***Black\_jack.py ------------------------ 1***

***Modulo\_crupier.py --------------------- 2***

***Modulo\_display.py -----------------3 , 4 , 5***

***Analisis requisitos:*** Los requisitos para este software son emular de manera fidedigna las reglas del black jack para poder jugar una partida con este software al igual que seria en un casino real.

Instalaciones pip necesarias: en cmd

pip install playsound

pip install pynput

***BLACK JACK***

main

Linea 1-37

Para empezar importamos **crupier** , esta clase sirve para crear objetos que representen las 'manos' de cartas , creamos con esta 2 'manos': la nuestra (player) y la de nuestro rival (dealer).

De este modulo importamos también la función **compara\_puntos()** que sirve para determinar quien gana según la pntuación y gestionar las fichas según apuesta y resultado de la partida.

Importamos **displays** , esta es la clase principal y contiene los displays de todas las elecciones del juego asi como los actos que conllevan estas elecciones, creamos con esta clase el objeto display

==> PRINCIPIO <== Linea 37-55

Utilizamos un bucle para poder volver a jugar otra ronda (se pregunta al final) , primero reseteamos la lista de cartas usadas y le damos a la carta 'as' un valor de 11 por defecto.

Luego (.reset\_hand) reseteamos las cartas de cada 'mano' por si venimos de rondas anteriores.

Finalmente repartimos dos cartas a cada manos según dictan las reglas de este juego.

==> SI TENEMOS 2 CARTAS CON EL MISMO NÚMERO <== Lineas 55-83

En el caso de tener 2 cartas con el mismo número según dictan las reglas del juego podemos elegir jugar con 2 manos, cada una de las 'iguales ' irán a una mano y se repartirá una nueva por mano.

Esta parte consta solo de 2 pasos, en el 1 mediante el 'if' comprobamos si las cartas tienen el mismo número y en caso de tenerlo mediante **display.manos\_bj()** mostramos un display que pregunte si queremos jugar con dos manos o seguir con una y nos devuelve la elección tomada 'hands'

A partir de aqui el programa se divide en 2 partes JUGAMOS CON 1 MANO // JUGAMOS CON 2 MANOS , según la elección tomada.

* En caso de jugar a 2 manos en linea 83-95 preparamos las manos según las reglas, una de las repetidas a cada mano y se reparten 2 nuevas
* Linea 96 -215 Hacemos un bucle que contenga las elecciones del juego a partir de aquí, pedir plantar o doblar. Dicho bucle solo se ejecutara una vez al jugar con una mano pero al jugar con 2, al final del bucle cambiara la mano 1 ya jugada por la mano 2 sin jugar , asi guardamos el resultado de esta ronda en mano 2 y podemos volver a ejecutar el bucle para la 2 mano utilizando las variables de mano 1.

***MODULO\_CRUPIER***

**Class crupier**

Esta clase representara la 'mano' de cada jugador mediante la variable lista de listas **self.\_hand**, constará de 2 métodos , **random\_card ()** al cual le pondremos como parametro un número, que sera la cantidad de cartas que nos 'repartiran' y una lista 'used' que utilizaremos para saber que cartas han sido utilizadas (en todas las 'manos') y que asi no vuelvan a repartirse

        Por otro lado tenemos **el método points ( )** al cual le introduciremos como parámetro el valor que hemos decidi-do para el 'as' durante el juego (Para que sepa como puntuarlo) y nos devolver  la puntuación de la mano (**self.\_hand**) según las reglas del Black Jack

        También estan los dos métodos cortos get\_hand y reset\_hand . El primero devuelve la lista de istas con las cartas 'nuestra mano' y el segudo resetea la mano para volver a empezar

        2 Metodos pequeños mas : add\_card( ) y delete\_card( ) , el primero añadira una carta (elegida por nosotros en lugar de random) y el segundo borrara una carta de nuestra 'mano'

def \_\_init\_\_ ( self ):

def random\_card (self,num\_cartas,used):

def points (self ):

def get\_hand (self):

def reset\_hand (self):

def add\_card ( self , card ):

def delete\_card (self):

**def compara\_puntos** (puntos\_player,puntos\_dealer,fichas,apuesta):

compara\_puntos es una función sin clase a la cual le introducimos como parámetros : los puntos de ambas manos , fichas y apuesta . Esta función nos devolverá un resultado según ganemos, perdamos o empatemos (cosa que dependerá de los puntos) y también nos devolverá las fichas , cuyo valor dependerá del resultado y de la apuesta realizada.

Esto lo hace de forma visual apoyandose en las pequeñas funciones de abajo que solo sirven dentro de compara\_puntos

\*

def resultado\_bj ():

def empatamos ():

def perdido ():

def ganado () :

def titulo\_mano ():

* ESTAS FUNCIONES SOLO PRINTEAN TITULOS ASCI

***MODULO DISPLAY***

**CLASS DISPLAYS:**

**INICIO**

**init ( )**

**key\_actions ( )**

**APUESTA**

**display\_bj\_apuesta ( )**

**actions\_apuesta ( )**

**2MANOS?**

**display\_2manos ( )**

**manos\_bj ( )**

**ELECCION**

**display\_bj\_options ( )**

**actions\_oprions ( )**

**AS??**

**display\_as ( )**

**as\_points ( )**

**CONTINUE?**

**display\_continue ( )**

**continue\_bj ( )**

**OTRAS**

**tablero ( )**

**print\_hand ( )**

==> Esta es **la estructura** de la clase 'displays' , es la clase principal del sotfware para el juego Black Jack y se encarga de mostrar y ejecutar todas las elecciones a lo largo del juego .

-En ***apuesta*** elegiremos la cantidad de fichas que apostamos.

-En ***2manos*** elegiremos (en caso de tener dos cartas de igual numero) si queremos jugar con dos manos , o seguir como siempre.

-En ***eleccion*** escogeremos entre (pedir,plantar,doblar o gestionar valor de 'as')

-En ***'as'*** escogeremos una vez dentro del menu que valor queremos (1 u 11) para nuestra carta.

-En ***continue*** elegimos si queremos jugar otra ronda o volver al menu de juegos.

- OTRAS: print\_hand nos printeara la lista de cartas (nuestra mano), se usara en tablero ( ) para mostrar nuestras cartas junto con las cartas del dealer y las puntuaciones correspondientes.

La usaremos donde sea necesario, coomo por ej en ELECCION asi ves las cartas y puntos que tienes para poder elegir mejor ( pedir,plantar...)

- (apuesta,2manos,eleccion,as?,continue?) funcionan las 5 igual . tienen 2 metodos un display que es la parte visual con una lista de listas que emula movimiento entre las casillas de opcciones

y otro metodo que es el principal que es el que organiza el display , el key logger y key\_actions (INICIO) asi conseguimos la parte visual con display, capturar la tecla con key loggeR y dependiendo de en que casilla estemos realizar distintas acciones con key\_actions.

**CLASS DISPLAYS:**

**INICIO:**

En esta parte (Inicio) pondremos el **método constructor** (Declara las variables) y el **método key\_actions** (lista,x). El primer parámetro (lista) corresponde a la lista que contiene las opciones.

Según el display en el que estemos y el segundo parámetro (x) es un número que diferenciará entre los distintos displays para que key\_actions realice distintas acciones al pulsar 'enter' según el display en el que nos encontremos.

Para emular el movimiento por la lista tendremos self.\_posx y self.\_posy que igualaremos a ' ◄ ' en la lista correspondiente para saber la casilla en la que nos encontramos.

De esta forma las opciones se elegirán mediante displays usando las direcciones y 'enter'.

**APUESTA:**

En esta parte (Apuestas) tendremos un método (**display\_bj\_apuesta**) que será la parte 'visual' , nos mostrará las distintas casillas con sus correspondientes opciones que seleccionaremos desplazando ' ◄ ' por las casillas y seleccionaremos con 'enter'.

Por último tenemos el método (**actions\_apuesta**) , este será el único que ejecutemos en el Main ya que gestionará los dos métodos anteriores yl el **key logger** (capta-teclas) y nos devolverA (apuesta,fichas) la apuesta realizada y las fichas restantes.

**2MANOS**

En esta parte (**2 manos** ) tendremos un método (**display\_2manos**) que será la parte 'visual' , nos mostrará las distintas casillas con sus correspondientes opciones que seleccionaremos desplazando ' ◄ ' por las casillas y seleccionaremos con 'enter'.

Por último tenemos el método (**manos\_bj**) , este será el único que ejecutemos en el Main ya que gestionará el método anterior y el **key logger** (capta-teclas) y nos devolverá (self.\_\_manos) que nos servirá para saber si doblamos mano o no.

**ELECCION**

En esta parte (eleccion pedir plantar... ) tendremos un método pequeño (**titulo\_puntos**) que printeará un titulo, tenemos otro método ( **display\_bj\_options** ) que que será la parte 'visual' , nos mostrará las distintas casillas con sus correspondientes opciones que seleccionaremos desplazando ' ◄ ' por las casillas y seleccionaremos con 'enter'.

Por último tenemos el método (**actions\_options**) , este será el único que ejecutemos en el Main ya que gestionará los dos métodos anteriores y el **key logger** (capta-teclas) y nos devolverá (self.\_resultado\_opciones) que será la opcción elegida según la cual luego efectuaremos distintas acciones (pedir,plantar etc..)

**AS??**

En esta parte ( GESTION AS\_CARD) tendremos un método (**display\_as**) que será la parte 'visual' , nos mostrará las distintas casillas con sus correspondientes opciones que seleccionaremos desplazando ' ◄ ' por las casillas y seleccionaremos con 'enter'.

Por último tenemos el método (**as\_points**) , este será el único que ejecutemos en el Main ya que gestionará el método anterior y el key logger (capta-teclas) y nos devolverá (self.\_\_valor\_as) que nos servirá para saber la puntuación de cada mano en función del valor elegido para nuestro 'as' .

**CONTINUE**

En esta parte ( CONTINUE ) tendremos un método (**display\_continue**) que será la parte 'visual' , nos mostrará las distintas casillas con sus correspondientes opciones que seleccionaremos desplazando ' ◄ ' por las casillas y seleccionaremos con 'enter'.

Por último tenemos el método (**continue\_bj**) , este será el único que ejecutemos en el Main ya que gestionará el método anterior y el **key logger** (capta-teclas) y nos devolverá (return self.\_continuar) de cuyo valor depende que volvamos a jugar una ronda o salgamos al menú de juegos , ya que este método se ejecuta en último lugar para resetear las variables necesarias