**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](https://drive.google.com/open?id=15-uGtrubTf_2ofQrKuNT5Yojk84_jEjn6zs_zSSQsJw). 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](https://drive.google.com/open?id=1Jcn8mAtA4PIwdB5DI9XPb_f6qeKGHGookN-FhD3HXu4).

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**](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](https://drive.google.com/open?id=1LtTrG0MuvHzrmX16AOh_VTV9-Wni7wtzWUAfV4e4oGI)).

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.

**Desarrollo del enunciado 4:**

1. Desarrollar una aplicación a una veterinaria que me permita mostrar las funcionalidades requeridas por la veterinaria “*Mi Pequeña Mascota”,* y además me permita saber los datos de cada mascota, de cada cliente, los medicamentos recetados y los mini cuartos, y ademas de las funcionalidades del tipo de servicio.

|  |  |
| --- | --- |
| Nombre: | R1 Calcular la masa corporal |
| Resumen: | Permitir calcular la masa corporal de las mascotas |
| Entradas: | Ninguna |
| Salida: | Muestra el índice de la masa corporal de la mascota |

|  |  |
| --- | --- |
| Nombre: | R2 Actualizar la información del cliente con su dirección y teléfono |
| Resumen: | Poder actualizar la información del usuario de acuerdo a su información |
| Entradas: | * La identificación del usuario * La dirección del usuario * El teléfono del usuario |
| Salida: | Se actualiza la información del usuario |

|  |  |
| --- | --- |
| Nombre: | R3 Agregar medicinas a las mascotas hospitalizados |
| Resumen: | Poder agregar medicinas a las mascotas hospitalizadas |
| Entradas: | * La identificación del usuario * El nombre de la mascota * El nombre del medicamento * La dosis del medicamento * El costo del medicamento * La frecuencia del medicamento |
| Salida: | Se agrega el medicamento al animal hospitalizado |

|  |  |
| --- | --- |
| Nombre: | R4 Agregar notas a las mascotas hospitalizados |
| Resumen: | Poder agregar notas a las mascotas que se encuentren hospitalizadas |
| Entradas: | * La identificación del usuario * El nombre de la mascota * Las notas |
| Salida: | Se agrego las notas a las mascotas hospitalizadas en los cuartos |

|  |  |
| --- | --- |
| Nombre: | R5 Agregar nuevos síntomas a las mascotas hospitalizados |
| Resumen: | Poder agregar un síntoma a la mascota hospitalizada en el minicuarto |
| Entradas: | * La identificación del usuario * El nombre de la mascota * El síntoma de la mascota |
| Salida: | Se agreo el síntoma de la mascota hospitalizada |

|  |  |
| --- | --- |
| Nombre: | R6 Calcular los ingresos por servicios |
| Resumen: | Permitir calcular los ingresos de los servicios |
| Entradas: | Ninguna |
| Salida: | Muestra los ingresos de cada servicio |

|  |  |
| --- | --- |
| Nombre: | R7 Calcular los ingresos totales de los servicios |
| Resumen: | Permitir calcular los ingresos totales de los servicios |
| Entradas: | Ninguna |
| Salida: | Se calcula los ingresos totales de todos los servicios |

|  |  |
| --- | --- |
| Nombre: | R8 Agregar al sistema nuevos servicios prestados por la veterinaria, es decir los servicios vendidos no nuevos tipos de servicios. |
| Resumen: | Poder agregar al sistema servicios vendidos al sistema |
| Entradas: | * El tipo de servicio * La mascota |
| Salida: | Se agrega al sistema los servicios vendidos |

|  |  |
| --- | --- |
| Nombre: | R9 Promedio de ingresos por servicios. |
| Resumen: | Permitir calcular el promedio de todos los servicios |
| Entradas: | Ninguna |
| Salida: | Se calcula el promedio de todos los servicios |

|  |  |
| --- | --- |
| Nombre: | R10 Promedio de ingresos de la veterinaria en una semana. |
| Resumen: | Permitir calcular los ingresos en la veterinaria de una semana |
| Entradas: | * El dia actual * El mes actual * El año actual |
| Salida: | Se calcula los ingresos por una semana |

|  |  |
| --- | --- |
| Nombre: | R11 Reporte de servicios prestados dada una fecha inicial y una fecha final. |
| Resumen: | Permitir dar un reporte de las fechas dentro de esas semanas |
| Entradas: | * Fecha inicial * Fecha final |
| Salida: | Muestra la informacion que este dentro de esa semana |

4.

|  |  |  |
| --- | --- | --- |
| RF | Código | |
| Clase | Método |
| Calcular la masa corporal | Pet | + calculateIBMOfAnimal() |
| Actualizar la información del cliente con su dirección y teléfono | Veterinary | + actualizePhoneAndAddressOfCLient(long,String,String) |
| Agregar medicinas a las mascotas hospitalizados | Veterinary,Room,Pet | +addMediceToHospitalization(long String,String,double,double,int):String  +addMedicamentsToPet(String,double, double,int):String  + addPetToHistoryCLinical(String,double,double,int) |
| Agregar notas a las mascotas hospitalizados | Veterinary,Room,Pet | +addNotesToPet(String): void  +addNotesToHospitalization(long,String String):void  +addNotesToHospitalizationFatality(String): void |
| Agregar nuevos síntomas a las mascotas hospitalizados | Veterinary,Room,Pet | +addNotesToSympTom(String):void  +addNewSymptom(long,String,String):void  +addNewSymptomHistory(String):void |
| Calcular los ingresos por servicios | Veterinary | +calculatedService1():double  +calculatedService2():double  +calculatedService3():double  +calculatedService4():double  +calculatedService5():double |
| Calcular los ingresos totales de los servicios | Veterinary | +calculatedCostTotalForTheService():double |
| Agregar al sistema nuevos servicios prestados por la veterinaria, es decir los servicios vendidos no nuevos tipos de servicios. | Veterinary, typeOfService | + addServiceWithPet(typeOfService,Pet):void |
| Promedio de ingresos por servicios. | Veterinary | +promediateCostOfServices():double |
| Promedio de ingresos de la veterinaria en una semana. | Veterinary | +calculatedCostForOneWeek(int,int,int) : double |
| Reporte de servicios prestados dada una fecha inicial y una fecha final. | Veterinary | +getReportedAboutAnimalWithInicialDate AndFinalDate(String, String):String |