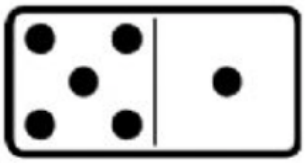
Deber de Herencia y Polimorfismo

Vas a diseñar un sistema de juego que permita a los jugadores jugar una versión simplificada del juego dominoes. Dominoes es un juego basado en fichas. Cada ficha tiene dos lados. Cada lado de la ficha tiene un número entre 1 y 6. En el juego cada jugador tiene un grupo de fichas. Al iniciar el juego uno de los jugadores lanza una ficha. El siguiente jugador en la mesa debe lanzar otra ficha que coincida con uno de los lados de la ficha ya en la mesa. El juego continúa hasta que alguno de los jugadores se quede sin fichas

lado1 lado2



En esta versión del juego las fichas se colocarán en la mesa ya sea al inicio o al final. Las fichas en la mesa se las va a llamar línea de juego

INICIO LÍNEA DE JUEGO FIN LÍNEA DE JUEGO

Ejemplo del juego:

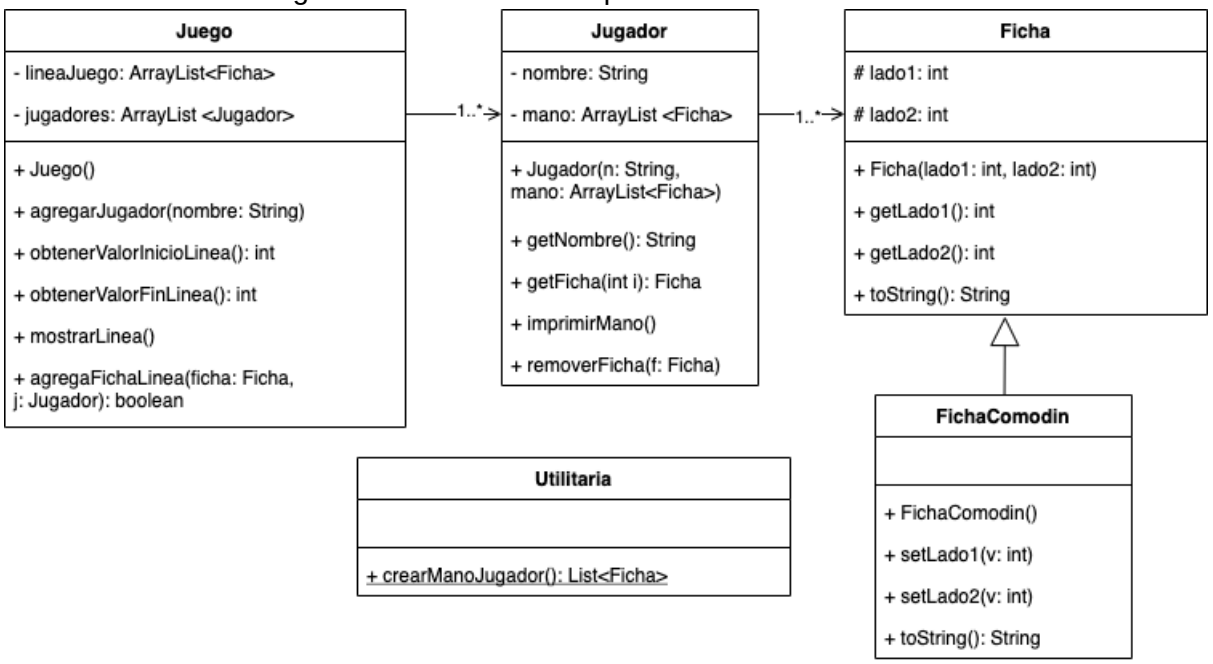
Jugador 0: Mano -> **5:2 - 0:1 - *-1:-1***
Línea de Juego ->
Índice de ficha para jugar (0 es el primero): 0
Movimiento Válido.
Nueva Línea de Juego -> **5:2**

Jugador 1: Mano -> **6:6 - 2:3 - 6:1**
Línea de Juego -> **5:2**
Índice de ficha para jugar (0 es el primero): 0
Ficha tenía 6:6 No puedo jugar esa ficha, inténtalo de nuevo
Índice de ficha para jugar (0 es el primero): 1
Movimiento Válido.
Nueva Línea de Juego -> **5:2 - 2:3**

Jugador 0: Mano -> **0:1 - *-1:-1***
Línea de Juego -> **5:2 - 2:3**
Índice de ficha para jugar (0 es el primero): 1
Movimiento Válido.
Posición: Inicio
Número Inicio: 6
Nueva Línea de Juego -> ***6:-1* - 5:2 - 2:3**

Jugador 1: Mano -> **6:6 - 6:1**
Línea de Juego -> ***6:-1* - 5:2 - 2:3**
Índice de ficha para jugar (0 es el primero): 0
Nueva Línea de Juego -> **6:6 -** ***6:-1* - 5:2 - 2:3**

A usted se le da el diagrama de clases de la aplicación:



Realice lo siguiente:

1. Cree la clase **Ficha** según el diagrama de clases. La clase **Ficha** modela una ficha del juego que tiene dos lados.
    a. El constructor recibe los valores de los lados y los inicializa.
    b. Los métodos get de los lados
    c. El método toString() retorna un String con el siguiente formato
    lado1:lado2

2. Cree la clase **FichaComodin** según el diagrama de clases. La clase **FichaComodin** modela una ficha especial del juego. Esta ficha al momento de ser creada sus lados debe tener los valores de -1.
    a. El constructor inicializa los valores de los lados con -1
    b. Los métodos set de los lados
    c. El método toString() retorna un String con el siguiente formato - debe usar el toString() del padre
    lado1:lado2

3. La clase Jugador tiene dos variables de instancia:
        - **nombre** que contiene el nombre del jugador
        - **mano** que tiene la lista de fichas que posee el jugador y con las cuales puede jugar.
    b. El constructor de la clase recibe el nombre y mano del jugador y crea uno nuevo

c. El método **getFicha** retorna la ficha de la mano del jugador en el índice pasado como parámetro. Si el índice es válido retorna la ficha. Si el índice es inválido retorna null
d. El método **removerFicha(f: Ficha)** remueve la ficha recibida de la mano del jugador.
e. El método **getNombre** retorna el nombre del jugador
f. El método **imprimirMano**() muestra en pantalla la mano del jugador con este formato:

lado1:lado2-lado1:lado2-lado1:lado2

4. En la clase Utilitaria va a implementar el siguiente comportamiento:
   a. El método estático **crearManoJugador(): ArrayList<Ficha>** es un método que genera al una lista de 5 objetos de tipo ficha con números aleatorio y una ficha comodín. El siguiente código genera un número aleatorio entre 1 y 6:

```
int numeroAleatorio = (int) (Math.random()*6+1);
```

5. Cree la clase **Juego** según el diagrama de clases.
   a. La clase **Juego** tiene dos variables de instancia:
      - **jugadores**: que contiene una lista de objetos jugadores en el orden que juegan
      - **lineaJuego**: que contiene una lista de las fichas que han sido jugadas hasta el momento.
   b. El **constructor** inicializa las variables **lineaJuego** y **jugadores** como ArrayList vacíos.
   a. El método **agregarJugador(nombre: String)** crea un nuevo objeto de tipo jugador y lo agrega a la lista de jugadores. El nombre del jugador es el nombre que recibe como parámetro y la mano la obtiene llamando al método **crearManoJugador()** de la clase Utilitaria.
   b. Implemente el método **obtenerValorInicioLinea()** que retorne el valor del lado1 de la ficha del inicio de la línea.
   c. Implemente el método **obtenerValorFinLinea()** que retorne el valor del lado2 de la ficha del fin de la línea.
   d. Implemente el método **mostrarLínea()** para que se muestre por pantalla los valores de las fichas en la línea como se ve a continuación:

ficha1 - ficha2 - .. fichaN

***6:-1* - 5:2 - 2:3**

   e. El método es **agregarFichaLinea(f: Ficha, j: Jugador): boolean** el método recibe la ficha que se quiere agregar a la línea de juego. El método realiza lo siguiente:
      - Si la ficha **NO** es de tipo FichaComodin
        - Si no hay fichas en **lineaJuego** se agrega la ficha a **lineaJuego.**
        - Si hay fichas en línea de juego:

a. La acción es válida si el **lado2** coincide **con el valor de inicio de la línea** o el **lado1** coincide **con el valor de fin de la línea.**
    i. En caso de ser la acción válida se agrega la ficha en **lineaJuego** en la posición que coincidió, se remueve la ficha de la mano del jugador y se retorna true
    ii. Si la acción no es válida se retorna false.

- Si la ficha **ES** de tipo **FichaComodin.**
    - Si no hay fichas en **lineaJuego** se agrega la ficha a **lineaJuego y se pide al jugador los valores del lado1 y lado2 de la ficha y se los actualiza.**
    - Si hay fichas en línea de juego:
        a. Se pide al jugador la posición donde lo quiere agregar en la línea: **Inicio** ó **Fin**.
        b. Si ingresa **Inicio** se pone la ficha al inicio de la línea, se pide el valor del lado1 y lo actualiza.
        c. Si ingresa **Fin** se pone la ficha al final de la línea, se pide el valor del lado2 y lo actualiza.