```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.
     *
     * @param cartasParaCoger Pila de cartas de la que extraer cartas para la mano.
```

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

74      *
75      *      1.Saca una carta jugable/apilable de su mano para poner sobre la pila de cartas
  tiradas.
76      *      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.
77      *
78      * @param cartasParaCoger Pila de cartas que sirve a este jugador para coger, si lo
  necesita, una nueva carta para su mano.
79      * @param cartasTiradas Pila de cartas sobre la que se debe apilar la carta
  jugable/apilable de la mano, si esta existe.
80      */
81     public void juegaâ€‹(PilaDeCartasDeUNO cartasParaCoger, PilaDeCartasDeUNO
  cartasTiradas) {
82
83         System.out.println("  Hay que sacar carta para un:
  "+cartasTiradas.verCartaParteSuperior().getIdentificador());
84         System.out.println("  Mi mano es: "+manoJugador.getMano());
85
86         CartaDeUNO CartaEnJuego = manoJugador.extraerCartaApilableSobreâ
  €‹(cartasTiradas.verCartaParteSuperior());
87
88
89         if(CartaEnJuego == null && !manoJugador.estÃ¡Llena()) {
90
91             System.out.print("  No tengo carta vÃ¡lida en la mano , cojo otra ...");
92
93             manoJugador.agregarCartaâ€‹(cartasParaCoger.extraerCartaParteSuperior());
94
95             CartaEnJuego = manoJugador.extraerCartaApilableSobreâ
  €‹(cartasTiradas.verCartaParteSuperior());
96
97             if(CartaEnJuego == null) {
98                 System.out.println("    Â¡Tampoco puedo jugar tras coger una carta!");
99
100            }else {
101                cartasTiradas.agregarCartaâ€‹(CartaEnJuego);
102                System.out.println("     Tengo carta vÃ¡lida en la mano
  "+CartaEnJuego.getIdentificador());
103            }
104        }
105
106         else {
107                cartasTiradas.agregarCartaâ€‹(CartaEnJuego);
108                System.out.println("  Tengo carta valida en la mano
```

```
   "+CartaEnJuego.getIdentificador());
109             }
110
111     }
112 }
113
114
```