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

***HILO CONTROLADOR***
buzonPregunta: Mailbox;
buzonCajaA: Mailbox;
buzonCajaB: Mailbox;
buzonAbandonoCaja: Mailbox;
arrayBuzonComunicacion: CommunicationScheme[];
impresora: CommunicationScheme;

tiempo: entero;
devuelveAsignacion: String;
idPreguntaCaja: entero;
idCajaA: entero;
idCajaB: entero;
idAbandono: entero;

cajaAOcupada: booleano;
cajaBOcupada: booleano;
activaImpresora: entero estatico;

Constructor HiloControlador(pregunta: Mailbox, cajaA: Mailbox, cajaB: Mailbox,
abandono: Mailbox,
                            array CommunicationScheme[], impresora:
                            CommunicationScheme)
begin
    buzonPregunta <- pregunta;
    buzonCajaA <- cajaA;
    buzonCajaB <- cajaB;
    buzonAbandonoCaja <- abandono;
    arrayBuzonComunicacion <- array;
    impresora <- impresora;

    cajaAOcupada <- false;
    cajaBOcupada <- false;
end

Procedimiento run()
begin

    send(impresora, activaImpresora);

    mientras true hacer
        select entre
            idPreguntaCaja <- receive(buzonPregunta);
            tiempo <- Random[1,10];
            si tiempo >= 5 hacer
                devuelveAsignacion <- tiempo.convertiraString() + ",A";
            sino devuelveAsignacion <- tiempo.convertiraString() + ",B";
            fin_si
            send(arrayBuzonComunicacion[idPreguntaCaja], devuelveAsignacion);
        or
            when !cajaAOcupada =>
            idCajaA <- receive(buzonCajaA);
            cajaAOcupada <- true;
            send(arrayBuzonComunicacion[idCajaA], "ok");
        or
            when !cajaBOcupada =>
            idCajaB <- receive(buzonCajaB);
            cajaBOcupada <- true;
            send(arrayBuzonComunicacion[idCajaB], "ok");
        or
            idAbandono <- receive(buzonAbandonoCaja);
            si idCajaA == idAbandono hacer
                cajaAOcupada <- false;
            sino cajaBOcupada <- false;
            fin_si
            send(arrayBuzonComunicacion[idAbandono], "ok");
        fin_select
    fin_mientras
end
```

```
72
73  ***HILO COMPRADOR***
74  buzonPregunta: CommunicationScheme;
75  buzonCajaA: CommunicationScheme;
76  buzonCajaB: CommunicationScheme;
77  buzonAbandonoCaja: CommunicationScheme;
78  arrayBuzonComunicacion: CommunicationScheme[];
79  impresora: CommunicationScheme;
80
81  id: entero;
82  tiempo: String;
83  caja: String;
84  mensajeAsignacion: String;
85  mensajeImprimir: entero;
86
87  Constructor HiloComprador(id: entero, pregunta: CommunicationScheme, cajaA:
    CommunicationScheme,
88                            cajaB: CommunicationScheme, abandono:
                              CommunicationScheme,
89                            array CommunicationScheme[], impresora:
                              CommunicationScheme)
90  begin
91      id <- id;
92      buzonPregunta <- pregunta;
93      buzonCajaA <- cajaA;
94      buzonCajaB <- cajaB;
95      buzonAbandonoCaja <- abandono;
96      arrayBuzonComunicacion <- array;
97      impresora <- impresora;
98  end
99
100 Procedimiento run()
101 begin
102     for i in 0..5 hacer
103         // PASO 1: REALIZA LA COMPRA
104         sleep(Random()*1000);
105
106
107         // PASO 2: SOLICITAR CAJA
108         // PRIMERO PREGUNTA POR LA CAJA QUE ESTÉ LIBRE
109         send(buzonPregunta, id);
110         mensajeAsignacion <- receive(arrayBuzonComunicacion[id]);
111         partes: String[];
112         partes <- split(mensajeAsignacion, ",");
113         tiempo <- partes[0];
114         caja <- partes[1];
115
116         // LUEGO SOLICITA ENTRAR EN ESA CAJA
117         si caja == "A" hacer
118             send(buzonCajaA, id);
119         sino send(buzonCajaB, id);
120         fin_si
121         receive(arrayBuzonComunicacion[id]);
122
123
124         // PASO 3: PAGAR EN CAJA
125         sleep(tiempo.convertiraInt()*1000);
126
127
128         // PASO 4: LIBERAR CAJA
129         send(buzonAbandonoCaja, id);
130         receive(arrayBuzonComunicacion[id]);
131
132
133         // PASO 5: IMPRIMIR POR PANTALLA
134         mensajeImprimir <- receive(impresora);
135         Imprimir("Persona " + (id+1) + " ha usado la caja " + caja + "\n");
136         Imprimir("Tiempo de pago=" + tiempo + "\n");
137         Imprimir("Thread.sleep(" + tiempo + ")\n");
138         Imprimir("Persona " + (id+1) + " liberando la caja " + caja + "\n");
139         send(impresora, mensajeImprimir);
140     fin_for
141 end
```

```
142
143 ***MAIN***
144 preguntaComprador: Mailbox;
145 cajaA: Mailbox;
146 cajaB: Mailbox;
147 abandonoCaja: Mailbox;
148 impresora: Mailbox;
149
150 arrayBuzon: arrayList de Mailbox;
151 compradores: arrayList de HiloComprador;
152
153 controlador: HiloControlador;
154
155 Procedimiento main()
156 begin
157     PreguntaComprador <- Creacion Mailbox;
158     cajaA <- Creacion Mailbox;
159     cajaB <- Creacion Mailbox;
160     AbandonoCaja <- Creacion Mailbox;
161     Impresora <- Creacion Mailbox;
162
163     arrayBuzon <- Creacion Mailbox[30];
164     for i in 0..30 hacer
165         arrayBuzon[i] <- Creacion Mailbox;
166     fin_for
167
168     controlador <- Creacion HiloControlador(preguntaComprador, cajaA, cajaB,
        abandonoCaja, arrayBuzon, impresora);
169     controlador.iniciar();
170
171     compradores <- Creacion HiloComprador[30];
172     for i in 0..30 hacer
173         compradores[i] <- Creacion HiloComprador(i, preguntaComprador, cajaA, cajaB,
            abandonoCaja, arrayBuzon, impresora);
174         compradores[i].iniciar();
175     fin_for
176 end
177
178
179 ***RECURSOS NO COMPARTIBLES Y CONDICIONES DE SINCRONIZACIÓN***
180 idCajaA, idCajaB, cajaAOcupada, cajaBOcupada son recursos no compartibles,
181 ya que al ser variables enteras que cambian de valor, debemos asegurarnos que nadie
    más usa la variable
182 mientras un hilo lo esté haciendo.
183
184 Además, la pantalla es otro recurso no compartible, no puede haber más de un proceso
    usándola a la vez.
185
186 En las condiciones de sincronización:
187     - Debemos asegurarnos que si un comprador solicita acceso a una caja, esta caja
        esté vacía.
188     - Debemos asegurarnos que solo haya un comprador imprimiendo cada vez.
189     - Cada vez que un comprador envía un send, debe ser devuelto un receive, a
        excepción de la impresora,
190         donde un comprador espera a recibir primero un receive (el cual indica que
            puede imprimir), para
191         posteriormente enviar un send diciendo que ha acabado de imprimir.
```