Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2023-2

Midterm

17 de octubre 2023

Instrucciones

- La evaluación consta de 20 preguntas de selección múltiple y 2 incisos de preguntas abiertas, es decir, de desarrollo.
- Se te proveerá de plantillas que deberás rellenar con tus datos y las respuestas correspondiente.
- Respecto a la selección múltiple, cada pregunta tiene únicamente 1 alternativa correcta. Responder con 2 o más alternativas implicará dejar inválida esa pregunta y se considerará incorrecta. Además, cada pregunta presenta el mismo puntaje y no se descontará por respuesta incorrecta.

Selección múltiple

1. Respecto a las *properties*, ¿cuál(es) de la(s) siguiente(s) afirmación(es) es/son **verdadera(s)**?

    I. Para que las *properties* sean válidas, es necesario que la *property* tenga el mismo nombre que un atributo privado al cual hace referencia.

    II. Las *properties* deben tener como mínimo un *getter* y un *setter* para poder funcionar correctamente, es decir, sin levantar ningún error en la ejecución del código.

    III. Las *properties* permiten encapsular comportamientos para manejar el acceso y modificación de datos dentro de una instancia.

    A) Solo I

    B) Solo II

    C) Solo III

    D) I y III

    E) II y III

2. Sin considerar saltos de línea, ¿qué imprimirá el siguiente código?

```
class CajaSecreta:

    def __init__(self, valor):
        self._valor_secreto = valor

    @property
    def valor(self):
        self.valor = self._valor_secreto + 1
        print(self._valor_secreto)
        return self._valor_secreto

    @valor.setter
    def valor(self, valor):
        print(valor)
        self._valor_secreto += valor

caja = CajaSecreta(5)
caja.valor
caja.valor += 5
```

A) Arroja error, pues la *property* tiene un nombre distinto al atributo de la instancia.

B) 6 11 12 23 28

C) 6 11 5

D) 6 11 16

E) Ninguna de las anteriores es correcta

3. Respecto a herencia, ¿cuál(es) de la(s) siguiente(s) afirmación(es) es/son **verdadera(s)**?

I. Corresponde a una relación de especialización entre dos clases.

II. Cuando una clase hija hace *overriding* de un método, ya no se puede acceder a la definición de dicho método de la clase padre.

III. En Python, una clase puede heredar de múltiples clases.

A) Solo II

B) Solo III

C) I y II

D) I y III

E) I, II y III

4. ¿Qué imprimirá el siguiente código?

```
class Padre:

    def __init__(self, valor):
        self.atr = valor

class Hijo(Padre):

    def __init__(self):
        self.atr = 25
        super().__init__(4)
        self.atr += 1

hijo = Hijo()
print(hijo.atr)
```

A) 5

B) 4

C) 25

D) 26

E) Error, pues el `__init__` de la clase debería recibir el `valor`.

5. ¿Cuál(es) de la(s) siguiente(s) afirmación(es) es/son **correctas** respecto a las **clases abstractas**?

I. Se pueden instanciar.

II. La herencia de clases abstractas no permite *override* de sus métodos.

III. El módulo `abc` permite definir clases abstractas en Python.

IV. Permiten herencia e implementación de sus métodos.

A) Solo IV

B) I y III

C) I, II y III

D) I, III y IV

E) II, III y IV

6. Dada la clase `C` que hereda tanto de `A` como de `B`, ¿cuál de los códigos del método `imprimir` toma precedencia?

```
class A:

    def imprimir(self):
        print('Soy A')

class B:

    def imprimir(self):
        print('Soy B')

class C(A, B):

    def __init__(self):
        super().__init__()

c = C()
c.imprimir()
```

A) Ambos códigos del método `imprimir` se combinan y se ejecutan juntos.

B) Arrojará un error de conflicto de nombres de método.

C) El código del método de la clase `A` toma precedencia.

D) El código del método de la clase `B` toma precedencia.

E) No se ejecuta ningún método.

7. ¿Cuál(es) de la(s) siguiente(s) afirmación(es) es/son **correctas** respecto a las **tuplas**?

I. Son estructuras de datos mutables.

II. Pueden desempaquetarse en variables independientes.

III. Se puede hacer *slicing* sobre una tupla.

IV. Se puede acceder a sus elementos a través de índices.

A) Solo I

B) Solo III

C) II y III

D) II, III, IV

E) I, II, III y IV

8. ¿Qué es lo **mínimo** que debes crear, para implementar una estructura **iterable** que se pueda recorrer directamente mediante el uso de `for`?

   A) Una clase Iterable con el método `__iter__`, y una clase Iterador con el método `__next__`.

   B) Una clase Iterable con el método `__iter__`, y una clase Iterador con los métodos `__iter__` y `__next__`.

   C) Una clase Iterable con el método `__next__`, y una clase Iterador con el método `__iter__`.

   D) Una clase Iterable con los métodos `__iter__` y `__next__`, y una clase Iterador con el método `__next__`.

   E) Una clase Iterable con los métodos `__iter__` y `__next__`, y una clase Iterador con los métodos `__iter__` y `__next__`.

9. Dada la siguiente lista de listas:

```
lista = [[1, 4], [2, 5], [3, 6]]
```

   ¿Cuál alternativa presenta un código que obtiene como resultado el valor 32?

   A) `reduce(lambda p, q: p + q[0] * q[1], lista)`

   B) `reduce(lambda p, q: p + q[0] * q[1], lista, 0)`

   C) `reduce(lambda p, q: p[0] + q[0] * q[1], lista, [0])`

   D) `reduce(lambda p, q: p[0] * p[1] + q[0] * q[1], lista)`

   E) `reduce(lambda p, q: p[0] * p[1] + q[0] * q[1], lista, [0, 0])`

10. ¿Qué resultado entrega el siguiente código?

```
mi_dic = {
    'Alice': 5,
    'Bob': 7,
    'Charlie': 1,
    'Denise': 3
}

var_1 = filter(lambda x: len(x) > 3, mi_dic.keys())
var_2 = filter(lambda x: (x * 10) != 0, mi_dic.values())

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

   A) `[('Alice', 5), ('Charlie', 7), ('Denise', 1), ('Alice', 3)]`

   B) `[('Alice', 5), ('Bob', 7), ('Charlie', 1), ('Denise', 3)]`

   C) `[('Alice', 5), ('Alice', 7), ('Alice', 1)]`

   D) `[('Alice', 5), ('Bob', 7)]`

   E) `[('Alice', 5), ('Charlie', 7), ('Denise', 1)]`

11. Se decide implementar un control remoto mediante una interfaz gráfica en `PyQt6`, el cual cuenta con un conjunto de botones con números del 0 al 9, cuyo *click* está conectado al método `getKey`.

    ¿Qué código **permite** obtener el número asociado a estos botones, cuando uno de estos es "clickeado" por el usuario?

    A)
    ```
    def getKey(self, event):
        return event.key()
    ```

    B)
    ```
    def getKey(self, event):
        return event.text()
    ```

    C)
    ```
    def getKey(self):
        sender = self.sender()
        return sender.key()
    ```

    D)
    ```
    def getKey(self):
        sender = self.sender()
        return sender.text()
    ```

    E) No es posible obtener el número asociado al botón.

12. ¿Cuál(es) de los siguientes códigos corresponde(n) a la **forma(s) válida(s)** de conectar una misma señal con las funciones `función_1` y `función_2`?

    I.
    ```
    señal.connect([función_1, función_2])
    ```

    II.
    ```
    señal.connect(función_1)
    señal.connect(función_2)
    ```

    III.
    ```
    señal.connect(función_1, función_2)
    ```

    A) Solo I

    B) Solo II

    C) Solo III

    D) I y III

    E) I, II y III

13. ¿Cuál o cuáles de estas funcionalidades **pertenecen** al *front-end* de un programa?

    I. Almacenar los puntajes en un archivo.

    II. Recibir un nombre de usuario.

    III. Validar la contraseña del usuario.

    IV. Reproducir una canción.

    V. Cargar el logo del programa.

    A) I y III

    B) I, II y III

    C) II y V

    D) II, IV y V

    E) II, III y V

14. ¿Cuál de las siguientes afirmaciones del método **run** en *threading* es **incorrecta**?

    A) Da la orden de comenzar la ejecución del *thread* de forma concurrente.

    B) Se ejecuta una única vez luego de inicializar el *thread*.

    C) Se ejecuta luego de llamar al método `start()`.

    D) Al heredar de la clase `Thread`, se puede personalizar el método mediante *override*.

    E) Contiene instrucciones que serán ejecutadas por la instancia del *thread* al iniciarse.

15. Supongamos que tenemos dos hilos de ejecución denominados `thread_1` y `thread_2`, que hacen uso de la misma instancia de un objeto `lock`. En el momento en que `thread_1` ha adquirido un recurso mediante una llamada a la función `lock.acquire()`.

    ¿Qué sucede cuando `thread_2` intenta también adquirir el mismo recurso mediante una llamada a la función `lock.acquire()`?

    A) Ambos *threads* comparten el recurso en simultáneo.

    B) `thread_2` adquiere inmediatamente el recurso, y ambos threads continúan su ejecución.

    C) `thread_1` se detiene y libera el recurso.

    D) `thread_2` finaliza al no poder ejecutar correctamente la función `lock.acquire()`.

    E) `thread_2` se detiene temporalmente en espera a que `thread_1` libere el recurso.

16. ¿Cuál afirmación es **correcta** sobre `Qthread` y `QTimer`?

A) El `QThread` es la clase análoga a Thread, mientras que `QTimer` es el análogo a time. Por lo tanto, aplicamos concurrencia con `QThread` y usamos `.sleep` con el `QTimer`.

B) Ambas son clases para realizar concurrencia, pero `QThread` se utiliza. exclusivamente cuando queremos aplicar herencia, mientras que `QTimer` se utiliza exclusivamente cuando queremos un método concurrente sin necesitar herencia.

C) El `QThread` era la versión antigua para aplicar concurrencia, pero fue deprecado, es decir, ya no funciona, y solo se ocupa la nueva versión: `QTimer`.

D) El `QThread` ejecuta de forma concurrente, mientras que el `QTimer` debe esperar a que no hayan otros `QTimers` corriendo para poder ejecutarse.

E) El `QThread` ejecuta de forma concurrente la función indicada 1 vez, mientras que el `QTimer` permite llamar periódicamente a la función indicada.

17. Sin considerar saltos de línea, ¿qué imprimirá el siguiente código en la consola?

```
1  def f1():
2      print('F1')
3
4  class Ventana(QWidget):
5      def __init__(self):
6          super().__init__()
7          timer = QTimer(self)
8          timer.setSingleShot(True)
9          timer.timeout.connect(f1)
10         timer.timeout.connect(self.f2)
11         timer.setInterval(1000)
12         timer.start()
13         self.show()
14
15     def f2(self):
16         print('F2')
17
18 app = QApplication([])
19 ventana = Ventana()
20 sys.exit(app.exec())
```

A) Arroja error, pues no se puede conectar una misma señal de *timeout* a dos funciones distintas

B) `"F1 F2"`

C) `"F2"`

D) `"F1"`

E) No imprime nada, pues había que hacer `timer.run()` en vez de `timer.start()`.

18. Respecto a las diferencias que existen entre `QMainWindow` y `QWidget`, ¿cuál(es) de la(s) siguiente(s) afirmación(es) es/son **verdadera(s)**?

    I. `QMainWindow` permite incluir barra de menú, barra de herramientas y barra de estado, mientras que el `QWidget` no da todas estas opciones por defecto.

    II. `QMainWindow` permite que hayan múltiples ventanas abiertas en la aplicación, mientras que el `QWidget` no permite que hayan múltiples ventanas abiertas en la aplicación.

    III. `QMainWindow` puede contener un `QWidget` para desplegar contenido dentro de la ventana, mientras que el `QWidget` no puede contener otros *widget* dentro de la ventana.

    A) Solo I

    B) Solo III

    C) I y II

    D) I y III

    E) II y III

19. ¿Cuál de las siguientes afirmaciones es **incorrecta** respecto a la **serialización** en Python?

    A) La serialización `pickle` convierte distintos tipos de datos en un formato binario para almacenamiento o transmisión.

    B) La serialización en `JSON` solamente es aplicable a números enteros y *strings*.

    C) Los objetos del tipo byte son inmutables, y una vez creados su contenido no puede ser modificado.

    D) Los `bytearray` son mutables y cada elemento de este se puede representar como un byte.

    E) La serialización en Python es un proceso reversible.

20. Tomando en cuenta el siguiente código, al levantar una excepción con el comando `raise`, es correcto afirmar que:

```
1 print('Comienza')
2 raise ValueError('Mensaje')
3 print('Error levantado')
4 print('Fin del codigo')
```

    A) El error es manejado automáticamente, se imprime el mensaje y el programa continúa su ejecución sin interrupción en la siguiente línea.

    B) El programa pausa su ejecución y se reinicia.

    C) No se ejecuta la siguiente línea del código automáticamente, sino que se detiene en el punto donde se levantó la excepción y finaliza.

    D) El flujo de ejecución del programa se mueve automáticamente a la última línea del programa, ignorando el código restante.

    E) El código completo no se ejecuta.

Preguntas desarrollo

Inciso 1 - Modelación OOP

Lea el siguiente texto y responda las preguntas correspondientes:

> *Entre el mes de octubre y noviembre se realizará una nueva edición de los Juegos Panamericanos en Santiago y tú, como amante de la programación y el deporte, has decidido hacer una simulación que modele a algunos de los deportistas con el fin de predecir su rendimiento en sus competencias.*
>
> *Sabes que cualquier deportista posee su propio nombre, edad y un país al que representan. Debido a su profesión, cada uno posee una cantidad de resistencia limitada, que corresponde a la energía que le permite realizar deporte y entrenar. La resistencia es un valor entero que no puede ser menor a 0 ni sobrepasar un valor máximo, que varía para cada deportista.*
>
> *Cada deportista tiene la opción de competir en su propia disciplina, proceso que varía de gran forma según el deporte en que se especialice, sin embargo, siempre termina gastando 20 puntos de resistencia e imprimiendo el mensaje "¡He finalizado de competir!". De la misma forma, el deportista puede descansar para recuperar 50 puntos de resistencia, y también puede entrenar para así aumentar su resistencia máxima. El valor en que aumenta la resistencia máxima dependerá de la disciplina que realiza el deportista.*
>
> *Para tu simulación, solo deberás considerar dos tipos de deportistas: Nadadores y Ciclistas. Los nadadores se especializan en un estilo de nado, información que deben poseer junto a sus datos personales, mientras que los ciclistas llevan registro del modelo de bicicleta que usarán en la competencia, y que por reglas del campeonato, no puede ser modificado una vez asignado. Cuando el nadador entrena, pierde 10 puntos de resistencia pero aumenta su resistencia máxima en 5 puntos, mientras que el ciclista se somete a un entrenamiento más duro, lo que hace que pierda 12 puntos de resistencia, pero a cambio suba su resistencia máxima en 7 puntos.*

Parte A

Usando conceptos de Programación Orientada a Objetos (POO), diseñe las clases que necesitaría un programa de Python que simule el contexto presentado anteriormente. Siga el siguiente formato para cada clase que cree:

1. Nombre de clase. Por cada clase que añada, indique:
    a. Si es abstracta o no.
    b. Si hereda de otra clase, en caso de que corresponda, indica de cuál clase hereda.
2. Atributos. Por cada atributo que añada, indique:
    a. Nombre del atributo.
    b. Tipo de dato: str, int, float, bool, instancia de algún objeto, etc.
    c. Si es una *property* o no.

    En caso de estar describiendo una clase que hereda de otra, no es necesario repetir los atributos de la superclase respectiva.
3. Métodos. Por cada método que añada, indique:
    a. Nombre del método.
    b. Argumentos que necesite.
    c. Tipo de dato que retorna.
    d. Descripción breve de lo que hace el método (máximo 2 líneas).

Parte B

Suponga que se añade un nuevo deporte al evento, el cual corresponde al Biatlón. En este deporte, los participantes deberán primero enfrentarse a una competencia de nado libre, seguida inmediatamente de una carrera en bicicleta.

Indique qué concepto de Programación Orientada a Objetos será útil para modelar a los competidores de esta disciplina, denominados Biatletas, y cómo se aplicaría en este caso. Además, indique un posible problema que podría encontrarse al aplicar este concepto y cómo evitarlo.

Inciso 2 - Análisis de código

Las **DCCompetencia** consiste en un juego cooperativo de 2 jugadores. Una vez comenzada la competencia, cada jugador responde diferentes preguntas, con un descanso entre cada pregunta. Estas preguntas serán evaluadas con puntaje de 1 a 10.

Una vez que el equipo supera los 500 puntos, el primer jugador en notar eso levantará una señal para indicar que la competencia finalizó. De este modo, ambos jugadores dejan de responder consultas y el puntaje del equipo se calcula como el puntaje total adquirido menos el tiempo que jugó cada competidor.

El siguiente código fue creado para intentar modelar esta competencia. Puede asumir que el código no tiene errores de sintaxis.

```
class DCCompetidorTeamA(threading.Thread):
    lock = threading.Lock()
    puntajeTeam = 0

    def __init__(self, nombre:str, evento_inicio: threading.Event,
                 evento_fin: threading.Event):
        super().__init__(name=nombre)
        self.tiempo = 0
        self.evento_inicio = evento_inicio
        self.evento_fin = evento_fin
        self.daemon = False

    def run(self):
        self.evento_inicio.wait()
        while not self.evento_fin.is_set():
            t = random.random() # Número decimal entre 0 y 1
            time.sleep(t)
            self.tiempo += t
            puntaje = DCC.responder_pregunta(self)

            with DCCompetidorTeamA.lock:
               DCCompetidorTeamA.puntajeTeam += puntaje
               if DCCompetidorTeamA.puntajeTeam > 500:
                  print("Listo, no más preguntas :D")

if __name__ == "__main__":
    evento_inicio = threading.Event()
    evento_fin = threading.Event()
    j1 = DCCompetidorTeamA("Anya", evento_inicio, evento_fin)
    j2 = DCCompetidorTeamA("Luffy", evento_inicio, evento_fin)

    j1.start()
    evento_inicio.set()
    j1.join()

    j2.start()
    j2.join()
    print(DCCompetidorTeamA.puntajeTeam - j1.tiempo - j2.tiempo)
```

Indique la veracidad o falsedad de las siguientes afirmaciones. Argumente, en máximo 5 líneas, cada respuesta **aplicando correctamente los términos de *threadings* y relacionando la argumentación realizada con el código entregado**. Solo se permite referenciar el código presentado, **no se puede asumir** que existen más clases que hagan otras acciones.

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

- **Afirmación:** Cada competidor libera el `DCCompetidorTeamA.lock` luego de actualizar el puntaje.
- **Respuesta:** Verdadera.
- **Argumentación:** En la línea 21 el código hace `with DCCompetidorTeamA.lock` para luego actualizar el puntaje dentro de dicho `with`. Este comando (`with`) se encarga de hacer `lock.release()` automáticamente una vez se finaliza de ejecutar el código dentro del bloque `with`. Por lo tanto, efectivamente se libera `DCCompetidorTeamA.lock` una vez actualizado el puntaje.

Finalmente, las afirmación que deberás evaluar y argumentar son:

1. El atributo "tiempo" debería estar dentro de un *lock*.
2. Las 2 instancias de jugadores no están ejecutando su código concurrentemente.
3. Aunque eliminemos los `.join()` del código, el programa sólo finalizará cuando ambos competidores terminen.
4. El código refleja fielmente que los jugadores competirán hasta superar los 500 puntos, luego el primer jugador en notar eso avisará que la competencia terminó y el método `run` finaliza su ejecución.