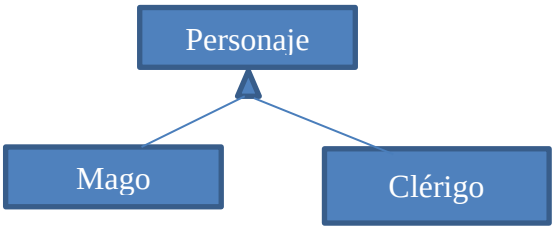
Laboratorio I
Implementación POO Básico

1. Ejercicio:

Se va a desarrollar un juego de rol, del cual se requiere programar en Java, parte del esquema de personajes. Para ello se nos han dado las siguientes directrices:

- Todos los personajes cuentan con los siguientes datos:
  - nombre: una cadena.
  - raza: una cadena que puede tomar los valores “humano”, “elfo”, “enano” u “orco”.
  - fuerza: un entero entre 0 y 20.
  - inteligencia: un entero entre 0 y 20.
  - puntos de vida máximos: un entero entre 0 y 100.
  - puntos de vida actuales: un entero entre 0 y puntos de vida máximos.

- Tenemos que programar 2 tipos de Personajes: los Magos y los Clérigos.
- Los Magos y Clérigos, por ser de tipo **Personaje**, tienen los atributos descritos anteriormente y adicionalmente tiene atributos y comportamientos propios que los distinguen entre ellos. En este caso confirmamos la existencia de una relación de herencia entre clases, lo cual representamos como se ve a continuación:



Se pide:

Apartado 1:

Escribe una clase **Personaje** que reúna los atributos mencionados en el enunciado. Dicha clase debe incluir:

- Un constructor **sin parámetros** para inicializar con valores por defecto de acuerdo al tipo de dato, cada uno de los atributos de la clase.
- Un constructor **con parámetros**, para inicializar con valores externos el estado de los atributos de la clase. En este caso se supone que los puntos de vida actuales son iguales a los máximos al inicializarse.
- Métodos **set y get** para todos los atributos de la clase.
- Método **imprimir()** que muestre por pantalla los datos del personaje.

Apartado 2:

Escribe la clase **Mago** teniendo en cuenta lo siguiente:

- Al crearse, un mago no puede tener en inteligencia un valor menor que 17 ni en fuerza un valor mayor que 15. Esta es una validación de datos de entrada que deben hacer antes de instanciar un objeto Mago.
- Un Mago se caracteriza por almacenar los nombres de los hechizos que ha memorizado. Un Mago normal sólo puede memorizar a la vez un máximo de 4 hechizos. Esta característica representa un atributo propio de los Magos, por lo que debe declararse como atributo en la clase correspondiente.
- Al crear la clase **Mago** se debe:
  - Indicar que es una clase que hereda de Personaje utilizando la palabra reservada **extends**. Para ello se debe colocar el encabezado como sigue: **public class Mago extends Personaje**.
  - Declarar el atributo hechizos, para ello se requiere implementar un arreglo de String: private String [] hechizos; //Sección atributos.
  - Definir un constructor **sin parámetros** para inicializar con valores por defecto de acuerdo al tipo de dato, cada uno de los atributos de la clase. Este constructor debe contener:
    - Llamado al constructor de la clase padre: Dado a que Mago hereda de Personaje, es necesario invocar al constructor de la clase padre para inicializar los atributos heredados. Para ello utilizar la sentencia **super()**. Tener en cuenta que en todo constructor el llamado a super() siempre debe estar al inicio del mismo.

        - Reserva del espacio de memoria para hechizos: Seguidamente se procede a reservar el espacio de memoria en el constructor para los hechizos**: hechizos = new String[4];**
        - Inicialización del arreglo de hechizos: Aquí se inicializan las posiciones del arreglo hechizos con blancos o cualquier carácter que les sirva posteriormente como referencia para identificar si la casilla está llena o vacía. Para el recorrido del arreglo utilizar **foreach** (investigar uso del mismo).
    - Definir un constructor **con parámetros**, para inicializar con valores externos el estado de los atributos de la clase. Este constructor debe contener:
        - Llamado al constructor de la clase padre: Invocar al constructor de la clase padre utilizando la sentencia **super().** En este caso el uso de **super()** debe contener como argumentos los atributos que recibe el constructor con parámetros invocado. Ejemplo: **super(Argumento1, argumento2, etc);**
        - Reserva del espacio de memoria para hechizos: Ver detalles en constructos sin parámetros.
        - Inicialización del arreglo de hechizos: Ver detalles en constructor sin parámetros.
    - No es necesario definir set/get de los atributos de personaje porque Mago hereda los existentes en la clase Personaje.
    - No es necesario set/get de hechizos porque se definirán métodos para cargar y vaciar dicho arreglo a medida que transcurre el juego.
    - Añadir los siguientes métodos:
        - **aprendeHechizo:** que tiene un parámetro de tipo String que representa el hechizo que se va a aprender. Este método añade un hechizo al array (deberá buscar un espacio libre en el array).
        - **lanzaHechizo:** que tiene como parámetro un objeto de tipo Personaje que será el personaje sobre el que recae el hechizo. Las acciones a tomar serán las de restar 10 de los puntos de vida actuales de dicho personaje y olvidar el hechizo (borrarlo del array).
    - Definir el método **imprimir()** para que se muestren los nuevos datos incluyendo la lista de hechizos.
        - Colocar a este método el mismo nombre y la misma cantidad de parámetros como el definido en la clase Personajes.
        - En la implementación del método se debe:
            - Invocar el método imprimir de la clase personaje, dado a que ese método permite imprimir los datos básicos de un personaje cualquiera. Para ello se escribe la instrucción **super.imprimir(),** lo que permite invocar este método de la clase padre. Y de esta manera le damos el segundo uso a la sentencia **super.**
            - Mostrar hechizos aprendidos por el Mago. Esto se hace recorriendo el arreglo de hechizos y mostrando lo que contiene.

Apartado 3:

Escribe la clase Clérigo teniendo en cuenta las siguientes restricciones:

- Al crearse, un clérigo no puede tener una fuerza con un valor menor que 18 y una inteligencia con un valor menor que 12 ni mayor que 16 ambos incluidos. Esta es una validación de datos de entrada que deben hacer antes de instanciar un objeto Clérigo.
- El clérigo reza a un dios para obtener el don de la curación. Este atributo es propio de clérigo. Realizar las adecuaciones a los constructores de clérigo para realizar la instancia de dicho objeto. Para el desarrollo de los constructores considerar todo lo explicado en el caso de Mago.
- Un clérigo tiene el don de curar, por lo tanto, la clase deberá tener un método curar que recibe como parámetro un objeto de tipo Personaje sobre el que recae la acción de curar y que aumenta en 10 los puntos de vida.
- Sobreescribe el método imprime para que muestre además de los datos básicos el nombre del Dios. Realizar adecuaciones como en la clase Mago.

Apartado 4:

Se pide:

a. Implementación en JAVA del modelo planteado. (Definición de atributos, constructor (con y Sin parámetros, métodos get/set, otros métodos etc.).
b. En la solución del ejercicio se debe reflejar: Sobreescritura de métodos, sobrecarga de métodos, sentencia super, **referencia this, clase abstracta.**
c. En el main imprimir la información asociada a Magos y Clérigos. Mostrar además los resultados de lo siguiente:
    - Crear 2 magos (A y B) y un clérigo (C) y en el que tengan que realizar las siguientes acciones.
    - Imprimir los datos de los tres personajes
    - El mago A aprende 2 hechizos.
    - El mago B aprende 1 hechizo.
    - Imprimir los datos de los magos
    - El mago A lanza un hechizo sobre el mago B

- El mago B lanza un hechizo sobre el mago A
- El clérigo cura al mago B
- El mago A lanza un hechizo sobre el mago B
- Imprimir los datos de los tres personajes

Consideraciones

Para el desarrollo de la práctica debe realizar lo que se indica a continuación:

1. Crear un proyecto nuevo con el nombre de LabI_XXXXX donde XXX es el nombre y apellido del estudiante.
2. Para la captura de datos por consola, utilizar la clase Scanner
3. Para mostrar mensajes por pantalla, usar la clase JoptionPane. (Investigar modo de uso)
4. Implementación en JAVA del modelo planteado: Definición de atributos, constructor (con y Sin parámetros, métodos get/set, otros métodos etc.).
5. En la solución del ejercicio se debe reflejar: Sobreescritura de métodos, sobrecarga de métodos, uso de la sentencia super, referencia this, identificar e implementar clase abstracta.
6. Mostrar los datos haciendo uso de polimorfismo.