Diseño, modelado dinámico

Eustaquio, un burrero viejo

Departamento de Computación
Facultad de Ciencias Exactas
Universidad de Buenos Aires

Enunciado

Eustaquio, un burrero viejo, saluda al canillita de la esquina de su casa, compra la sexta y parte rumbo al hipódromo de Palermo.

Antes que la carrera comience, Eustaquio se acerca a la ventanilla 7, siempre apuesta en la misma por cábala, y le indica al cajero que desea apostar $1 a la yegua “O Primera o Mortadela” al primer lugar, $2 al caballo “Lole” al segundo lugar y $4 al caballo “Checho Batista” al tercer lugar. El cajero anota las apuestas de Eustaquio y recibe el dinero del apostador. A continuación guarda el dinero en la caja y el ticket de apuesta en el fichero de apuestas pendientes. Luego, el cajero le entrega el recibo a Eustaquio, tanto el recibo como el ticket tienen el mismo número de apuesta.

Finalizada la carrera, Eustaquio se acerca a la ventanilla en busca de sus ganancias (“O Primera o Mortadela” salió afortunadamente primera). El cajero recibe el recibo del apostador, lo compara contra et ticket en el fichero de apuestas pendientes y se fija los resultados de la carrera. Por último, tras corroborar que todo está en orden marca el ticket de apuesta como pagado, lo guarda en el fichero de apuestas pagadas y le paga a Eustaquio el monto correspondiente.

1. Modele mediante diagramas de secuencias la situación planteada, utilice la cantidad de diagramas que considere necesarios.

2. Construya diagramas de clases, completando atributos, métodos y visibilidad en cada clase.

A trabajar ☺ …

Resolución

Escenario: *Eustaquio, un burrero viejo, saluda al canillita de la esquina de su casa, compra la sexta y parte rumbo al hipódromo de Palermo.*

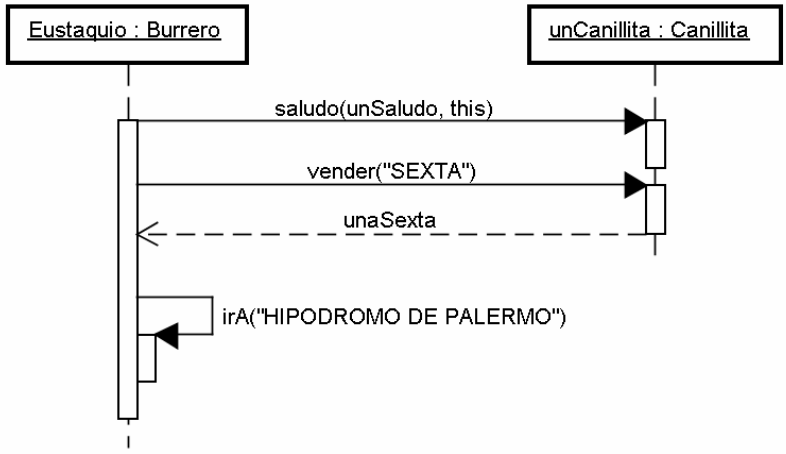

figura 1

El canillita recibe saludo(unSaludo, this) de Eustaquio con los objetos unSaludo y this (es decir, Eustaquio) para que el canillita sepa quién lo saludó . Ya que el que saluda es Eustaquio, no es apropiado que el canillita reciba el mensaje saludar(), éste no saluda. Otras variantes válidas podrían ser recibirSaludo(unSaludos, this), recibir(unSaludo, this) y si alguien argumentara que efectivamente al canillita no le interesa saber quién lo saludó podría obviarse this.

No se muestra cómo el canillita hace para obtener una sexta, podría crearla, pedírsela a un puesto de diarios o bien quizá tener diez sextas en su bolso y pedírsela a éste pero dado que ahora no sabemos nada al respecto no lo modelamos.

Alguien podría preguntarse por la paga de la sexta. Ya que no sabemos nada sobre esto sólo modelamos que el canillita puede vender en particular una sexta, quien sabe, quizá Eustaquio pague todo lo comprado a fin de mes con lo cual no sería necesario que pague en el momento de la compra.

Otra opción al mensaje self es que Eustaquio le envíe un mensaje al hipódromo para que pase a tenerlo.

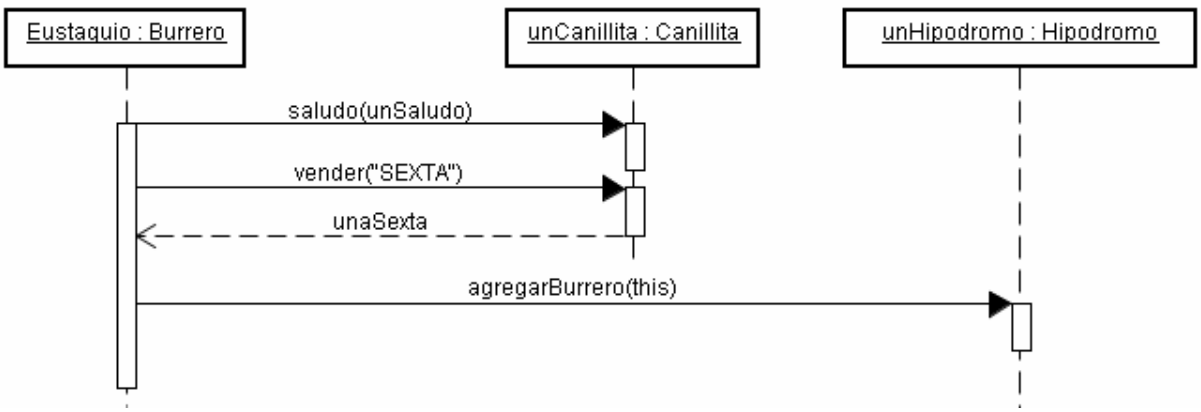

figura 2

Si nos quedamos con la ***figura 1*** dejamos sin modelar en este diagrama qué ocurre cuando Eustaquio recibe el mensaje irA(“HIPODROMO D PALERMO”) que él mismo se mandó. Nada impide, si es de nuestro interés, que luego modelemos en otro diagrama qué ocurre ante la recepción de tal mensaje, por ejemplo:

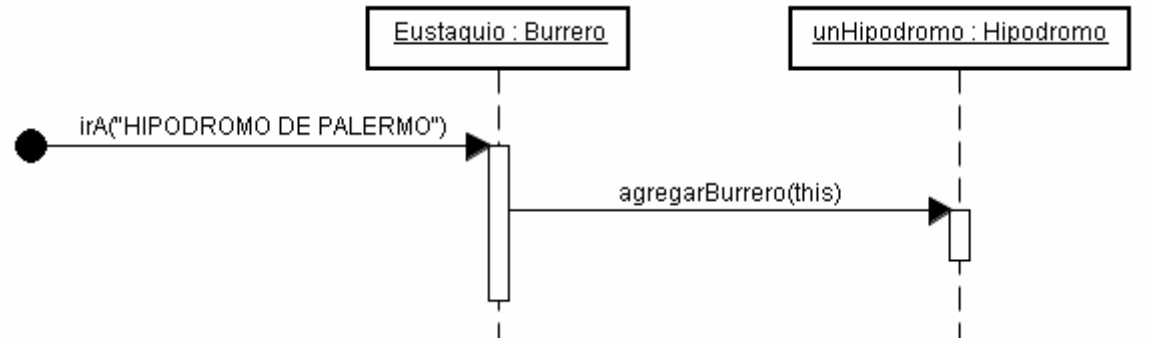

figura 3

Respecto a nombres de algunos tipos de mensajes

- Para acciones de dar y recibir un objeto usaremos un *sustantivo* o *frase nominal* que etiquete el concepto de lo que se está dando o recibiendo. Ejemplo:

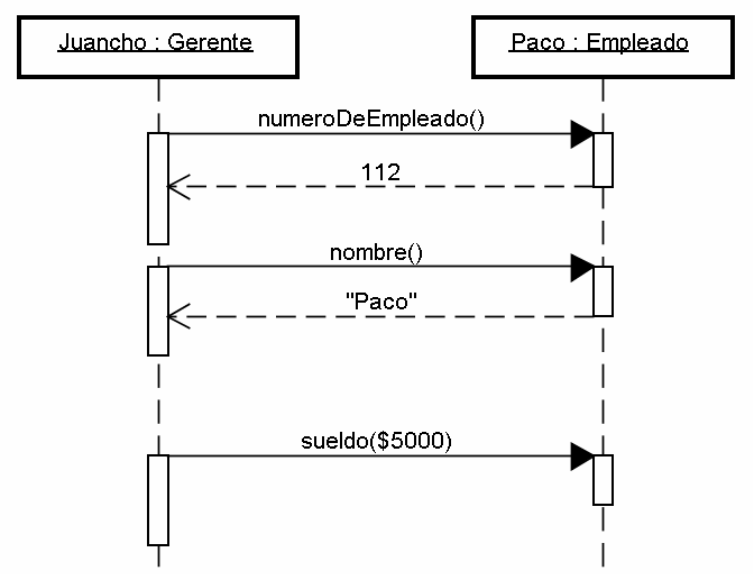

figura 4

En los primeros dos mensajes Paco está dando a conocer su número de empleado y su nombre mientras que en el tercero está recibiendo el sueldo que él tendrá de ahora en más.

De todos modos, podremos tener, por ejemplo, nombres de mensajes tales como darNumeroDeEmpleado(), darNombre() y recibirSueldo($5000) o sus versiones en inglés con prefijos get y set respectivamente conocidos como getters y setters.

- Para acciones usaremos *verbos en infinitivo* tal como el mensaje vender(“SEXTA”) que recibe el canillita.

Escenario: *Eustaquio se acerca a la ventanilla 7, siempre apuesta en la misma por cábala, y le indica al cajero que desea apostar $1 a la yegua "O Primera o Mortadela" al primer lugar, $2 al caballo "Lole" al segundo lugar y $4 al caballo "Checho Batista" al tercer lugar. El cajero anota las apuestas de Eustaquio y recibe el dinero del apostador. A continuación guarda el dinero en la caja y el ticket de apuesta en el fichero de apuestas pendientes. Luego, el cajero le entrega el recibo a Eustaquio, tanto el recibo como el ticket tienen el mismo número de apuesta*.

Notamos que tendremos un objeto unCajero de tipo Cajero. Notamos también que este cajero tiene más responsabilidades que un cajero común y corriente ya que sabe, por ejemplo, recibir apuestas y pagar las ganancias. Esto último no parece ser parte del comportamiento de todo cajero con lo cual sería inapropiado que nuestro cajero sea sólo (se comporte como) un cajero, nuestro cajero tiene el comportamiento de todo cajero y además sabe hacer otras cosas relacionadas con apuestas. Llamaremos cajero de apuestas a este tipo de cajero.

Respecto a qué es una apuesta, por ahora vamos a tomar como tal (recordar que estamos en fase exploratoria) a cada elección de caballo, monto a aportar y posición. Otra variante podría ser considerar una apuesta como el conjunto de caballos elegidos con sus montos apostados y posiciones.

Respecto al fichero, las responsabilidades guardar, sacar y obtener parecen ser parte del comportamiento de un contenedor, en este caso un fichero, pero obtener un ticket a partir de un número de apuesta no es algo que todo fichero deba saber hacer, tendremos entonces un fichero de tickets.

Primer enfoque

Podemos pensar que, dado que un burrero tiene comportamiento de apostador sabe entonces crear apuestas con lo cual será éste el encargado de armar las apuestas y dárselas al cajero. Nos queda lo siguiente:

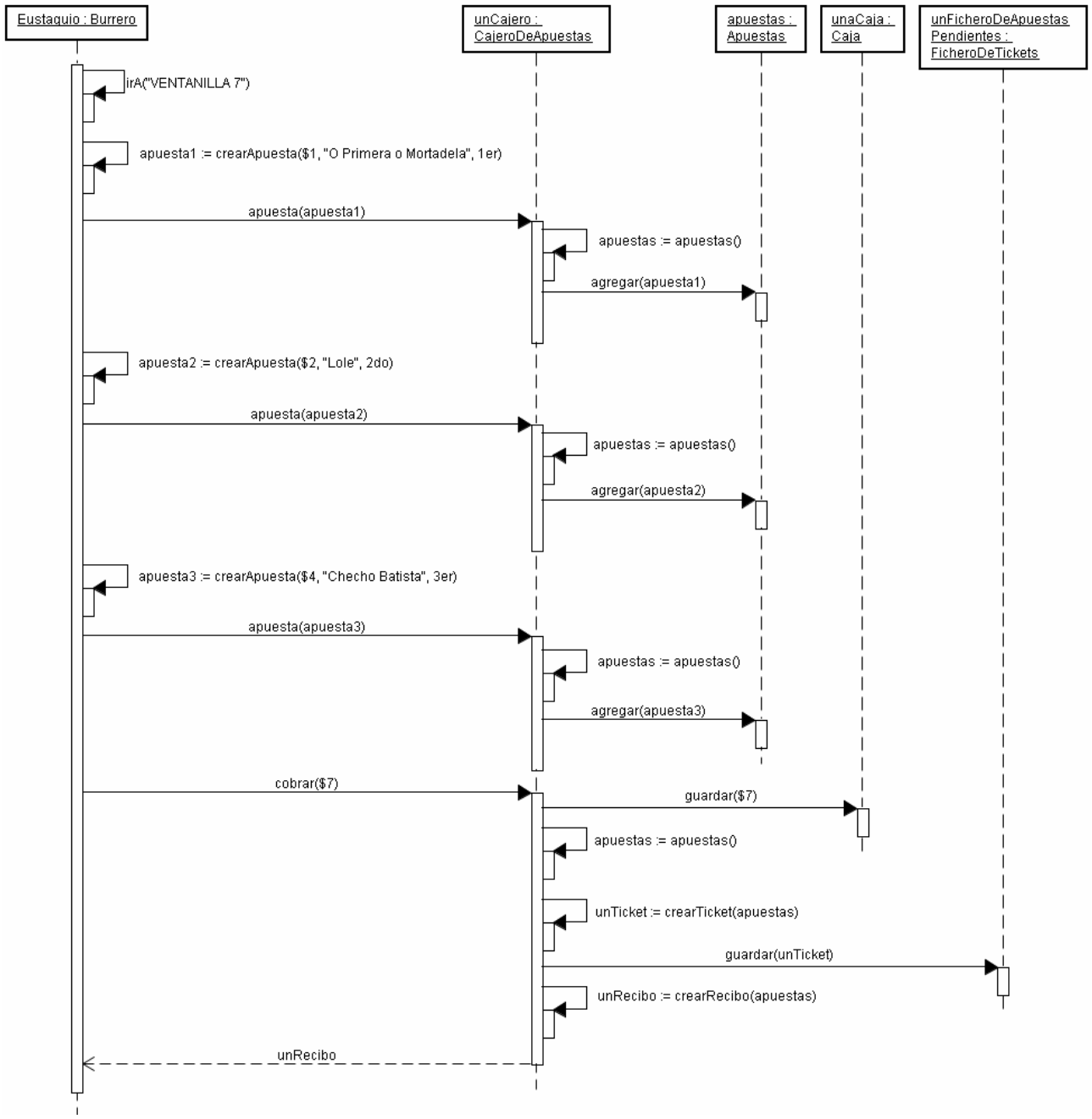

Figura 5 – primer enfoque

El objeto apuestas es el encargado de contener las apuestas que el cajero va recibiendo. Usamos este objeto ya que luego se creará un recibo y un ticket con lo apostado por Eustaquio.

Respecto al pago de las apuestas, otra opción es que la cajera le diga a Eustaquio que pague.

Segundo enfoque

El cajero podría recibir todas las apuestas en un único mensaje apuestas(apuestas).

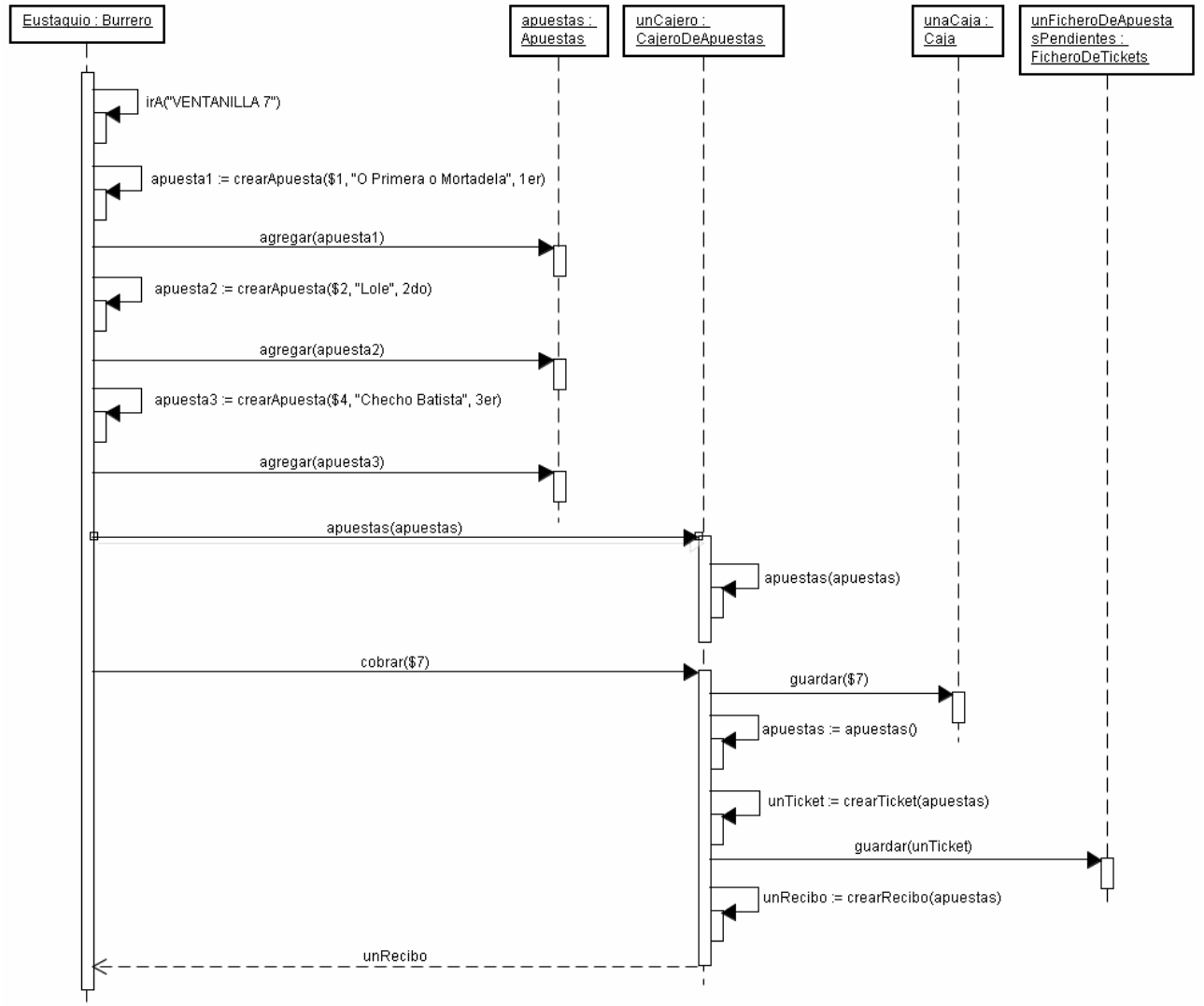

figura 6 – segundo enfoque

Tercer enfoque

Otro enfoque válido es que el burrero Eustaquio le pase al cajero los datos de nombre de caballo, posición y monto a apostar y sea este último quien se encargue de la creación y armado de la/as apuesta/as.

Respecto a la creación de objetos

Otra opción, en lugar del mensaje self crearApuesta(...), es que el objeto unCajero le envíe el mensaje Apuesta(...) a la clase Apuesta que es un objeto , las clases son objetos también. Notar que sólo mostramos el nombre de la clase Apuesta.

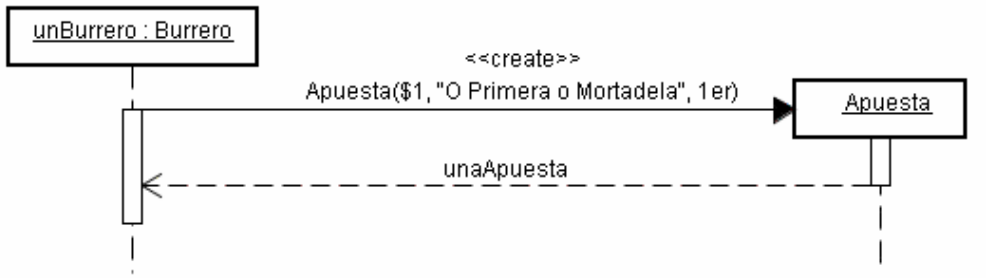

figura 7

<<create>> es el estereotipo del mensaje mostrando justamente que el mensaje es de creación. El estereotipo de un mensaje es opcional.

Uno podría pensar que Eustaquio apuesta a partir de la recepción del mensaje apostar() ya que parece parte de su comportamiento. La pregunta que nos hacemos es quién le envía tal mensaje. No nos preocupamos por esto en este escenario, sólo nos preocupa qué sucede cuando él recibe el mensaje apostar().

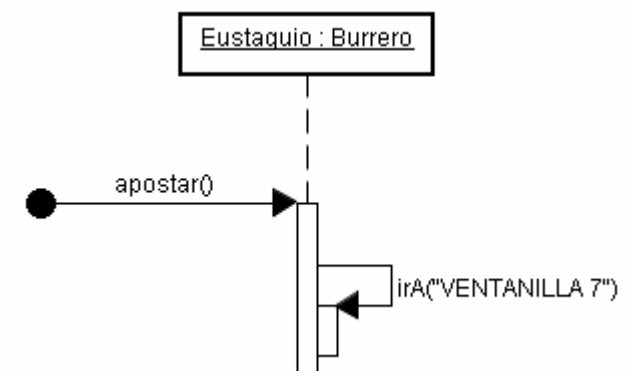

figura 8

Escenario: *Eustaquio se acerca a la ventanilla en busca de sus ganancias ("O Primera o Mortadela" salió afortunadamente primera). El cajero recibe el recibo del apostador, lo compara contra et ticket en el fichero de apuestas pendientes y se fija los resultados de la carrera. Por último, tras corroborar que todo está en orden marca el ticket de apuesta como pagado, lo guarda en el fichero de apuestas pagadas y le paga a Eustaquio el monto correspondiente.*

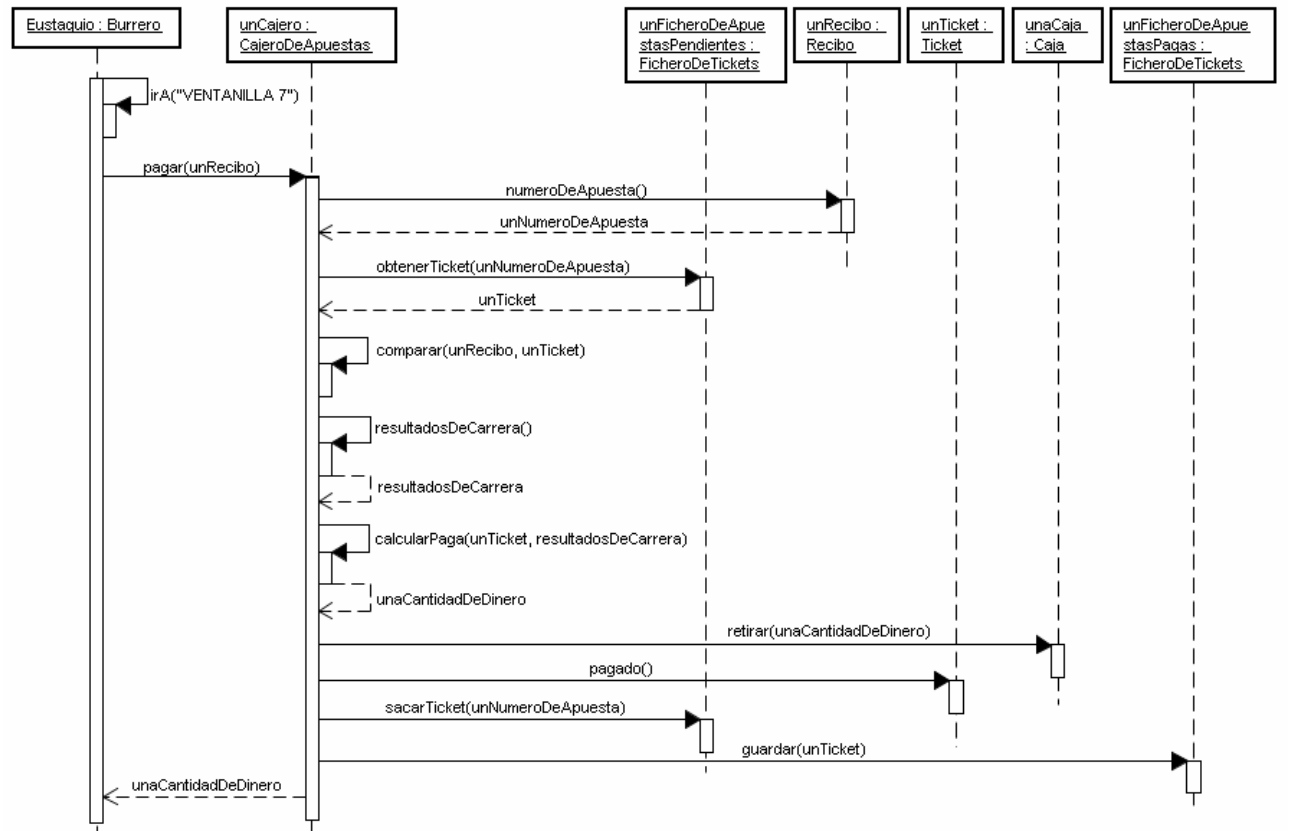

figura 11

Notar que mostramos otra forma de mostrar la respuesta de mensajes self.

Respecto al mensaje comparar(unRecibo, unTicket), si el recibo no se hubiera correspondido con el ticket entonces el escenario habría sido otro. Este nuevo escenario podrá ser mostrado en otro diagrama de secuencias o bien podremos mostrar esta alternativa en el mismo diagrama, en clases posteriores veremos cómo notar esto último. Recordar que lo que deseamos modelar no es más que una situación particular, un escenario.

Respecto a las clases

A medida que vamos descubriendo el comportamiento de cada objeto vamos completando su protocolo. Luego pasamos a las clases que fabricarán los objetos. A continuación se muestran algunos ejemplos de clases para el primer enfoque: