```
***COMPONENTES DEL GRUPO***
Joaquín Gálvez Díaz (Portavoz)
Jorge Urbelz Alonso-Cortés


***MONITOR MAQUINA***
l: ReentrantLock privado
colaMaquina: Condition privado
maquinas[]: array de enteros privado

l <- ReentrantLock(true);
colaMaquina <- l.newCondition();
maquinas[] <- entero[3];

Constructor MonitorMaquina()
begin
      for i in 0 … maquinas.length hacer
            maquinas[i] <- -1;
      fin_for
end

privado indexOf(n: entero): entero
begin
      index: entero
      index <- -1;
      for i in 0 … maquinas.length hacer
            si maquinas[i] == n entonces
                  index <- i;
                  devolver index;
            fin_si
      fin_for
      devolver index;
end
```

```
34 publico solicitarMaquina(id: entero, tiempoMaquina: entero,
35 tiempoMesa: entero, monitorMesa: MonitorMesa): entero provoca
36 InterruptedException
37 begin
38       l.bloquear();
39       mesaAsignada, numeroMesa, numeroMaquina, maquina: int
40       mesaAsignada <- -1;
41       intentar
42             maquina <- indexOf(id);
43             si maquina >= 0 hacer
44                   mesaAsignada <-
45 monitorMesa.mesaMenorTiempoEspera();
46                   numeroMaquina <- maquina + 1;
47                   imprimir("Cliente " + id + " ha solicitado su
48 servicio en la máquina: " + numeroMaquina + "\n");
49                   imprimir("Tiempo en solicitar el servicio: " +
50 tiempoMaquina + "\n");
51                   numeroMesa <- mesaAsignada + 1;
52                   imprimir("Será atendido en la mesa: " +
53 numeroMesa + "\n");
54                   imprimir("Tiempo en la mesa: " + tiempoMesa);
55                   numeroMesa <- 1;
56                   imprimir("Tiempo de espera en la mesa" +
57 numeroMesa + " = " + monitorMesa.getTiempoColaMesa()[0]);
58                   numeroMesa <- numeroMesa + 1;
59                   for i in 1 … hacer
60 monitorMesa.getTiempoColaMesa().length hacer
61                         imprimir(", mesa" + numeroMesa + " = " +
62 monitorMesa.getTiempoColaMesa()[i]);
63                         numeroMesa <- numeroMesa + 1;
64                   fin_for
65                   imprimir("\n\n");
66       monitorMesa.introducirClienteColaMesa(mesaAsignada,
67 tiempoMesa);
68                   maquinas[maquina] <- -1;
69                   colaMaquina.signal();
```

```
70
71              fin_si
72              sino
73                    mientras(indexOf(-1) == -1) hacer
74                          colaMaquina.await();
75                    fin_mientras
76                    maquinas[indexOf(-1)] <- id;
77              fin_sino
78        finalmente
79              l.desbloquear();
80        devolver mesaAsignada;
81  end
82
83
84  ***MONITOR MESA***
85  l: ReentrantLock privado
86  colaMesa[]: array de Condition
87  tiempoColaMesa[], clienteColaMesa[]: array de enteros
88
89  l <- ReentrantLock(true);
90  colaMesa[] <- Condition[4];
91  tiempoColaMesa[] <- entero[4];
92  clienteColaMesa[] <- entero[4];
93
94  Constructor MonitorMesa()
95  begin
96        for i in 0 … colaMesa.length hacer
97              tiempoColaMesa[i] <- 0;
98              clienteColaMesa[i] <- -1;
99              colaMesa[i] <- l.newCondition();
100       fin_for
101 end
102
```

```
103  publico getTiempoColaMesa(): entero[]
104  begin
105        l.bloquear();
106        intentar
107              devolver Arrays.copyOf(tiempoColaMesa,
108  tiempoColaMesa.length);
109        finalmente
110              l.desbloquear;
111  end
112
113  public introducirClienteColaMesa(mesaAsignada, tiempoMesa:
114  entero)
115  begin
116        l.bloquear();
117        intentar
118              tiempoColaMesa[mesaAsignada] <-
119  tiempoColaMesa[mesaAsignada] + tiempoMesa;
120        finalmente
121              l.desbloquear();
122  end
123
124  publico mesaMenorTiempoEspera(): entero
125  begin
126        l.bloquear();
127        tiempoActual, mesaActual: entero;
128        tiempoActual <- tiempoColaMesa[0];
129        mesaActual <- 0;
130        intentar
131              for i in 1 … tiempoColaMesa.length hacer
132                    si tiempoColaMesa[i] < tiempoActual entonces
133                          tiempoActual <- tiempoColaMesa[i];
134                          mesaActual <- i;
135                    fin_si
136        finalmente
```

```
137             l.desbloquear();
138       devolver mesaActual;
139 end
140
141 publico solicitarMesa(id, mesaAsignada, tiempoMesa: entero)
142 provoca InterruptedException
143 begin
144       l.bloquear();
145       intentar
146             si clienteColaMesa[mesaAsignada] == id entonces
147                   clienteColaMesa[mesaAsignada] <- -1;
148                   tiempoColaMesa[mesaAsignada] <-
149 tiempoColaMesa[mesaAsignada] – tiempoMesa;
150                   colaMesa[mesaAsignada].signal();
151             fin_si
152             sino
153                   mientras clienteColaMesa[mesaAsignada] > -1
154 hacer
155                         colaMesa[mesaAsignada].await();
156                   fin_mientras
157                   clienteColaMesa[mesaAsignada] <- id;
158             fin_sino
159       finalmente
160             l.desbloquear();
161 end
162
163
164 ***HILO CLIENTE*** EXTIENDE THREAD
165 id, tiempoMaquina, tiempoMesa: entero;
166 monitorMaquina: MonitorMaquina;
167 monitorMesa: MonitorMesa;
168
169 Constructor HiloCliente(id, tiempoMaquina, tiempoMesa: entero,
170 monitorMaquina: MonitorMaquina, monitorMesa: MonitorMesa)
```

```
begin
      id <- id;
      tiempoMaquina <- tiempoMaquina;
      tiempoMesa <- tiempoMesa;
      monitorMaquina <- monitorMaquina;
      monitorMesa <- monitorMesa;
end

publico run()
begin
      mesaAsignada: entero;
      intentar
            monitorMaquina.solicitarMaquina(id, tiempoMaquina,
tiempoMesa, monitorMesa);
            Thread.dormir(tiempoMaquina);
            mesaAsignada <- monitorMaquina.solicitarMaquina(id,
tiempoMaquina, tiempoMesa, monitorMesa);
            monitorMesa.solicitarMesa(id, mesaAsignada,
tiempoMesa);
            Thread.dormir(tiempoMesa);
            monitorMesa.solicitarMesa(id, mesaAsignada,
tiempoMesa);
      capturar(e: InterruptedException)
            e.imprimirTrazaPila();
end


***MAIN***
publico estático main(args[]: array de Strings)
begin
      r: Random;
      monitorMaquina: MonitorMaquina;
      monitorMesa: MonitorMesa;
      clientes[]: array de HiloCliente;
```

```
205
206         r <- Random();
207         monitorMaquina <- MonitorMaquina();
208         monitorMesa <- MonitorMesa();
209         clientes[] <- HiloCliente[50];
210
211         for i in 0 … clientes.length hacer
212               clientes[i] <- HiloCliente(i, r.nextInt(1000),
213   r.nextInt(1000), monitorMaquina, monitorMesa);
214               clientes[i].start();
215         fin_for
216
217   end
218
219
220
221
222
223
```