realiza la operación solicitada por el usuario y se muestra de nuevo el menú.

**DESARROLLO**

**1.**

| **Nombre** | **R. 1. Registrar a un cliente persona** |
|---|---|
| **Resumen** | El usuario debe ingresar la información de un cliente humano al sistema para poder ser utilizada dentro del programa |
| **Entradas** | |
| -Nombre<br>-ID<br>-Dirección<br>-Teléfono de contacto | |
| **Resultados** | |
| Registrar la información de un cliente humano en el sistema. | |

| **Nombre** | **R. 2. Registrar a un cliente mascota** |
|---|---|
| **Resumen** | El usuario debe ingresar la información de un cliente mascota al sistema para poder ser utilizada dentro del programa |
| **Entradas** | |
| -Nombre<br>-Tipo de animal<br>-Edad<br>-Peso | |
| **Resultados** | |
| Registrar la información de un cliente mascota en el sistema. | |

| **Nombre** | **R. 3. Hospitalizar a un cliente mascota** |
|---|---|
| **Resumen** | El usuario debe ingresar la información de un cliente mascota al sistema para poder ser utilizada dentro del programa y así hospitalizar una mascota, creandole una historia clínica |
| **Entradas** | |
| -Fecha de ingreso del cliente mascota para la historia clinica<br>- Síntomas presentados del cliente mascota para la historia clinica<br>- El posible diagnóstico del cliente mascota para la historia clinica<br>-Los medicamentos recetados del cliente mascota para la historia clinica | |
| **Resultados** | |
| Hospitalizar a un cliente mascota en el sistema. | |

| **Nombre** | **R. 4. Registrar a un medicamento recetado** |
|---|---|
| **Resumen** | El usuario debe ingresar la información de un medicamento recetado al sistema para poder ser utilizada dentro del programa |
| **Entradas** | |
| -El nombre<br>-La dosis<br>-El costo por dosis<br>-La frecuencia con la cual se debe administrar al cliente mascota | |
| **Resultados** | |
| Registrar la información de un medicamento recetado en el sistema. | |

| **Nombre** | **R. 5. Mostrar la información de todas las historias clínicas de los clientes mascotas hospitalizados actuales** |
|---|---|
| **Resumen** | El programa debe mostrar la información de todas las historias clínicas de los clientes mascotas hospitalizados actuales |

| Entradas |
| --- |
| <Ninguno> |
| **Resultados** |
| Mostrar la información de todas las historias clínicas de los clientes mascotas hospitalizados actuales |

| **Nombre** | **R. 6. Mostrar los datos de contacto del dueño de un cliente mascota hospitalizado** |
| --- | --- |
| **Resumen** | El programa debe mostrar los datos de contacto del dueño de un cliente mascota hospitalizado a partir del nombre de su dueño o del nombre del animalito |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Mostrar los datos de contacto del dueño de un cliente mascota hospitalizado | |

| **Nombre** | **R. 7. Calcular el costo total de una hospitalización** |
| --- | --- |
| **Resumen** | El programa debe calcular el costo de una hospitalización por día de acuerdo a la tabla, y el costo de medicamentos de acuerdo a la cantidad de dosis aplicada al cliente mascota x el precio de cada dosis. Sumar ambos valores para sacar el costo total de la hospitalización |
| **Entradas** | |
| <Ninguna> | |
| **Resultados** | |
| Costo total de la hospitalización | |

| **Nombre** | **R. 8. Permitir dar de alta a un cliente mascota que haya estado hospitalizado** |
|---|---|
| **Resumen** | El programa debe permitir dar de alta y al momento, guardar la historia clínica en un historial de historias clínicas y mostrarlo, después eliminar la relación de la historia clínica con el mini cuarto |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Dar de alta a un cliente mascota que haya estado hospitalizado | |

| **Nombre** | **R. 9. Mostrar datos de hospitalización de un cliente mascota** |
|---|---|
| **Resumen** | El programa debe seguido al dar de alta, mostrar los datos de hospitalización de un cliente mascota |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Mostrar datos de hospitalización de un cliente mascota | |

| **Nombre** | **R. 10. Poner disponible un minicuarto** |
|---|---|
| **Resumen** | El programa debe poner como disponible el minicuarto que estaba siendo ocupado justo después de dar de alta y mostrar los datos de hospitalización del cliente mascota |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Disponible el minicuarto que estaba siendo ocupado | |

| **Nombre** | **R. 11. Mostrar ingresos por concepto de hospitalizaciones** |
|---|---|
| **Resumen** | El programa debe sumar todos los cobros hechos por concepto de hospitalizaciones para poder ser mostrado posteriormente |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Mostrar ingresos por concepto de hospitalizaciones | |

| **Nombre** | **R. 12. Mostrar el número del mini cuarto que ocupa un cliente mascota hospitalizado** |
|---|---|
| **Resumen** | El programa debe mostrar el número del mini cuarto que ocupa un cliente mascota hospitalizado según en su nombre. |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Mostrar el número del mini cuarto que ocupa un cliente mascota hospitalizado | |

| **Nombre** | **R. 13. Mostrar el historial de historias medicas de una mascota** |
|---|---|
| **Resumen** | El programa debe mostrar el historial de historias medicas de una mascota |
| **Entradas** | |
| -ID del dueño<br>-Nombre del dueño<br>-Nombre de la mascota | |
| **Resultados** | |

| |
|---|
| Mostrar el historial de historias medicas de una mascota |

| **Nombre** | **R. 14. Calcular los ingresos por servicios** |
|---|---|
| **Resumen** | El programa debe poder calcular el ingreso total por concepto de servicios prestados |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Ingreso total por concepto de servicios prestados | |

| **Nombre** | **R. 15. Calcular los ingresos totales de la veterinaria** |
|---|---|
| **Resumen** | El programa debe poder calcular el ingreso total por concepto de servicios prestados y hospitalizaciones |
| **Entradas** | |
| <Ninguno> | |
| **Resultados** | |
| Ingreso total por concepto de servicios prestados y hospitalizaciones | |

| **Nombre** | **R. 16. Agregar al sistema nuevos servicios vendidos por la veterinaria** |
|---|---|
| **Resumen** | El usuario debe ingresar la información de un servicio prestado al sistema para poder ser utilizada dentro del programa |
| **Entradas** | |
| -Tipo de servicio<br>-ID del dueño de la mascota<br>-Nombre de la mascota | |

| |
|---|
| -Fecha de la prestación del servicio |
| **Resultados** |
| Registrar un servicio prestado en el sistema. |

| | |
|---|---|
| **Nombre** | **R. 17. Mostrar promedio de ingresos de la veterinaria en una semana** |
| **Resumen** | El programa debe poder calcular el promedio de ingresos de la veterinaria en una semana dada una fecha inicial |
| **Entradas** | |
| -Fecha inicial de una semana | |
| **Resultados** | |
| Promedio de ingresos de la veterinaria en una semana dada una fecha inicial. | |

| | |
|---|---|
| **Nombre** | **R. 18. Mostrar reporte de servicios prestados dada una fecha inicial y una fecha final.** |
| **Resumen** | El usuario debe ingresar la información de un cliente humano al sistema para poder ser utilizada dentro del programa |
| **Entradas** | |
| -Fecha inicial<br>-Fecha final | |
| **Resultados** | |
| Reporte de servicios prestados dada una fecha inicial y una fecha final. | |

**REQUERIMIENTOS NO FUNCIONALES**

| | |
|---|---|
| **Nombre** | **RNF 1. 8 mini cuartos** |
| **Resumen** | La veterinaria cuenta únicamente con 8 mini cuartos. |

| **Nombre** | **RNF 2. Una mascota por mini cuarto** |
|---|---|
| **Resumen** | Cada mini cuarto solo puede alojar a una mascota. |

| **Nombre** | **RNF 3. Historial medico permanente** |
|---|---|
| **Resumen** | Cada vez que se crea una nueva historia médica deberá anexarse la anterior, haciendo así que la historia médica más reciente contenga a las anteriores en la mayoría de casos del programa |

**Descripción del problema:** La veterinaria "Mi pequeña mascota" no tiene una forma sencilla de administrar los procesos básicos y servicios que ofrece.
La veterinaria busca facilitar la forma en: como hospitaliza sus mascotas, como lleva sus ingresos, como lleva un registro de historiales clinicos, como crea historiales clinicos.

**TRAZABILIDAD**

| REQUERIMIENTO FUNCIONAL | CÓDIGO | |
|---|---|---|
| | CLASE | MÉTODO |
| R. 1. Registrar a un cliente persona | Main<br>Veterianary<br>Person | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+registerANewClient():void<br>+convertChoise2Type(int):String<br>+createPerson(Person, ArrayList<Pet>):void<br>+createPet(ArrayList<Pet>):void |
| R. 2. Registrar a un cliente mascota | Main<br>Veterianary<br>Person | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+registerANewClient():void<br>+convertChoise2Type(int):String<br>+createPerson(Person, ArrayList<Pet>):void<br>+createPet(ArrayList<Pet>):void |
| R. 3. Hospitalizar a un cliente mascota | Main<br>Veterianary<br>MiniRoom<br>Person<br>Pet<br>MedRecord<br>DateIn<br>ReqMed | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+hospitalizeAPet():void<br>+miniRoomAvailable():boolean<br>+getAvailable():boolean<br>+checkOwner(String, String, String)boolean<br>+showClientsInfo(String, String):String<br>+getName():String |

| | | |
|---|---|---|
| | | +getId():String<br>+showMyinfo():String<br>+showPetsinfo():String<br>+retrievePet(String, String, String):Pet<br>+getName():String<br>+getId():String<br>+givePet(String):Pet<br>+getName():String<br>+startHospitalizeVet(String, String, String, MedRecord, ArrayList<ReqMed>):void<br>+getName():String<br>+getId():String<br>+startHospitalizePers(String, MedRecord, ArrayList<ReqMed>):void<br>+getName():String<br>+addMedRec(MedRecord, ArrayList<ReqMed>):void<br>+setPetInfo(String):void<br>+showPetsinfo():String<br>+setOwnerInfo(String):void<br>+contactInfo():String<br>+addMeds(ArrayList<ReqMed>):void<br>+calculatingFee():double<br>+getDay():int<br>+getMonth():int<br>+getYear():int<br>+getType():String<br>+getWeight():double<br>+priceMed():double |
| R. 4. Registrar a un medicamento recetado | Main<br>Veterianary<br>MiniRoom<br>Person<br>Pet<br>MedRecord | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+hospitalizeAPet():void<br>+miniRoomAvailable():boolean<br>+getAvailable():boolean<br>+checkOwner(String, String, String)boolean<br>+showClientsInfo(String, String):String<br>+getName():String<br>+getId():String<br>+showMyinfo():String<br>+showPetsinfo():String<br>+retrievePet(String, String, String):Pet<br>+getName():String<br>+getId():String<br>+givePet(String):Pet<br>+getName():String<br>+startHospitalizeVet(String, String, String, MedRecord, ArrayList<ReqMed>):void<br>+getName():String<br>+getId():String<br>+startHospitalizePers(String, MedRecord, ArrayList<ReqMed>):void |

| | | |
|---|---|---|
| | | +getName():String<br>+addMedRec(MedRecord, ArrayList<ReqMed>):void<br>+setPetInfo(String):void<br>+showPetsinfo():String<br>+setOwnerInfo(String):void<br>+contactInfo():String<br>+addMeds(ArrayList<ReqMed>):void<br>^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^<br>Primera forma es al momento de hospitalizar una mascotas<br>----------<br>Segunda forma es después de haber hospitalizado una mascota<br>vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv<br>+editMedRecFromAPet():void<br>+checkOwner(String, String, String)boolean<br>+showClientsInfo(String, String):String<br>+getName():String<br>+getId():String<br>+showMyinfo():String<br>+showPetsinfo():String<br>+retrieveNumberMiniRoom(String):int<br>+getPet():String<br>+getNumberOfMiniRoom():int<br>+typeSelectionMssg():void<br>+plainLine():void<br>+locatePersonWithPet2AddStuff(String, String, String, int, String, String, ArrayList<ReqMed>):void<br>+getName():String<br>+getId():String<br>+locatePet2AddStuff(String, int, String, String, ArrayList<ReqMed>):void<br>+getName():String<br>+addStuff2MedRec(int, String, String, ArrayList<ReqMed>):void<br>+getSymptoms():String<br>+setSymptoms(String):void<br>+getDiagnosys():String<br>+setDiagnosys (String):void<br>+addNewMed(ReqMed):void |
| R. 5. Mostrar la información de todas las historias clínicas de los clientes mascotas hospitalizados actuales | Main<br>Veterinary<br>MiniRoom<br>Pet<br>MedRecord<br>ReqMed<br>DateIn | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+showMedRecsFromHPet():void<br>+medRecsFromHPet():String<br>+getAvailable():boolean<br>+getHostage():Pet<br>+showAllRecords():String<br>+medRecordInfo():String<br>+convertDateToString():String<br>+showMedsinfo():String |

| | | |
|---|---|---|
| R. 6. Mostrar los datos de contacto del dueño de un cliente mascota hospitalizado | Main<br>Veterinary<br>MiniRoom<br>Person<br>Pet | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+contactInfoByNameOrPet():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showContactInfo(int, String, String):String<br>+checkIfItsHospitalized(int, String):String<br>+getName():String<br>+getId():String<br>+contactInfo():String<br>+checkIfItsHospitalized(int, String):String<br>+getName():String<br>+getId():String<br>+getOwner():String<br>+getName():String<br>+reviewPet(String):boolean<br>+getName():String<br>+contactInfo():String |
| R. 7. Calcular el costo total de una hospitalización | MedRec<br>ReqMed<br>DateIn | +calculatingFee():double<br>+getDay():int<br>+getMonth():int<br>+getYear():int<br>+getType():String<br>+getWeight():double<br>+priceMed():double |
| R. 8. Permitir dar de alta a un cliente mascota que haya estado hospitalizado | Main<br>Veterinary<br>MiniRoom<br>Person<br>Pet<br>DateIn<br>MedRec<br>ReqMed | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+dischargeAPet():void<br>+checkOwner(String, String, String):boolean<br>+showClientsInfo(String, String):String<br>+getName():String<br>+getId():String<br>+showMyinfo():String<br>+showPetsinfo():String<br>+retrieveNumberMiniRoom(String):int<br>+getPet():String<br>+getNumberOfMiniRoom():int<br>+showAPetMedRecs(String, String, String):String<br>+getName():String<br>+getId():String<br>+givePet(String):Pet<br>+showAllRecords():String<br>+medRecordInfo():String<br>+convertDateToString():String<br>+showMedsinfo():String<br>+removePet(String, String, String):void<br>+getName():String |

| | | |
|---|---|---|
| | | +getId():String<br>+givePet(String):Pet<br>+getName():String<br>+cutItOff():void<br>+setStatus(String):void<br>+setDateOut(DateIn):void<br>+getPet():String<br>+getOwner():String<br>+setAvailable(boolean):void<br>+setPet(String):void<br>+setOwner(String):void<br>+setHostage(Pet):void |
| R. 9. Mostrar datos de hospitalización de un cliente mascota | Main<br>Veterinary<br>Person<br>Pet<br>DateIn<br>ReqMed<br>MedRec | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+showAllMedRecs4Pet():void<br>+checkOwner(String, String, String):boolean<br>+showClientsInfo(String, String):String<br>+getName():String<br>+getId():String<br>+showMyinfo():String<br>+showPetsinfo():String<br>+showAPetMedRecs(String, String, String):String<br>+getName():String<br>+getId():String<br>+givePet(String):Pet<br>+getName():String<br>+showAllRecords():String<br>+medRecordInfo():String<br>+convertDateToString():String<br>+showMedsinfo():String |
| R. 10. Poner disponible un minicuarto | Veterianary<br>MiniRoom<br>Person<br>Pet<br>MedRecord | +removePet(String, String, String):void<br>+getName():String<br>+getId():String<br>+givePet(String):Pet<br>+getName():String<br>+cutItOff():void<br>+setStatus(String):void<br>+setDateOut(DateIn):void<br>+getPet():String<br>+getOwner():String<br>+setAvailable(boolean):void<br>+setPet(String):void<br>+setOwner(String):void<br>+setHostage(Pet):void |

| | | |
|---|---|---|
| R. 11. Mostrar ingresos por concepto de hospitalizaciones | Main<br>Veterinary<br>Person<br>Pet<br>MedRec | +showIncome():void<br>+gatherAllFees():double<br>+myBill():double<br>+gatherCosts():double<br>+getFee():double |
| R. 12. Mostrar el número del mini cuarto que ocupa un cliente mascota hospitalizado | Main<br>Veterinary<br>MiniRoom | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+showMiniRoomNumberByPetsName():void<br>+retrieveNumberMiniRoom(String):int<br>+getPet():String<br>+getNumberOfMiniRoom():int |
| R. 13. Mostrar el historial de historias medicas de una mascota | Main<br>Veterinary<br>Person<br>Pet<br>MedRec<br>ReqMed<br>DateIn | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+checkOwner(String, String, String):boolean<br>+showClientsInfo(String, String):String<br>+getName():String<br>+getId():String<br>+showMyinfo():String<br>+showPetsinfo():String<br>+showAPetMedRecs(String, String, String):String<br>+getName():String<br>+getId():String<br>+givePet(String):Pet<br>+showAllRecords():String<br>+medRecordInfo():String<br>+convertDateToString():String<br>+showMedsinfo():String |
| R. 14. Calcular los ingresos por servicios | Main<br>Veterinary<br>Person<br>Pet<br>Service<br>MedRec | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+showIncome():void<br>+gatherAllHospitalizationFees():double<br>+feeServicesVeterinary(int):double<br>+feeServicesPeople(int):double<br>+petsFeeServices(int):double<br>+servicesFees(int):double<br>+feeServices(char):double<br>+getType():char<br>+getPrice():double |

| | | |
|---|---|---|
| R. 15. Calcular los ingresos totales de la veterinaria | Main<br>Veterinary<br>Person<br>Pet<br>Service<br>MedRec | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+showIncome():void<br>+totalIncome():double<br>+gatherAllHospitalizationFees():double<br>+feeServicesVeterinary(int):double<br>+feeServicesPeople(int):double<br>+petsFeeServices(int):double<br>+servicesFees(int):double<br>+feeServices(char):double<br>+getType():char<br>+getPrice():double<br>+myBill():double<br>+gatherCosts():double<br>+getFee():double |
| R. 16. Agregar al sistema nuevos servicios vendidos por la veterinaria | Main<br>Veterinary<br>Person<br>Pet<br>Service | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+registerAService():void<br>+convertChoise2Char(int):char<br>+retrievePet(String, String, String):Pet<br>+findPerson(String, String):Person<br>+getName():String<br>+givePet(String):Pet<br>+getName():String<br>+startServiceVet(String, String, Pet, Service):void<br>+startServicePers(Pet,Service):void<br>+addService(Service):void |
| R. 17. Mostrar promedio de ingresos de la veterinaria en una semana | Main<br>Veterinary<br>Person<br>Pet<br>Service<br>DateIn | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+showAverageIncomeByWeek():void<br>+weeklyAverageIncome(DateIn):double<br>+petsServicesWeeklyIncome(DateIn):double<br>+weeklyIncome(DateIn):double<br>+calculateDayOfTheYear():int<br>+getDateJob():DateIn<br>+getYear():int<br>+dateJobDayOfYear():int<br>+getPrice():double<br>+petsServicesWeeklyIncomeCounter(DateIn):int<br>+weeklyIncomeCounter(DateIn):double<br>+calculateDayOfTheYear():int<br>+getDateJob():DateIn<br>+getYear():int<br>+dateJobDayOfYear():int |

| | | |
|---|---|---|
| R. 18. Mostrar reporte de servicios prestados dada una fecha inicial y una fecha final | | +menu():void<br>+typeSelectionMssg():void<br>+plainLine():void<br>+showMenuOptions():void<br>+serviceReportsBetweenDates():void<br>+datesServiceReports(DateIn, DateIn):String<br>+myPetsServiceReports(DateIn, DateIn):String<br>+serviceBetweenDates(DateIn, DateIn):String<br>+toString():String<br>+calculateDayOfTheYear():int<br>+getDateJob():DateIn |