```java
package juego;

import entorno.Entorno;

public class Juego extends InterfaceJuego {
    // El objeto Entorno que controla el tiempo y otros
    private Entorno entorno;
    boolean juegoPerdido = false;
    boolean juegoGanado = false;

    // Variables y métodos propios de cada grupo

    // Creación del arreglo de vigas
    static Viga suelos[] = new Viga[] {

            new Viga(1), new Viga(2), new Viga(3), new Viga(4), new Viga(5), new Viga(6)

    };

    static Escaleras escaleras[] = new Escaleras[] { new Escaleras(0, suelos), new Escaleras(1, suelos),
            new Escaleras(2, suelos), new Escaleras(3, suelos), new Escaleras(4, suelos) };

    Donkey donkeyKong = new Donkey();
    Personaje jugador = new Personaje(suelos[0]);
    Mensajes terminal = new Mensajes();
    Puntaje puntuador = new Puntaje();

    int contador = 0;

    Barril barriles[] = new Barril[]

    { new Barril(suelos[suelos.length - 3]), null, null, null, null, null, null, null, null, null, null, null, null,
            null, null, null, null, null, null, null, null

    };

    // ...

    Juego() {
        // Inicializa el objeto entorno
        this.entorno = new Entorno(this, "Donkey - Grupo Pereira - Sanchez - Tula - V1", 800, 600);

        // Inicializar lo que haga falta para el juego
        // ...

        // Inicia el juego!
        this.entorno.iniciar();

    }

    /**
     * Durante el juego, el método tick() será ejecutado en cada instante y por lo
     * tanto es el método más importante de esta clase. Aquí se debe actualizar el
     * estado interno del juego para simular el paso del tiempo (ver el enunciado
     * del TP para mayor detalle).
     */
    public void tick() {
        // Procesamiento de un instante de tiempo
        // ...
```

```
62
63          // Ejecuta la función dibujar por cada miembro del arreglo de vigas.
64          for (int i = 0; i < suelos.length; i++) {
65              suelos[i].dibujar(entorno);
66          }
67
68          // Ejecuta la función dibujar por cada miembro del arreglo de escaleras.
69          for (int i = 0; i < escaleras.length; i++) {
70              escaleras[i].dibujar(entorno);
71          }
72
73          // Ejecuta la función dibujar para donkey
74          donkeyKong.gorilear(entorno, contador);
75
76          // Ejecuta la función dibujar para el jugador
77          jugador.dibujar(entorno, contador, escaleras);
78
79          puntuador.dibujar(entorno);
80
81          // Ejecuta la función que analiza si el jugador está EN una escalera.
82          if (jugador.estaEnEscalera() == false) {
83              jugador.estoyCercaDeEscalera(entorno, escaleras, suelos);
84          }
85
86          // Ejecuta la función que analiza si el jugador está EN el medio de un salto o
87          // caida.
88          jugador.saltando(entorno, contador, suelos);
89
90          // Contador de tiempo, medido en ticks
91          contador = contador + 1;
92
93          // Si donkey decidió arrojar, se crea un nuevo barril en la primera posición
94          // NULL del arreglo de barriles.
95          if (donkeyKong.decidir(contador)) {
96              int creados = 0;
97
98              for (int i = 0; i < barriles.length && creados == 0; i++) {
99                  if (barriles[i] == null) {
100                     barriles[i] = new Barril(suelos[suelos.length +
    donkeyKong.arribaOabajo()]);
101                     creados = 1;
102                 }
103             }
104         }
105
106         // Ejecuta la función dibujar por cada elemento no NULL del arreglo de barriles
107         // , también analiza si un barril debe destruirse.
108
109         for (int i = 0; i < barriles.length; i++) {
110             if (barriles[i] != null) {
111
112                 barriles[i].dibujar(entorno, contador, suelos);
113                 if (barriles[i].deboDestruirme(entorno, suelos)) {
114                     barriles[i] = null;
115                 }
116
117             }
118         }
119
120         // Es la función que indica si el jugador está tocando algún barril y por lo
121         // tanto game over si es verdadero.
122
```

```
123         if (jugador.tocando(barriles) && juegoPerdido == false) {
124             juegoPerdido = true;
125         }
126 
127         for (int i = 0; i < barriles.length; i++) {
128             if (barriles[i] != null) {
129 
130                 if (jugador.saltandoBarril(barriles[i])) {
131                     puntuador.saltarbarril();
132                 }
133 
134             }
135         }
136 
137         if (juegoPerdido) {
138             terminal.dibujar("perder", entorno);
139             jugador.morir();
140             donkeyKong.noMasViolencia();
141         }
142 
143         if (jugador.ganar(entorno, suelos) && juegoGanado == false) {
144             juegoGanado = true;
145             puntuador.ganar();
146         }
147 
148         if (juegoGanado) {
149             terminal.dibujar("ganar", entorno);
150             donkeyKong.noMasViolencia();
151         }
152 
153     }
154 
155                     ("unused")
156     public static void main(String[] args) {
157         Juego juego = new Juego();
158     }
159 }
160 
```