Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Primer Semestre de 2015

Guía Black Jack

Rodrigo Toro Icarte

Introducción

El objetivo de este documento es que te familiarices con los conceptos de Programación Orientada a Objetos. Para ello, tendrás que programar el juego conocido como **Black Jack**. Sin embargo, por tratarse de tu primer programa utilizando clases, te guiaré paso a paso con lo que debes hacer.

Reglas

El **Black Jack** es un juego de cartas inglesas muy común en casinos y bares de mala muerte. Se juega en mesas circulares donde un grupo de entre 1 y 7 personas juegan contra un *crupier* (representante del casino). La dinámica del juego es la siguiente:

- Todos los jugadores ingresan sus apuestas.
- El crupier reparte 2 cartas a cada jugador (visibles por todos).
- El crupier reparte 2 cartas para él, una visible y la otra dada vuelta (i.e. los jugadores solo pueden ver una de las dos cartas del crupier).
- En orden, cada jugador puede:
  - **Pedir una carta**: Mientras la mano del jugador sume menos de 21, podrá pedir una carta al crupier.
  - **Plantarse**: Decidir no pedir más cartas.
- Cuando todos los jugadores hayan pedido cartas, el crupier revelará su mano. Mientras su mano sea menor o igual a 16, pedirá cartas. Cuando supere los 16, se plantará (estas son ordenes del casino).
- Finalmente, se comparan la suma de las manos de cada jugador contra la mano del crupier. Gana quien tenga la mayor mano que sea menor o igual a 21.
  - Si gana el jugador, el casino le paga un monto igual a la apuesta del jugador. Sin embargo, si la mano del jugador suma 21 con sus dos cartas iniciales, tiene un *black jack*. En este caso, el casino le pagará 1,5 veces su apuesta inicial.
  - Si la mano del crupier es mayor, el casino se queda con la apuesta del jugador.
  - Si es un empate, no ocurre nada.

Para **sumar una mano** se suma el valor de cada una de las cartas en ella. La J, Q y K valen 10, mientras que el As vale 11 o 1, dependiendo de lo que sea más conveniente para la mano actual.

Casos especiales

Luego de que cada jugador recibe sus dos cartas, antes de pedir, puede realizar una de las siguientes jugadas::

- **Doblar**: Esto significa que el jugador dobla su apuesta sujeto a pedir una única carta extra.
- **Dividir**: Si el jugador recibe dos cartas del mismo valor, puede dividir su juego en dos manos independientes. Para esto debe asociar una apuesta idéntica a la original a su segundo juego.
- **Asegurar**: Si la carta visible del crupier es un As, se puede apostar a que el crupier tendrá un *black jack*. El valor apostado no puede superar la mitad de la apuesta original del jugador. Si el crupier efectivamente tiene black jack, se le pagará al jugador el doble de lo que apostó.
- **Abandonar**: Esto significa que el jugador decide retirarse del juego, perdiendo la mitad de lo apostado.

Tu programa

Debes programar un jugador de black jack. Al iniciar el juego, ingresa los jugadores con sus nombres y el dinero total que tienen. Luego permíteles participar, o no, en una nueva mesa de black jack. Cada mesa debe seguir las reglas ya discutidas del juego. Las decisiones de cada jugador son pedidas mediante `input()`, pero las decisiones del crupier deben ser automáticas (programadas por ti). Al terminar la mesa, se saldan las apuestas y se permite comenzar una nueva mesa. Solo pueden participar quienes aún tengan dinero para apostar.
Para resolver este problema sigue los siguiente pasos:

1. Define una clase `Carta`, que tenga 2 atributos: `pinta` y `valor` (ambos definidos en su constructor).
2. Para entrar en confianza, crea 3 cartas en tu programa principal: 3♠, A♡ y el K♢.
3. ¿Qué pasa si haces `print` de una carta?
4. Sobrecarga el método `__str__(self)` de `Carta`. Has que retorne un string con el valor y el número de la carta. ¿Qué pasa ahora si haces `print` de una carta?
5. Crea la clase `Mazo` con el atributo `mazo`, que es una lista de objetos de la clase `Carta`. En su constructor, agrega las cartas al mazo en forma ordenada (usando fors).
6. Agrega el método `mostrar(self)` a `Mazo`. En él recorre las cartas del mazo y muéstralas en consola.
7. Crea un nuevo mazo (en tu programa principal) y llama a `mostrar()` para chequear que todas las cartas se hayan agregado correctamente.
8. Utiliza la función `shuffle(l)` de la librería `random` para desordenar el mazo. Notar que `shuffle(l)` desordenará la lista `l` directo, por lo no retorna nada.
9. Vuelve a mostrar el mazo para chequear que las cartas fueron correctamente desordenadas.
10. Define la clase `Jugador` con 4 atributos: `nombre`, `dinero`, `mano` (lista vacía) y `apuesta` (con valor de 0).
11. Agrega a `Jugador` el método `asignar_apuesta(self,d)`, que asigna `d` a la apuesta del jugador.
12. Agrega a `Jugador` el método `recibir_carta(self,c)`, que agrega la carta `c` a la mano del jugador.
13. En tu código principal crea un nuevo jugador con tu nombre y el dinero que llevas en tus bolsillos.
14. Agrega a tu jugador las siguientes cartas (mediante recibir_carta(c)): 3♠, A♡ y el K♢.
15. Agrega el método `contar_mano(self)` a `Jugador`, que retorna el valor de la mano actual del jugador. Para ello, podrías incluir a `Carta` el método `get_valor(self)` tal que retorna el valor de la carta (llevando a 10 los `J`, `Q` y `K`), y retornando `"A"` para los ases). Notar que el valor del *Ases* es 1 u 11 según de la mano.
16. Imprime el resultado de Llamar a `contar_mano()` con el jugador que habías creado (debería imprimir 14).
17. Sobreescribe `__str__(self)` de `Jugador` para que retorne un string con su nombre, apuesta, mano actual y el valor de su mano.
18. En tu programa principal imprime al jugador y chequea que se muestre correctamente.
19. Agrega a la clase `Mazo` el método `dar_carta(self, jugador, num)`, que quita del mazo las primeras `num` cartas y las agrega a la mano de jugador (mediante `recibir_carta(c)`).
20. En tu código principal define la función `agregar_usuarios()`, que retorna una lista con los usuarios que van a jugar (sus datos se agregan mediante `input`).
21. Implementa la función `jugar(usuarios, mazo)`, que permite jugar una mesa de black jack a la lista de usuarios. Aquí deberás pedir las apuestas, permitir que cada jugador juegue (incluyendo al crupier), y saldar las apuestas al finalizar el juego. Por ahora, no consideres los *casos especiales* (doblar, dividir, asegurar y abandonar), ni el caso en que el mazo se termina.
22. Permite que los usuarios sigan jugando hasta que queden sin dinero, o decidan retirarse.
23. **[Opcional]** Agrega los casos especiales y maneja el caso en que el mazo se acaba (debes crear un nuevo mazo que no contenga las cartas en juego).