Autores: Pau Sanchez Valdivieso y Albert Espín Román

Documentación de la práctica 1

Ejercicio 1

Clase Time:

Representa un tiempo con hora, minuto y segundo, que son atributos de la clase.

Tiene como funciones setters y getters de los atributos (privados), así como :

* print\_time(self); muestra el tiempo en formato estándar, increment(self, seconds).
* increment(self, seconds); incrementa el tiempo unos dados segundos.
* time\_to\_int(self): convierte y retorna el tiempo a segundos.
* add\_time(self, t2): suma a un tiempo otro tiempo.
* def is\_after(self, other): indica si el tiempo del objeto propio es posterior a otro
* ej1(self): ejecuta el código del enunciado.

Explicación del código de la función ej1(), línea por línea:

def ej1():

*Se crea un objeto Time, t1, inicializando sus atributos mediante el constructor definido*

t1=Time(22,15,55)

*Muestra los atributos del objeto t1, mostrados separados por ":" como un tiempo*

t1.print\_time()

*Se crea un objeto Time, t2, inicializando sus atributos mediante el constructor definido*

t2=Time(1,22,34)

*Muestra los atributos del objeto t1, mostrado separado por ":" como un tiempo*

t2.print\_time()

*Muestra el tiempo t1 una vez se ha incrementado con el valor del tiempo t2*

t1.add\_time(t2).print\_time()

*Incrementa el valor de t1 en 1000 segundos, distribuidos según el convenio de segundos, minutos y horas*

t1.increment(1000)

*Muestra el valor de t1 una vez incrementado*

t1.print\_time()

Ejercicio 2

Clase Ej2\_FormulaUno:

Explicación de las funciones:

* estadisticaFormulaUno(lista,h,m,s): dado un tiempo y una lista de tiempos da cuantos tiempos de la lista son menores.
* ej2(h=1,m=40,s=0): muestra, dada una lista de pilotos que se crea dentro del ejercicio, cuantos tiempos de esos pilotos son menores que un tiempo pasado por parámetro (01:40:00), mediante la función estadisticaFormulaUno.

Ejercicio 3

Clase Card:

Esta clase implementa una carta de UNO, con los atributos privados color y number, que cuentan con sus propios getters; el constructor inicializa estos valores. Cuenta con una sobrecarga de \_\_str\_\_ para mostrar el color y el número de la carta objeto, además de una función check\_card para comparar dos cartas y determinar si su color o número coincide, así como una función test para pruebas.

Clase Player:

Esta clase implementa un jugador de UNO, con los atributos privados name y cards, la lista de sus cartas, valores que el constructor inicializa; se sobrecarga \_\_str\_\_ para mostrar los datos y \_\_len\_\_ para mostrar el número de cartas; cuenta con la función can\_play\_card para determinar si alguna de sus cartas es compatible con una pasada por parámetro, una función play\_a\_card para lanzar una carta a la pila y select\_card para seleccionarla, así como un test para comprobar el funcionamiento de la clase.

Clase Deck:

Esta clase implementa el mazo del juego UNO, y cuenta con dos veces todas las combinaciones posibles de cartas, creadas en el constructor para el atributo privado cards; se sobrecarga \_\_len\_\_ para devolver el número de cartas, \_\_getitem\_\_ para devolver una carta dado un índice; se borra una carta en una posición con la función remove, se reparte una carta al azar a un jugador con deal\_one\_card, se reparten N cartas a todos los jugadores con deal y se puede testear con test.

Clase Discard\_Pile:

Esta es la clase que representa la pila de cartas jugadas, recogidas en el atributo privado discard\_pile, con la particularidad que en el juego UNO interesa la última carta, visible, que se puede retornar con show\_last\_card; se pueden añadir cartas con el método append; se sobrecarga \_\_len\_\_ para devolver el número de cartas en la pila y se puede testear el objeto de la clase con test.