```java

/**
 * Esta clase modela un jugador de UNO. Cada jugador dispone de un nombre y
 * de una mano de cartas capaz de almacenar un nÃºmero mÃ¡ximo de cartas.
 *
 *
 *
 */
public class JugadorDeUNO {

    private final  ManoDeUno manoJugador;
    private final String nombreJugador;


    /**
     * Constructor de la clase. Recibe un nombre y un nÃºmero mÃ¡ximo de cartas
     * que puede albergar su mano.
     *
     * @param nombre Nombre del jugador. Ejemplo: "Jugador 1".
     * @param nÃºmeroMÃ¡ximoDeCartas Nombre del jugador. Ejemplo: "Jugador 1".
     */

    public JugadorDeUNO(String nombre,int nÃºmeroMÃ¡ximoDeCartas) {

        manoJugador = new ManoDeUno (nÃºmeroMÃ¡ximoDeCartas);
        nombreJugador = nombre;
    }


    /**
     * Devuelve el nombre del jugador.
     * @return Nombre del jugador.
     */

    public String getNombre() {

        return nombreJugador;
    }

    /**
     * Indica si la mano de este jugador se ha quedado sin cartas.
     * @return true Si la mano de este jugador se ha quedado sin cartas.
     */

    public boolean sinCartasEnLaMano() {

        boolean noTieneCartas = false;

        if(manoJugador.estÃ¡VacÃ-a()) {
            noTieneCartas = true;
        }

        return noTieneCartas;
    }

    /**
     * Este mÃ©todo permite a este jugador extraer de la pila de cartas para coger,
  recibida como argumento, hasta un maximo de 'numeroDeCartasACoger' cartas que el juego le
  indica para aÃ±adirlas a su mano.
     * El jugador cogerÃ¡ tantas cartas como haya disponibles en la pila hasta un mÃ¡ximo
  de 'numeroDeCartasACoger' cartas o hasta que su mano estÃ© llena.
     *
```

```
60      * @param cartasParaCoger Pila de cartas de la que extraer cartas para la mano.
61      * @param numeroDeCartasACoger NÃºmero mÃ¡ximo de cartas que puede intentar extraer
  este jugador (si estÃ¡n disponibles en la pila de cartas).
62      */
63     public void cogeCartasâ€‹(PilaDeCartasDeUNO cartasParaCoger, int numeroDeCartasACoger)
  {
64
65         for(int i=0; i<numeroDeCartasACoger; i++) {
66             if(!manoJugador.estÃ¡Llena()  && cartasParaCoger.hayCartasDisponibles()){
67
68                 manoJugador.agregarCartaâ€‹(cartasParaCoger.extraerCartaParteSuperior());
69             }
70         }
71     }
72
73     /**
74      * Este mÃ©todo contiene las acciones que realiza este jugador cuando le toca su turno.

75      *
76      *      1.Saca una carta jugable/apilable de su mano para poner sobre la pila de cartas
  tiradas.
77      *      2.Si no dispone de carta jugable/apilable y la mano no estÃ¡ llena, extrae una
  carta de la pila de cartas para coger, la aÃ±ade a la mano e intenta de nuevo sacar una
  carta jugable/apilable de su mano para poner sobre la pila de cartas tiradas.
78      *
79      * @param cartasParaCoger Pila de cartas que sirve a este jugador para coger, si lo
  necesita, una nueva carta para su mano.
80      * @param cartasTiradas Pila de cartas sobre la que se debe apilar la carta
  jugable/apilable de la mano, si esta existe.
81      */
82     public void juegaâ€‹(PilaDeCartasDeUNO cartasParaCoger, PilaDeCartasDeUNO
  cartasTiradas) {
83
84         System.out.println("  Hay que sacar carta para un:
  "+cartasTiradas.verCartaParteSuperior().getIdentificador());
85         System.out.println("  Mi mano es: "+manoJugador.getMano());
86
87         CartaDeUNO CartaEnJuego = manoJugador.extraerCartaApilableSobreâ
  €‹(cartasTiradas.verCartaParteSuperior());
88
89
90         if(CartaEnJuego == null && !manoJugador.estÃ¡Llena()) {
91
92             System.out.print("  No tengo carta vÃ¡lida en la mano , cojo otra ...");
93
94             manoJugador.agregarCartaâ€‹(cartasParaCoger.extraerCartaParteSuperior());
95
96             CartaEnJuego = manoJugador.extraerCartaApilableSobreâ
  €‹(cartasTiradas.verCartaParteSuperior());
97
98             if(CartaEnJuego == null) {
99                 System.out.println("    Â¡Tampoco puedo jugar tras coger una carta!");
100
101            }else {
102                cartasTiradas.agregarCartaâ€‹(CartaEnJuego);
103                System.out.println("     Tengo carta vÃ¡lida en la mano
  "+CartaEnJuego.getIdentificador());
104            }
105        }
106
107         else {
108                cartasTiradas.agregarCartaâ€‹(CartaEnJuego);
```

```
109                 System.out.println("  Tengo carta valida en la mano
    "+CartaEnJuego.getIdentificador());
110             }
111 
112     }
113 }
114 
115 
116 
```