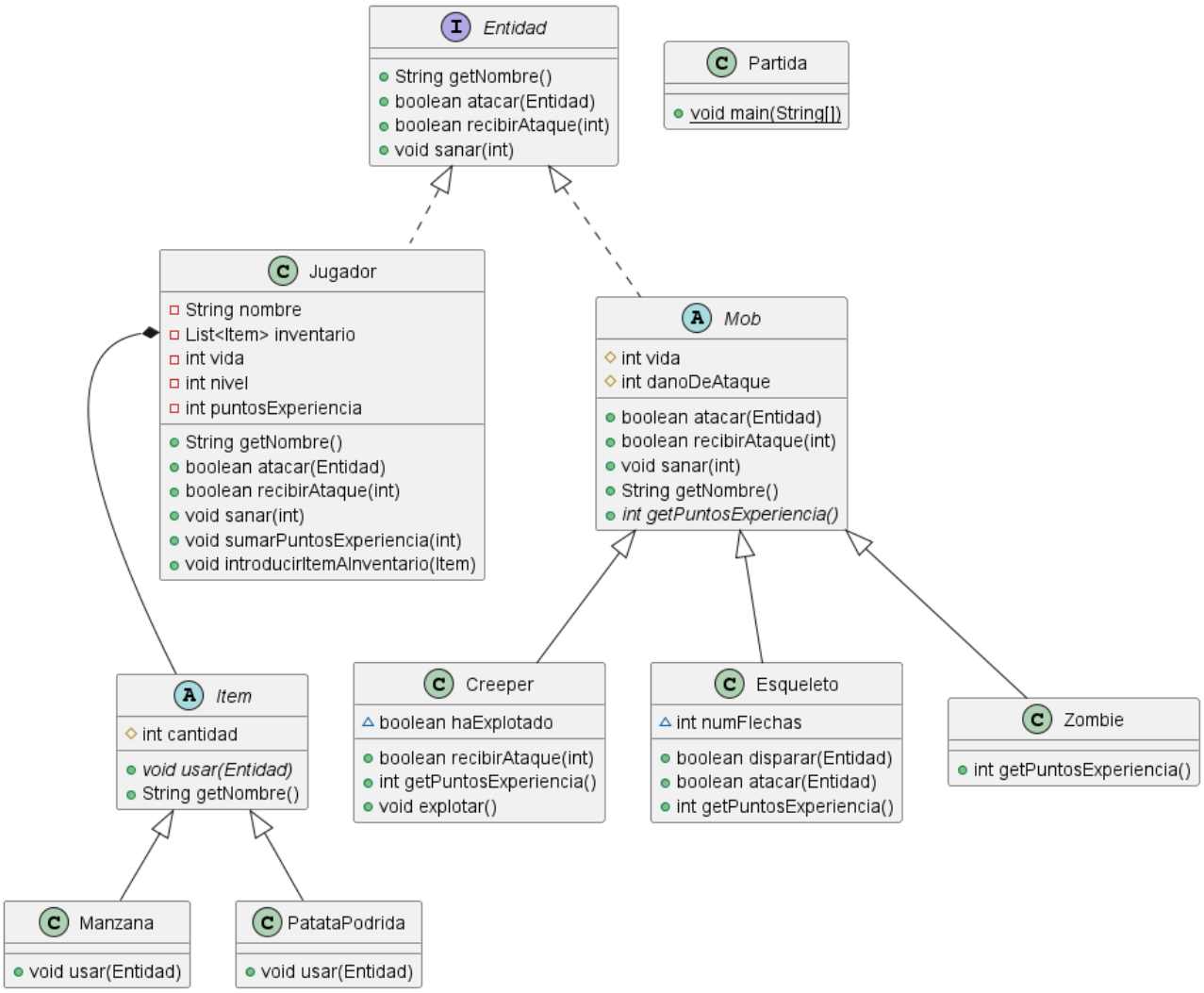
maría ana sanz CENTRO INTEGRADO	Ciclo: **C.F.G.S. Desarrollo de Aplicaciones Multiplataforma** Módulo: **Programación 1º** Evaluación: **3ª - UT7 Práctica** Fecha: 03 - abril - 2023
Nombre y Apellidos:	**Nota:**

Para este examen se deberá modificar el proyecto ExamenUT7 proporcionado con el fin de generar las clases abstractas e interfaces requeridas, adaptando todo el código necesario para ello. Puedes utilizar cualquier IDE aunque se sugiere Intellij IDEA.

No olvides poner tu nombre detrás de la etiqueta *@author*

El proyecto ExamenUT7 es un breve simulacro del juego de Minecraft. El desarrollador que ha creado este programa quiere poder ampliarlo e incluir distintos tipos de enemigos (mobs) y objetos (items) sin que deba modificar mucho código. Para ello, ha realizado una consulta a su profesor de programación y juntos han concluido que el **modelo que quieren llegar a obtener** es el siguiente:



Parte 1: Modificando el inventario del jugador

Esta parte consistirá en modificar el inventario del jugador para que acepte cualquier tipo de objeto. Para ello realizaremos las siguientes modificaciones al código:

- Generar la clase abstracta **Item**: Se deberá crear una clase abstracta que corresponda con el modelo de datos que se especifica en el esquema UML arriba descrito. Deberá contener las variables y métodos indicados. Dado que todavía no existe la interfaz **Entidad**, el método usar(Entidad) deberá definirse como usar(Object). (4 puntos)
- Actualizar las clases **Manzana** y **PatataPodrida**: Se deberán actualizar estas clases para que hereden de la clase **Item** previamente generada. Se deberán eliminar aquellos métodos que ya no sean necesarios y se deberán indicar los métodos sobrescritos con la anotación correspondiente. (6 puntos)
- Actualizar inventario de la clase **Jugador**: Se deberá modificar la variable global "inventario" para que acepte los objetos de tipo **Item**. A continuación, se deberá modificar el método recibirAtaque() para que trabaje con clases de tipo **Item**. (6 puntos)

Parte 2: Unificando entidades

Dado que queremos unificar el comportamiento de todas las entidades del juego, generaremos una interfaz que unifique estos aspectos. Para ello:

- Generar la interfaz **Entidad**: Se deberá crear una interfaz que corresponda con el modelo de datos que se especifica en el esquema UML arriba descrito. Deberá contener los métodos indicados. (4 puntos)
- Actualizar la clase **Item**: Se deberá modificar el método usar(Object) para que trabaje con objetos de tipo **Entidad**. Después, se deberán modificar aquellas clases que heredan este método y modificar el código correspondiente para que trabajen con objetos de tipo **Entidad**. (8 puntos)
- Actualizar las clases **Jugador**, **Creeper**, **Esqueleto** y **Zombie**: Se deberán actualizar estas clases para que hereden de la clase **Entidad** previamente generada. Se deberán indicar los métodos sobrescritos con la anotación correspondiente. Se deberán modificar los métodos atacar(Object) en todas las clases que hereden de **Entidad** para que trabaje con clases de tipo **Entidad**. (12 puntos)
- Generar la clase abstracta **Mob**: Se deberá crear una clase abstracta que corresponda con el modelo de datos que se especifica en el esquema UML arriba descrito. Deberá contener las variables y métodos indicados. (4 puntos)
- Actualizar las clases **Creeper, Esqueleto y Zombie**: Se deberán actualizar estas clases para que hereden de la clase **Mob** previamente generada. Se deberán eliminar aquellos métodos que ya no sean necesarios y se deberán indicar los métodos sobrescritos con la anotación correspondiente. (10 puntos)

Parte 3: Modificando la partida

Una vez que tenemos el modelo actualizado al diseño UML que se ha planteado arriba, nos falta actualizar la ejecución del juego para que funcione correctamente en base a las modificaciones realizadas. Para ello:

- Actualizar la clase **Partida**: Se deberán modificar el método main() para que en vez de emplear el tipo de objeto **Object**, emplee el tipo de objeto que corresponde en cada apartado del código. (12 puntos)

Parte 4: Implementando interfaces

Para finalizar implementaremos una serie de interfaces:

- Actualizar la clase **Jugador**: Se deberá modificar esta clase para que implemente la interfaz que nos permita realizar clones. Una vez realizada esta modificación, se deberá generar un clon del jugador "Steve" en la línea 17 de la clase **Partida** (puede variar si has realizado alguna modificación, pero está indicado con un TODO) y renombrar a dicho clon para que se llame "Alex" e incluirlo en la lista de jugadores. Para ello, se deberá emplear una de las interfaces vistas en clase, no vale obtenerlo de cualquier manera. (10 puntos)
- Ordenar los mobs por daño de ataque: Se deberá modificar la línea 30 de la clase **Partida** (puede variar si has realizado alguna modificación, pero está indicado con un TODO) para obtener el mob que mas daño de ataque tenga en vez de uno aleatorio. Para ello, se deberá emplear una de las interfaces vistas en clase, no vale obtenerlo de cualquier manera. (8 puntos)
- Ordenar los jugadores por cantidad de vida: Se deberá modificar la línea 34 de la clase **Partida** (puede variar si has realizado alguna modificación, pero está indicado con un TODO) para obtener el jugador que tenga más vida en vez de uno aleatorio. Para ello, se deberá emplear una de las interfaces vistas en clase, no vale obtenerlo de cualquier manera. (6 puntos)
- Mostrar los jugadores por nivel y experiencia: Se deberá modificar la línea 82 de la clase **Partida** (puede variar si has realizado alguna modificación, pero está indicado con un TODO) para visualizar un listado con los jugadores ordenador en base a su nivel y experiencia (¡Emplear ambos parámetros!). Para ello, se deberá emplear una de las interfaces vistas en clase, no vale obtenerlo de cualquier manera. (10 puntos)

Puntuación	
PARTE 1	
Generar la clase abstracta Item	4
Actualizar las clases Manzana y PatataPodrida	6
Actualizar inventario de la clase Jugador	6
PARTE 2	
Generar la interfaz Entidad	4
Actualizar la clase Item	8
Actualizar las clases Jugador, Creeper, Esqueleto y Zombie	12
Generar la clase abstracta Mob	4
Actualizar las clases Creeper, Esqueleto y Zombie	10
PARTE 3	
Actualizar la clase Partida	12
PARTE 4	
Actualizar la clase Jugador	10
Ordenar los mobs por daño de ataque	8
Ordenar los jugadores por cantidad de vida	6
Mostrar los jugadores por nivel y experiencia	10
TOTAL	
TOTAL	100