

PARTE PRÁCTICA [6,5 PUNTOS]:

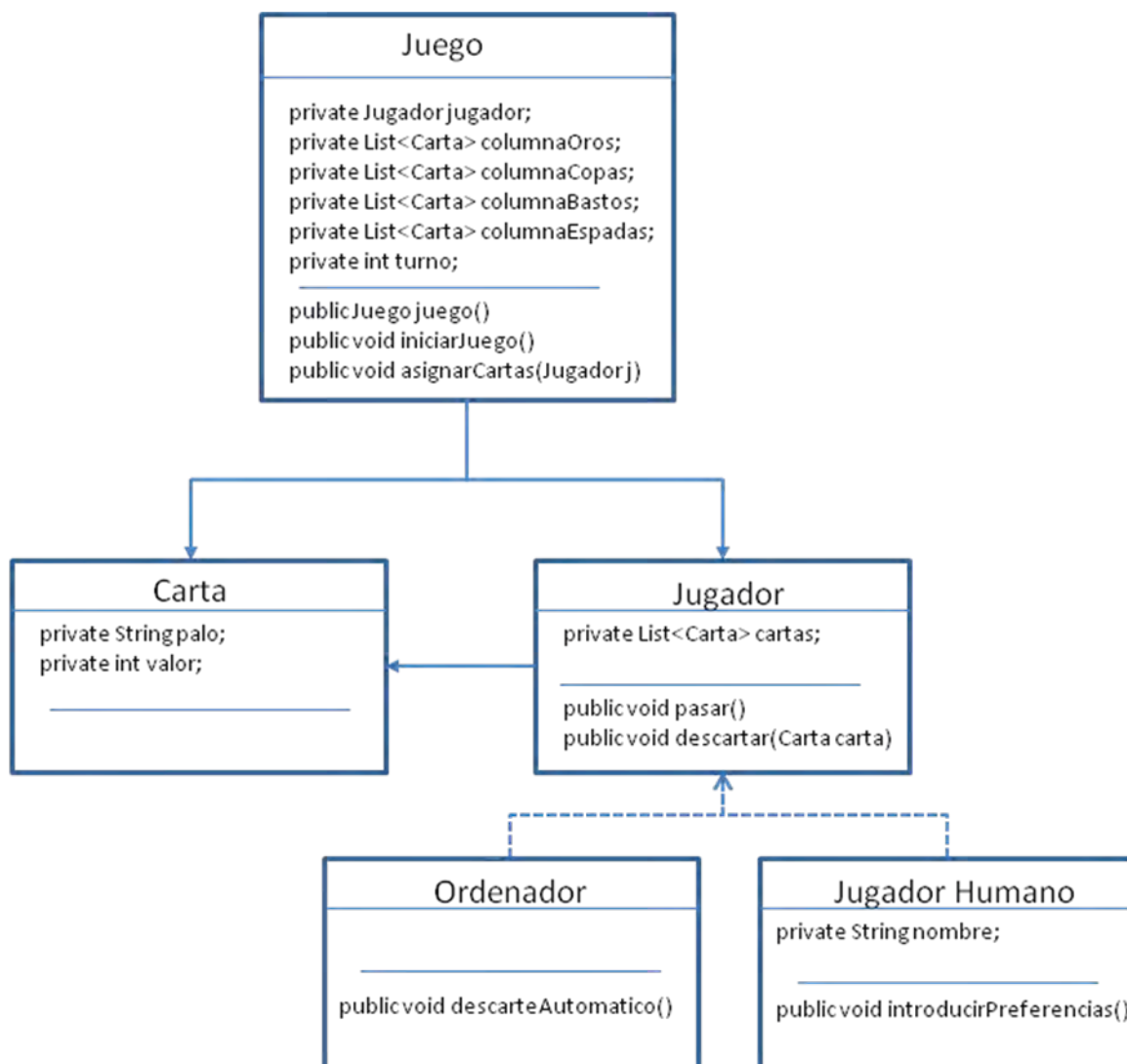
El juego del Cinquillo Solitario es una variedad del popular Cinquillo en el cual un jugador puede jugar de manera online contra el ordenador. El juego se inicia con el reparto de todas las cartas de una baraja española que consta de 48 naipes o cartas, clasificados en cuatro palos (oros, bastos, copas y espadas) y numerados del 1 al 12. El objetivo del juego consiste en descartarse (quedarse sin cartas) antes que el oponente.

El jugador que posee el cinco de oros lo coloca boca arriba encima de la mesa y de esta forma empieza el turno de descartes. En turnos alternativos, cada jugador puede descartarse de máximo un naipe. Solo se pueden colocar cincos o todas aquellas cartas que siguen en progresión ascendente o descendente a las que hay en la mesa y sean del mismo palo. Es decir, si por ejemplo solamente está colocado el cinco de oros en la mesa, los jugadores solo podrán colocar el seis o el cuatro de oros o un cinco de otro palo.

Si un jugador no puede colocar ninguna carta pasa, y le toca el turno al siguiente jugador. Nunca se puede pasar si se puede colocar alguna carta. El primer jugador que consigue colocar todas sus cartas sobre la mesa es el ganador.

En cuanto a la dinámica del juego, uno de los contrincantes será un jugador humano (introducimos sus datos y sus preferencias por el teclado) y el otro contrincante será el propio ordenador.

- a) **[1,5 puntos]** Diseñe las clases necesarias que permita desarrollar el juego del Cinquillo Online utilizando un paradigma orientado a objetos. Debe hacerse uso de los mecanismos de la programación orientada a objetos siempre que sea posible y un diseño que permita la reutilización del código y facilite su mantenimiento.



- b) **[1,5 puntos]** Implemente un método que defina el funcionamiento del ordenador, teniendo en cuenta que todos sus procesos tienen que hacerse automáticamente sin la intervención del usuario.

```
public void descartar(){

    ArrayList<ArrayList<Carta>> columnas = new ArrayList<ArrayList<Carta>>();
    columnas = juego.getCartas();
    Carta cartaMenor = null;
    Carta cartaMayor = null;
    boolean descarte = false;

    if(esPosibleDescarte(columnas)){

        for (ArrayList<Carta> columna : columnas) {

            if(columna.size()>0){

                String palo = columna.get(0).getPalo();
                cartaMayor = columna.get(0);
                cartaMayor = columna.get(columna.size());

                for (Carta carta : super.getCartas()) {

                    if(carta.getPalo().equals(palo)){
                        if(carta.getValor()+1==cartaMenor.getValor()){
                            descartar(carta);
                            descarte = true;
                            break;
                        }else if(carta.getValor()-1==cartaMayor.getValor()){
                            descartar(carta);
                            descarte = true;
                            break;
                        }
                    }
                }
            }
            if(descarte){
                break;
            }
        }
    }

    if(!descarte){
        for (Carta carta : super.getCartas()) {
            if(carta.getValor()==5){
                descartar(carta);
            }
        }
    }

    }else{
        super.pasar();
    }

}
```

- c) **[1,5 puntos]** Proporcione un método que muestre la lógica del juego, definiendo la información necesaria para establecer el uso de clases, interacciones entre elementos, declaración y uso de variables y métodos necesarios, etc.

```
public void iniciarJuego(){

    boolean cartasPendientes = true;

    Jugador jugadorHumano = new Jugador();
    Jugador jugadorOrdenador = new Jugador();

    asignarCartas(jugadorHumano);
    asignarCartas(jugadorOrdenador);

    while(cartasPendientes){

        if(jugadorHumano.iniciarTurno()){
            System.out.println("Jugador Humano Ganador");
            break;
        }else if(jugadorOrdenador.iniciarTurno()){
            System.out.println("Ordenador Ganador");
            break;
        }
    }

}
```

- d) **[2,0 puntos]** Indiqué qué modificaciones son necesarias introducir en la aplicación para permitir la participación de varios jugadores humanos (hasta 4). Para ello el juego en lugar de constar de partidas individuales en las cuales gana el jugador que antes se descarta, para a ser una partida formada por un conjunto de rondas. El ordenador deberá llevar un registro de los puntos que cada jugador ha conseguido en cada ronda. El jugador que consigue descartarse primero logrará 3 puntos, el jugador o jugadores que se quede con un mayor número de cartas al finalizar la ronda obtendrá 0 puntos. El resto obtendrá 1 punto. La partida finaliza cuando un jugador consiga llegar al menos a los 10 puntos, ganando el que más puntos tenga en caso de superar esta puntuación varios jugadores. En caso de empate se jugará una ronda extra para decidir el ganador.

- Para la participación de varios jugadores, tan solo es necesario tener una instancia de cada jugador y asignar los turnos en un orden lógico. En lugar de tener una variable con la instancia del jugador, sería necesaria una lista de jugadores.
- Para que el juego pueda anotar el tanteo de las partidas, se debería crear una clase “Tanteo” que almacene por cada una de las rondas de la partida, los puntos conseguidos por cada jugador. También sería necesario que almacenara los puntos que cada jugador ha conseguido de manera global. Además, para implementar la asignación de puntos, habría que crear un método que al final de cada partida analice las cartas que los jugadores no han conseguido descartar. Este método actualizaría la información de los puntos de cada jugador y debe dar por finalizada la partida en caso de que un jugador llegase a la puntuación máxima. La ronda extra también sería responsabilidad de este método y de la información almacenada en la clase Tanteo.