Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2024-1

Midterm

12 de abril 2024

1. Selección múltiple [50 %]

Cada alternativa correcta otorgaba **3 décimas** a esta sección del *midterm*.

1. E
2. A
3. C
4. E
5. B
6. C
7. C
8. A
9. A
10. D
11. D
12. C
13. A
14. B
15. A
16. D
17. E
18. B
19. B
20. B

2. Desarrollo [50 %]

2.1. Pregunta 1 (Modelación OOP) [6 puntos]

Inciso 1 (2 puntos):

Respuesta: El orden de los prints sería el siguiente:

1. AyudanteHibrido
2. AyudanteCatedra
3. Ayudante
4. Estudiante
5. Humano
6. Mamifero
7. Docente
8. Humano
9. Mamifero
10. AyudanteCorrector
11. Ayudante
12. Estudiante
13. Humano
14. Mamifero
15. Docente
16. Humano
17. Mamifero

Desglose puntaje:

- Se entregan **2 puntos** si se escriben todos los prints en el orden correcto.
- Se entrega **1 punto** si hay un error en la escritura de los prints de máximo 1 línea.
- Se entregan **0 puntos** en cualquier otro caso.

Inciso 2 (1 punto):

Respuesta: El problema existente es la forma en que están escritos los métodos inicializadores de las superclases en cada subclase del código. Al llamar explícitamente a los inicializadores de cada superclase, se están haciendo múltiples llamadas redundantes a las superclases (veáse la pregunta anterior donde Humano y mamífero se activan 4 veces cada uno). Esto se llama problema del diamante.

La solución es reemplazar las llamadas explícitas por un `super()` y así Python se encargará de generar el MRO de tal forma que solo llame una vez a cada superclase.

Desglose Puntaje:

- **0.3 puntos** por mencionar que el problema se denomina problema del diamante
- **0.4 puntos** por explicar en qué consiste el problema en este caso.
- **0.3 puntos** por indicar la solución para este caso (usar `super()`)

Inciso 3 (1 punto):

Las clases abstractas son:

- **Mamífero (Obligatoria)**: No es posible tener mamíferos genéricos sin especie, por lo que debe ser abstracta.
- **Humano (Opcional)**: Dependerá de si se entrega una justificación adecuada.
- **Ayudante (Opcional)**: Dependerá de si se entrega una justificación adecuada.

Desglose Puntaje:

- Clases indicadas (0.5 puntos):

- **0.5 puntos** si dice Mamífero y cualquiera de las opcionales (0, 1 o las 2).
- **0.25 puntos** si no dice Mamífero, pero dice una o más de las opcionales.
- **0 puntos** si no indica ni mamífero ni alguna de las opcionales.

- Justificación (0.5 puntos):
  - **0.5 puntos** si justifica correctamente Mamífero y todas las opcionales que haya puesto.
  - **0.25 puntos** si identificó 2 o más clases (Mamífero y 1 o 2 de las opcionales), pero justificó mal una
  - **0.25 ptos** si no identificó Mamífero, pero identificó 1 o 2 opcionales y las justificó bien.
  - **0 puntos** en cualquier otro caso.

Inciso 4 (2 puntos):

Respuesta: El atributo Hambre debería ir en Mamífero, la clase más alta, debido a que todos los mamiferos tienen la capacidad de sentir hambre, por lo que todas las subclases deberían tener ese atributo. Sin embargo, se puede permitir que se asigne en Humano si se entrega una buena justificación de por qué ahí y no en mamífero.

Sobre cómo modelarla, Hambre debería ser programada como una property, la cual posee un getter y un setter que evite que su valor salga de los rangos pedidos por el enunciado.

Desglose Puntaje:

- **1 punto** por colocar el atributo en la clase correcta y justificar adecuadamente.
- **1 punto** por indicar que el atributo debe ser una property que restringe valores y tiene un setter.

2.2. Pregunta 2 (Análisis de código) [6 puntos]

A diferencia de la pregunta anterior, los cuatro incisos tienen el mismo desglose de puntaje. **Desglose Puntaje:**

- **0.5 puntos** por responder verdadero/falso de manera consistente con la justificación
- **1 punto** por justificar correctamente.
  - **1 punto** si entrega una justificación completa, haciendo referencia al código y contenidos del cursos relacionados a la pregunta.
  - **0.5 puntos** si entrega una justifición que hace referencia al código y contenidos del cursos, pero no considera todos los contenidos necesarios para la pregunta.

Inciso 1 (1.5 puntos):

Alternativa 1: Verdadero, un `deque` tiene el método `append`, por lo que `agregar_a_la_cola` funciona correctamente. Además, en `escuchar_siguiente_cancion` se utiliza `popleft()`, que también corresponde a un método de `deque`.

Alternativa 2: Falso, el método `agregar_a_la_cola` funcionará correctamente si `self.cola` es una lista o cualquier otra estructura de datos que tenga el método append. Por lo tanto `self.cola` no **debe** ser un `deque` y puede ser otra estructura.

Inciso 2 (1.5 puntos):

Verdadero, si bien un stack no es un tipo de dato que esté implementado en Python, las listas de Python pueden ser utilzadas como stack haciendo `append` y `pop`.

Inciso 3 (1.5 puntos):

Alternativa 1: Falso, el atributo `self.favoritos` es un `set`, lo que significa que no tiene el método `appened`. Esto hará que el código falle, por lo que no se agregará la canción a favoritos.

Alternativa 2: Falso, el atributo `self.favoritos` es un `set`, lo que significa que no puede guardar repetidos, por lo que la segunda vez que se intente agregar la canción, no se modificará el `set`.

Inciso 4 (1.5 puntos):

Verdadero, las `named_tuples` permiten acceder a sus atributos a través de sus nombres o de sus índices. Observando la creación de `Cancion`, podemos ver que su primera entrada será nombre y la segunda artista, haciendo que tengan los índices 0 y 1, respectivamente, y que ambos `print` sean idénticos.