



Guía de Estudio

Control 5

Juan Alvarez - Nelson Baloian

Kurt Schwarze - Erich Reimberg - Andrés Muñoz



CC10A - 2003

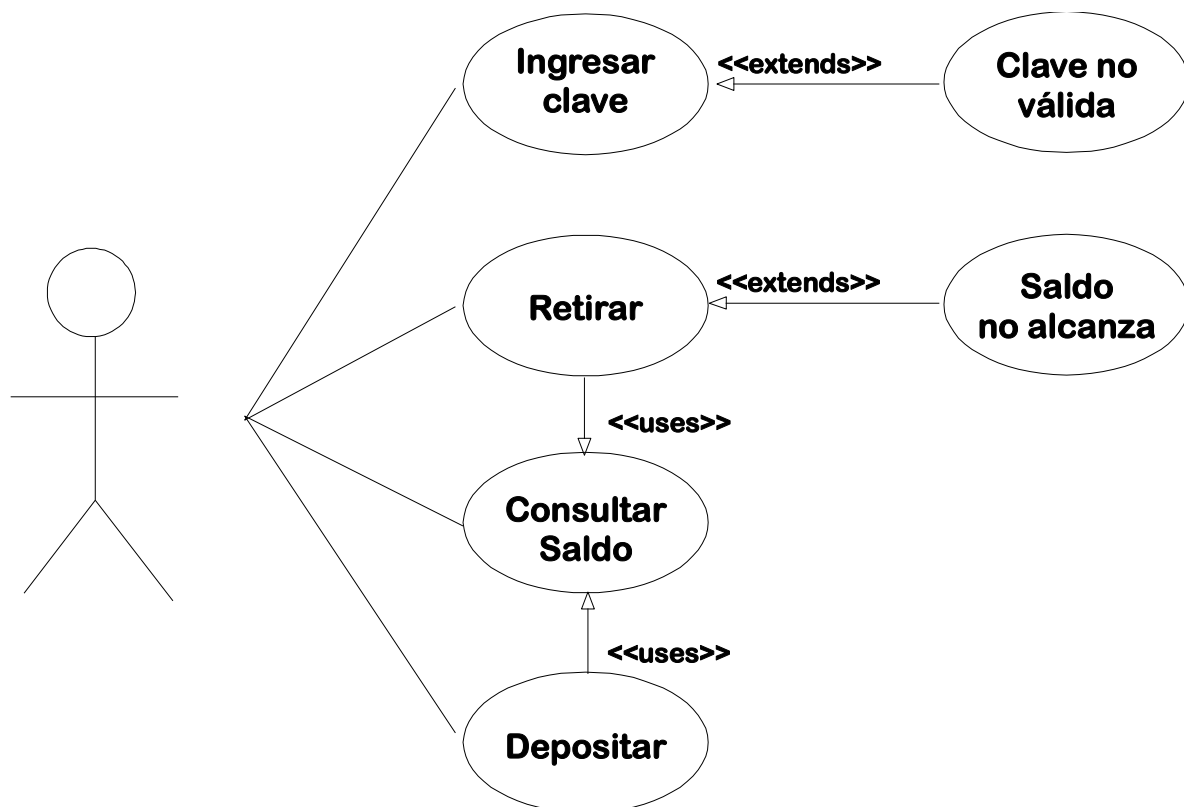
Tabla de Contenidos

TABLA DE CONTENIDOS	2
1. CAJERO AUTOMÁTICO	3
2. VENDOMÁTICA	4
3. APUESTAMÁTICO	5
4. VIDEOJUEGO	6
5. EJERCICIO UML - AÑO 2001	7
6. CENTRAL TELEFÓNICA	11
7. LA BIBLIOTECA	12
8. EL VUELO	14
9. FUTBOL ON-LINE	15
10. EJERCICIO UML - AÑO 2002	18
11. JUEGO DE DAMAS	21
12. EL ASCENSOR (PARTE 2)	25
13. CARRITO DE COMPRAS	28

1. Cajero automático

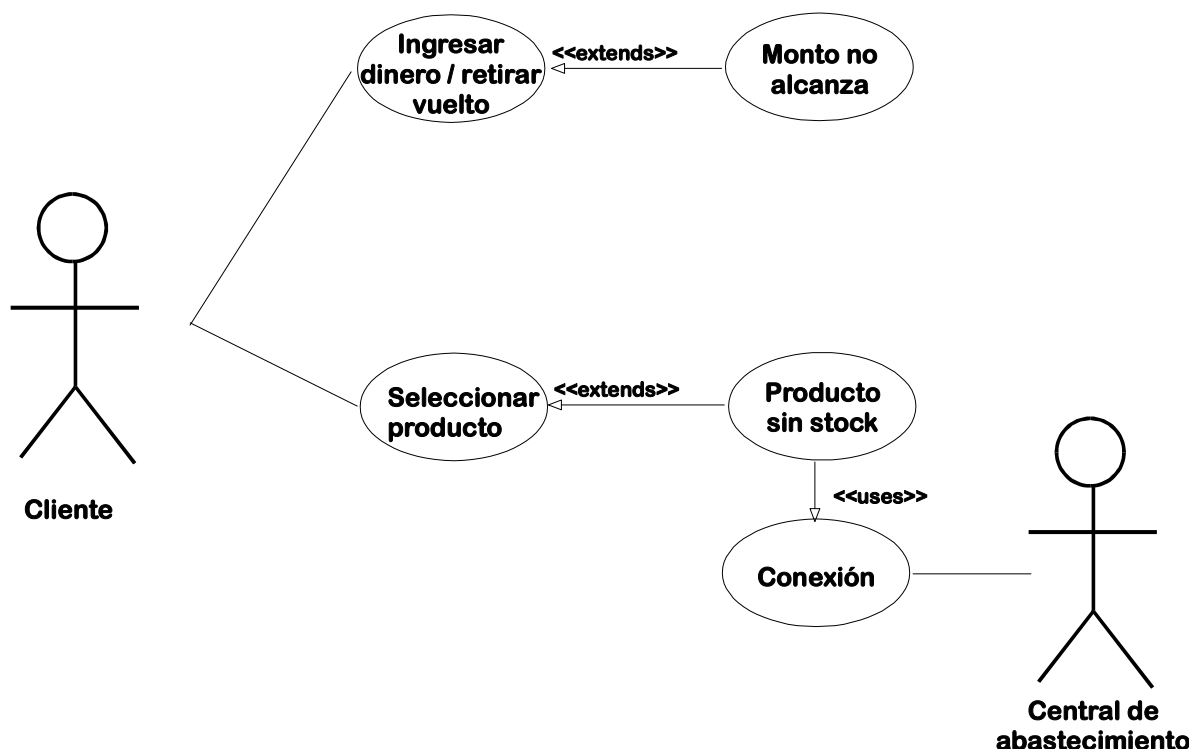
El banco EstafaBank necesita ayuda para modelar el sistema que hará funcionar sus nuevos cajeros automáticos portátiles. Estos, del porte de un teléfono público, le permitirán al usuario realizar sólo las operaciones más simples: retirar, depositar y consultar saldo (ni soñar con movimientos entre cuentas o compras de tarjetas de prepago telefónico). Para ello ten en consideración que:

- Se pide ingresar la clave del usuario posteriormente al paso de la tarjeta por la ranura.
- No se puede retirar más fondos de los que realmente hay, notificando de esta situación al usuario.



2. Vendomática

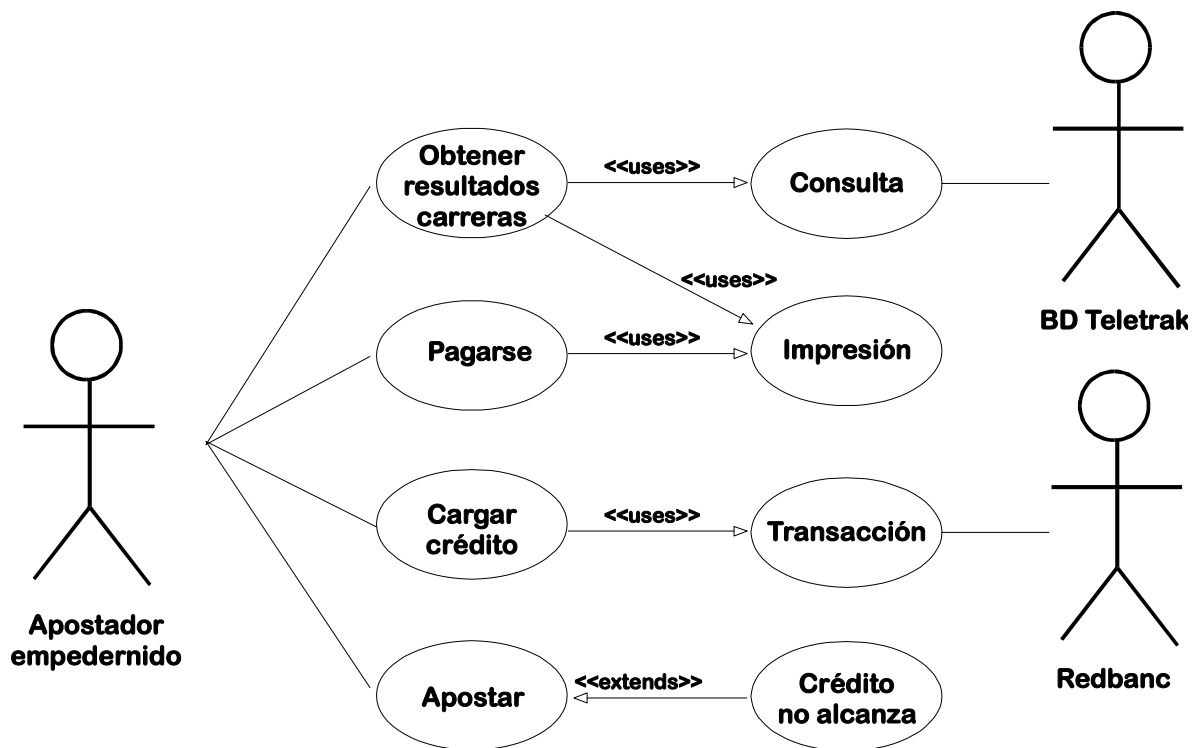
La empresa Nerdcafé tiene planes para instalar una nueva máquina vendomática "inteligente" en la facultad. Inteligente porque cuando detecte que un cliente intenta comprar un producto agotado, se conectará automáticamente a la central de abastecimiento y dará aviso para realizar la reposición. Además, como buena vendomática, debe dar vuelto y no dejar que la hagan lesa pagando menos del precio de lo que está vendiendo.



3. Apuestamático

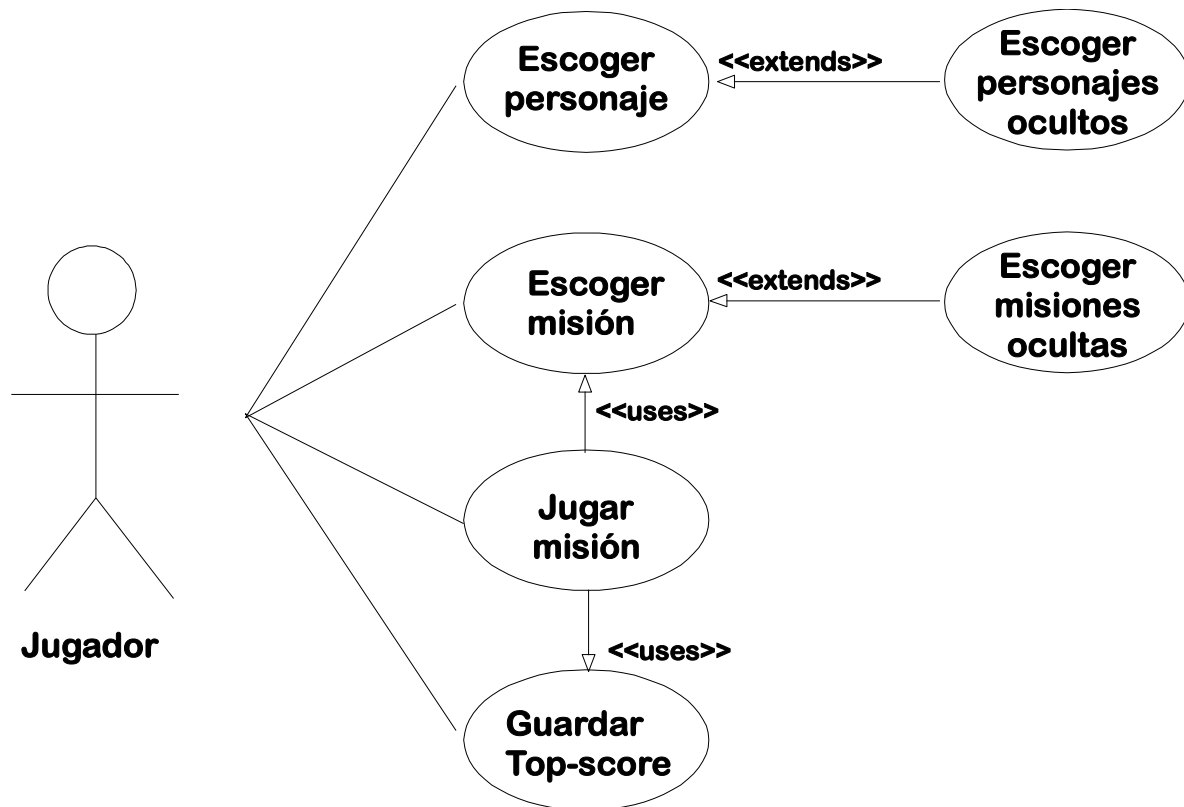
Esta es la última papita para los apostadores empedernidos: una máquina que les permite obtener información de caballos / carreras / premios, cargar crédito de dinero desde su cuenta corriente (accesible vía RedBanc), realizar apuestas y hasta imprimir un boleto que es cambiable por efectivo en la caja del local de apuestas (ya que volver a depositarla es incentivo para que no la gaste).

- No se aceptan apuestas que involucren más dinero que el del crédito actual
- El crédito que el apostador desee cargar debe solicitarse al servidor de redbanc mediante una conexión.
- Tanto la obtención de información como el pago de apuestas utilizan la impresora incluida en el apuestamático.
- La información de carreras/caballos/apuestas se mantiene en un computador con la base de datos de Teletrak



4. Videojuego

Esto es una invención de los años 70, que para ser revividos dentro de un computador hogareño deben utilizarse los llamados "emuladores". Para construir uno se te pide comenzar por diseñar los casos de uso del sistema (suponiendo que es una máquina arcade original) en que el jugador puede escoger un personaje, una misión, jugar la misión y, si logra un buen desempeño, ingresar su "top-score". También se pide incluir los casos en que el jugador conoce del tema y activa las claves para acceder a los personajes y misiones ocultas del juego.



5. Ejercicio UML - Año 2001

1. Indique si la afirmación es (V)erdadera o (F)alsa

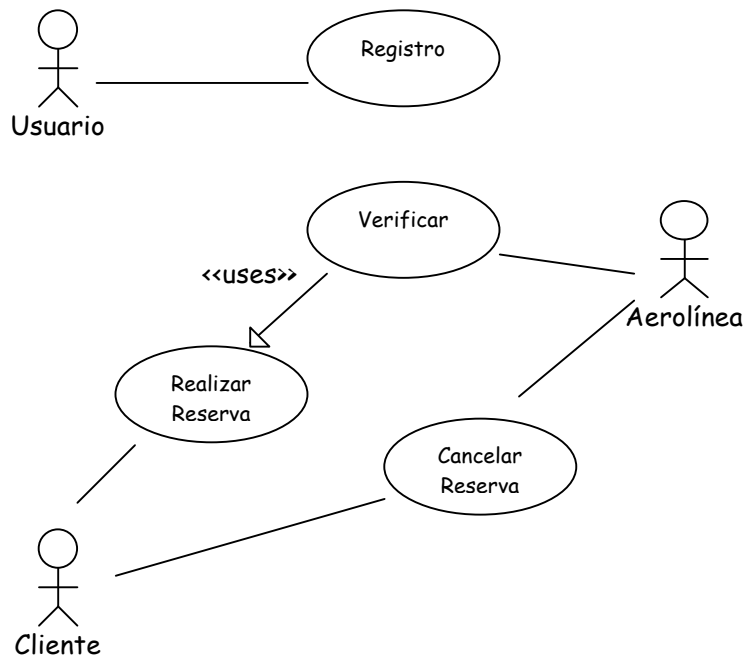
- F** UML es una metodología de desarrollo de software que posee pequeños ciclos iterativos iguales. (1 punto)
- F** Los Casos de Uso ayudan a definir las componentes del sistema como las ve el programador, dejando la perspectiva del usuario en segundo plano. (1 punto)
- V** Los Diagramas de Estado muestran las distintas "etapas" que un objeto del sistema va pasando mientras se ejecuta el software. (1 punto)
- V** De los Diagramas de Interacción se pueden obtener los principales métodos de las clases. (1 punto)
- F** UML sirve para dibujar diagramas más bonitos. (1 punto)
- V** El lenguaje de "The Three Amigos" (Jacobson, Boosh y Rumbaugh) sirve para apoyar el desarrollo de software. (1 punto)
- F** Los Diagramas de Clases detallan el sentido de la ejecución del programa, indicando que componentes se ejecutan después de otra. (1 punto)

2. Diagrama de Casos de Uso

Identifique los actores (2 puntos) y dibuje el Diagrama de C.U. (5 puntos) que represente un software que permita realizar la reserva de boletos de avión en una agencia turística, considerando los siguientes procesos del negocio (especificación de C.U.):

- (a) Todo cliente debe registrarse en el software antes de reservar. (usuario)
- (b) El cliente puede hacer una reserva con un día y hora, para que el sistema se comunice con el software de la aerolínea deseada a verificar el estado del vuelo. Si no hay disponibilidad, el cliente puede seleccionar otro vuelo.
- (c) El cliente puede cancelar una reserva con 48 horas de anticipación mínimo al sistema. Si es así, la reserva se cancela en la aerolínea que se hizo dejando disponibilidad para otro cliente.
- (d) Un agente de viajes puede realizar la función del cliente en caso de que sea desde una oficina física, registrando al mismo cliente y le entrega una clave para que se comuniquen con el sistema.

Este diagrama es solo una solución guía. No es la única.



Algunas pautas para corregir:

- Solo hay 3 actores. Si no reconocen los 3 actores, descontar 0.5 por cada actor que no vean o 0.5 por actor de más. Es cruel, pero no reconocerán más de 4 actores.
- Los procesos base representables en casos de uso son: Registro (1 punto y obvio), Reserva (2 puntos) y Cancelar (2 puntos). El caso de uso Verificar es solo un caso especial en el caso de realizar reserva. También es posible que Cancelar sea un «extends» de Reserva.
- No es necesario que usen «uses» o «extends»

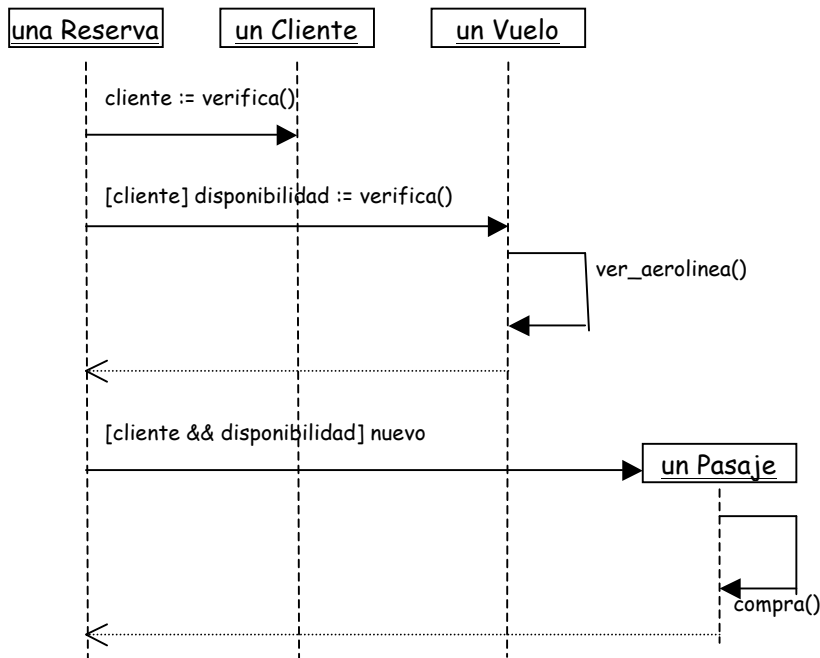
Respuesta de la Pregunta 2

3. Diagrama de Interacción

Identifique los objetos (2 puntos) y dibuje el Diagrama de Secuencia (5 puntos) para el proceso de reserva de avión del problema anterior, considerando que:

- (a) Se debe verificar si el usuario es cliente.
- (b) Se debe verificar la disponibilidad en la aerolínea deseada para la fecha y hora señalada.
- (c) Una vez que se verifica todo, se cursa la Reserva.

Al igual que en la Preg 2, esta es solo una solución (aunque es más fácil que la lógica sea obvia de revisar):



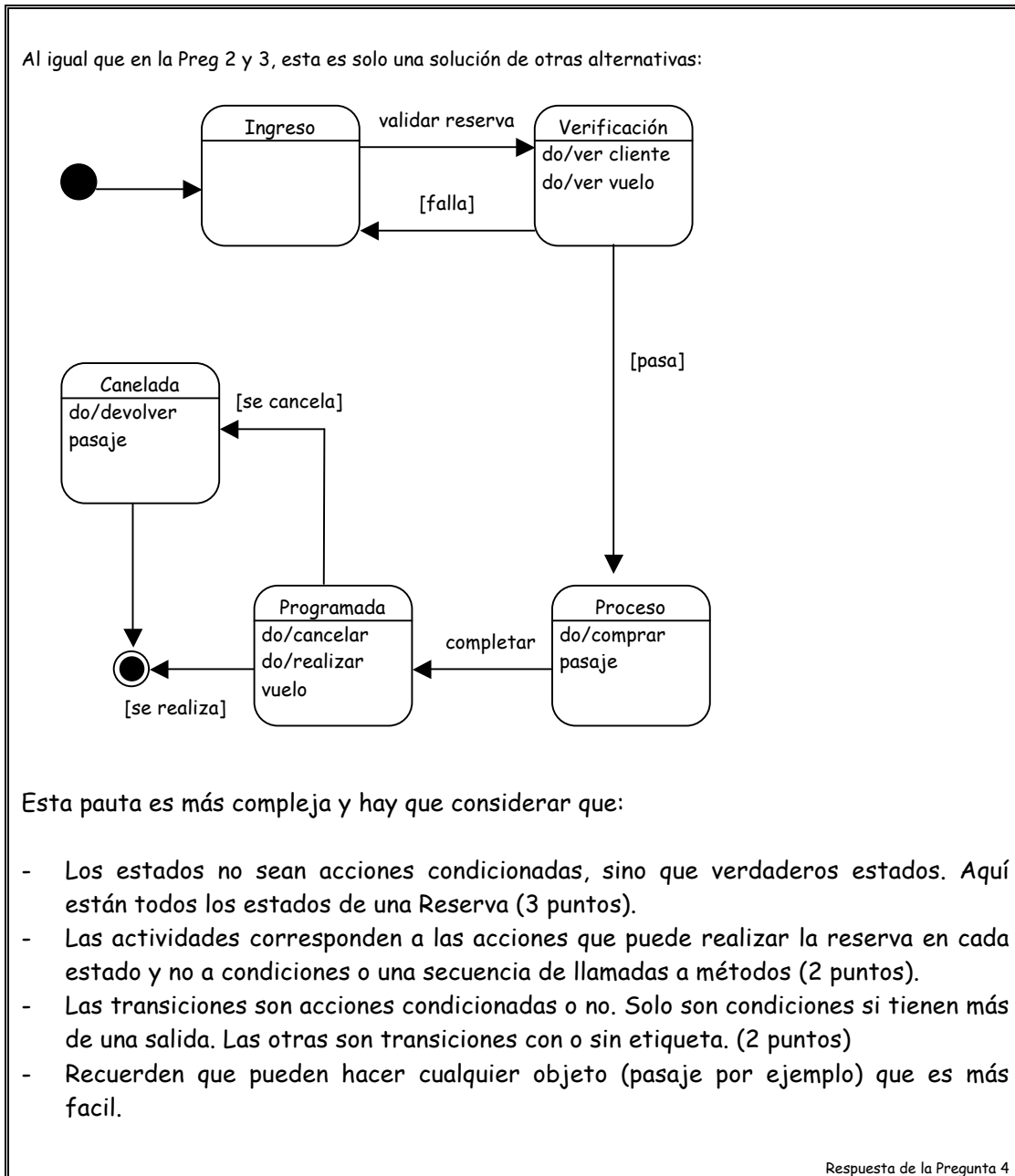
Ese funcionamiento también se basa en:

- Los objetos base son: Reserva (1 punto), Pasaje (1 punto). Cliente y Vuelo son objetos que pueden ser creados para que cada uno verifique las condiciones.
- Lo importante es que en el diagrama aparezcan las verificaciones y condiciones (a) de 1 punto (b) de 1 punto y (c) de 1 punto.
- Los puntos restantes son por darse cuenta que Pasaje se crea solo cuando todo es verdadero y éste debe comprar físicamente la reserva (2 puntos).
- ver_aerolínea() y compra() son autodelegaciones para llamar al software externo.

Respuesta de la Pregunta 3

4. Diagramas de Estado

Elija uno de los objetos dibujados en el problema 3 y dibuje su Diagrama de Estados (7 puntos). Si no hizo la pregunta 3, suponga que tiene la clase **Reserva** y piense qué estados posee.



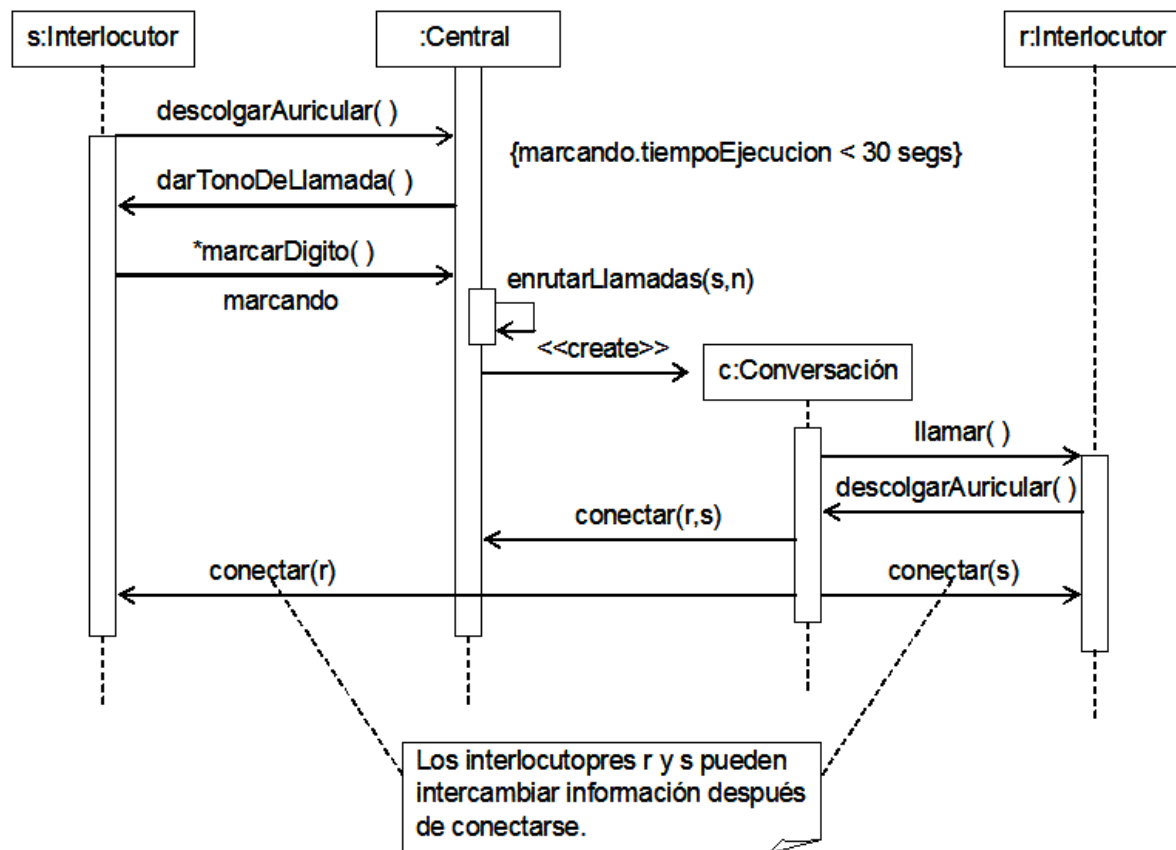
6. Central Telefónica

Gentileza del Profesor Kurt Schwarze

Se quiere modelar una llamada a través de una central telefónica.

Para esto se tienen cuatro objetos involucrados: dos interlocutores (s y r), una central y una conversación. La secuencia empieza cuando un interlocutor envía un mensaje a la central al descolgar al auricular. La central da el tono de llamada, y el interlocutor marca el número al que desea llamar. El tiempo de marcado debe ser menor que 30 segundos.

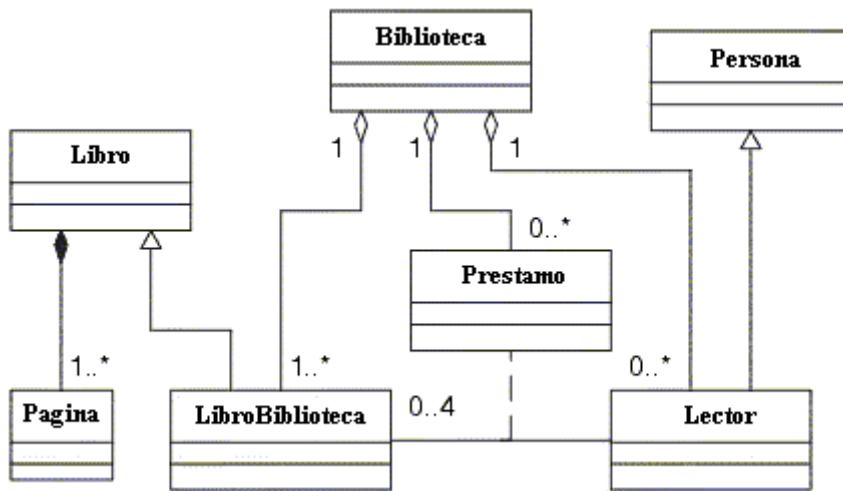
Se pide dibujar el diagrama de interacción (solo el de secuencia) para esta situación.



7. La Biblioteca

Gentileza del Profesor Kurt Schwarze

Se quiere modelar el comportamiento de una biblioteca con diagramas de clases. Este comportamiento se puede modelar de la siguiente forma:



En donde los diamantes indican los conceptos de **Agregación** y **Composición**.

La agregación es simple de leer, puesto que en el caso de la clase Biblioteca y la clase Lector se puede decir que biblioteca tiene como "integrantes" a los lectores, sin embargo ambos pueden vivir por separado (rombos sin rellenar).

La composición es una "agregación más fuerte" y quiere encarecer la necesidad de una clase de la otra, en el caso del diagrama, tenemos que si el Libro desaparece, sus Páginas también son destruidas con él (rombo negro).

Veamos el código en Java que representa este diagrama de clases:

```
public class Libro{
    public Pagina pags[];
}

public class LibroBiblioteca extends Libro{
    private Lector usuario;
}

public class Persona{
}

public class Lector extends Persona{
    private LibroBiblioteca lb[4];
}

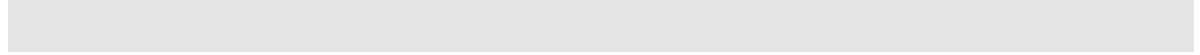
public class Prestamo{
```

Guía de Estudio: Control 5

CC10A 2003;Error! Argumento de modificador desconocido.

```
}  
  
public class Biblioteca{  
    public LibroBiblioteca coleccion[];  
    public Prestamo prestados[];  
    public Lector lect[];  
}
```

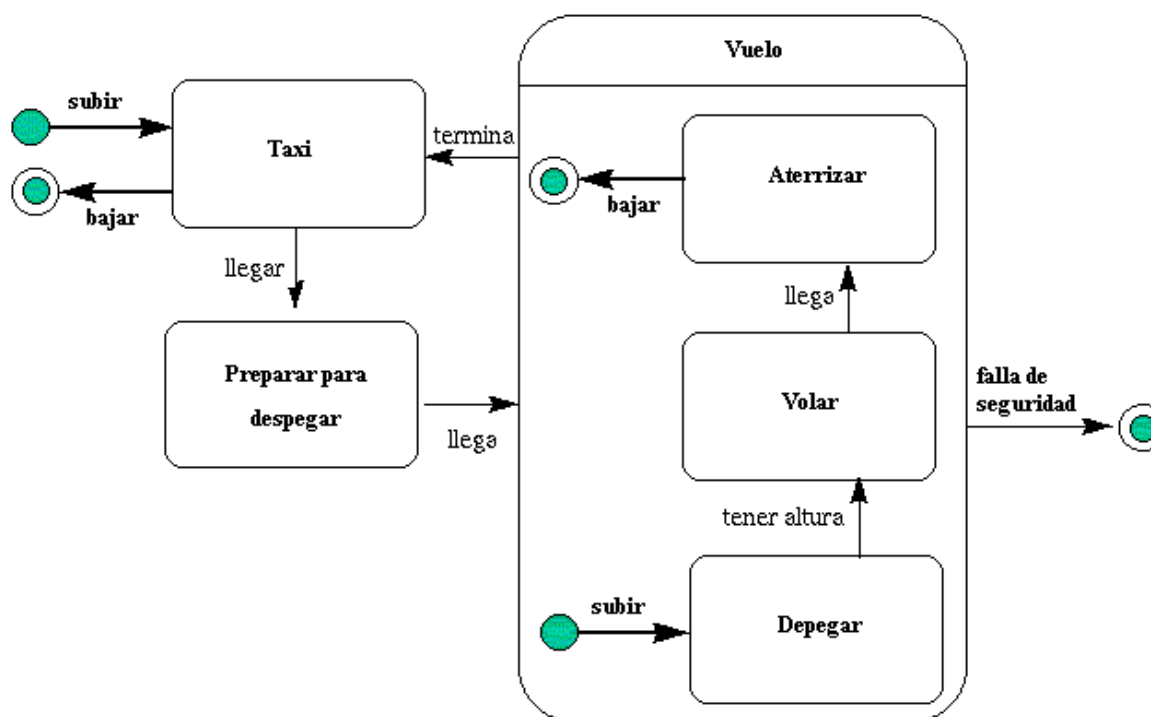
Estas son las clases que son generadas. Para agregar funcionalidades a los metodos de las clases, es necesario ver el diagrama de estado correspondiente a cada clase.



8. El Vuelo

Gentileza del Profesor Kurt Schwarze

Se desea modelar con un diagrama de estados un Vuelo. Esto es, desde que se toma el taxi hasta que despegamos el avión.



Vemos que el diagrama puede poseer cero o más puntos de salida, pero solo un punto de partida.

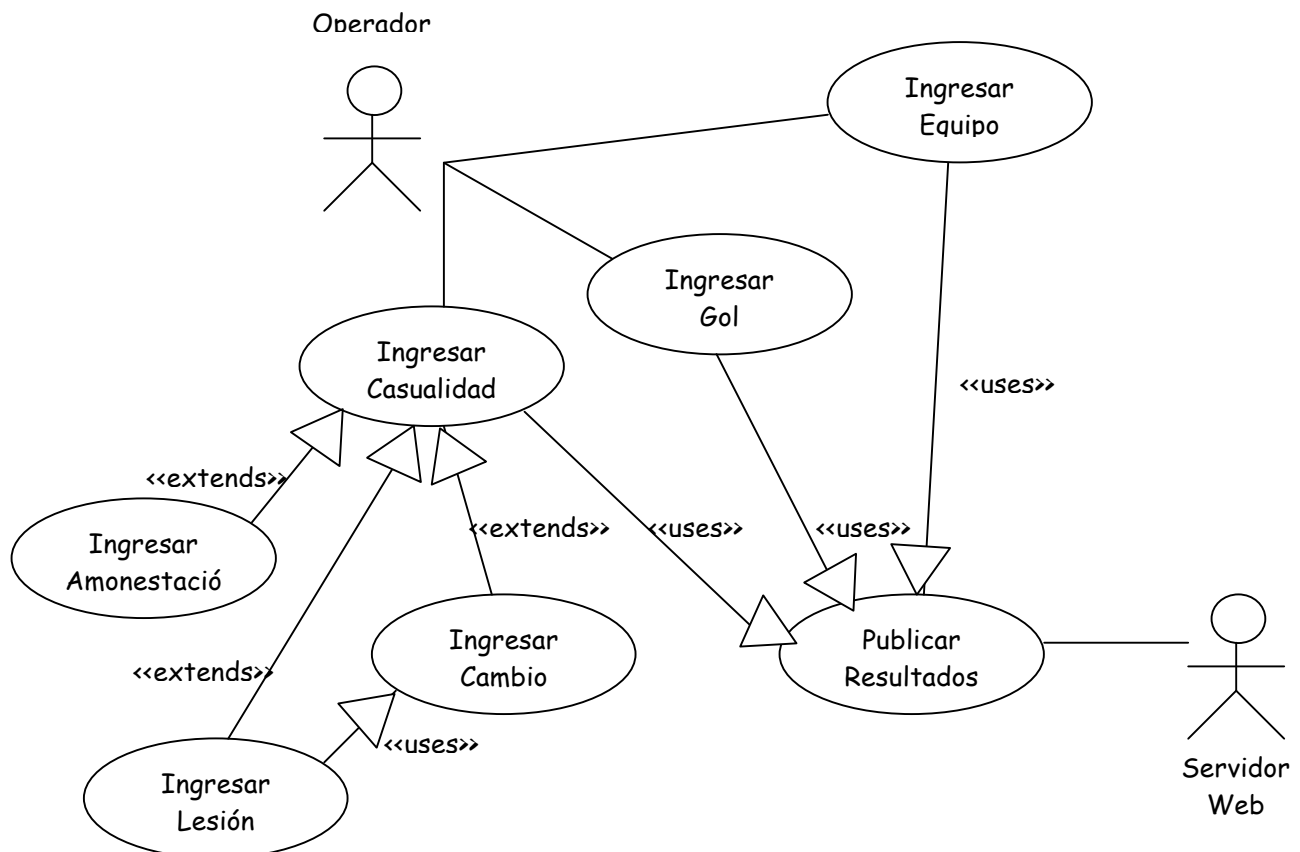
9. Fútbol On-Line

La ANFP quiere comprar un software para mantener en línea los resultados de los partidos de fútbol en un servidor web existente. Este software debe ser operado por unos especialistas que se encuentran en la caseta de transmisión del estadio, y sería alimentado con los siguientes datos:

- Al inicio del software, ingresa los nombres de los equipos y la nómina de jugadores.
- Durante el partido se van almacenando los goles indicando el minuto, el jugador y equipo que convirtió el equipo.
- También se pueden ingresar casualidades como tarjetas amarillas, tarjetas rojas, lesiones y cambios en la formación del equipo.

Considere que el servidor web está fuera del sistema a modelar.

- (a) Identifique los casos de uso y los actores que permitan dibujar un diagrama básico del sistema.



Guía de Estudio: Control 5

CC10A 2003¡Error! Argumento de modificador desconocido.

(b) Escoja el proceso Ingresar Equipo del sistema y escriba un diagrama de Interacción, identificando los objetos que participan en ese proceso.

(c) **Propuesto:** Elija otro proceso y haga el diagrama de interacción.

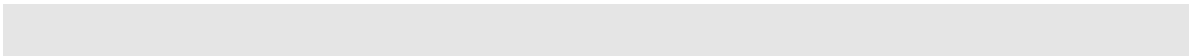
(d) Para el objeto Jugador, dibuje un diagrama de estados.

(e) **Propuesto:** Escoja otro objeto y haga el diagrama de estados.

Guía de Estudio: Control 5

CC10A 2003>Error! Argumento de modificador desconocido.

(f) Dibuje el diagrama de clases que representa la mayor parte del sistema.



10. Ejercicio UML – Año 2002

Problema 1

Responda (V)verdadero o (F)falso:

1. Los diagramas de estado son para analizar el estado del sistema a lo largo de la ejecución.
2. Los diagramas de secuencia son para saber en qué orden se ejecutan las acciones dentro de un proceso.
3. Las clases Java del sistema son representadas en un diagrama de clase.
4. Al diseñar el sistema es importante siempre dibujar diagramas con el enfoque de implementación, porque es más fácil de construir.
5. Los diagramas de colaboración son para mostrar cómo se comunican los actores del sistema.
6. Si un programador tiene que escribir un módulo del sistema, debe usar el diagrama de estados para obtener las clases.
7. Las especificación de un proceso (en un diagrama de Casos de Uso) sirve para describir cómo se comporta éste.

F

V

V

F

F

F

V

Problema 2

Dado el siguiente sistema de control de salidas de buses, en su especificación de casos de uso:

Proceso de Inscripción de Máquina: En este proceso, el operador ingresa un bus identificado por su patente, chofer, sobrecargo, capacidad de pasajeros y distribución de asientos y queda guardado en la base de datos del sistema.

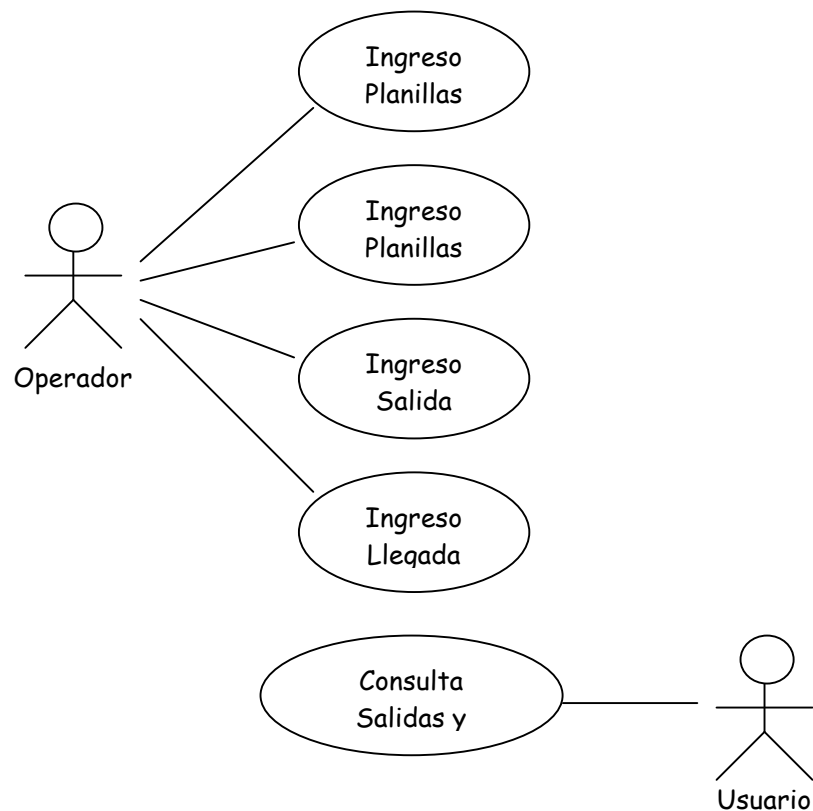
Proceso de Ingreso de Planilla: En este proceso, el operador indica las patentes de los buses que deben salir, andén y el horario de salida de éste. Esto se hace 1 vez al día y se planifican todas las salidas del día.

Proceso de Ingreso de Salida: En este proceso, el operador ingresa la patente del bus que va saliendo y el sistema guarda la hora de llegada. Además, el sistema actualiza que el andén en el cuál estaba ahora está vacío.

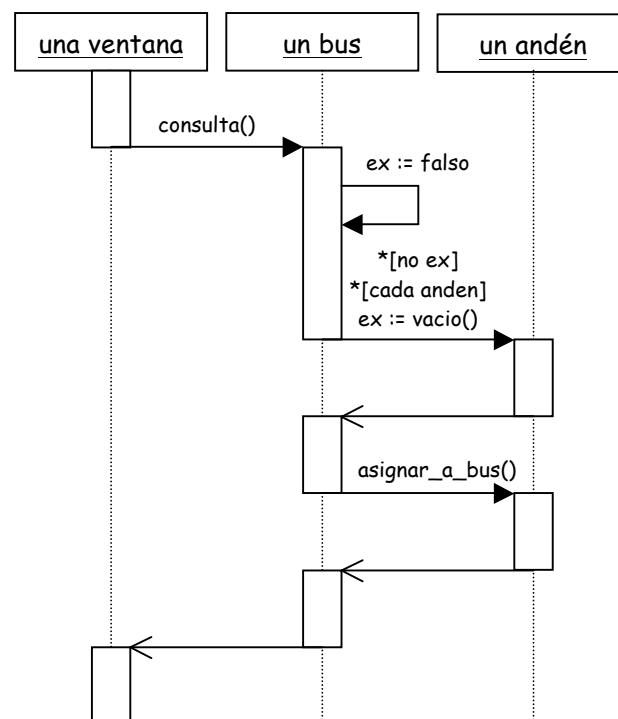
Proceso de Ingreso de Llegada: En este proceso, el operador ingresa la patente del bus que viene llegando y el sistema guarda la hora de llegada. Además, el sistema devuelve el andén en el cuál debe estacionarse el bus (andén vacío).

Proceso de Consulta de Salida y Llegadas: En este proceso, el usuario ve una planilla obtenida desde la base de datos con todas las próximas salidas (próxima hora) y las llegadas que han ocurrido en esa última media hora.

El diagrama de casos de uso sería el siguiente:



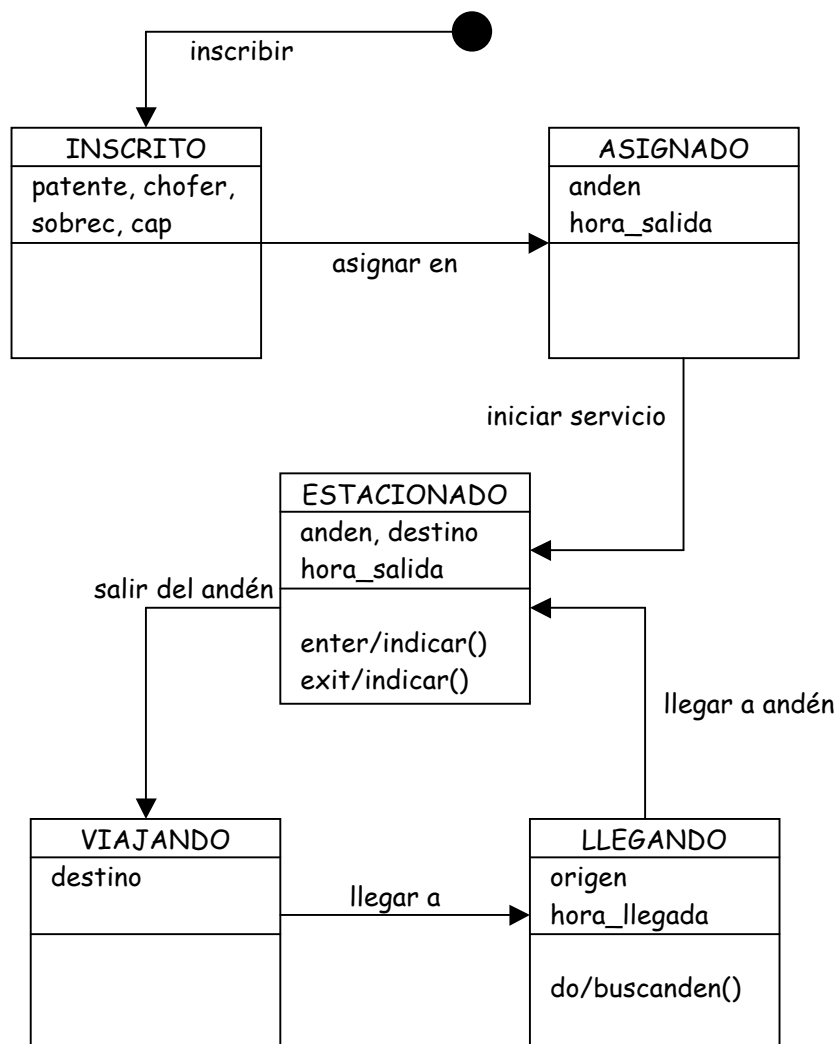
- (a) Dibuje el diagrama de Secuencia para el proceso de Ingreso de Llegada. Recuerde que si no hay andén vacío, el bus debe quedar en espera hasta que lo haga.



Guía de Estudio: Control 5

CC10A 2003;Error! Argumento de modificador desconocido.

- (b) Dibuje el diagrama de Estados del objeto Bus durante todo el sistema, es decir, desde que es ingresado hasta que sale y llega al terminal de buses.



11. Juego de Damas

El **tablero de damas** está compuesto de 64 **celdas** en una grilla de 8x8. El objetivo del juego es *capturar* todas las **piezas** de tu oponente. Los **contendientes** usan su turno para *mover* una de sus piezas del tablero a una celda vacía. Una **movida válida** es una movida diagonal hacia adelante en una celda hacia una vacía o una movida hacia delante saltando sobre una pieza del oponente hasta una celda vacía. La última movida captura la pieza saltada. La pieza saltada es *removida* del tablero.

Después de capturar la pieza del oponente, se puede repetir el mismo procedimiento de captura mientras sea posible capturar otra pieza.

Si un **jugador** logra mover una de sus piezas hasta el borde del tablero del oponente, entonces esta pieza es *promovida* a **Dama**. Una Dama tiene poderes especiales: además de poder moverse diagonalmente hacia adelante, puede también hacerlo hacia atrás.

El juego se gana cuando el jugador ha capturado todas las piezas de su oponente (se puede empatar si un jugador no puede moverse más o ninguno puede comer piezas del otro).

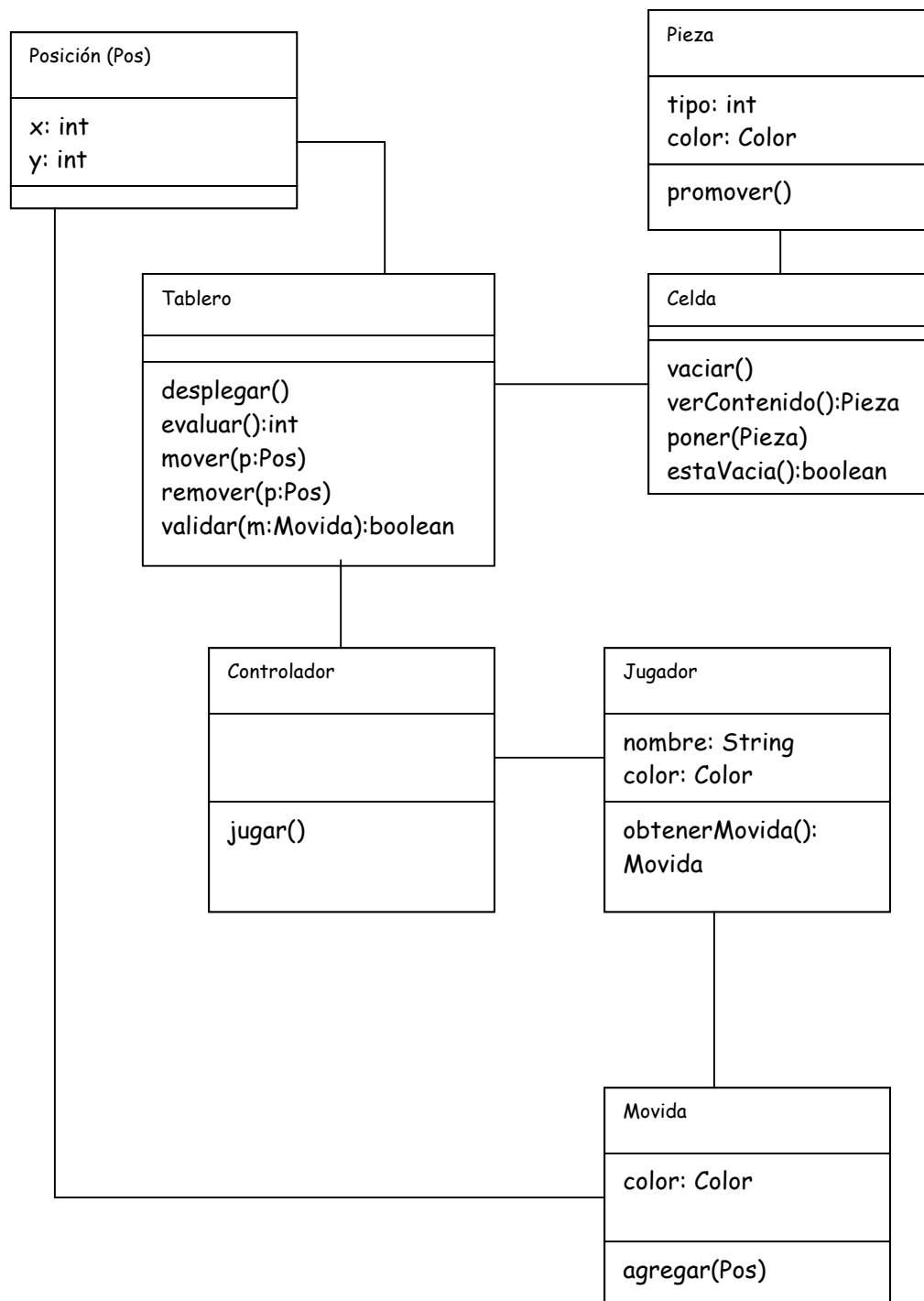
Dado que el juego ocurre en una simulación en el computador, se necesitará un **controlador** del juego que le *pregunta* a cada jugador de turno su movida. Cuando se reciba una movida de un jugador, se solicita al tablero que valide la movida. Si ésta es válida entonces la pieza del jugador actual es movida en el tablero. Si se captura alguna pieza ésta es removida del tablero. El tablero es *desplegado* y se *evalúa* el nuevo estado del juego. Este proceso se repite hasta que un jugador gane al dejar a su oponente sin piezas o se llegue a una posición de bloqueo.

¿Qué hacer para que el sistema permita jugar varias veces ?

Guía de Estudio: Control 5

CC10A 2003 | Error! Argumento de modificador desconocido.

1. Especificar el diagrama de clases que represente este juego.



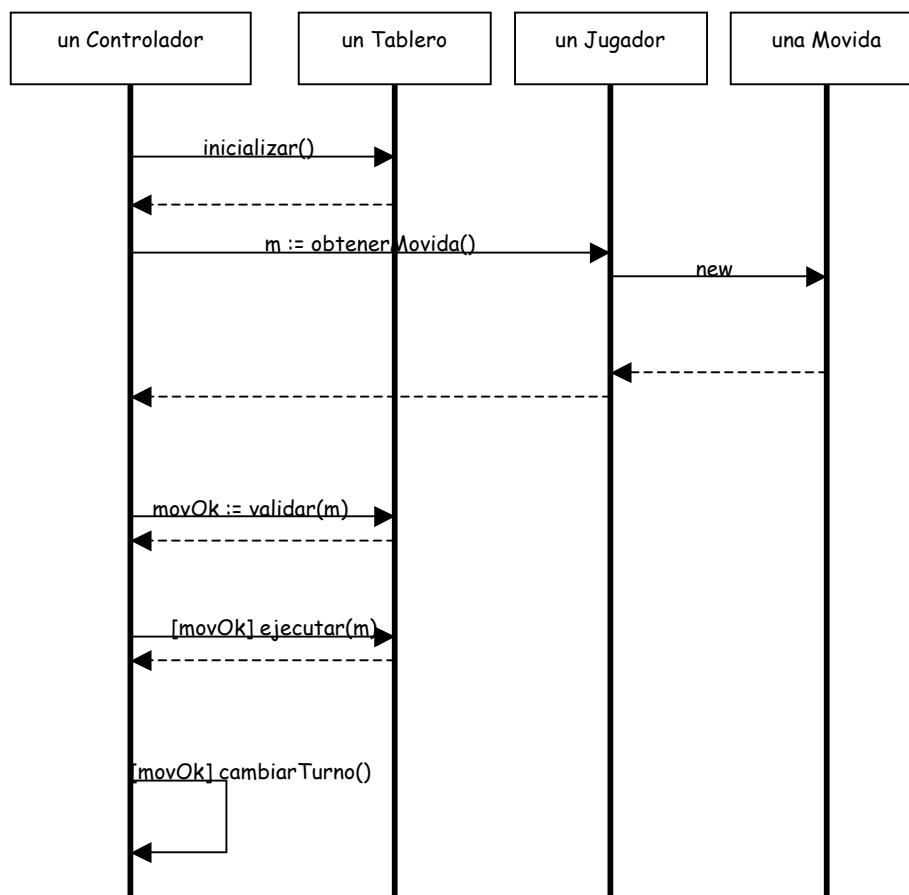
Guía de Estudio: Control 5

CC10A 2003 Error! Argumento de modificador desconocido.

2. Dibujar el diagrama de secuencia para la inicialización y la primera jugada, en base a los siguientes segmentos de código de la implementación.

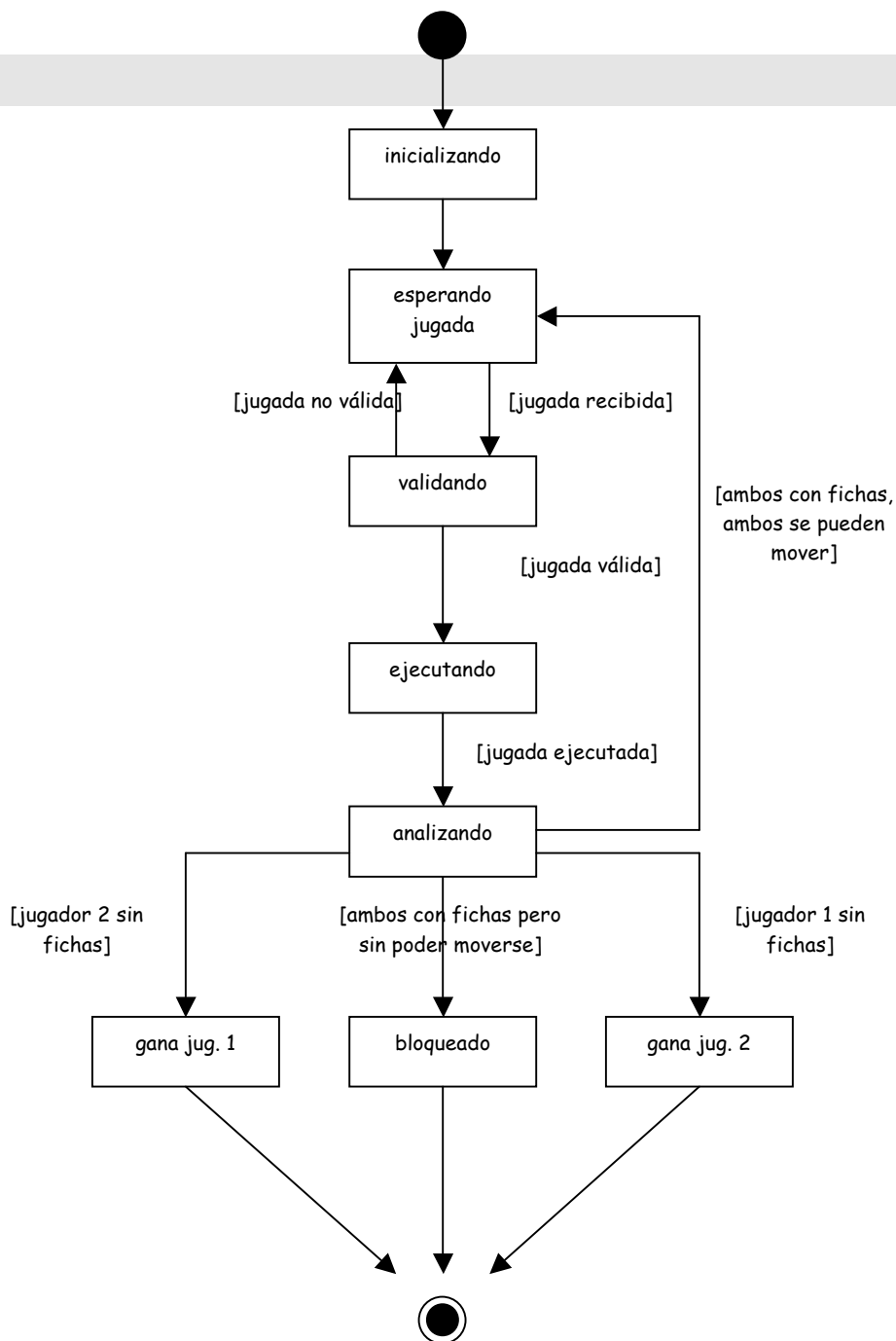
```
class Controlador
// ...
T.inicializar()
Movida m = Jturno.obtenerMovida() ;
boolean movOK = T.validar(m) ;
if (movOK) {
    T.ejecutar(m) ;
    cambiarTurno() ;
}
// ...

class Jugador {
...
public Movida obtenerMovida() {
    Movida m = new Movida() ;
    ...
    return m ;
}
...
}
```



3. Dibujar el diagrama de estados del sistema considerando que:

- Comienza el jugador 1
- La movida entregada por el jugador pasa por una validación: en caso de ser anulada se vuelve a esperar la movida del mismo jugador; si no, se ejecuta.
- Después de cada ejecución, se analiza si el juego puede continuar o si alguno de los jugadores ganó (o sea, no le quedan fichas a su oponente).



12. El Ascensor (Parte 2)

Para el problema del ascensor se han dibujado los diagramas de estado y clase que se muestran a continuación. Se le pide a usted implementar el código en java.

DIAGRAMA DE ESTADOS

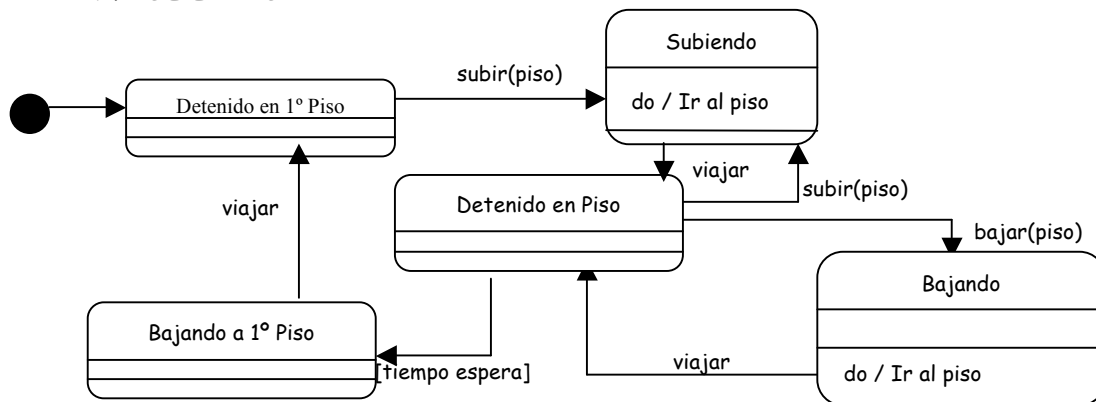
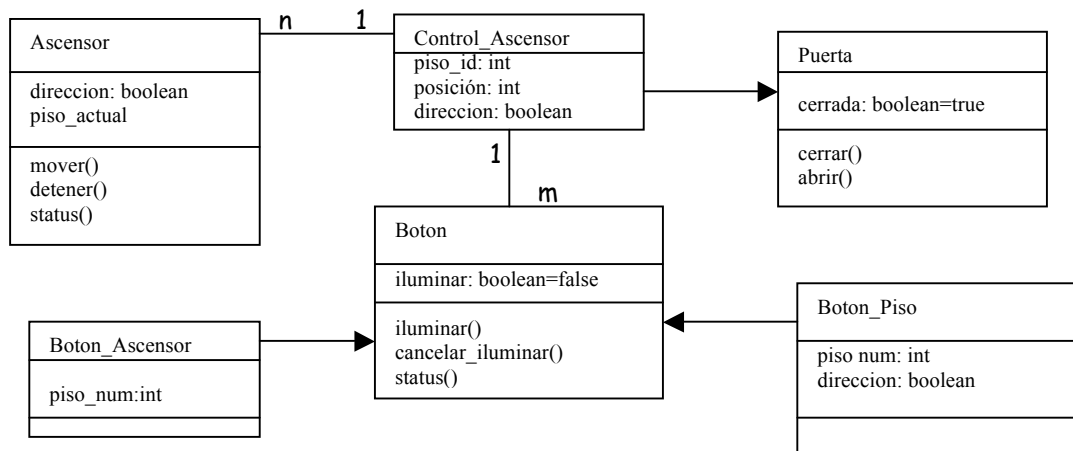


DIAGRAMA DE CLASE



Solución

```

class Ascensor{
    public int direccion;
    public int piso_actual;

    public Ascensor(){
    }

    public void mover(){
        piso_actual+=direccion;
    }

    public void detener(){
        direccion=0;
    }

    public int status(){
        return direccion;
    }
}
    
```

Guía de Estudio: Control 5

CC10A 2003;Error! Argumento de modificador desconocido.

```
}

class Puerta{

    boolean cerrada;
public Puerta(){
    cerrada=true;
}

    public void cerrar(){
        cerrada=true;
    }

    public void abrir(){
        cerrada=false;
    }
}

class Boton{
    boolean iluminar;

    public Boton(){
        iluminar=false;
    }

    public void iluminar(){
        iluminar =true;
    }

    public void cancelar_iluminar(){
        iluminar=false;
    }

    public boolean status(){
        return iluminar;
    }
}

class Boton_Ascensor extends Boton{

    public int piso_num;

    public Boton_Ascensor(int p_num){
        piso_num= p_num;
    }
}

class Boton_Piso extends Boton{
    public int piso_num;
    boolean direccion;

    public Boton_Piso(int p_num, boolean dir){
        super();
        piso_num=p_num;
        direccion =dir;
    }
}

class c_a{

    int n= 10;
    long T_ultimo;
    Boton[] bp=new Boton_Piso[n];
    Boton[] ba=new Boton_Ascensor[n];
    Cola c= new Cola();
}
```

Guía de Estudio: Control 5

CC10A 2003;Error! Argumento de modificador desconocido.

```
Ascensor a =new Asensor();

static public void main(String args[]){

    for(int i=0;i<n;i++){
        bp[i]=new Boton_Piso(i);
        ba[i]=new Boton_Piso(i);
    }
    while(true){
        //vemos que pisos han sido apretaods y los agregamos a la cola
        for(int i=0;i<n;i++){
            if(bp[i].status() & (bp[i].piso_num-
a.piso_actual)/a.direccion=1){
                cola.encolar(bp[i]);
                bp[i].cancelar_linuminar();
            }
            if(ba[i].status() & ((ba[i].piso_num-
a.piso_actual)/a.direccion=1){
                cola.encolar(bp[i]);
                ba[i].cancelar_linuminar();
            }
        }
        if (cola.vacia() && TimeAhora()-T_ultimo< T_espera){
            cola.encolar(1);
            direccion = -1;
        }

        //recorremos los pisos
        while(!cola.vacia()){
            while(a.piso_actual!=c.sacar()){
                a.mover()
            }
            puerta.abrir();
            //se suben/bajan
            puerta.cerrar();
            T_ultimo=TiempoAhora();
        }

        a.detener();
    }
}
```

13. Carrito de Compras

Amazon.com ha decidido que su sistema de compras esta obsoleto, por lo que desea implementarlo utilizando java. Ustedes voluntariosamente se ofrecen para programarlo, (y de paso ganar mucho \$\$\$). Pero antes de contratarlos Amazon desea ver los diagramas de UML, mas especificamente hablando el Diagrama de Clase, para determinar si es que los contrata o no.

Amazon desea que su sistema tenga las siguientes funcionalidades:

- Un cliente puede tener varios carrito de compras, cuando desea comprar algo lo sube a algun carrito, y cuando cambia de opinión lo puede sacar, pudiendo comprar más de un item del mismo producto.
- Una vez que el carrito contiene todo lo que el cliente desea, este puede solicitar la compra de los productos.
- Los clientes pagan el contenido de un carrito utilizando su tarjeta de credito, la cual es verificada previamente.
- Cada cliente tiene que especificar su nombre, dirección de envío, dirección de cobro, email. Algunos clientes son preferenciales, a quienes se les hace un porcentaje de descuento.

