

```

1 package Modelo;
2
3 import Interfaz.Escritor;
4 import static java.lang.Thread.sleep;
5 import java.util.ArrayList;
6 import java.util.concurrent.ArrayBlockingQueue;
7 import java.util.concurrent.BrokenBarrierException;
8 import java.util.concurrent.CyclicBarrier;
9 import java.util.logging.Level;
10 import java.util.logging.Logger;
11
12 /**
13  *
14  * @author marco
15  */
16 public class Soga {
17     private int tamEquipo;
18     private Escritor escritor;
19     private Paso paso;
20     private ArrayBlockingQueue<Ninno> colaSoga;
21     private ArrayBlockingQueue<Ninno> colaSogaEquipoA;
22     private ArrayBlockingQueue<Ninno> colaSogaEquipoB;
23     private ArrayList<Monitor> monSoga = new ArrayList<>();
24     private CyclicBarrier barreraSoga;
25
26     public Soga(int p_tamEquipo, Escritor p_escritor, Paso p_paso){
27         tamEquipo = p_tamEquipo;
28         colaSoga = new ArrayBlockingQueue(tamEquipo*2);
29         colaSogaEquipoA = new ArrayBlockingQueue(tamEquipo);
30         colaSogaEquipoB = new ArrayBlockingQueue(tamEquipo);
31         barreraSoga = new CyclicBarrier(1+tamEquipo*2);
32         escritor = p_escritor;
33         paso = p_paso;
34     }
35
36     public void sogamirar(Ninno ninno){
37         paso.mirar();
38         if (colaSoga.size()+colaSogaEquipoA.size()+colaSogaEquipoB.size()<tamEquipo*2){
39             try {
40                 colaSoga.put(ninno);
41             } catch (InterruptedException ex) {
42                 Logger.getLogger(Soga.class.getName()).log(Level.SEVERE, null, ex);
43             }
44             escritor.addMsg(ninno.getMtId() + " se pone a la cola de SOGA");
45             paso.mirar();
46             try {
47                 barreraSoga.await();
48             } catch (InterruptedException ex) {
49                 Logger.getLogger(Soga.class.getName()).log(Level.SEVERE, null, ex);
50             } catch (BrokenBarrierException ex) {
51                 Logger.getLogger(Soga.class.getName()).log(Level.SEVERE, null, ex);
52             }
53         }
54     }
55
56     public void sogamirar(Monitor mon) {
57         paso.mirar();
58         monSoga.add(mon);
59         escritor.addMsg(mon.getMtId() + " llega a la SOGA");
60         paso.mirar();
61         while (mon.getContadorActividades() > 0) {
62             for (int i = 0; i < tamEquipo * 2; i++) {
63                 paso.mirar();
64                 Ninno n;
65                 try {
66                     n = colaSoga.take();
67                     if (Math.random() < 0.5) {
68                         if (!colaSogaEquipoA.offer(n)) {
69                             colaSogaEquipoB.put(n);
70                         }
71                     } else {
72                         if (!colaSogaEquipoB.offer(n)) {
73                             colaSogaEquipoA.put(n);
74                         }
75                     }
76                 }

```

```

75     }
76     } catch (InterruptedException ex) {
77         Logger.getLogger(Soga.class.getName()).log(Level.SEVERE, null, ex);
78     }
79 }
80 escritor.addMsg(mon.getMiId() + " inicia el enfrentamiento en SOGA");
81 paso.mirar();
82 try {
83     sleep(7000);
84 } catch (InterruptedException ex) {
85     Logger.getLogger(Soga.class.getName()).log(Level.SEVERE, null, ex);
86 }
87 paso.mirar();
88 if (Math.random() < 0.5) {
89     escritor.addMsg(mon.getMiId() + " termina el enfrentamiento en SOGA a favor del EQUIPO A");
90     for (Ninno ninno : colaSogaEquipoA) {
91         ninno.substractActividad(2);
92         colaSogaEquipoA.remove(ninno);
93         escritor.addMsg(ninno.getMiId() + " gana en SOGA");
94     }
95     for (Ninno ninno : colaSogaEquipoB) {
96         ninno.substractActividad(1);
97         colaSogaEquipoB.remove(ninno);
98         escritor.addMsg(ninno.getMiId() + " pierde en SOGA");
99     }
100 } else {
101     escritor.addMsg(mon.getMiId() + " termina el enfrentamiento en SOGA a favor del EQUIPO B");
102     for (Ninno ninno : colaSogaEquipoA) {
103         ninno.substractActividad(1);
104         colaSogaEquipoA.remove(ninno);
105         escritor.addMsg(ninno.getMiId() + " pierde en SOGA");
106     }
107     for (Ninno ninno : colaSogaEquipoB) {
108         ninno.substractActividad(2);
109         colaSogaEquipoB.remove(ninno);
110         escritor.addMsg(ninno.getMiId() + " gana en SOGA");
111     }
112 }
113 paso.mirar();
114 try {
115     barreraSoga.await();
116 } catch (InterruptedException ex) {
117     Logger.getLogger(Soga.class.getName()).log(Level.SEVERE, null, ex);
118 } catch (BrokenBarrierException ex) {
119     Logger.getLogger(Soga.class.getName()).log(Level.SEVERE, null, ex);
120 }
121 mon.substractActividad();
122 }
123 monSoga.remove(mon);
124 paso.mirar();
125 }
126
127 public String getCola(){
128     String msg = "";
129     for (Ninno ninno:colaSoga){
130         msg += ninno.getMiId() + " ";
131     }
132     return msg;
133 }
134 public int getTamCola(){
135     return colaSoga.size();
136 }
137 public int cuantosNinnosCola(){
138     return colaSoga.size();
139 }
140
141 public String getColaEquipoA(){
142     String msg = "";
143     for (Ninno ninno:colaSogaEquipoA){
144         msg += ninno.getMiId() + " ";
145     }
146     return msg;
147 }
148
149 public String getColaEquipoB(){

```

```
150     String msg = "";
151     for (Ninno ninno:colaSogaEquipoB){
152         msg += ninno.getMiId() + " ";
153     }
154     return msg;
155 }
156
157 public String getMon(){
158     String msg = "";
159     for (Monitor mon:monSoga){
160         msg += mon.getMiId() + " ";
161     }
162     return msg;
163 }
164 }
```