

Nombre: Juan Martín Iglesias	Padrón: 107018
Corrigió: Valdez Santiago	Nota: 8 (ochos)

Cada ítem del examen se va a calificar con un valor entre 0 y 1. Para aprobar el examen se deberán cumplir todas las siguientes condiciones:

- En el primer ejercicio se debe tener al menos un puntaje de 0,6.
- Se debe sumar como mínimo el 60% del puntaje total obtenido mediante la ponderación de cada ejercicio (ver el coeficiente de ponderación en cada ejercicio).

1 - Ejercicio de modelado (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos. Ponderación: 70%

En el sistema de alquiler de mascotas AlgoPupis, tenemos los siguientes elementos::

- Usuarios: las personas anotadas en el sistema de alquiler.
- Mascotas: Las mascotas a alquilar, poseen una ubicación y un estado de energía (%) que indica que tanto están dispuestos a ser mascotas hasta necesitar descansar.

Además los usuarios pueden ser de tipo:

- Estándar: el sistema les ofrece mascotas con estado de energía de 30% a 70%.
- Intensos: el sistema les ofrece también mascotas con estado de energía de 60% a 100%.

Las mascotas pueden ser de tipo:

- AlgoPupi estándar: pueden ser alquiladas por cualquier usuario.
- AlgoPupi superEnergético: sólo pueden ser alquiladas por usuarios Intensos.

Suponer la existencia de una clase Ubicacion¹ a quien puede enviarse un mensaje del siguiente tipo:

unaUbicacion obtenerDistanciaEnMetrosA: otraUbicacion.

El alumno pensará y modelará: (colocar la respuesta a cada uno de estos puntos en una hoja separada)

- ◆ **Modelo 1.** Un diagrama de secuencia para obtener los AlgoPupis en un radio de 12 metros de distancia que pueden ser alquilados por un usuario. Contamos con 2 AlgoPupi estándar y 2 AlgoPupi superEnérgicos (uno con 65% y el otro con 90% de energía) todos ubicados a 10 mts. del cliente. *Incluir en el diagrama la inicialización de los objetos.*
- ◆ **Modelo 2.** Un diagrama de clases que muestre las clases que intervienen en el Modelo 1, y las relaciones entre ellas.
- ◆ **Modelo 3.** Dos pruebas automatizadas que modelen el caso anterior (en pseudocódigo).

Para evaluar conceptos teóricos: (10% cada pregunta) (contestar sobre esta hoja, no es necesario justificar)

2 - Un método abstracto sirve para implementar la visibilidad privada en Smalltalk

Verdadero Falso

X (2/3)

3 - En general, la delegación provoca dependencias menos acopladas que la herencia

Verdadero Falso

✓

4 - Una clase abstracta no puede tener métodos concretos

Verdadero Falso

✓

¹ Ubicacion puede tener el contrato que se necesite de ser necesario.



USER

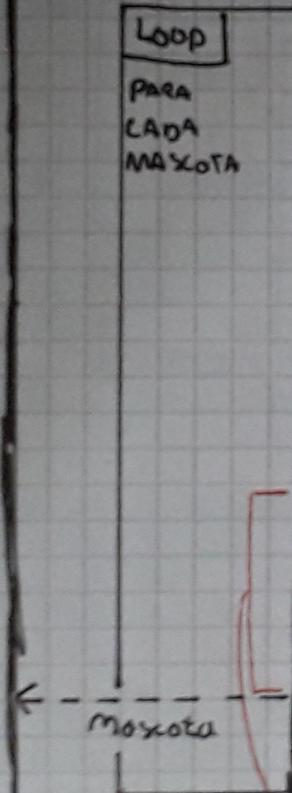
alquiler:
Algo Pups

obtener, MascotaAlq.
obtener(MascotaAlq., d:int
:int)

Mascota:
Mascota SuperEnergética

Usuario:
UsuarioIntenso

Ubicación Usuario:
Ubicación



DJO Con
los mensajes is-dp y Tell don't ask
Sofia mejor delegarlo
en el Objeto y que
este luego pregunte
a la mascota

Juan Martín Iolestan
10 70 18

Nº 1

En este ejemplo monto con un bucle intenso y dentro de la memoria puse, una super energética para mover cada piso.

AlgoPups

- + obtener_Mascotas_Negociables (Usuario, int): array de Mascotas
- + cargar_Ubicacion (Usuario)
- + cargar_Mascota (Mascota)

Abstrac >> Usuario

- energiaMin: int
- energiaMax: int
- + Obtener_Distancia_EnMetros (Ubicacion): int
- + testa_Energia_En_Porcentaje (int): bool
- + puede_Si_Requiere_Intenso (bool): bool

Abstrac >> Mascota

- energiaMax: int
- requiere_Intenso: bool
- + Obtener_Ubicacion (): arr.Ubicacion
- + Consultar_Energia (): int
- + requiere_Intenso (): bool

Usuario Intenso

- Energia Min: 80
- Energia Max: 100
- + obt_Dist_EnMe (Ub): int
- + testa_En_Per (int): bool
- + Puede_SiReqInt (bool): bool

Usuario Estándar

- energia Min: 30
- energia Max: 70
- + obt_Dist_EnMe (Ub): int
- + testa_En_Per (int): bool
- + Puede_SiReqInt (bool): bool

Mascota Estándar

- energiaMax: int
- requiere_Intenso: False
- + obt_Ub (): Ubicacion
- + cons_En (): int
- + req_Intenso (): bool

Mascota Super Energética

- energiaMax: int
- requiere_Intenso: True
- + obt_Ub (): Ubicacion
- + cons_En (): int
- + req_Intenso (): bool

return true
(siempre puede)

si lo
puede
si es
estandar
(clase macth)

if false == False
elijo true

Ubicación

- + obtener_Distancia_EnMetros (Ubicacion): int

2070 18

TEST 1: Usuario Estándar Recibe Los Mascotas Dibujos

alquiler = new AlgoPupis

alquiler. cargar Usuario (Usuario Estándar)

alquiler. cargar Mascota (Mascota Estándar, $\text{no energy} = 40$)

- : mascota1 → estandar $e=40$
- : mascota2 → SE $e=90$
- : mascota3 → estandar $e=60$
- : mascota4 → estandar $e=100$
- : SE $e=65$

SUPONIENDO
QUE TODAS ESTAN
EN RANGO DE
DISTANCIA

mascotas

mascotas Posibles Objetivo = [mascota1, mascota2] ✓

assert(alquiler. obtener Mascotas Alquilables (UsuarioEstándar, 12) == mascotas Posibles Objetivo) ✓

TEST 2: Usuario Intenso Recibe Los Mascotas Dibujos

alquiler = new AlgoPupis

alquiler. cargar Usuario (Usuario Intenso)

CARGA LAS MISMAS MASCOTAS QUE TEST 1 ✓

mascotas Posibles Objetivo = [mascota2, mascota3, mascota4]

assert(alquiler. obtener Mascotas Alquilables (UsuarioIntenso, 12) == mascotas Posibles Objetivo) ✓