APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

Duración: tres horas: 60 minutos -> teoría -> 4 puntos. 120 minutos -> práctica -> 6 puntos

Antes de comenzar:

- **El examen se realizará en una sesión individual creada con el nombre y el apellido del alumno presente en la pestaña Clases online de Blackboard. Comprobar antes de realizar el examen que la sesión de cada usuario está siendo grabada y que se está compartiendo audio, vídeo y pantalla. Ante cualquier duda contactar con el profesor Alberto Gil de la Fuente o Antonio Artés García en la sesión de dudas del examen.**
- **La sesión de dudas del examen se encuentra en el siguiente enlace: https://eu.bbcollab.com/guest/67389fe32f4546f59f189945907e5b16. Se recomienda estar presente durante toda la sesión e interactuar con los profesores en el chat privado de esta sesión, no en las sesiones individuales donde no se va a contestar ninguna pregunta.**
- **Antes de comenzar el examen debe compartirse la pantalla, la cámara y el micrófono. Se debe mostrar el teléfono móvil a la cámara y dejarlo en una zona a la espalda del estudiante dentro del plano de la cámara o, en su defecto, a una distancia suficiente para que no se pueda coger con el brazo sin levantarse de la silla. Se debe mostrar el entorno del examen antes de comenzar.**
- **El resto de dispositivos electrónicos deben estar guardados en la mochila y apagados o sin sonido.** En caso contrario debe indicarse al profesor los motivos por los que debe tenerse el teléfono con sonido antes de comenzar el examen.
- **No se permite la utilización de dos monitores durante el examen, por lo que se debe mostrar el entorno del ordenador sin ningún monitor cerca de él.**
- **Se permite el uso de calculadora del ordenador, aunque su utilización no es necesaria para la realización del examen.**
- **No se permite la utilización de apuntes, teoría, la escritura en papel, el acceso a recursos externos, hablar, prestando especial atención a la lectura del enunciado del examen en voz alta, o cualquier otra actividad que no sea la escritura mediante el teclado del equipo de las respuestas en la plantilla de respuestas facilitada en Word.**
- **El uso de cualquier dispositivo electrónico o el plagio o copia conllevará el suspenso del examen y de la evaluación ordinaria, acudiendo directamente a la evaluación extraordinaria según el artículo 9 del reglamento sobre pruebas de evaluación y su revisión:**

  Artículo 9.º. De la utilización de métodos ilícitos para la superación de las pruebas de evaluación. "Cualquier evidencia de plagio, de copia del examen de un compañero, o cualquier intento de obtener de forma fraudulenta las respuestas a las preguntas de una prueba de evaluación, será penalizada con el suspendo en la convocatoria ordinaria y extraordinaria y podrá suponer la apertura de expediente y la aplicación de las correspondientes sanciones, pudiendo llegar a sustanciarse en la expulsión de la Universidad del alumno o alumnos implicados, conforme a lo establecido en los arts. 29 y siguientes del Reglamento del Alumnado de esta Universidad."

- **No contestar en esta hoja, abrir el documento Word answers_template para responder a las preguntas teóricas. El documento de answers_template debe ser renombrado a <ApellidosEstudiante>_<NombreEstudiante>_answers.pdf**
- **No se podrá abandonar la silla durante la realización del examen.**
- **No se podrá detener o interrumpir la cámara, la pantalla o el micrófono en ningún momento del examen.**
- **Una vez finalizada la teoría se podrá empezar la parte práctica, a los 30 minutos del comienzo del examen y enviando un mensaje en el chat de la sesión privada del usuario en el momento en el que se acabe la parte teórica y se pase a la parte práctica.**
- **La parte práctica se debe entregar en la actividad correspondiente y en el documento .zip que contenga los diferentes módulos y sus clases según el enunciado, así como el ejercicio 6 de la parte teórica en caso de seleccionarlo para su solución.**
- **Cualquier sospecha sobre el incumplimiento de las normas aquí descritas puede concluir en la revisión oral del examen, en la realización de un nuevo examen oral o en el suspenso de la asignatura y posible apertura de expediente académico si corresponde, según la gravedad de las acciones.**

Entregar esta hoja con el nombre y apellidos del alumno, DNI y grupo.

APELLIDOS	NOMBRE		GRUPO	CALIFICACIÓN
ASIGNATURA Algoritmos y Estructuras de Datos	FECHA 18/01/21	DNI	**GIN 02**	

Resultados de aprendizaje que se evalúan en este examen:

- Entender los conocimientos básicos de algorítmica y complejidad computacional.
- Ser capaz de realizar análisis de complejidad de algoritmos.
- Ser capaz de programar en un lenguaje de programación utilizando estructuras de datos comunes.
- Ser capaz de implementar algoritmos básicos.
- Ser capaz de probar adecuadamente los programas desarrollados.

Entregar esta hoja con el nombre y apellidos del alumno, DNI y grupo.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI	**GIN 02**	

Ejercicio 1 (0.5 puntos) Tiempo estimado: 5 minutos. Responda a las cuestiones

La calificación máxima de esta parte tipo test es de 0.5 puntos. Tenga en cuenta que:

- Para cada pregunta, sólo existe una respuesta correcta.
- Cada respuesta correcta sumará 0.10 puntos.
- Cada respuesta incorrecta restará 0.10 puntos.
- Las preguntas no contestadas ni suman ni restan puntuación.

1) En relación al estado de un objeto, para preservar el principio de encapsulación:

    a) Los atributos de nuestra clase deben permanecer públicos, para permitir un acceso total a la información que almacenamos en los objetos. Sin embargo, aquellos métodos que realicen operaciones internas (y no deban ser utilizados), deben establecerse como privados.

    b) Debemos establecer la visibilidad más restrictiva (por ejemplo, privada) en los atributos de una clase. Así, cualquier software que utilice nuestro objeto, sólo accederá al estado de los objetos mediante los métodos que le hayamos permitido utilizar.

2) Los métodos de una clase no pueden devolver objetos:

    a) Verdadero

    b) Falso

3) ¿Cuál de las siguientes características de la programación orientada a objetos está relacionada con la reutilización de código?

    a) Abstracción

    b) Polimorfismo

4) El tiempo de acceso a un dato en una lista enlazada es:

    a) O(n)

    b) O(1)

5) Un algoritmo de ordenación que implemente el método de inserción, tiene una complejidad en el peor caso de:

    a) $O(n^2)$

    b) $O(n^3)$

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

Ejercicio 2 (2.5 puntos) Tiempo estimado: 25 minutos. Se debe responder a las 5 preguntas de forma obligatoria. Respuestas correctas suman 0.5 puntos.

Responda a las cuestiones

1) ¿Cuáles son las limitaciones del análisis O para medir la complejidad computacional?

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

2) Explica las diferencias entre una LinkedList y un arrayList. ¿Cuándo conviene utilizar cada una de estas estructuras? ¿Cuál es el orden de complejidad de una operación de (1) inserción, de (2) acceso aleatorio, de (3) búsqueda de un dato, y de (4) crecimiento de la estructura?

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

3) Explica las diferencias entre una pila y una cola. ¿Cuándo conviene utilizar cada una de estas estructuras? ¿Cuál es el orden de complejidad de una operación de (1) inserción, de (2) acceso aleatorio, y de (3) búsqueda de un dato?

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

4) Explique el método de ordenación denominado burbuja mejorada y describe las iteraciones sobre el siguiente conjunto de datos. No se solicita ningún tipo de código Python o pseudocódigo:

50 20 84 13 22 16 89 85

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

5) Dadas dos funciones y su número de operaciones:

A = 8192n
B = $8n^3$

Calcular a partir de qué tamaño de entrada A es más eficiente que B.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

Ejercicio 3 (1 punto). Tiempo estimado: 20 minutos. Calcular de forma recursiva el elemento más grande presente en una lista de números según la siguiente definición max_recursive(lista). Se solicita el código fuente en Python por lo que se puede abrir el fichero recursive_max.py, pero no está permitida la utilización de recursos externos para su solución.

El algoritmo recursivo se define como:
Caso base: longitud de la lista es un elemento. Devuelve el único elemento de la lista
Caso general: llamada a la función máximo de dos números enviando el último elemento de la lista y el resto de la lista como parámetros.

En cualquier caso, el algoritmo lanzará una excepción si el parámetro de entrada recibido no es una lista o es una lista vacía.

Se recomienda utilizar las operaciones básicas de estructuras de datos para trabajar con listas. La función para obtener y eliminar el último elemento de una lista es pop(). Se proporciona la función max_2_numbers(number1, number2) que devuelve el valor mayor y recibe como parámetros dos números enteros. No se permite la utilización de la función max de la librería estándar de Python.

Para la realización de este ejercicio se facilita el fichero max_recursive.py donde debe realizarse la función.

Además, se debe responder a las siguientes preguntas:

¿Qué tipo de recursividad se está empleando?

¿Cuál es la complejidad computacional del algoritmo?

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

Ejercicio 4 (2.5 puntos) Tiempo estimado: 60 minutos.

Existen multitud de juegos de videoconsola, móvil o navegador de internet que se encuentran basados en una competición entre dos equipos liderados por un entrenador. Cada equipo se encuentra formado no solo por el entrenador, sino también por un conjunto de combatientes. Sin embargo, solamente los combatientes son los que compiten, al estilo del conocido juego "Pokemon".

En este ejercicio, vamos a hacer una versión programada en Python para que dos jugadores puedan jugar a este juego. Los jugadores se encontrarán representados por los entrenadores de cada uno de los equipos.

No es necesario implementar el docString correspondiente a las funciones y métodos desarrollados, aunque se recomienda hacer el diagrama de flujo de los métodos en papel de forma previa a su resolución.

En base a estas especificaciones se solicita que:

a) Programe la clase Warrior.

i) Cada guerrero va a estar caracterizado por:
- Tipo de guerrero. En esta primera versión del juego, solamente 4 tipos de guerrero van a existir (Boxeador, Gladiador, UFC - Ultimate Fighting Championship, y MMA - Mixed martial arts).
- Tipo de arma que lleva a cabo el guerrero. En esta primera versión del juego, solamente 4 tipos de arma van a existir (Puñetazo, Patada, Codazo, Espada).
- Puntos de salud que tiene el guerrero. Deben de estar entre 1 y 100.
- Índice de ataque del guerrero. Deben de estar entre 1 y 10. En base al tipo de arma los valores son: Puñetazo:4, Patada:6, Codazo: 8, Espada: 10.
- Índice de defensa del guerrero. Deben de estar entre 1 y 10.
- Nota: Es obligatorio el uso de enumerados para los atributos del tipo de guerrero y tipo de arma. El índice de ataque del guerrero se recomienda que contenga el valor del tipo de arma. Se recuerda que el enumerado en Python tiene el formato "Clave / Valor".

ii) Incluya los atributos de esta clase y establezca la visibilidad adecuada (público, privado, protegido). Añada cualquier atributo y/o método que considere necesario.

iii) Programe un **constructor** que reciba los datos necesarios para crear un guerrero. El método **debe verificar** el tipo y valor de cada uno de los parámetros y **lanzar la excepción** correspondiente cuando no se cumplan los requisitos.

iv) Programe los métodos **setters y getters** para la clase en función de lo que necesite. Si no necesita algún o ningún getter y/o setter, **argumente por qué** en un comentario del módulo.

v) Programe el método **is_alive(self)** de la clase Warrior. Este método sirve para saber si el guerrero está vivo.

vi) Programe el método **fight_attack(self, Warrior warrior_to_attack)**. Método que implementa el ataque del guerrero usando un golpe sobre otro guerrero. Este método se basa en el método **fight_defense(self, int points_of_damage)** del guerrero atacado. Se aplicará el índice de ataque del guerrero atacante como representación del golpe dado.

vii) Programe el método **fight_defense(self, int points_of_damage)**. Este método implementa la defensa del guerrero de un golpe de otro guerrero. Este método actualiza el atributo de puntos de salud que tiene el guerrero en base a los puntos de daño recibidos menos el índice de defensa del propio guerrero. Si el índice de defensa es mayor que los puntos de ataque recibidos, el método devolverá false y el guerrero no sufrirá ningún daño.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI		

viii) Pruebe los objetos de la clase Warrior con los casos de prueba que se le han pasado. Modifíquelos si lo ve necesario, pero añada una justificación de cada uno de los cambios. Se penalizarán los cambios realizados en los casos de prueba proporcionados.

Para ayudar en el desarrollo de este ejercicio, **se le hace entrega de un UML parcialmente completo** de la posible implementación de este juego. Se puede modificar o realizar un UML completamente distinto, el cuál es necesario explicar brevemente en un documento de texto o en los comentarios de un fichero .py. En caso de realizar un UML diferente, no se aplicarán los criterios en base a los casos de prueba facilitados, que es probable que no funcionen sin una adaptación previa.

Se facilitan también los archivos vacíos dónde **deberán estar implementadas las clases** que se piden y que tienen que ser completadas por el alumno. En dichas clases, están ya añadidos los casos de prueba de cada una de ellas.

b) Programe una clase Coach.

i) Cada entrenador va a estar caracterizado por:
   (1) Un nombre que lo identifique (por ejemplo, Raúl, Luis, María,..).
   (2) La lista de guerreros que forman su equipo. Al menos debería existir un guerrero.

ii) Incluya los atributos de esta clase y establezca la visibilidad adecuada (público, privado, protegido). Añada cualquier atributo y/o método que considere necesario.

iii) Programe un **constructor** que reciba los datos necesarios para crear un entrenador. El método debe **verificar el tipo y valor** de cada uno de los parámetros y **lanzar la excepción** correspondiente cuando no se cumplan los requisitos.

iv) **No es un requisito necesario** la programación de métodos para añadir/eliminar guerreros de un entrenador ya existente.

v) Programe los métodos setters y getters para la clase en función de lo que necesite. Si no necesita algún o ningún getter y/o setter, argumente por qué en un comentario del módulo.

vi) Programe el método **is_undefeated(self)**. Este método sirve para saber si el entrenador está aún sin ser derrotado. Para ello evaluará que al menos uno de sus guerreros esté vivo.

vii) Programe el método **surrender(self)**. Este método sirve para que el propio Coach se rinda. Para ello tiene que poner los puntos de vida de cada uno de sus guerreros a 0.

viii) Programe los métodos **str** y **repr** para que cada uno muestre la información tal y como se especifica a continuación.
   El método **str** debe mostrar la información:
   (1) [WARRIOR_TYPE_NAME] with a [WEAPON_TYPE_NAME] and health: [HEALTH_POINTS]
   El método **repr** debe mostrar la información:
   (2) [ID]\t[WARRIOR_TYPE_NAME]\t[WEAPON_TYPE_NAME]

ix) Pruebe los objetos de la clase Coach con los casos de prueba que se le han pasado. Modifíquelos si lo ve necesario, pero añada una justificación de cada uno de los cambios. Este punto será evaluado inversamente proporcional al número de cambios realizados en los casos de prueba.

Para ayudar en el desarrollo de este ejercicio, **se le hace entrega de un UML parcialmente completo** de la posible implementación de este juego. Se puede modificar o realizar un UML completamente distinto, el cuál es necesario explicar brevemente en un documento de texto o en los comentarios de un fichero .py. En caso de realizar un UML diferente, no se aplicarán los criterios en base a los casos de prueba facilitados, que es probable que no funcionen sin una adaptación previa.

Se facilitan también los archivos vacíos dónde **deberán estar implementadas las clases** que se piden y que tienen que ser completadas por el alumno. En dichas clases, están ya añadidos los casos de prueba de cada una de ellas.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI	**GIN 02**	

Ejercicio 5 (1.75 puntos) Tiempo estimado: 20 minutos.

Se propone una extensión del juego propuesto en el ejercicio 4 en la que, en vez de jugar con un solo tipo de guerrero, se pueda jugar con tres tipos de guerrero diferentes.

Los tres tipos de guerrero que se van a implementar son Guerrero defensivo, Guerrero atacante y Super Guerrero.

En base a estas especificaciones en este ejercicio se solicita que:

a) Programe una clase que implemente un **guerrero defensivo** (Defensive Warrior).
   - Este guerrero va a poseer una característica nueva: Índice de defensa especial del guerrero. Debe de estar entre **11 y 20.**
   - La defensa de este guerrero se implementa de tal forma que **cuando sea este guerrero atacado exista un 50% de probabilidad de poder defenderse mediante una defensa normal o con una defensa especial**. Este 50% se puede calcular utilizando el módulo **random**.
   - En el caso de que el índice de defensa aplicado sea mayor que el índice de ataque recibido, se entenderá como que no hay sufrido un ataque.

b) Programe una clase que implemente un **guerrero ofensivo** (Offensive Warrior).
   - Este guerrero va a poseer una característica nueva: Índice de ataque especial del guerrero. Debe de estar entre **11 y 20**.
   - El ataque de este guerrero se implementa de tal forma que cuando **este guerrero esté atacando exista un 50% de probabilidad de poder atacar mediante un ataque normal o con un ataque especial**.
   - Tenga en cuenta que, si **se cambia el arma**, se **recomienda también cambiar tanto el índice de ataque normal como el índice de ataque especial del guerrero**. Se puede utilizar un valor aleatorio entre 11 y 20 al cambiar de arma para modificar el índice especial del guerrero.

c) Programe una clase que implemente un **super guerrero** (Super Warrior).
   - Este guerrero es la **combinación del guerrero defensivo y guerrero ofensivo** en un mismo guerrero. Este guerrero implementará el método de ataque del guerrero ofensivo y el método de defensa del guerrero defensivo (ver UML).
   - La utilización de la **herencia múltiple** para la implementación del super guerrero es **totalmente opcional**. **No se penalizará la no utilización de ésta**.

Reutilice tanto código como sea posible del ejercicio 4 para hacer esta extensión del juego.

No es necesario implementar el docString correspondiente a las funciones y métodos desarrollados, aunque se recomienda hacerlo para facilitar la comprensión por parte del estudiante.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 18/01/21	DNI	**GIN 02**	

Ejercicio 6 (1.75 puntos) Tiempo estimado: 30 minutos.

La aplicación programada en Python va a comenzar leyendo las características de los guerreros a partir de un fichero csv seleccionado. Cada jugador va a tener como máximo 3 guerreros. Una vez formados ambos equipos se podrá empezar la partida.

En este último ejercicio se pide que se implemente el módulo principal del juego. Este módulo principal va a implementar la partida que van a jugar los jugadores, los cuales se encuentran identificados como los entrenadores de los combatientes.
En este **módulo main**, lo primero que se hace es obtener la configuración deseada de los guerreros por parte de cada entrenador. Hay que tener en cuenta, que cada entrenador solamente va a tener tres combatientes al iniciar la partida.

Las **características de cada uno de los grupos de combatientes** que va a poseer cada entrenador van a ser introducidas **a través de archivos CSV**. Los dos archivos CSV a utilizar por en este ejercicio han sido proporcionados:

- **coach_1_warriors.csv**
- **coach_2_warriors.csv**

Se deben crear los guerreros leyendo los ficheros CSV para cada uno de los entrenadores (ver Flowchart).

Seguidamente, una vez que se tienen las configuraciones de cada uno de los equipos, el combate puede comenzar entre los entrenadores.

Los combates entre guerreros se van a realizar secuencialmente. Además, un combate se entiende como aquel llevado a cabo por un guerrero de cada uno de los equipos. Por lo tanto, en cada turno un guerrero de uno de los equipos va a atacar a otro guerrero del otro equipo. Una vez que dos guerreros entran en combate, no se para este combate hasta que uno de los dos guerreros es derrotado, momento en el que el siguiente guerrero del entrenador debe entrar al combate, solicitando al entrenador que seleccione un guerrero de aquellos que estén vivos.

Los entrenadores van a elegir el orden en que los guerreros son puestos en combate. Para ello, se debe listar al jugador la lista de guerreros vivos y no seleccionados aún y éste debe introducir el id de cada guerrero. Se recomienda crear una estructura de datos independiente que contenga únicamente los guerreros del entrenador vivos y que se vayan sacando de dicha estructura una vez hayan sido derrotados/muertos. Una vez que el guerrero es derrotado, no puede volver a entrar en combate.

Cuando uno de los dos entrenadores tenga ya todos sus guerreros derrotados al acabar el turno, se acaba el juego, y se indica no solo quién es el ganador, sino también las estadísticas de cada equipo en base a los puntos de vida de los guerreros. En el caso de que los dos acaben sin guerreros en el mismo turno, el juego indicará un empate. En cada turno los dos guerreros atacan con independencia de si en ese mismo turno es derrotado.

Recuerde utilizar las clases implementadas tanto en el ejercicio 4 como en el ejercicio 5 para implementar todos los componentes de este juego.

El diagrama de flujo de este módulo principal le ha sido proporcionado a modo de guía.