

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2025-1

Midterm

11 de Abril 2025

1. Selección múltiple [50 %]

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

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

2. Desarrollo [50 %]

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

Clases (2 puntos)

Desglose puntaje

- **1.6 puntos** por definir correctamente las clases que poseen herencia (`MarkAlpha`, `MarkBeta`, `MarkGamma`, `SuperHeroe`, `SuperVillano`).
- **0.2 puntos** por definir correctamente la clase `Mark`.
- **0.2 puntos** por definir correctamente la clase `Realidad`.

Posible Respuesta

Cuadro 1: Tabla de clases

Clase	¿Es Abstracta?	¿Presenta herencia?	Hereda de
`Mark`	No	No	-
`MarkAlpha`	No	Sí	Mark, SuperHeroe
`MarkBeta`	No	Sí	Mark, SuperVillano
`MarkGamma`	No	Sí	Mark, SuperHeroe, SuperVillano
`SuperHeroe`	Sí	Sí	ABC
`SuperVillano`	Sí	Sí	ABC
`Realidad`	No	No	-

Atributos (2 puntos)

Desglose puntaje

- **0.2 puntos** por indicar uno o más atributos de cualquier clase.
- **0.9 puntos** por definir correctamente los atributos de las clases poseen `properties` (`Mark`, `MarkAlpha`, `MarkBeta`).
- **0.9 puntos** por definir correctamente los atributos de las clases que no poseen `properties` (`MarkGamma`, `Realidad`).

Posible Respuesta

Cuadro 2: Tabla de atributos `Mark`

Nombre	Tipo de dato	¿Es property?
`realidad_origen`	`Realidad`	No
`vida_base`	`int` o `float`	No
`fuerza`	`int` o `float`	No
`agilidad`	`int` o `float`	No
`defensa`	`int` o `float`	No
`nivel_de_poder`	`int` o `float`	Sí

Cuadro 3: Tabla de atributos `MarkAlpha`

Nombre	Tipo de dato	¿Es property?
`nombre_accesorio`	`str`	No
`poder_accesorio`	`int` o `float`	No
`nivel_de_poder`	`int` o `float`	Sí

Cuadro 4: Tabla de atributos `MarkBeta`

Nombre	Tipo de dato	¿Es property?
`material_piel`	`str`	No
`factores_piel`	`dict`	No
`nivel_de_poder`	`int` o `float`	Sí

Cuadro 5: Tabla de atributos `MarkGamma`

Nombre	Tipo de dato	¿Es property?
`figura_simbolo`	`str`	No
`intensidad_simbolo`	`int` o `float`	No

Cuadro 6: Tabla de atributos `Realidad`

Nombre	Tipo de dato	¿Es property?
`nombre`	`str`	No
`en_guerra`	bool	No
`descripcion`	`str`	No

Métodos (2 puntos)

Desglose puntaje

- **1 punto** por definir correctamente los métodos de las clases poseen 1 solo método (`Mark`, `MarkAlpha`, `MarkBeta`, `SuperHeroe`, `SuperVillano`).
- **1 punto** por definir correctamente los métodos de las clases poseen 2 solo método (`MarkGamma`).

Posible Respuesta

Cuadro 7: Métodos de `Mark`

Nombre	Argumentos y sus tipos	Tipo de dato que retorna	Descripción breve
`atacar_enemigo`	enemigo: `Mark`	`None`	Método encargado de debilitar al enemigo.

Cuadro 8: Métodos de `MarkAlpha`

Nombre	Argumentos y sus tipos	Tipo de dato que retorna	Descripción breve
`atacar_enemigo`	enemigo: `Mark`	`None`	Método encargado de debilitar al enemigo, además llama a `encarcelar_enemigo` de `SuperHeroe`.

Cuadro 9: Métodos de `MarkBeta`

Nombre	Argumentos y sus tipos	Tipo de dato que retorna	Descripción breve
`atacar_enemigo`	enemigo: `Mark`	`None`	Método encargado de debilitar al enemigo, además llama a `eliminar_enemigo` de `SuperVillano`.

Cuadro 10: Métodos de `MarkGamma`

Nombre	Argumentos y sus tipos	Tipo de dato que retorna	Descripción breve
`atacar_enemigo`	enemigo: `Mark`	`None`	Método encargado de debilitar al enemigo, además llama a los métodos `eliminar_enemigo` y `encarcelar_enemigo` correctamente.
`canalizar_energia`	`None`	`None`	Método encargado de canalizar la energía, debe "encender el símbolo", por lo que aumenta su intensidad de 0 a algún otro número.

Cuadro 11: Métodos de `SuperHeroe`

Nombre	Argumentos y sus tipos	Tipo de dato que retorna	Descripción breve
`encarcelar_enemigo`	enemigo: `Mark`	`None`	Método encargado de encarcelar al enemigo.

Cuadro 12: Métodos de `SuperVillano`

Nombre	Argumentos y sus tipos	Tipo de dato que retorna	Descripción breve
`eliminar_enemigo`	enemigo: `Mark`	`None`	Método encargado de eliminar al enemigo.

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

2.2.1. Parte A [3 puntos]

Desglose puntaje

1. (1 pt.) `crear_sistema`.
   - (0.5 pt.) Usa `namedtuple`, o diccionario con los tres atributos.
   - (0.5 pt.) Para la instancia, usa dos `deques` vacíos y una forma de contar la cantidad de preferentes seguidos.
2. (1 pt.) `recibir_cliente`.
   - Agrega usuario a la `deque` correcta del sistema.
3. (1 pt.) `atender_cliente`.
   A) (0.5 pt.) Atiende a preferente.
      - (0.25 pt.) Hace correctamente los checkeos para ver si es válido hacer la operación. Se acepta tanto checkear `preferentes_seguidos` distinto de 5 como checkear que sea menor a 5.
      - (0.25 pt.) Actualiza los valores correctamente (ejemplo: si `preferentes_seguidos` es un `int`, y lo intenta de actualizar directamente en una `namdetuple` o tupla, se da la mitad del puntaje).

   B) (0.5 pt.) Atiende a normal.
      - (0.25 pt.) Hace correctamente los checkeos para ver si es válido hacer la operación.
      - (0.25 pt.) Actualiza los valores correctamente (ejemplo: si `preferentes_seguidos` es un `int`, y lo intenta de actualizar directamente en una `namdetuple` o tupla, se da la mitad del puntaje).

Posible Respuesta

Un código Python correcto:

```python
from collections import namedtuple, deque

def crear_sistema():
    Sistema = namedtuple('sistema', [
        'preferentes', 'normales', 'preferentes_seguidos'
    ])
    return Sistema(deque(), deque(), [0])

def recibir_cliente(sistema, cliente, es_prioritario):
    if es_prioritario:
        sistema.preferentes.append(usuario)
    else:
        sistema.normales.append(usuario)

def atender_cliente(sistema):
    if len(sistema.preferentes) > 0 and sistema.preferentes_seguidos[0] < 5:
        sistema.preferentes_seguidos[0] += 1
        sistema.preferentes.popleft()
    elif len(sistema.normales) > 0:
        sistema.preferentes_seguidos[0] = 0
        sistema.normales.popleft()
```

2.2.2. Parte B [3 puntos]

Desglose puntaje

1. (0.5 pt.) la estructura que guarda el *output* de `obtener_personas()` es adecuada, y la función se llama una sola vez.
   - (0.25 pt.) Se llama a `obtener_personas()` una sola vez.
   - (0.25 pt.) Se usa una estructura de datos no secuencial adecuada (es `set`, ya que `dict` no es adecuada al necesitar más información de la que se desea almacenar) para guardar las personas.
2. (0.5 pt.) `distribucion_jugadores`.
   - (0.25 pt.) Es un diccionario o lista.
   - (0.25 pt.) Guarda sets en su interior.
3. (1 pt.) Emular competencia entre jugadores de una ronda.
   - (0.5 pt.) Iterar de a pares sobre toda la estructura.
   - (0.5 pt.) Agregar los ganadores y perdedores adecuadamente.
4. (1 pt.) Emular rondas.
   - (0.4 pt.) Actualiza adecuadamente las personas que siguen jugando (los que aún no han perdido).
   - (0.1 pt.) Se juega hasta que quede un ganador.
   - (0.4 pt.) Agregar a `distribucion_jugadores` según victorias.
   - (0.1 pt.) Se cuentan y actualizan correctamente las victorias.

Posible Respuesta

Un código Python correcto:

```python
personas = set(obtener_personas())
victorias = 0
distribucion_jugadores = dict()

while len(personas) != 1:
    derrotados = set()
    ganadores = set()
    una_persona = None

    for persona in personas:
        if una_persona is None:
            una_persona = persona
            continue

        otra_persona = persona
        ganador = obtener_ganador(una_persona, otra_persona)

        if ganador == una_persona:
            derrotados.add(otra_persona)
            ganadores.add(una_persona)
        else:
            ganadores.add(otra_persona)
            derrotados.add(una_persona)

        una_persona = None

    distribucion_jugadores[victorias] = derrotados
    victorias += 1
    personas = ganadores

distribucion_jugadores[victorias] = personas
```