Práctica: Juego Blackjack

En esta práctica vamos a crear el juego de cartas Blackjack, también conocido como veintiuno. Si bien ya realizamos un ejercicio similar, en este caso vamos a hacerlo aplicando las técnicas de programación orientada a objetos. Además, si aplicamos la norma de que haya pocas clases, que realicen entrada/salida, podremos posteriormente añadirle una interfaz gráfica de forma sencilla.

Objetivo

El objetivo del juego consiste en obtener 21 puntos o tener más puntos que el resto de jugadores sin pasarse de 21 mediante la suma de los valores de las cartas.

Valor de las cartas

El Blackjack se juega con baraja francesa, este tipo de baraja y tiene 52 cartas repartidas en 4 palos (tréboles, diamantes, corazones y espadas), lo que da un total de 13 cartas por palo (A,2,3,4,5,6,7,8,9,10,J,Q,K). Para obtener la suma de los valores de las cartas hay que fijarse en el valor numérico de estas, sin importar el palo, de la siguiente forma:

- Las cartas numéricas (2,3,4,5,6,7,8,9,10) suman su valor numérico.
- Las figuras (J,Q,K) suman 10.
- El AS (A) puede sumar 11 o 1 según convenga al jugador.

Desarrollo del juego

1. En primer lugar los jugadores realizan su apuesta.
2. La banca reparte 2 cartas descubiertas a cada jugador.
3. La banca se reparte a sí misma 2 cartas la primera descubierta y la segunda cubierta.
4. Si algún jugador ha obtenido como suma 21 en las 2 primeras cartas gana automáticamente.
5. Una vez realizadas las apuestas y repartidas las cartas, cada jugador tiene la opción de pedir más cartas o plantarse y quedarse con las que tiene. Si pide una nueva carta y la suma total supera 21 ha perdido.
6. Si todos los jugadores se han pasado de 21, gana la banca. Si alguno de los jugadores se ha plantado, es turno de la banca que tiene las siguientes restricciones:
    - Si su puntuación inicial es 16 o menor debe pedir carta obligatoriamente.
    - Si su puntuación es 17 o más deberá plantarse independientemente de las jugadas que tengan cada uno de los jugadores.

Premios

Aquellos jugadores que superen la suma total de la banca sin pasarse de 21, recibirán un premio igual a lo apostado (relación 1 a 1), excepto para aquellos jugadores que consigan Blackjack (sumar 21) que recibirán un premio igual a 1,5 veces lo apostado (relación 3 a 2).

Simplificando el modelo

Aunque en una mesa real de blackjack se pueden sentar hasta 7 jugadores, en nuestro caso lo vamos a dejar en un jugador contra la banca (CPU) para simplificar el juego.

Se deja como ampliación, el implementar el resto de jugadores, que jugarán de forma automática. Puedes utilizar JavaFaker para ponerle nombres a los jugadores y darle más realismo.

Ejemplos de juego

Podéis echarle un vistazo algunas implementaciones online del juego, como por ejemplo:

- http://www.minijuegos.com/juego/blackjack
- http://showcase.codethislab.com/games/blackjack/
- http://html5blackjack.net/

Creando el juego

Los siguientes pasos son orientativos y no tienen porqué tomarse al pie de la letra, es el propio alumno el que deberá decidir si quiere seguirlos o prefiere implementar otra solución que le resulte más cómoda. El único requisito es que la funcionalidad del juego descrita se cumpla utilizando correctamente la programación orientada objetos.

1. Diseñar e implementar la clase **Carta**, y las estructuras asociadas necesarias, para representar una carta de la baraja francesa. Recuerda que este tipo de baraja tiene 52 cartas repartidas en 4 palos (tréboles, diamantes, corazones y espadas), lo que da un total de 13 cartas por palo (A,2,3,4,5,6,7,8,9,10,J,Q,K). Cada carta tiene un valor tal y como se ha comentado ya anteriormente. Resumiendo, de una carta necesitamos saber:
   a) El número
   b) El palo
   c) Su valor en el juego Blackjack
2. A partir de las estructuras de datos definidas en el apartado anterior, diseñar e implementar la clase **Baraja** que tendrá un único atributo “cartas” que será un Array de 52 instancias (13 cartas de cada palo) de la clase **Carta**.
3. Después de cada mano, necesitaremos un método que recoja las cartas repartidas y las mezcle/ baraje de forma sencilla y eficiente. Si bien podríamos trabajar directamente con la clase **Baraja**, existen mejores soluciones que simplifican el proceso de recoger las cartas y barajar, como por ejemplo el concepto de **Mazo**.

   Mientras que el concepto de Baraja es algo “estático a lo largo del tiempo” y las cartas no tienen una posición definida, es decir, una baraja francesa tiene 52 cartas y eso no cambia, el concepto de Mazo es algo dinámico que varía a lo largo del tiempo y la posición de las cartas

sí afecta ya que representa las cartas con las que se van a jugar cada mano, es decir, su posición determina el orden de aparición en el juego.

4. Implementa el siguiente **menú**:

   ****

   ***** BLACKJACK *****

   ****

   1. Nueva partida

   2. Mostrar estadísticas

   0. Salir

   - Al seleccionar **Nueva partida** deberá realizar las siguientes acciones:

     b) Reiniciar el saldo de la CPU (banca) al valor por defecto.

     b) Reiniciar el saldo del jugador al valor por defecto.

     c) Reiniciar la apuesta a la apuesta por defecto.

     d) Iniciar una nueva mano. La CPU barajará y repartirá cartas a los jugadores y empezará la lógica del juego:

     - Recoger las cartas, que simplemente será vaciar las cartas del jugador y las cartas de la CPU, y reiniciar el mazo.
     - Barajar las cartas del mazo.
     - Repartir cartas. 2 para el jugador y 2 para la CPU. Puedes hacer uso del método **Thread.sleep**(milisegundos) para añadir un pequeño retardo en el reparto de cartas dándole un efecto de animación. A partir de aquí se desencadena la lógica del juego representada mediante estados.

   - Al pulsar M**ostrar estadísticas** mostrará el número de manos ganadas por la CPU y el número de manos ganadas por el Jugador.

5. Llegados a este punto, dispones de las estructuras más básicas necesarias para crear nuestro juego pero probablemente necesites algunas más, como por ejemplo **Jugador** y una clase **Blackjack** que se encargue de gestionar toda lógica del juego. Es tarea del alumno identificar las clases restantes necesarias y los métodos y atributos que deberán tener cada una de ellas.

6. Para implementar la **lógica del juego del jugador** deberemos:

   a) Cada vez que el jugador pide carta debemos comprobar si la suma del valor total de las cartas es menor, mayor o igual que 21.

      - Si es menor, se le preguntará si desea otra carta o prefiere plantarse.
      - Si es igual, hay que tener en cuenta que, si el jugador ha obtenido 21 en las dos primeras cartas, directamente ha ganado. De lo contrario se pasa el turno a la CPU para ver si iguala.

- Si es mayor ha perdido.

7. Para implementar la **lógica del juego de la CPU** deberemos realizar las siguientes acciones:
   a) Si su puntuación inicial es 16 o menor debe pedir carta obligatoriamente.
   b) Si su puntuación es 17 o más deberá plantarse independientemente de las jugadas que tengan cada uno de los jugadores.
8. Al finalizar cada mano se deberán realizar las siguientes acciones:
   a) Cuando gane el jugador se deberán actualizar los saldos del jugador y de la CPU y mostrar el mensaje “Jugador gana!!!”, y en función del saldo que le quede a la CPU, finalizar la mano actual y empezar otra nueva o finalizar la partida si la CPU se ha quedado sin dinero o no tiene suficiente para afrontar la apuesta mínima de la siguiente mano.
   b) Cuando gane la CPU se deberán actualizar los saldos del jugador y de la CPU y mostrar el mensaje “CPU gana”, y en función del saldo que le quede al Jugador, finalizar la mano actual y empezar otra nueva o finalizar la partida si el Jugador se ha quedado sin dinero o no tiene suficiente para afrontar la apuesta mínima de la siguiente mano.

Ampliaciones:

- Implementar la visualización de las cartas mediante ASCII Art con colores (+ 1 punto)
- Añadir que el usuario tenga la opción de doblar apuesta después del reparto de las dos primeras cartas, siempre y cuando no se haya pasado. (+0,5 puntos)
- Implementar el juego Jugador VS CPU de forma automática, es decir, que el ordenador tome las decisiones de pedir carta o plantarse. Para ello se debe diseñar varias estrategias y en cada mano seleccionar la que se considere mejor en función de las cartas visibles de la CPU. (+ 1 punto)
- Implementar la opción multijugador, hasta 7 Jugadores (+1 punto)