Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2024-2

Midterm

03 de Octubre 2024

Instrucciones

- La evaluación consta de dos partes: 20 preguntas de selección múltiple y 2 preguntas de desarrollo.
- Cada una de las secciones (alternativas y desarrollo) son independientes y valen un 50 % de la nota final. Para calcular la nota de cada sección, se considera que todas las preguntas de alternativas valen lo mismo, mientras que la sección de desarrollo indica cuánto vale cada pregunta.
- Recibirás una hoja de respuestas que deberás rellenar con tus datos y las respuestas correspondientes a cada alternativa o pregunta de desarrollo. Sólo se corregirá la hoja de respuestas. **Ten mucha precaución de anotar correctamente tus datos**.
- Cada pregunta de selección múltiple tiene únicamente 1 alternativa correcta. Responder con 2 o más alternativas implicará dejar inválida esa pregunta y se considerará incorrecta. Cada pregunta presenta el mismo puntaje y no se descontará por respuesta incorrecta.
- Debes completar con tus datos personales en cada hoja utilizada. Luego debes entregar de vuelta **todas** las hojas de respuesta (tanto para alternativas como de desarrollo) de la prueba, aunque estas se encuentren en blanco.
- Solo podrás retirarte de la sala una vez hayan transcurrido 40 minutos desde que inició de la evaluación.
- Durante la evaluación se realizarán 2 rondas de preguntas. Estas preguntas únicamente serán respondidas por los profesores del ramo.
- En caso de que sea necesario, podrás solicitar hojas blancas para apoyar al desarrollo de la prueba. Para esto levanta la mano y espera que un ayudante se acerque a tu puesto.

Selección múltiple

1. Respecto a la importación de módulos en Python, ¿cuál de las siguientes afirmaciones es **incorrecta**?

   A) Es posible cambiar el nombre de la variable que contiene las definiciones de un módulo.

   B) No es posible importar clases desde otros archivos.

   C) Cada archivo con extensión `.py` es un módulo, si tiene sentencias y definiciones.

   D) Aporta a una mejor organización y legibilidad de código.

   E) Es posible importar sólo algunas definiciones de un módulo.

2. ¿Cuál de los siguientes comandos de terminal utiliza una **ruta absoluta**?

   A) `cd ../../Syllabus/Tareas/`

   B) `mkdir ../Syllabus/Tareas/Tareas/T1`

   C) `cd /usuario/Escritorio/Syllabus/`

   D) `python ./Syllabus/Actividades/AC1/main.py`

   E) `ls ../Syllabus/Tareas/Tareas/T2`

3. Respecto al desempaquetamiento de estructuras de datos, ¿cuál de las siguientes líneas de código **no corresponde** a un uso correcto de operadores de desempaquetado?

   A) `print(*['valor 1', 'valor 2', 10, True])`

   B) `datos = {*('a', 1), *('b', 2)}`

   C) `datos = [*(1,2), 4, 5]`

   D) `datos = {**{'a': 1, 'b': 2}, **{'c': 3}}`

   E) `datos = **{'a': 1, 'b': 2, 'c': 3}`

4. Asumiendo que `producto` es un diccionario, la expresión "`'identificador' in producto`" retornará:

   A) La posición de la llave *'identificador'*.

   B) **True** si *'identificador'* se encuentra entre los valores de `producto`.

   C) **True** si *'identificador'* se encuentra entre las llaves de `producto`.

   D) **True** si *'identificador'* se encuentra entre las llaves de `producto`, o bien, entre sus valores.

   E) El valor asociado a la llave *'identificador'*.

5. Para el siguiente código, ¿cuál será el ***output* esperado**?

```
from collections import deque

def obtener_deseado(a: int, b: int, *args) -> int:
    if a < b:
        args = deque(args)
        a = args.popleft()
        return obtener_deseado(a, b, *args)
    return a

obtener_deseado(2, 5, 4, 6, 8)
```

A) 2

B) 5

C) 4

D) 6

E) 8

6. ¿Cuál o cuáles de las siguientes filas de esta tabla **conectan correctamente** las diferencias entre atributos de instancia y atributos de clase?

	Atributo de instancia	**Atributo de clase**
I.	Sólo se relaciona a una instancia particular de una clase.	Es compartida por todas las instancias de la clase.
II.	Al modificar el atributo, se modifica inmediatamente para todas las instancias de la clase.	No es posible modificar un atributo de clase.
III.	Se debe referenciar a la instancia para usarlo.	Se puede referenciar a la instancia o a la clase a la que pertenece para usarlo.

A) I y II

B) I y III

C) II y III

D) I, II y III

E) Ninguna de las filas conecta correctamente las diferencias.

7. Para el siguiente código, ¿cuál será el ***output* esperado**?

```
from typing import Iterable

class Equipo:
    id_equipo = 0

    def __init__(self) -> None:
        self.id = Equipo.id_equipo
        Equipo.id_equipo += 1

    def registrar_puntajes(self, resultados: Iterable, *args, **kwargs) -> None:
        self.resultados = set([*resultados, *args, *kwargs.values()])

def resultados_finales(equipo_a: Equipo, equipo_b: Equipo) -> set:
    metrica_1 = equipo_a.resultados | equipo_b.resultados
    metrica_2 = equipo_a.resultados & equipo_b.resultados
    return metrica_1 - metrica_2


mis_amigos = Equipo()
mis_rivales = Equipo()

mis_amigos.registrar_puntajes({0, 1, 4, 6}, ptje_extra = 2)
mis_rivales.registrar_puntajes({1, 3, 2, 7}, ptje_extra = 6)

puntajes = resultados_finales(mis_amigos, mis_rivales)

print(puntajes)
```

A) `{0, 4}`

B) `{0, 3, 4, 7}`

C) `{1, 2, 6}`

D) `{0, 1, 2, 3, 4, 6, 7}`

E) `{0, 2, 3, 4, 6, 7}`

8. Suponga que debe modelar mediante OOP una tienda de venta de libros físicos. ¿Cuál de los siguientes conjuntos de clases sería el **más recomendable** para modelar mediante herencia?

A) `CajaRegistradora` (superclase) y `Empleado` (subclase).

B) `CajaRegistradora` (superclase) y `Catálogo` (subclase).

C) `Estante` (superclase) y `Libro` (subclase).

D) `Libro` (superclase), `Novela` (subclase) y `Enciclopedia` (subclase).

E) `Catálogo` (superclase), `Novela` (subclase) y `Enciclopedia` (subclase).

9. ¿Qué conceptos de Programación Orientada a Objetos **se ven aplicados** en el siguiente código?

```
class ClaseA:
    def __init__(self, numero: str) -> None:
        self._numero = numero

    @property
    def numero(self) -> str:
        return self._numero

    @numero.setter
    def numero(self, cantidad_a_agregar) -> None:
        self._numero += cantidad_a_agregar

class ClaseB(ClaseA):
    def __init__(self, numero):
        super().__init__(numero)

    def numero(self):
        print('¿que ocurre?')

c_b = ClaseB(5)
print(c_b.numero(10))
```

I. Herencia

II. Polimorfismo

III. *Properties*

IV. Clases abstractas

A) I y II

B) I y III

C) I y IV

D) I, II y III

E) I, II, III, IV

10. Respecto al código de la pregunta anterior, ¿cuál es el **resultado de su ejecución**?

A) `5`

B) `10`

C) `15`

D) `¿que ocurre?`

E) Ocurre un error de ejecución.

11. Para el siguiente código, ¿cuál o cuáles de las siguientes afirmaciones son **correctas**?

```
from abc import ABC, abstractmethod

class Enemigo(ABC):
    def atacar(self):
        print("Enemigo atacando.")

    @abstractmethod
    def defender(self):
        pass

class Esqueleto(Enemigo):
    def atacar(self):
        return -15

    def defender(self):
        print("Esqueleto se defiende con su escudo.")

class Guerrero(Enemigo):
    def atacar(self):
        print("Guerrero ataca con su lanza.")

bill_el_arquero = Esqueleto()
```

I. La clase `Enemigo` se encuentra en el MRO de `Esqueleto`.

II. Al ejecutar "`bill_el_arquero.atacar()`", se imprimirá *"Enemigo atacando."*

III. Si se intenta instanciar `Guerrero` con "`soldado = Guerrero()`", se producirá una excepción.

A) Solo I

B) I y II

C) I y III

D) II y III

E) I, II, III

12. ¿Cuál de los siguientes tipos de excepciones **no** son posibles de atrapar en el mismo archivo donde está el error con un bloque **`try/except`**?

A) `TypeError`

B) `FileNotFoundError`

C) `NameError`

D) `IndentationError`

E) `AttributeError`

13. Según lo visto en el curso sobre Listas Ligadas, ¿cuál o cuáles de las siguientes afirmaciones son **correctas** sobre una Lista Ligada con **más de 1 nodo**?

    I. La `cola` y la `cabeza` referencian el mismo nodo.

    II. La `cola` referencia al primer nodo almacenado mientras que la `cabeza` referencia al último.

    III. Para recorrer y alterar la Lista Ligada, es necesario iterar secuencialmente sobre los nodos, usando sus respectivas referencias que los conectan.

    A) Solo I

    B) Solo II

    C) Solo III

    D) I y III

    E) II y III

14. Respecto a Iteradores e Iterables, ¿cuál o cuáles de las siguientes afirmaciones son **correctas**?

    I. Un Iterable no posee un estado interno de iteración mientras que un Iterador sí.

    II. Todo Iterable puede acceder a sus elementos mediante indexación.

    III. Un Iterador tiene el método `len()` implementado por defecto.

    IV. Un Generador es un tipo de Iterable.

    A) Solo I

    B) I y II

    C) I y IV

    D) II y III y IV

    E) I, II, III y IV

15. ¿Cuál de las siguientes afirmaciones es **falsa** respecto al **paradigma de programación funcional en Python**?

    A) En programación funcional, el valor de salida de una función solo debería depender de su valor de entrada.

    B) En programación funcional, una función no debe modificar de forma permanente ninguno de sus parámetros de entrada.

    C) No es posible entregar instancias de clases como argumento a las funciones con este paradigma.

    D) Python permite trabajar con el paradigma OOP y el paradigma funcional en el mismo programa.

    E) Una función "impura" es una cuyo resultado depende del contexto en el que se ejecuta dicha función.

16. Para el siguiente código, ¿cuál será **el valor** de la variable `numero_ganador`?

```
cifras_seleccionados = (i for i in range(2, 12))

trivia_lunes = [5, cifras_seleccionados]
trivia_martes = trivia_lunes

for cifra in trivia_lunes[1]:
    if cifra == 11:
        break
    print(cifra)

numero_ganador = trivia_martes[1].__next__()
```

A) El código se ejecuta hasta el **`break`**, por lo que `numero_ganador` no tendrá un valor asignado.

B) `2`

C) `11`

D) `12`

E) El código levanta una excepción, por lo que `numero_ganador` no tendrá un valor asignado.

17. Considere el siguiente código en Python. Sin considerar saltos de línea, ¿qué **imprime** este código?

```
def funcion(valor):
    try:
        print("A")
        if valor > 0:
            raise ValueError
    except ValueError:
        print("B")
        funcion(valor - 1)
    else:
        print("C")
    finally:
        print("D")

funcion(1)
```

A) `A B A C D`

B) `A B D A C`

C) `A B C A C`

D) `A B D A C D`

E) `A B A C D D`

18. ¿Cuál de las siguientes alternativas es **falsa** respecto a las **funciones lambda**?

A) Pueden ser definidas de forma anónima, es decir, sin un nombre asociado.

B) Una correcta aplicación de ellas es usarlas únicamente donde fueron creadas.

C) Pueden recibir cualquier tipo de dato como parámetro de entrada.

D) No es necesario poner explícitamente el comando **`return`** para que retornen un valor.

E) Debido a su simpleza, no pueden recibir más de un parámetro a la vez.

19. Considere el siguiente código de una función generadora:

```
1  def funcion_generadora():
2      contador = 0
3
4      while True:
5          valor_recibido = yield contador
6          print("Hemos recibido {}".format(valor_recibido))
7
8          if not valor_recibido:
9              valor_recibido = 0
10
11         print("Sumaremos {} a nuestro contador".format(valor_recibido))
12         contador += valor_recibido
13
14 generador = funcion_generadora()
15 resultado = ? # Completar
```

¿Con cuál o cuáles de las siguientes opciones se podría **reemplazar** el `?` en la línea 15 **sin levantar una excepción** en el programa?

I. `generador.send(3)`

II. `generador.send(None)`

III. `next(generador)`

A) Solo III

B) Solo I y II

C) Solo I y III

D) Solo II y III

E) I, II y III

20. ¿Cuál es el ***output*** del siguiente código?

```
fonda = {
    'Choripan': 4,
    'Sopaipillas': 3,
    'Anticucho': 6,
    'Asado': 0,
}

var_1 = filter(lambda x: len(x) > 0, fonda.keys())
var_2 = map(lambda x: x * 2 , filter(lambda x: x % 2 == 0, fonda.values()))

print(list(zip(var_1, var_2)))
```

A) `[('Choripan', 4), ('Sopaipillas', 3), ('Anticucho', 6), ('Asado', 0)]`

B) `[('Choripan', 4), ('Sopaipillas', 3), ('Anticucho', 6)]`

C) `[('Choripan', 8), ('Sopaipillas', 12), ('Anticucho', 0), ('Asado', 8)]`

D) `[('Choripan', 8), ('Sopaipillas', 12), ('Anticucho', 0)]`

E) `[('Choripan', 8), ('Sopaipillas', 6), ('Anticucho', 12)]`

Preguntas desarrollo

Pregunta 1 - Modelación OOP

Lea el siguiente texto y luego desarrolle las preguntas correspondientes:

> La empresa emergente JapanEx está realizando en Latinoamérica diversos eventos relacionadas con Japón. Estos eventos se distinguen por tener nombre único, fecha de realización, dirección, capacidad máxima del recinto y una lista de todas las personas que asisten. En la actualidad, la organizadora solo gestiona dos tipos de eventos: ferias y conciertos, pero eventualmente desea expandirse a más tipos de eventos.
>
> Por un lado, las ferias deben definir un límite máximo de expositores y un costo base para colocar un puesto como expositor. Además, el costo de arriendo de un puesto en las ferias se calcula dinámicamente, teniendo en cuenta el costo base y la cantidad de expositores que ya se han registrado hasta la fecha.
>
> Por otro lado, los conciertos poseen una lista con los diferentes tipos de entrada. Cada tipo de entrada tiene un nombre único, la cantidad de entradas que todavía no se venden y el costo de la entrada, el cual es único y diferente para cada tipo de entrada.
>
> Finalmente, es importante considerar que hay distintos tipos de personas. Todas ellas poseen un nombre y la cantidad de dinero en efectivo que llevan al evento. Además, deben tener la posibilidad de asistir a cualquiera de estos 2 eventos, aunque la forma en que lo hacen difiere según el tipo de persona y el evento al que asisten. Por el momento, la organizadora distingue dos tipos de personas que serán excluyentes entre si: artistas y visitantes. Estos tipos no poseen información específica, solo se diferencian en la forma que visitan cada evento. Además, en caso de empezar a distinguir nuevos tipos de personas, todas estas deben tener definido la forma que visitan cada evento.
>
> - **Artistas**: Si visitan una feria, deben pagar el monto para poner un puesto. Mientras que si visitan un concierto, no deben pagar nada.
> - **Visitantes**: Si visitan una feria, no pagan nada. Mientras que si visitan un concierto, deben comprar la entrada más cara que esté disponible y que su dinero pueda financiar.
>
> En ambos casos, esta acción nos responderá con un *booleano* que indica si la persona pudo asistir al evento (`True`) o no (`False`).

Ahora, desarrolle las siguientes preguntas:

1. **(5.0 puntos)** Confeccione un diagrama de clases que modele correctamente la situación anteriormente descrita. El diagrama debe utilizar el formato visto en el curso. Recuerde que un diagrama de clases incluye: los atributos (nombre, tipo de dato y si es *property* o no), métodos (nombre, qué recibe y qué tipo de dato retorna), y las relaciones entre clases (herencia y contención).

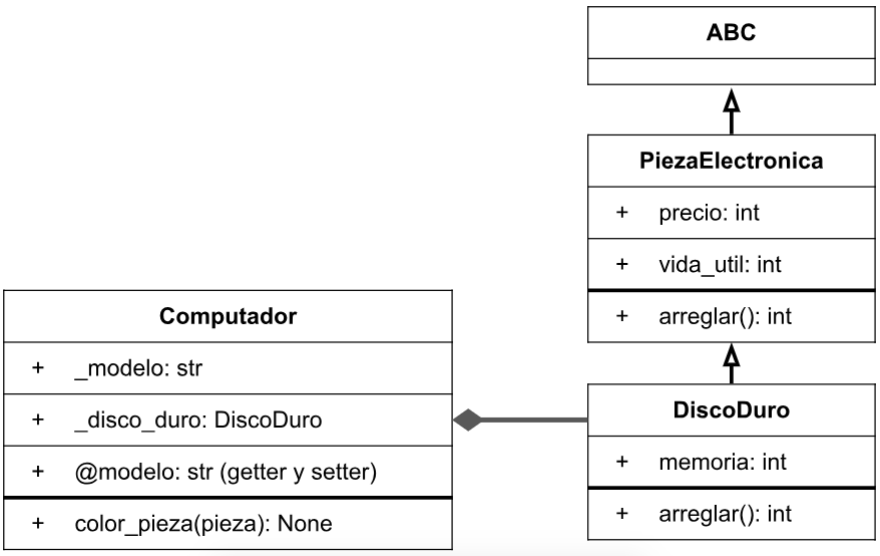
2. **(1.0 punto)** La organizadora quiere incluir un nuevo tipo de evento: *"AnimeExpo"* que es un tipo de feria en donde las personas pueden asistir a un concierto en el mismo evento. En otras palabras, es un nuevo evento que posee las características de los 2 tipos eventos que actualmente gestiona la organizadora. Para incluir este nuevo tipo, se propuso modelarlo de la siguiente forma:

```
class AnimeExpo(Visitante, Feria):
    def __init__(self, *args, **kwargs):
        Visitante.__init__(self, *args, **kwargs)
        Feria.__init__(self, *args, **kwargs)
```

   Lamentablemente, el código anterior presenta 2 errores. Indique cuáles son y por qué son errores. Justifique su respuesta con los contenidos del curso y en un máximo de 4 líneas.

Ejemplo: Diagrama de Clases

A continuación, se muestra un diagrama de clases respetando el formato visto en el curso:



Este diagrama modela a un computador que contiene un disco duro, el cual es un tipo de pieza electrónica. Además, toda pieza electrónica hereda de la clase `ABC`.

Pregunta 2 - Análisis de código

El siguiente código corresponde a la modelación de una lista de reproducción:

```
from collections import namedtuple
from random import shuffle
from typing import Self


Canción = namedtuple('Canción', 'nombre,artista,album,popularidad')

class ListaReproducción:
    def __init__(self, canciones: list, forma_reproducción: str = '') -> None:
        self.canciones = canciones
        self.forma_reproducción = forma_reproducción

    def __iter__(self) -> IteradorListaReproducción:
        return IteradorListaReproducción(self.canciones, self.forma_reproducción)

class IteradorListaReproducción:
    def __init__(self, canciones: list, forma_reproducción: str) -> None:
        self.canciones = canciones.copy()

        if forma_reproducción == 'aleatorio':
            shuffle(self.canciones)
        elif forma_reproducción == 'populares':
            filter(lambda c: c.popularidad > 10 ** 6, self.canciones)

    def __iter__(self) -> Self:
        return self

    def __next__(self) -> Canción:
        if not self.canciones:
            raise StopIteration('Lista reproducción vacía')
        return self.canciones[0]


lista_reproducción = ListaReproducción([
    Canción('Cortes de Papel', 'Tortuganónima', 'Imago', 54000),
    Canción('Plantasia', 'Mort Garson', 'Mother Earth\'s Plantasia', 16000000),
    Canción('Mercury', 'Sufjan Stevens y ...', 'Planetarium', 7000000),
    Canción('The Lightning Strike', 'Snow Patrol', 'A Hundred Million Suns', 12000000),
    Canción('let me out', 're:code', '', 1200),
])
lista_reproducción.forma_reproducción = 'populares'

for canción in lista_reproducción:
    print(f'Escuchando {canción.nombre} de {canción.artista}')
print('¡Escuchaste toda la lista de reproducción!')
```

Indique la veracidad o falsedad de las siguientes afirmaciones. Argumente, en máximo 4 líneas, cada respuesta **aplicando correctamente los términos de estructuras de datos y relacionando la argumentación realizada con el código entregado**. Solo se permite referenciar el código presentado, es decir, **no se puede asumir** que existen más funciones, clases o variables fuera de las contenidas en el código anterior.

A continuación se presenta una afirmación con su respuesta y una argumentación acorde a la instrucción del párrafo anterior.

0. Al comentar la línea 41, se reproducirán todas las canciones según su posición en la lista de reproducción.

   (V)/ F

   Dado que no se sobrescribe el valor del atributo `forma_reproducción`, este mantendrá su valor original (`''`), por lo que al instanciar el iterador no se ejecuta ninguna de las sentencias del **`if`**/**`elif`**, lo que evita que la lista de canciones sea reordenada y filtrada.

Finalmente, las afirmaciones que deberás evaluar y argumentar son:

1. **(1.5 puntos)** Si se ejecuta la excepción **`StopIteration`** mientras se ejecuta la línea 43, entonces se detendrá la ejecución del programa.

2. **(1.5 puntos)** A partir de la ejecución del código entregado, se imprimirán todas las canciones de la lista de reproducción y al final el mensaje *`'¡Escuchaste toda la lista de reproducción!'`*.

3. **(1.5 puntos)** A partir de la ejecución del código entregado, solo se reproducirán las canciones que tengan una popularidad mayor a 1 millón.

4. **(1.5 puntos)** Cuando se ejecuta la línea 43, se llaman y ejecutan los siguientes métodos en este orden:

   1) `ListaReproducción.__iter__()`

   2) `IteradorListaReproducción.__iter__()`

   3) `IteradorListaReproducción.__next__()`

Nombre completo: _____

RUT: _____ Sección: _____ N° lista: _____

IIC2233 - Pregunta 1 (Modelación OOP)

1. Dibuja tu diagrama a continuación:

2. _____

_____

_____

_____

Nombre completo: _____

RUT: _____ Sección: _____ N° lista: _____

IIC2233 - Pregunta 2 (Análisis de código)

Para cada afirmación marca si es Verdadero (V) o Falso (F) a la **izquierda**, y luego justifica tu respuesta a la **derecha**.

1. V / F _____

2. V / F _____

3. V / F _____

4. V / F _____