JUEGO SIETE Y MEDIA

Antonio Fco Sánchez Caparrós

José Miguel Toro Canillas

♣ ♠ ♥ ♦ Carta:

Atributos

- palos: array que establece cuales son los valores que puede tomar un palo de la carta
- numeros: son los números que puede tener la carta

- palo: más adelante le daremos alguno de los valores del array palos
- valor: al igual que palo, más adelante se le establecerá un valor del array numeros
- puntos: es la puntuación que tiene cada carta

Constructor

Se le asigna un valor aleatorio del array numeros al String numero y un valor aleatorio de palos al String palo.

Con el switch podemos evaluar qué carta es, y darle un valor al double puntos.

Getters

Más adelante nos harán falta los getters de valor, palo y puntos, con lo que son los que establecemos.

toString

Si queremos imprimir la carta necesitamos un toString, además, en este le damos formato a la salida, que varía su color dependiendo del palo.

equals

Para poder comparar la carta con otras necesitamos este método, con lo que también lo establecemos.

🃏 Baraja

Atributos

- cartas: un ArrayList del objeto Carta.
- cartasUsadas: un entero que indica cuántas cartas se han sumado a la baraja.

Métodos

barajarBaraja() {

Este método genera una carta (aleatoria, por definición de su constructor), y la añade al ArrayList cartas, una vez añadida genera 39 más y las añade a cartas, pero no sin antes comparar si la carta ya está añadida a la baraja ya, mediante el método contains (por esto necesitábamos en la clase Carta el método equals).

}

repartirUnaCarta() {

Este método es del tipo Carta, con lo que devuelve una carta. Lo primero que hace es sumar 1 al índice de cartas que tenemos, para así devolver la carta siguiente a la anterior, y después devuelve la carta que se encuentre en la posición que indique este índice.

}

NOTA

Esta clase no tiene constructor, ya que el propio método de barajarBaraja conseguimos lo que queremos, que es crear una serie de 40 cartas desordenadas, ya que en ningún momento se usan de manera ordenada.

🖐️ Mano

Atributos

- puntuacionJugador: va acomulando la puntuación que el jugador tiene a lo largo de la ronda.
- mano: un ArrayList del objeto Carta, con el que vamos añadiendo cartas a la mano del jugador.

Getters

En este caso solo necesitaremos el getter de la puntuación, para usarla tanto en la clase Jugador como en la App principal.

Métodos

recibirCarta(Carta carta) {

Este método añade una carta (por ahora objeto no definido) al ArrayList que supone la mano del jugador, además de añadir los puntos de esta misma a la puntuación del jugador.

}

mostrarMano() {

Repite un bucle tantas veces como la larga sea la mano del jugador. En este bucle imprime la carta, y cuando sale de este, imprime la puntuación actual del jugador.

}

NOTA

En esta clase no hay toString, ya que el método mostrarMano ya está prácticamente haciendo esta función.

🧑🏾 Jugador

Atributos

- monedero: este double establece cuál es el dinero con el cual el jugador cuenta.
- apuesta: indica cuál es la cantidad que el jugador está dispuesto a apostar.
- mano: del tipo Mano, para usar sus métodos

Constructores

Al primero se le pide el monedero del jugador, lo usaremos para construir al jugador, el segundo no pide nada, ya que de la banca no nos interesa su monedero.

Getters

Solo necesitamos el getter del monedero para usarlo en la App principal.

Setters

También necesitamos el setter del monedero, para cambiarlo a lo largo de la partida.

Métodos

apostar(double apuesta) {

> Este método es un booleano que devuelve verdadero si la apuesta es válida, es decir, si la cantidad a apostar es menor o igual que el monedero con el que el jugador cuenta.

}

recibirCartaJugador(Carta carta) {

Básicamente usa el método de su mano para recibir una carta.

}

puntuacionJugador() {

Es casi que un getter, pero lo llamo método porque usa otro getter (el del objeto Mano), para no llevar a confusión con los nombres.

}

mostrarManoJugador() {

Tan solo hago uso del método de su objeto Mano para mostrar la mano.

}

📱 AppJuego

- jugador: es tipo Jugador, y su constructor es con el monedero incluido.
- banca: otro Jugador pero sin monedero, ya que no nos interesa.
- seguirJugando: es un booleano que a lo largo de TODO el código nos va a ir indicando si seguimos jugando o si se acaba la partida o por lo menos una parte de esta.
- primeraVez: otro booleano, este indica si es la primera vez que el jugador apuesta, para preguntarle cuánto dinero TIENE para apostar, frente a cuánto dinero QUIERE apostar.
- monederoJugador: es parecido al getter del monedero de Jugador, pero este lo usamos para hacer los cálculos más sencillos.

Con esto ya podemos empezar a jugar. El juego inicia con un mensaje de bienvenida y preguntando al jugador cuánto dinero TIENE para apostar (**línea 12**), y entonces entra en un bucle while, que se repetirá siempre que nuestra condición de seguirJugando sea verdadera (**línea 18**).

Obviamente vamos a entrar por primera vez ya que seguirJugando lo hemos iniciado como verdadero.

Una vez dentro establecemos el double cantidadApuesta, más adelante la usaremos. Evaluamos si es la primera vez que el jugador apuesta o no, en caso de serlo, se pide por teclado la cantidad que tiene para apostar y se introduce en el setter del monedero (**línea 24**), además de convertir el booleano primeraVez como falso.

En caso de no ser la primera vez que se juega, se settea el monedero a la cantidad acomulada en la ronda anterior.

De nuevo en la **línea 38** se evalúa si se quiere o no seguir jugando (o si se puede o no), y se crea y se baraja entonces la baraja. Cambiamos el valor de seguirJugando a false, para poder evaluarlo en el siguiente bucle do while.

En la **línea 48** empieza el bucle do while que evalúa si la cantidad apostada por el jugador es válida para ser apostada, es decir, si es mayor a su monedero o

no. En caso de no serlo, vuelve a preguntar la cantidad a apostar, en caso de serlo, se cambiará el estado de seguirJugando para salir de este do while (**línea 66**).

Para repartir cartas al jugador debemos hacer un bucle do while, que una vez más exprime la usabilidad del booleano seguirJugando (**línea 74 a línea 138**).

En este, repartimos una carta al jugador, y también, mostramos la mano del jugador. Si la puntuación del jugador (**línea 84**), es mayor que 7.5, seguir jugando será false, ya que habremos perdido, se imprime un mensaje que indica que la puntuación es mayor y se sale del bucle con el break.

En caso de ser la puntuación igual a 7.5, la lógica es igual pero el mensaje es distinto, salimos porque hemos ganado, no porque hemos perdido.

En el caso en el que no pase ni una ni otra, es decir, en el caso en el que la puntuación sea todavía menor a 7.5, le preguntaremos al jugador que quiere hacer, si seguir jugando o si plantarse.

Esto lo hacemos con el switch (**línea 123 a línea 138**), si decide seguir jugando, esta variable será verdadera, si decide plantarse, la variable será falsa y dará la puntuación por pantalla. Además lo pide siempre que la decisión no sea 1 o 2, de lo contrario, pide que lo introduzca de nuevo.

Ahora, evalúa si la banca tiene que jugar o no; si la puntuación del jugador es menor a 7.5, la banca juega, sino, ni siquiera juega (**línea 142**)

En caso de jugar, es muy parecido al jugador, pero sin hacer salidas por pantalla, solo cuando acabe.

Si la banca tiene menos que un 7.5, y además es menor o igual que la puntuación del jugador, seguirá jugando, en caso contrario, deja de jugar (**línea 144 a línea 169**).

Entonces, imprime la puntuación de la banca (**línea 172**).

Es ahora cuando evalúa realmente quién ha ganado. Esto se evalúa comparando la distancia a la que está la puntuación del jugador y de la banca del 7.5 (**línea 178 a línea 206**).

En caso de ganar, se le suma la cantidad apostada al monedero del jugador, en caso contrario, se le resta.

Imprime el monedero del jugador, y si este es mayor a 0, se le pregunta si quiere volver a apostar o si quiere retirar su dinero, de nuevo en un bucle para comprobar que la opción sea alguna válida.

En caso de que el monedero sea 0, habrás acabado el juego, y se te recordará que CUANDO APUESTAS SIEMPRE PIERDES.