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 (Incluyendo los contratos definidos para los métodos que cumplen con los requerimientos funcionales propuestos en el laboratorio 3 y el presente laboratorio 4).
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. Returns -1 if the height is zero  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.
*@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 por 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.

LISTA DE POSIBLES REQUERIMIENTOS FUNCIONALES

Nombre	**R.# 1. Registrar a los clientes humanos y a sus respectivas mascotas**
Resumen	El programa debe permitir que el usuario registre los datos de sus clientes humanos y a sus mascotas
Entradas: • **Datos del dueño** • **Datos de la mascota**	
Resultados: Los datos del cliente humano y su mascota han sido registrados	

Nombre	**R.# 2. Conocer la disponibilidad de un minicuarto**
Resumen	El programa permite que el usuario saber la disponibilidad del minicuarto para seguidamente poder hospitalizar a una mascota.
Entradas:	
Resultados: Se ha verificado la disponibilidad de un minicuarto.	

Nombre	**R.#3. Crear historia clínica de las mascotas**
Resumen	El programa permite que el usuario ingrese la información de la mascota para poder hacer futuras operaciones con ella.
Entradas: • **Datos de la mascota** • **Datos del dueño** • **Fecha de ingreso** • **Síntomas de la mascota** • **Posible diagnostico** • **Medicamentos**	
Resultados: La historia clínica de la mascota ha sido creada.	

Nombre	**R.# 4. Dar de alta a la mascota**
Resumen	El programa permite que el usuario dé de alta a la mascota en cuanto esta se sienta mejor.
Entradas: • Nombre de la mascota a dar de alta.	
Resultados: La relación de la mascota con el minicuarto que se le había asignado ha sido eliminada.	

Nombre	**R.# 5. Realizar informe de las historias clínicas de las mascotas.**
Resumen	El programa realiza un informe de las historias clínicas de las mascotas hospitalizadas en el momento de las consultas del reporte.
Entradas:	
Resultados: El informe de la historia clínica de la mascota ha sido creado.	

Nombre	**R.# 6. Calcular el costo de una hospitalización**
Resumen	El programa debe calcular el costo de la hospitalización de un animal dependiendo del día en que se esté.
Entradas: • Peso del animal • Tipo de animal • Valor por día	
Resultados: Costo de la hospitalización	

Nombre	**R.# 7. Conocer los ingresos por hospitalizaciones.**
Resumen	El programa permite que el usuario conozca la información de los ambientes de los canguros y de los dragones.
Entradas:	
Resultados: Los ingresos por hospitalizaciones han sido cargados.	

Nombre	**R.# 8. Saber el número del cuarto en el cual está ubicada cada mascota**

Resumen	El programa permite que el usuario conozca la ubicación del cuarto de la mascota ingresando su nombre.
Entradas: • Nombre de la mascota	
Resultados: El nombre del cuarto en el cual está la mascota.	

Nombre	**R.# 9. Consultar el historial de una mascota.**
Resumen	El programa permite que el usuario consulte el historial de una mascota teniendo en cuenta las hospitalizaciones que esta haya tenido, cada vez que vaya a crear una nueva historia clínica y en caso de que ya exista, debe anexarla a la anterior.
Entradas:	
Resultados: La historia clínica ha sido anexada.	

Nombre	**R.# 10. Calcular el índice de masas corporal de una mascota**
Resumen	El programa permite que el usuario calcule el índice de masa corporal de una mascota
Entradas:	
Resultados: El índice de masa corporal de la mascota ha sido calculado.	

Nombre	**R.# 11. Actualizar los datos básicos de un cliente.**
Resumen	El programa permite que el usuario actualice los datos básicos de un cliente, como lo es su teléfono y dirección.
Entradas: • Nuevo teléfono • Nueva dirección	
Resultados: Los datos del cliente han sido actualizados.	

Nombre	R.# 12. Agregar nueva medicina al historial médico de una mascota.
Resumen	El programa permite que el usuario agregue nueva medicina al historial médico de una mascota durante la hospitalización.
Entradas: • Nombre de la nueva medicina • Dosis de la medicina • Precio de la medicina • Frecuencia de la medicina	
Resultados: La medicina ha sido agregada al historial médico.	

Nombre	R.# 13. Agregar nuevas notas al historial médico de una mascota.
Resumen	El programa permite que el usuario agregue nuevas notas al historial médico de una mascota durante la hospitalización.
Entradas: • Nuevas notas	
Resultados: La nueva nota ha sido agregada al historial médico.	

Nombre	R.# 14. Agregar nuevos síntomas al historial médico de una mascota.
Resumen	El programa permite que el usuario agregue nuevos síntomas al historial médico de una mascota durante la hospitalización.
Entradas: • Nuevos síntomas	
Resultados: Los nuevos síntomas ha sido agregada al historial médico.	

Nombre	R.# 15. Calcular los ingresos.
Resumen	El programa permite que el usuario calcule los ingresos por servicios que se presentaron en la veterinaria
Entradas:	
Resultados: Los ingresos basados en servicios han sido calculados.	

Nombre	**R.# 16. Calcular los ingresos totales de la veterinaria.**
Resumen	El programa permite que el usuario calcule los ingresos totales que se presentaron en la veterinaria
Entradas:	
Resultados: Los ingresos totales han sido calculados.	

Nombre	**R.# 17.Calcular el promedio de ingresos por servicios.**
Resumen	El programa permite que el usuario calcule el promedio de los ingresos por servicios.
Entradas:	
Resultados: el promedio por ingresos de servicios ha sido calculado.	

Nombre	**R.# 18.Calcular el promedio de ingresos en la veterinaria en una semana.**
Resumen	El programa permite que el usuario calcule el promedio de los ingresos por servicios en una semana.
Entradas: • **Dia** • **Mes** • **Año**	
Resultados: el promedio por ingresos de servicios en una semana por una fecha dada por el usuario ha sido calculado.	
Nombre	**R.# 19.Generar reporte de servicios prestados dada una fecha final y una inicial.**
Resumen	El programa permite que el usuario vea un reporte de los servicios prestados en una fecha que él ingrese.
Entradas: • **Dia inicial** • **Mes inicial** • **Año inicial** • **Dia final** • **Mes final** • **Año final**	
Resultados: el reporte ha sido generado.	

TRAZABILIDAD

REQUERIMIENTO	MÉTODOS	CLASES
R.# 1. Registrar a los clientes humanos y a sus respectivas mascotas	+setPetClient.add(ArrayList<Pet> petClient):void +setOwner.add(Client owner):void	Pet Client
R.# 2. Conocer la disponibilidad de un minicuarto	+ShowRoomsAvaliables():String	Veterinary
R.#3. Crear historia clínica de las mascotas	+setThePetRecord(Pet thePetRecord):void +setThePet(MedicalRecord thePet):void +setTheClient(Client theClient):void	MedicalRecord MedicalHistory
R.# 4. Dar de alta a la mascota	+setStatus(String status):void +setPetRoom(Pet petRoom):void	MedicalRecord Room Main
R.# 5. Realizar informe de las historias clínicas de las mascotas.	+ showInformationHistorial():String	Veterinary
R.# 6. Calcular el costo de una hospitalización	+calculateCostOfHospitalization():String +getTotalDays() +priceMedicineTotal	Veterinary MedicalRecord Medicine
R.# 7. Conocer los ingresos por hospitalizaciones.	+ calculateEarnings()	Veterinary
R.# 8. Saber el número del cuarto en el cual está ubicada cada mascota	+showInformation():String	Veterinary
R.# 9. Consultar el historial de una mascota.	+ showInformationHistorial()	Veterinary
R.# 10. Calcular el índice de masas corporal de una mascota	+ calculateBmiOfAPet(): String	Pet

R.# 11. Actualizar los datos básicos de un cliente.	+ changeDataBasic(String clientToChange, String NewAdress, int newPhone):String + updateTheBasicDataOfTheClient(String NewAdress, int newPhone):void	Veterinary Client
R.# 12. Agregar nueva medicina al historial médico de una mascota.	+ addNewMedicine2(String nameOwner, String namePet, String name, String dose, double price, int frecuency): void + addNewMedicine(String name, String dose, double price, int frecuency):void	Veterinary MedicalHistory
R.# 13. Agregar nuevas notas al historial médico de una mascota.	+ addNewNotes(String nameOwner, String namePet, String noteDiagnosis):void + addNewNotesToAMedicalRecord(String noteDiagnosis):void	Veterinary MedicalHistory
R.# 14. Agregar nuevos síntomas al historial médico de una mascota.	+ addNewSymptomToAMedicalHistory(String nameOwner, String namePet, String symptomToAdd):void + addNewSymptom(String symptomToAdd):void	Veterinary MedicalHistory
R.# 15. Calcular los ingresos	+ calculateEarningsForServices():String	Veterinary
R.# 16. Calcular los ingresos totales de la veterinaria	+ calculateTotalEarningsOfVeterinary():String	Veterinary
R.# 17.Calcular el promedio de ingresos por servicios.	+ calculateTheAverageForServicess():String	Veterinary
R.# 18.Calcular el promedio de ingresos en la veterinaria en una semana	+ IncomeBetweenAWeek(int day, int month, int year):String	Veterinary
R.# 19.Generar reporte de servicios prestados dada una fecha final y una inicial	+ reportWithADateGiven(int dayI, int monthI, int yearI, int dayF, int monthF, int yearF):String	Veterinary