UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Materia:	Programación II		
Nivel:	2º Cuatrimestre		
Tipo de Examen:	Primer Parcial		
Apellido(1):		Fecha:	24 oct 2024
Nombre/s(1):		Docente a cargo(2):	Baus /Quiroz
División(1):	121-2	Nota(2):	
DNI(1):		Firma(2):	

(1) Campos a completar solo por el estudiante en caso de imprimir este enunciado en papel.
(2) Campos a completar solo por el docente en caso de imprimir este enunciado en papel.

Sistema Zoológico

Nos encargan implementar un sistema básico para un Zoológico.
En el zoológico se administra la información de diferentes tipos de animales como mamíferos, aves y reptiles. Cada animal cuenta con un nombre y su edad. Los animales además tienen un peso y un tipo de dieta (**HERBIVORO, CARNIVORO, OMNIVORO).**
Las aves cuentan además con la envergadura de las alas y los reptiles con tipo escama y regulación temperatura. Tanto los mamíferos como las aves deben poder vacunarse llamando a su método vacunar.

- **agregarAnimal()** : El empleado del zoológico debe poder agregar todo tipo de animales al mismo. Se deberá lanzar una excepción personalizada si ya existe el animal. (dos animales son iguales si tienen el mismo nombre y la misma edad)

- mostrarAnimales() : Muestra todos los animales donde se pueden observar los siguientes atributos.(dedúzcalos de la imagen de ejemplo):

```
Mamifero [peso=12.5, dieta=CARNIVORO]
Reptil [tipoEscama=queratinosa, regTemperatura=ectotermia]
Ave [envergaduraAlas=54.7
```

-vacunarAnimales() : Vacunar todos los animales “vacunables”. Informar los que no se puedan vacunar.

A partir del enunciado anterior se solicita:

- Realizar el diagrama de clases completo en papel.
- Realizar el código fuente en Java que resuelva las funcionalidades solicitadas.

Objetivos de Aprobación No Directa (Calificación de 4 a 5 puntos):

1. **Diagrama de clases**:
   - El diagrama de clases debe reflejar correctamente las relaciones entre las clases
   - Se deben mostrar atributos como el nombre y la edad en la clase Animal, y los atributos específicos de cada subclase.
   - Debe incluir la herencia correctamente.
   - Los métodos mencionados en el enunciado (vacunar, agregarAnimal, etc.) deben estar representados.
2. **Clases y Herencia**:
   - El código debe incluir la clase base y las subclases.
   - Cada clase debe tener los atributos solicitados.
   - El uso de herencia debe estar implementado correctamente.
3. **Métodos básicos**:
   - El método agregarAnimal() debe agregar animales al Zoo y lanzar una excepción si ya existe uno con el mismo nombre y edad.
   - El método mostrarAnimales() debe mostrar todos los animaless con sus atributos.
   - El método vacunarAnimales() debe permitir vacunar animales vacunables.
4. **Excepciones**:
   - El manejo de excepciones debe ser básico, pero efectivo. Debe lanzarse una excepción personalizada cuando se intenta agregar un animal duplicado.
5. **Funcionamiento general**:
   - El programa debe compilar y ejecutarse sin errores.
   - Las funciones principales deben estar correctamente implementadas y verificables a través de ejemplos simples.

Objetivos de Aprobación Directa (Calificación de 6 a 10 puntos):

Diagrama de clases detallado:

- El diagrama de clases no solo debe mostrar las relaciones entre clases y métodos, sino que también debe incluir relaciones como composiciones o agregaciones (si aplica).
- Deben incluirse visibilidades correctas para atributos y métodos (privado, público, protegido).
- El diagrama debe demostrar un correcto entendimiento de la arquitectura del sistema.

Implementación completa y estructurada:

- El código debe utilizar correctamente los principios de POO, como encapsulamiento y uso de modificadores de acceso adecuados.
- El uso de enumerados (`enum`) debe estar correctamente implementado.
- Deben existir comentarios claros y descriptivos en el código, explicando las partes más complejas.
- El código debe incluir métodos adicionales útiles (como un método `toString()` para cada tipo de animal).

Manejo avanzado de excepciones:

- El manejo de excepciones debe ser más robusto. Por ejemplo, se espera la definición de excepciones personalizadas con mensajes claros y manejos múltiples de casos excepcionales.
- Uso adecuado de bloques `try-catch` y mensajes descriptivos.

Métodos y funcionalidades extras:

- Se espera que el método vacunarAnimales`()` no solo vacune, sino que maneje de manera elegante qué animales pueden vacunarse y cuáles no, con reportes claros.

Eficiencia y uso de colecciones adecuadas:

- El programa debe usar colecciones.

Funcionalidad completa y ejemplos:

- El código debe incluir un conjunto de pruebas o un menú interactivo que permita verificar el correcto funcionamiento del sistema.
- Todo debe funcionar de manera fluida sin errores y debe cubrir todos los casos solicitados en el enunciado, incluyendo la lectura y la validación de publicaciones duplicadas.