

En la unidad 4 de nuestro curso hemos trabajado sobre la aplicación de mecanismos de reutilización como son la herencia y el polimorfismo. Estas técnicas enriquecen el comportamiento de las clases modeladas como parte de una solución en el diagrama de clases. Además, aprendimos como utilizar las interfaces para independizar los contratos funcionales de las implementaciones, con el fin de hacer más flexible y fácil, los cambios sobre el programa construido.

En la unidad 5 de nuestro curso hemos aprendido a utilizar el concepto de matriz como elemento de modelaje, pudiendo así agrupar los elementos del modelo de la solución en una estructura contenedora de dos dimensiones de tamaño fijo y estructuras contenedoras lineales de tamaño variable como elementos de modelado que permiten manejar una secuencia de objetos.

La siguiente tarea integradora involucra una actividad en la cual se requiere aplicar todos los conocimientos adquiridos en estas unidades y verificar de esta manera el cumplimiento de los objetivos que han sido planteados para las unidades 4 y 5 descritas en el programa del curso.

Actividades

Para llevar a cabo este ejercicio es necesario realizar las actividades de cada una de las etapas de desarrollo de software:

1. Análisis del problema, para producir la lista de requerimientos funcionales usando el formato visto en clase.
2. Diseño de la solución. Elabore un diagrama de clases que modele la solución del problema de acuerdo con las buenas prácticas y los patrones de diseño revisados hasta el momento en el curso. Su diagrama debe incluir el paquete modelo y el de interfaz de usuario **y debe estar en inglés.**
3. Realice un diagrama de objetos teniendo en cuenta su diagrama de clases.
4. Implementación en Java. Incluya en la implementación, los comentarios descriptivos sobre los atributos y métodos de cada clase. Los nombres de las clases, los atributos, las relaciones, los métodos y las variables **deben estar en inglés.**
5. Documentación en JavaDoc (Debe entregarse el JavaDoc generado y ubicarlo en la carpeta docs).

6. Usar GitHub como repositorio de código fuente y documentación utilizando la estructura de carpetas aprendida en clase. Recuerde que se debe evidenciar su avance a través de los días en el desarrollo de su tarea.

7. Subir a moodle los puntos anteriores el plazo máximo es

Recuerde que puede encontrar la Rúbrica de la tarea integradora en el siguiente [enlace](#).

Nota:

- Usted debe entregar la URL de su repositorio GitHub 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:

petCenter/
src/
bin/
docs/

- Dentro de los directorios **src/** y **bin/** estarán presentes estos directorios(representando cada uno de sus paquetes):

ui/
model/





- El directorio src (source code) contiene sus clases .java dentro del directorio ui/ y model/. Por otro lado el directorio bin (binary files) contiene los archivos .class en el directorio ui/ y model/. El directorio docs tendrá toda la documentación de análisis y diseño
- Su código debería compilar de acuerdo con lo explicado en la diapositiva 15 de esta presentación: <http://tinyurl.com/y3bd9bg2>

A continuación, encontrará un enunciado que narra de forma detallada la situación problemática que se espera usted solucione.

Enunciado

El centro veterinario necesita de una ampliación de la solución realizada para la tarea integradora porque desea habilitar la zona de hospitalización y guardería.

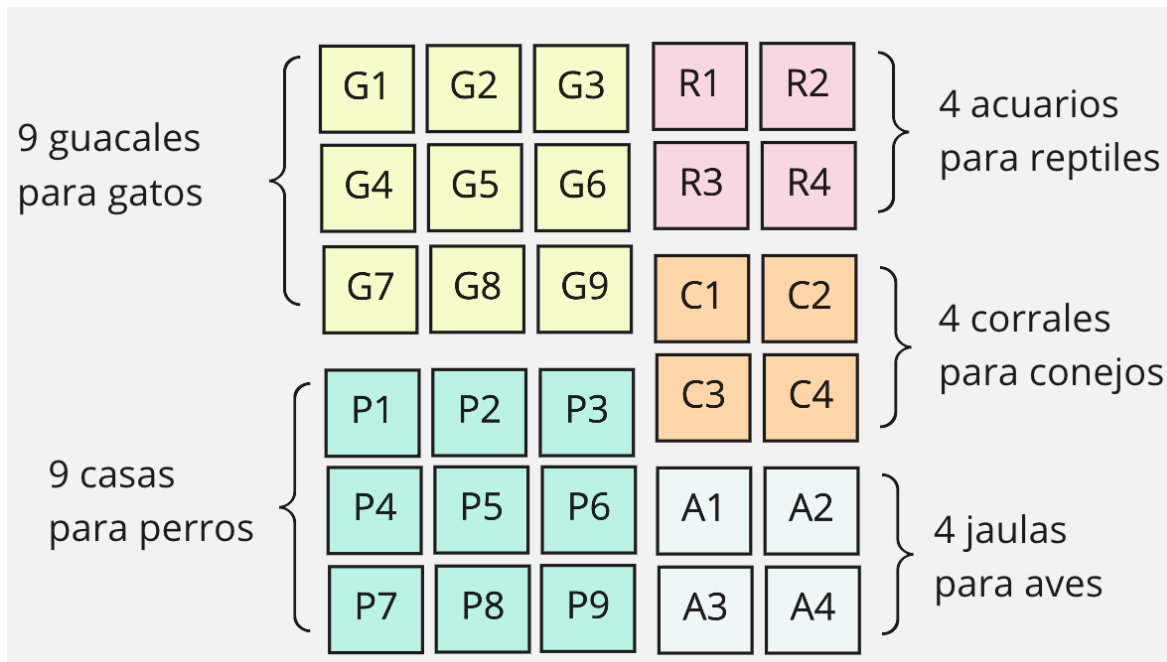
La zona de hospitalización y guardería está conformada por 30 hábitats organizados de la siguiente forma:

 Zona de gatos, 9 hábitats	 Zona de reptiles, 4 hábitats
 Zona de perros, 9 hábitats	 Zona de conejos, 4 hábitats  Zona de aves, 4 hábitats

Cada uno de los hábitats tiene un identificador, unas dimensiones (largo y ancho en metros) y un tipo de uso (vacío, ocupado-sano, ocupado-enfermo)

Además, de forma particular, la zona de gatos está compuesta por 9 guacales. De cada guacal se registra además la altura y el peso máximo soportado en el guacal. Por otra parte, la zona de perros tiene 9 casitas, cada una con una cantidad de juguetes a disposición del perrito que la ocupa. Los hábitats de la zona de reptiles son acuarios que tienen un tipo (para anfibios, y solo tierra) y un material para poder albergar distintos tipos de reptiles. En la zona de conejos encontramos cuatro corrales que se caracterizan por tener unas plantas para que el conejo se sienta cómodo. Cada corral tiene entonces un tipo de planta y una cantidad de esas plantas a disposición del animalito. Por último, en la zona de aves hay 4 jaulas que pueden ser colgadas (para canarios o loros) o en tierra, para aves de corral como gallinas, las jaulas se caracterizan también por una altura y la cantidad de aves que puede albergar.

Teniendo en cuenta la descripción de los hábitats, la guardería del centro veterinario estaría organizada de la siguiente forma:



Requerimientos no funcionales

1. Su programa debe preguntar al inicio si se desea acceder al servicio de emergencias veterinarias o si se quiere acceder al servicio de guardería-hospitalización (cada servicio tendrá un menú independiente). Ejemplo:

Al ejecutar un programa aparece un mensaje como el siguiente:

```
--Elija el servicio al que desea ingresar:
(1) Emergencias Veterinarias
(2) Guardería de mascotas
```

Dependiendo del tipo de servicio seleccionado se muestra el menú de emergencias veterinarias (tarea integradora 2) o el menú de guardería de mascotas.

2. Proponga una solución usando los conceptos de herencia y polimorfismo.

Nuevos requerimientos funcionales

Extienda su proyecto de tal forma que cumpla los siguientes requerimientos:

1. En el menú de emergencias veterinarias el programa debe permitir el ingreso a hospitalización de una mascota que ha sido atendida por alguno de los veterinarios. Para hacer la hospitalización es necesario:
 - Verificar que hay cupo en la zona que corresponde de acuerdo al tipo de mascota.

- Registrar la mascota un hábitat libre y cambiar el estado del hábitat a ocupado-enfermo.
- Terminar el registro notificando al usuario el identificador del hábitat en el que se aloja una mascota.
- 2. En el menú de guardería, el programa permite registrar el ingreso de una mascota en la guardería. Para hacer el ingreso es necesario:
 - Preguntar al usuario los datos de la mascota, del dueño de la mascota y la cantidad de días en que se usará el servicio.
 - Verificar que hay cupo en la zona que corresponde de acuerdo al tipo de mascota.
 - Registrar la mascota un hábitat libre y cambiar el estado del hábitat a ocupado-sano.
 - Terminar el registro notificando al usuario el identificador del hábitat en el que se aloja una mascota.
- 3. En el menú de guardería, dado el nombre de una mascota, mostrar si está en la guardería y reportar la zona, el identificador del hábitat, y si está enferma o sana.
- 4. En el menú de guardería, el programa imprime en la consola el mapa de la guardería mostrando el estado de cada hábitat.
Por ejemplo: Convención: ID Estado (E= ocupado enfermo, S=ocupado sano, V= vacío)
Es decir, en el ejemplo en gatos 1 hay una mascota enferma.

```
G1E - G2E - G3V - R1V - R2V  
G4V - G5V - G6V - R3V - R4V  
G7S - G8S - G9S - C1S - C2S  
P1E - P2E - P3V - C3V - P4V  
P4V - P5V - P6V - A1V - A2V  
P7S - P8S - P9S - A3S - A4S
```

- 5. En el menú de guardería, dado el identificador de un hábitat el programa muestra en la consola la información del hábitat y de la mascota que está alojada. Si no hay una mascota se imprime solo la información del hábitat y un mensaje anunciando al usuario que el hábitat está vacío.
- 6. Mostrar al usuario las siguientes estadísticas:
 - Porcentaje de ocupación por cada zona en la guardería.
 - Porcentaje de ocupación general de la guardería.
 - Porcentaje de mascotas sanas y porcentaje de mascotas hospitalizadas.