Breve manual de diseño de la práctica del Gestor de Banca del Monopoly

Tabla de contenidos

1- Introducción....2
2- Clases propuestas....3
3- Operativa del juego....5
3.1 Inicio de la aplicación....6
3.2 Operativa del traductor....8
3.3 Operativa del bucle principal del programa....9
3.4 doOperation sobre una propiedad....10
4- Conclusiones....14

Versión 1.1.77

18:24 - 05/12/23

1 Introducción

El siguiente documento pretende guiar al estudiante por un diseño adecuado para implementar la práctica MonopolyBank.

El diagrama de actividad de la Figura 1 muestra a grandes rasgos la lógica asociada a la aplicación MonopolyBank.

Obsérvese que este diagrama no contempla la lógica de bajo nivel asociada a cada operación. Este diagrama tampoco refleja la estructura del juego.

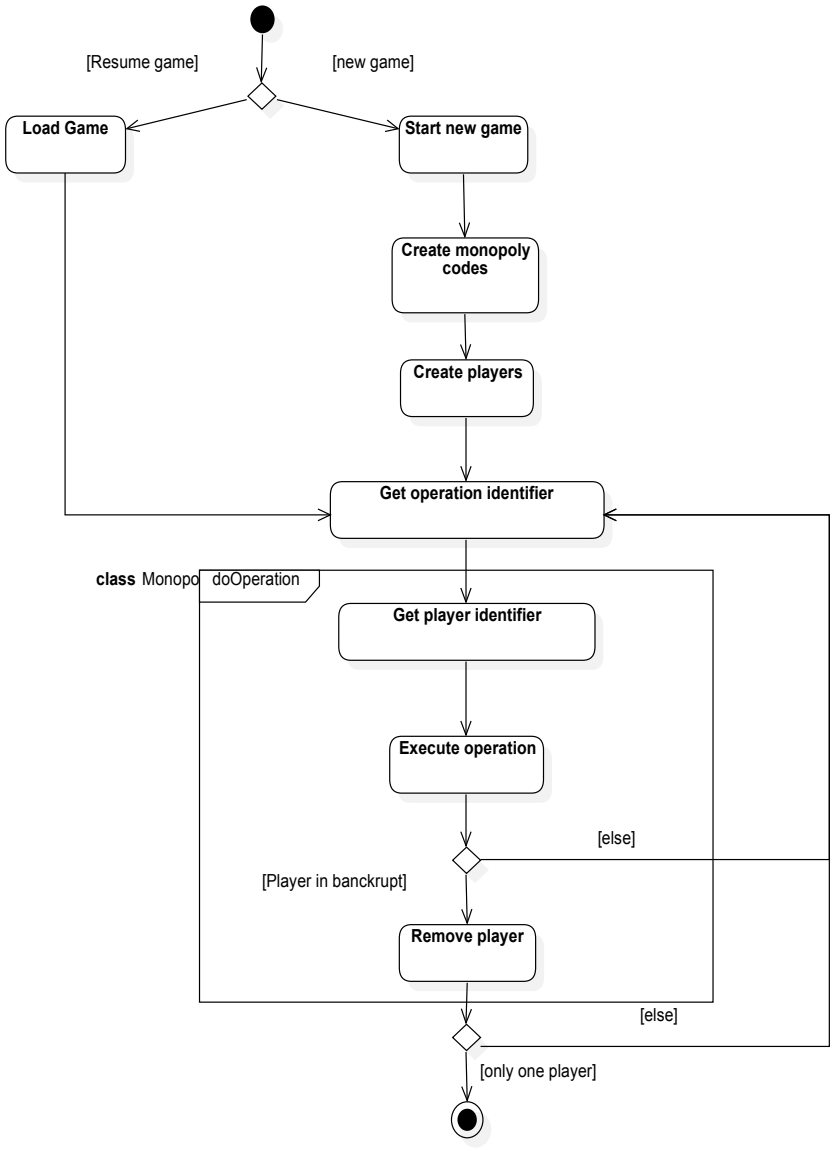

Figura 1.- Diagrama de actividad con la lógica de alto nivel asociada a la práctica.

2 Clases propuestas

La Figura 2 presenta las clases principales que gestionan la operativa. A continuación se resume la responsabilidad de cada clase de este diagrama.

- `MonopolyBank`.- Encargada de iniciar el programa.
- `GameManager`.- Encargada de iniciar un juego nuevo o continuar un juego antiguo.
- `Game`.- Encargada de gestionar una partida.
- `MonopolyCode`.- Encargada de representar la información común de todas las tarjetas y propiedades.
- `Player`.- Encargada de representar gestionar la información de un jugador.

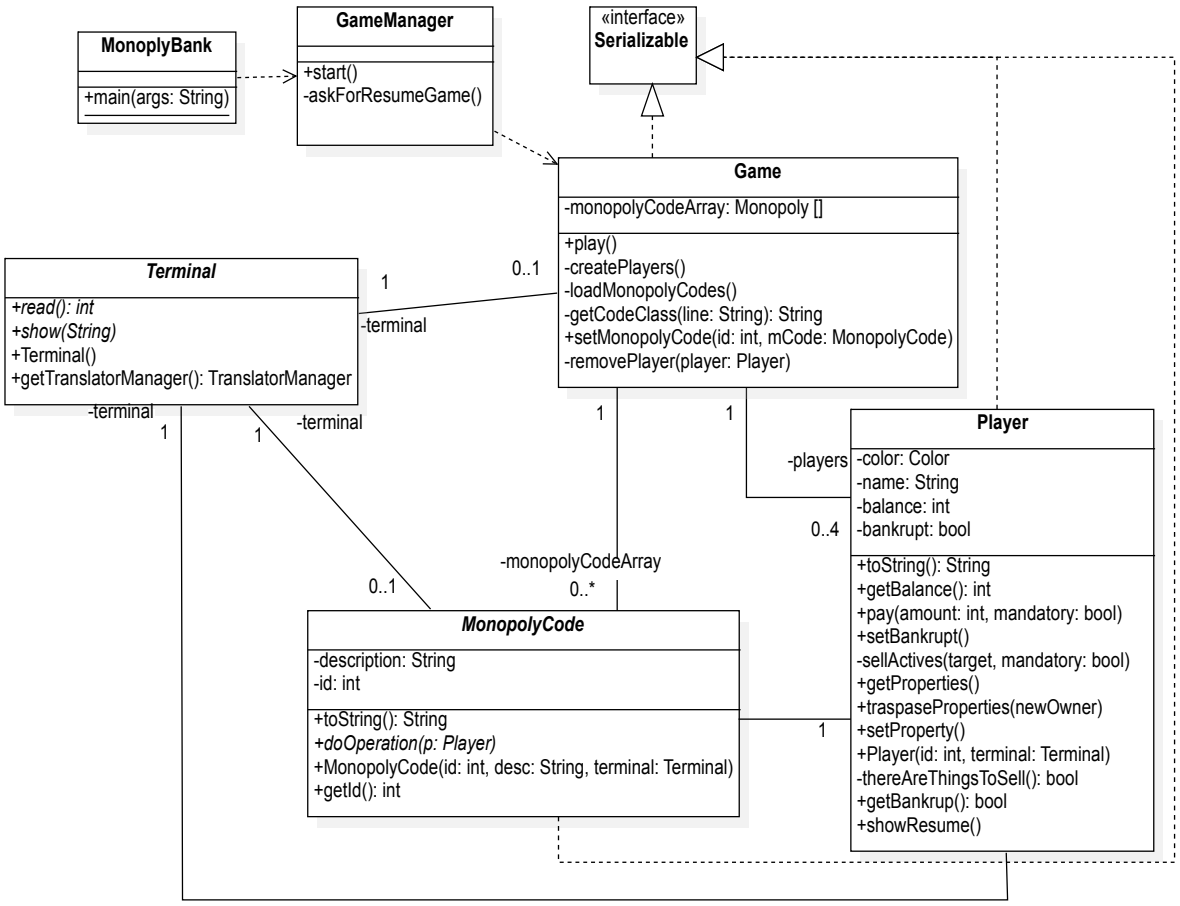

Figura 2.- Diagrama de clases principal.

La Figura 3 presenta las clases que derivan de `MonopolyCode` y que se encargan de gestionar la comunicación con el usuario. A continuación se resume la responsabilidad de cada clase de este diagrama.

- `PaymentCharge.-` Gestiona la información y las operaciones relacionadas con las tarjetas de suerte, las tarjetas de comunidad y con las casillas especiales del tablero (pago de

impuesto, cobra por pasar por la salida...). Al fin y al cabo, todas estas operaciones consisten en cobrar o pagar una cantidad fija.

- `RepairsCard`.- Gestiona la información y las operaciones relacionadas con las tarjetas de pago por reparaciones en casas y hoteles.
- `Property`.– Representa la parte común de todas las propiedades.
- `Transport`.– Gestiona la información y las operaciones relacionadas con las propiedades de tipo transporte.
- `Service`.– Gestiona la información y las operaciones relacionadas con las propiedades de tipo servicio.
- `Transport`.– Gestiona la información y las operaciones relacionadas con las propiedades de tipo transporte.

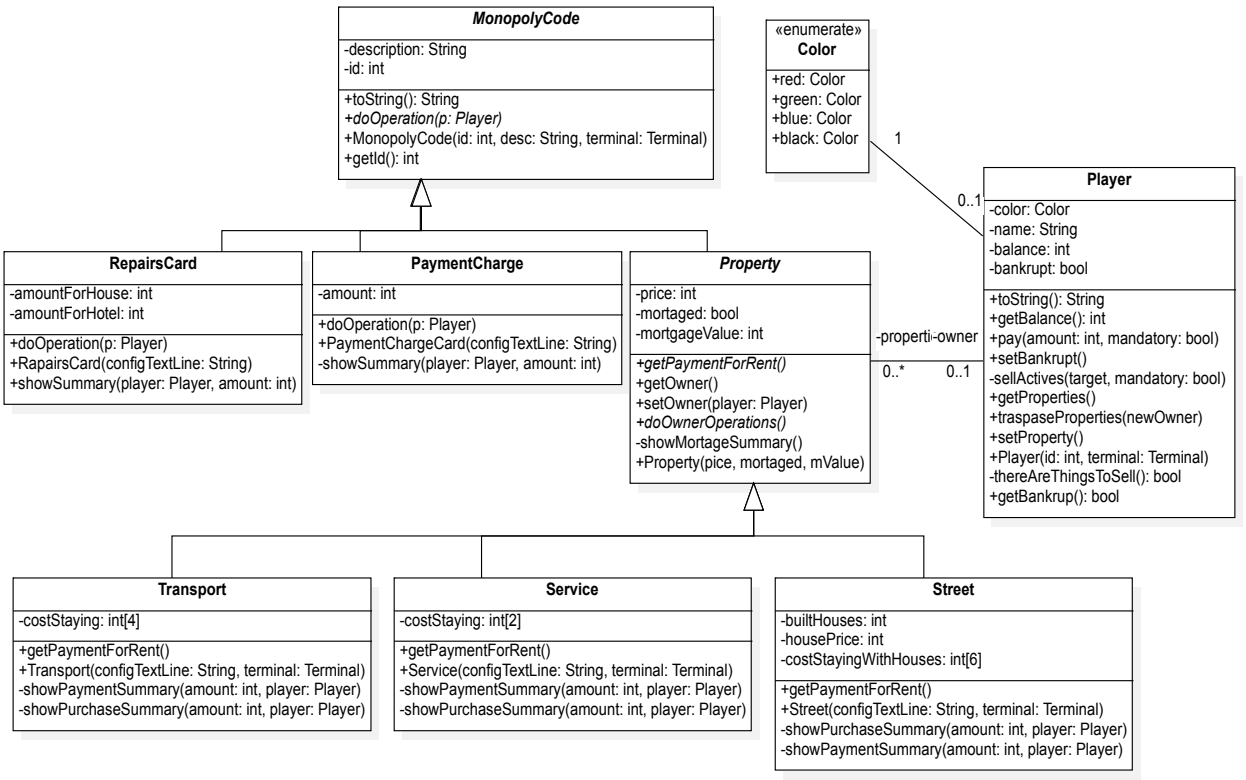

Figura 3.- Diagrama de clases que refleja el almacenamiento de la información sobre las diferentes tarjetas y cartas del juego del Monopoly.

Finalmente, la Figura 4 presenta las clases relacionadas con la entrada salida y la traducción.

- `Translator`.- Encargada de traducir las cadenas desde el idioma en el que estén escritas las cadenas en el código a los diferentes idiomas requeridos.
- `TranslatorManager`.- Encargada de gestionar los diferentes traductores del juego.
- `Terminal`.- Define las operaciones que debe tener cualquier sistema que sirva para gestionar la comuniación del programa con los usuarios. Esta clase abstracta deja abierta la

puerta a implementar la comunicación con el programa de diferentes formas. Por ejemplo con un terminal de texto, o con una pistola lectora de códigos de barras y una pantlla.

- `TextTerminal`.- Implementa una terminal basada en texto y la consola de ordenador.

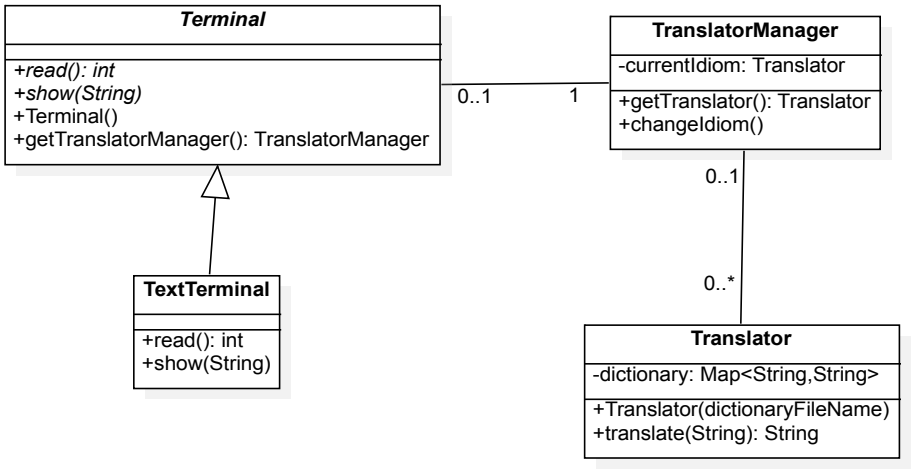

Figura 4.- Diagrama de clases principal.

3 Operativa del juego

Partiendo de estas clases, el diagrama de Actividad de la Figura 1 quedaría segmentada en las clases y métodos que se ven el diagrama de la Figura 5.

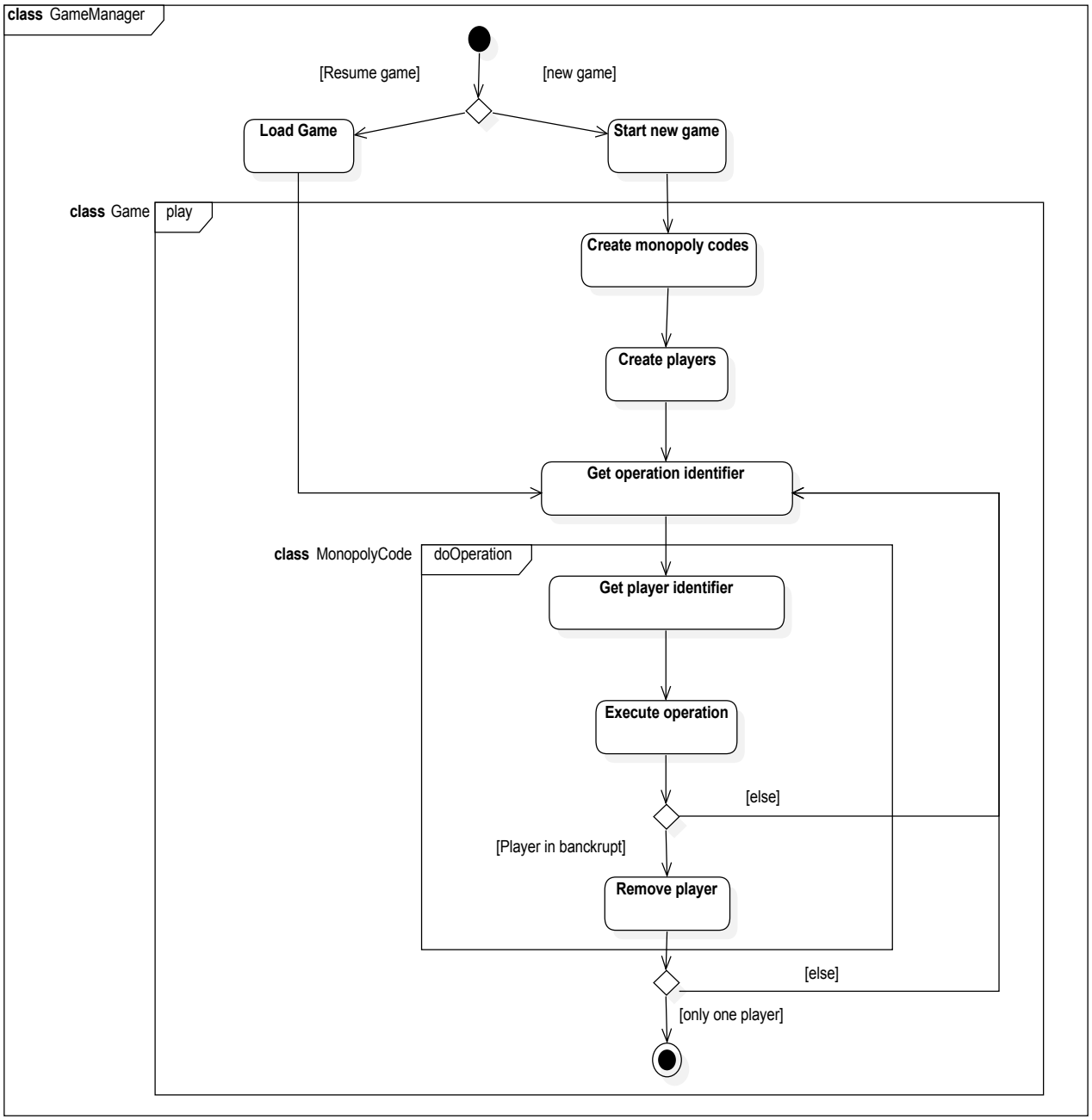

Figura 5.- Diagrama de actividad con la lógica de alto nivel asociada a la práctica repartida en las clases que lo implementarían.

3.1 Inicio de la aplicación

El diagrama de secuencia de la Figura 6 ilustra el inicio de la aplicación.

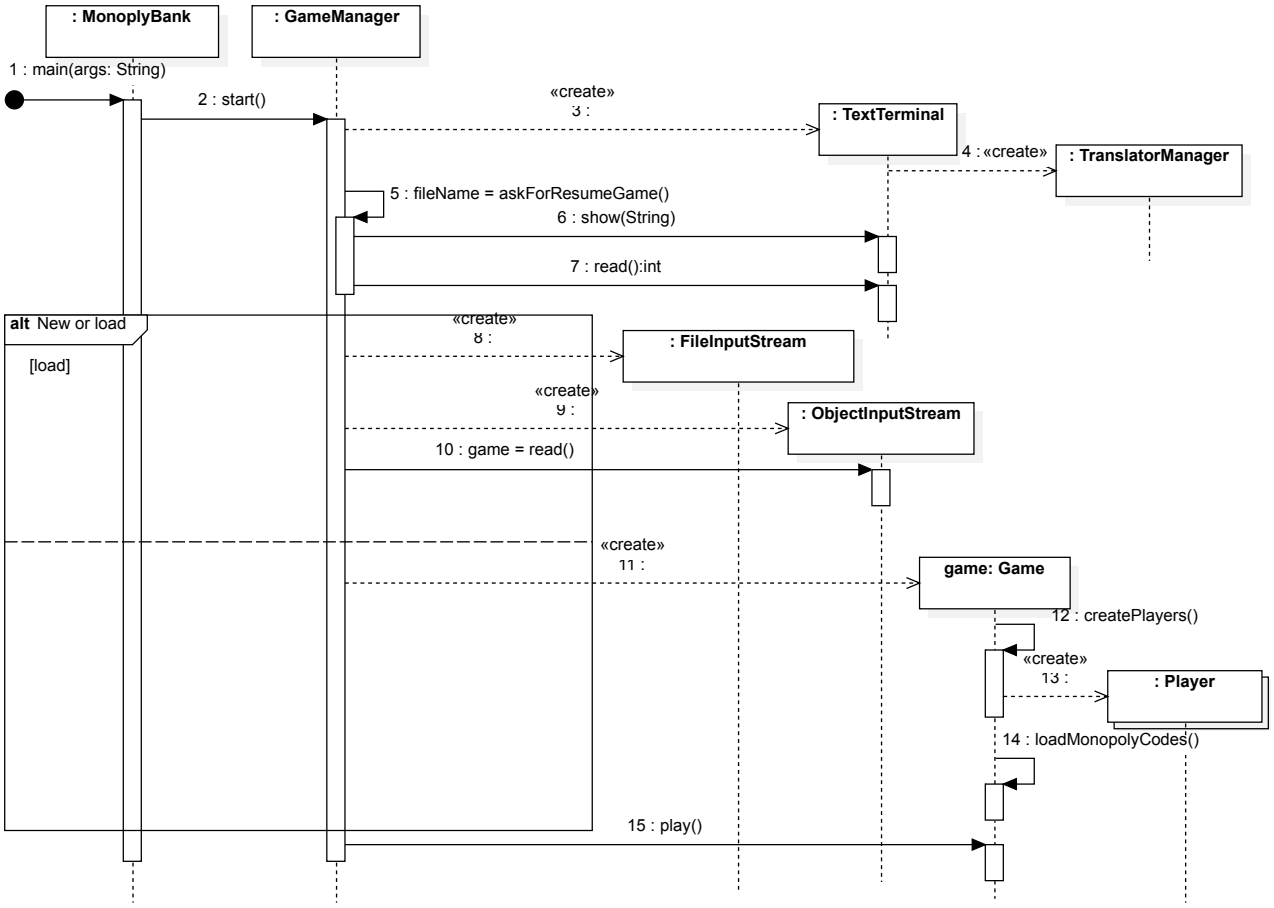

Figura 6.- Diagrama de secuencia que refleja el inicio del programa.

En el inicio de la aplicación es muy importante el fichero de configuración de propiedades, tarjetas y casillas especiales. Este fichero, que se suministra con la práctica con el nombre `MonopolyCode.txt`, sigue el siguiente formato:

`<id>;<clase>;<texto>`

En el caso de las tarjetas de suerte, comunidad y casillas especiales, la clase corresponde al texto `PAYMENT_CHARGE`. Luego, el texto puede contener una cantidad en euros positiva o negativa que corresponde a la cantidad a cobrar o pagar.

En el caso de las propiedades, la clase puede tomar diferentes valores: `TRANSPORT, SERVICE, STREET`. Mientras que el texto se compone de diferentes valores enteros separados por el símbolo de punto y coma.

La Figura 6 se complementa con el diagrama de secuencia de la Figura 7 que refleja la carga de la información, desde el fichero de configuración, de las tarjetas de suerte, las casillas especiales y las propiedades.

En este caso, debe observarse que todas las clases que derivan `MonopolyCode` tienen un constructor que recibe como parámetro cada cadena de configuración que se lea del fichero de configuración.

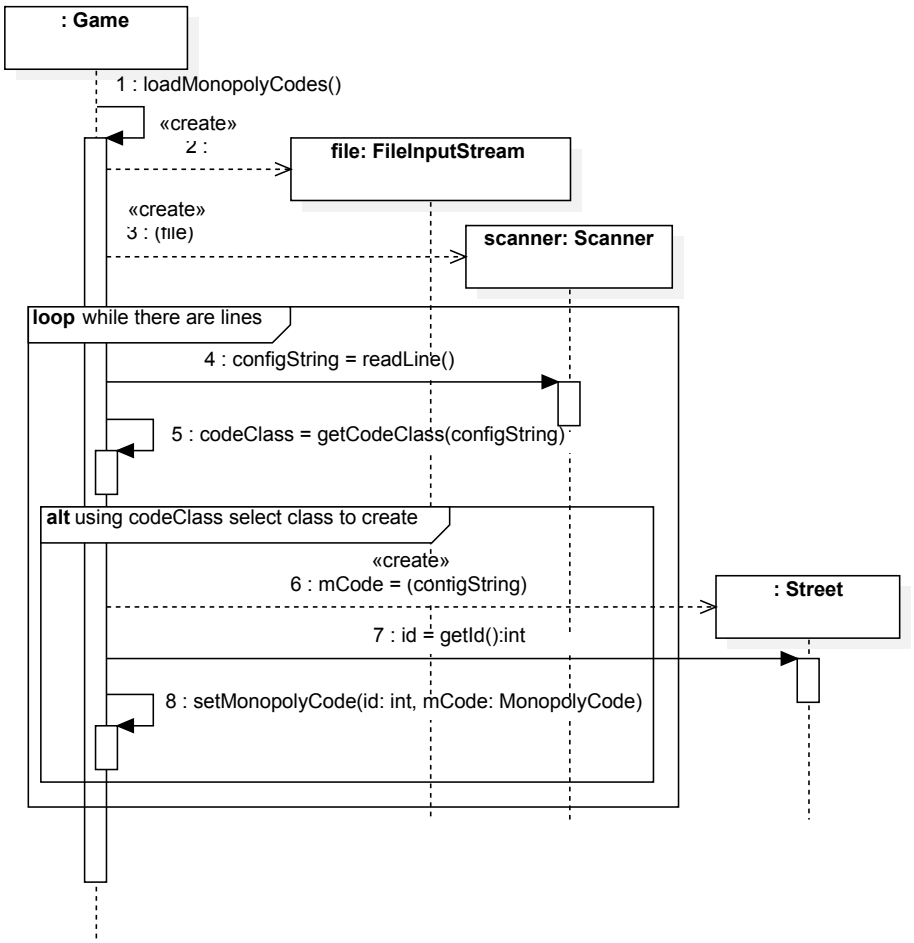

Figura 7.- Diagrama de secuencia que refleja la carga de los códigos y configuración de tarjetas y propiedades.

3.2 Operativa del traductor

El diagrama de la Figura 8 muestra el funcionamiento del sistema de traducción. Obsérvese que, previamente se ha inicializado el `TranslatorManager`, el cual habrá cargado todos los diccionarios, y se habrá seleccionado el idioma para la partida.

Por cada idioma deberá existir un fichero y un traductor. Cada fichero de diccionario consta de todas las frases que pueden aparecer en el programa en español y en otro idioma.

La única clase que habla con el traductor es el terminal. En realidad

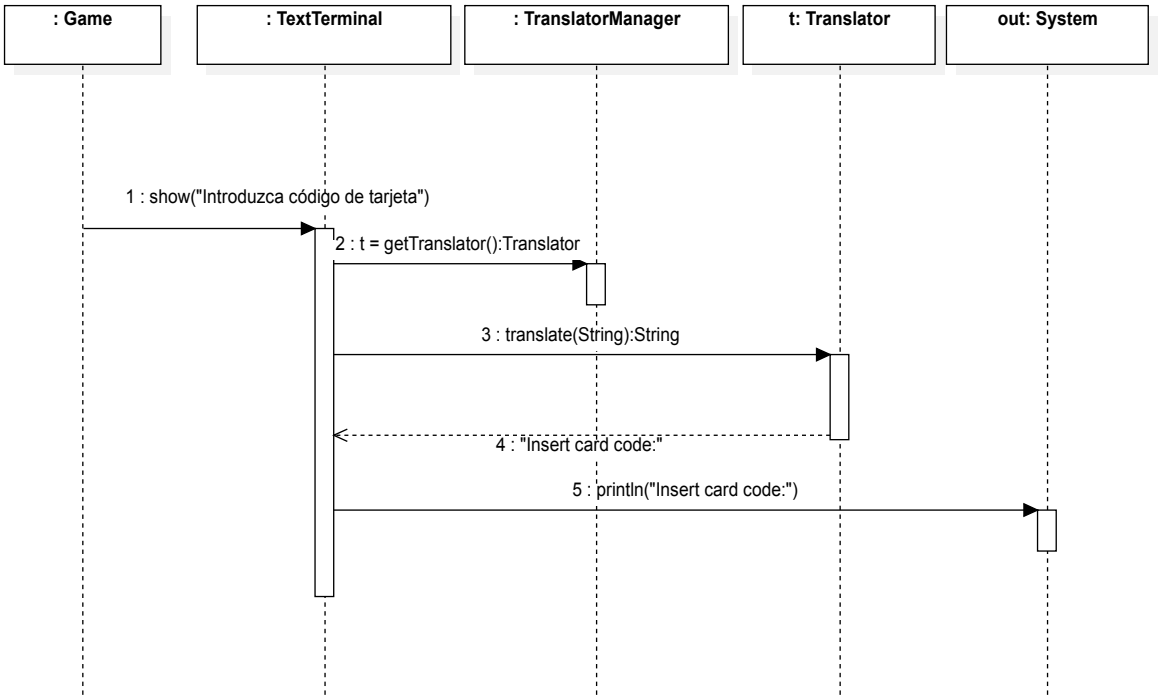

Figura 8.- Diagrama de secuencia que refleja la traducción de una frase.

3.3 Operativa del bucle principal del programa

El diagrama de la Figura 9 ilustra las operaciones que ocurren dentro del bucle principal del diagrama que se ejecuta en el método `play` de `Game`. Para no complicar aún más el diagrama, en esta figura no se profundiza en la llamada a `doOperation` de los objetos derivados de la clase `MonopolyCode`.

Básicamente, la llamada a `doOperation` puede desencadenar operaciones de compra (de propiedades, casas, quitar hipoteca), de recibir dinero (por tarjetas de suerte, o casillas especiales del tablero como la de salida) o de pago (por caer en casillas del tablero, por caer en propiedades de otros o por mensajes en las tarjetas de suerte). En las siguientes secciones se analizan algunos de estos casos.

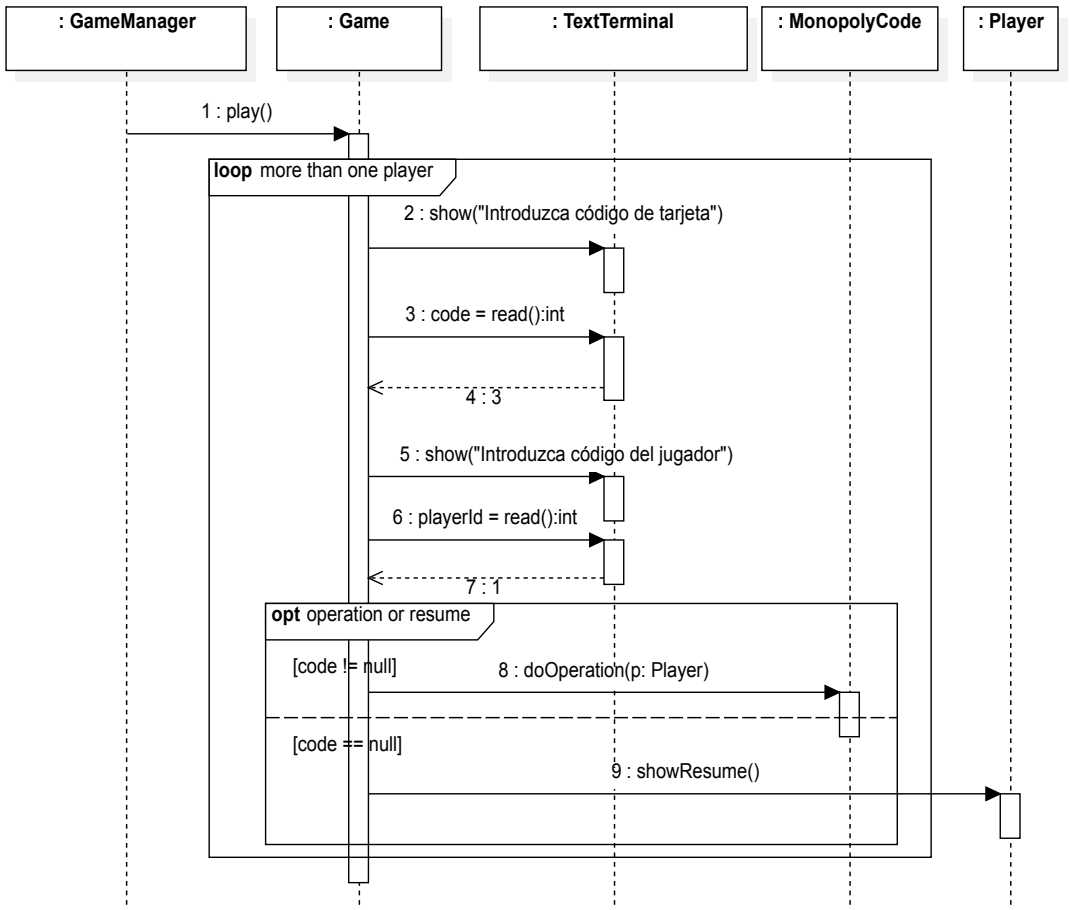

Figura 9.- Diagrama de secuencia que refleja el bucle principal del diagrama que se ejecuta en el método play de Game.

3.4 doOperation sobre una propiedad

La Figura 10 muestra la operativa que se desencadena en el `doOperation` de una propiedad. Obsérvese que dicho `doOperation` recibe el código de un jugador, mientras que la información de la propiedad ya la conoce.

A continuación, el diagrama de secuencia de la Figura 11 refleja las principales acciones que ocurren cuando se llama a `doOperation` de una propiedad de tipo Calle. Este diagrama no profundiza en lo que ocurre cuando se llama al método `getPaymentForRent()` ni tampoco en lo que ocurre dentro del método `pay()` de `Player`.

Lo que ocurre en el método `getPaymentForRent()` se describe en la Figura 12. En este caso se utiliza como ejemplo la caída en una propiedad de tipo `Service`. El caso de caída en una propiedad de tipo `Transport` o de tipo `Street` sería similar.

Por otro lado, el diagrama de la Figura 13 profundiza en lo que ocurre al llamar al método `pay` de `Player`. En la Figura 13 el parámetro `mandatory` tiene el objetivo de obligar a la venta porque el jugador ha caído en una casilla en la que hay que pagar. Si el parámetro `mandatory` es false y el jugador no tiene dinero la operación de pago no se realiza, y el jugador tendría que vender

propiedades antes de reintentarla. Este diagrama de la Figura 13 se complementa con el de la Figura 14 que explica lo que hay que hacer cuando el `Player` no tiene suficiente dinero para afrontar el pago y tiene que vender casas o hipotecar propiedades.

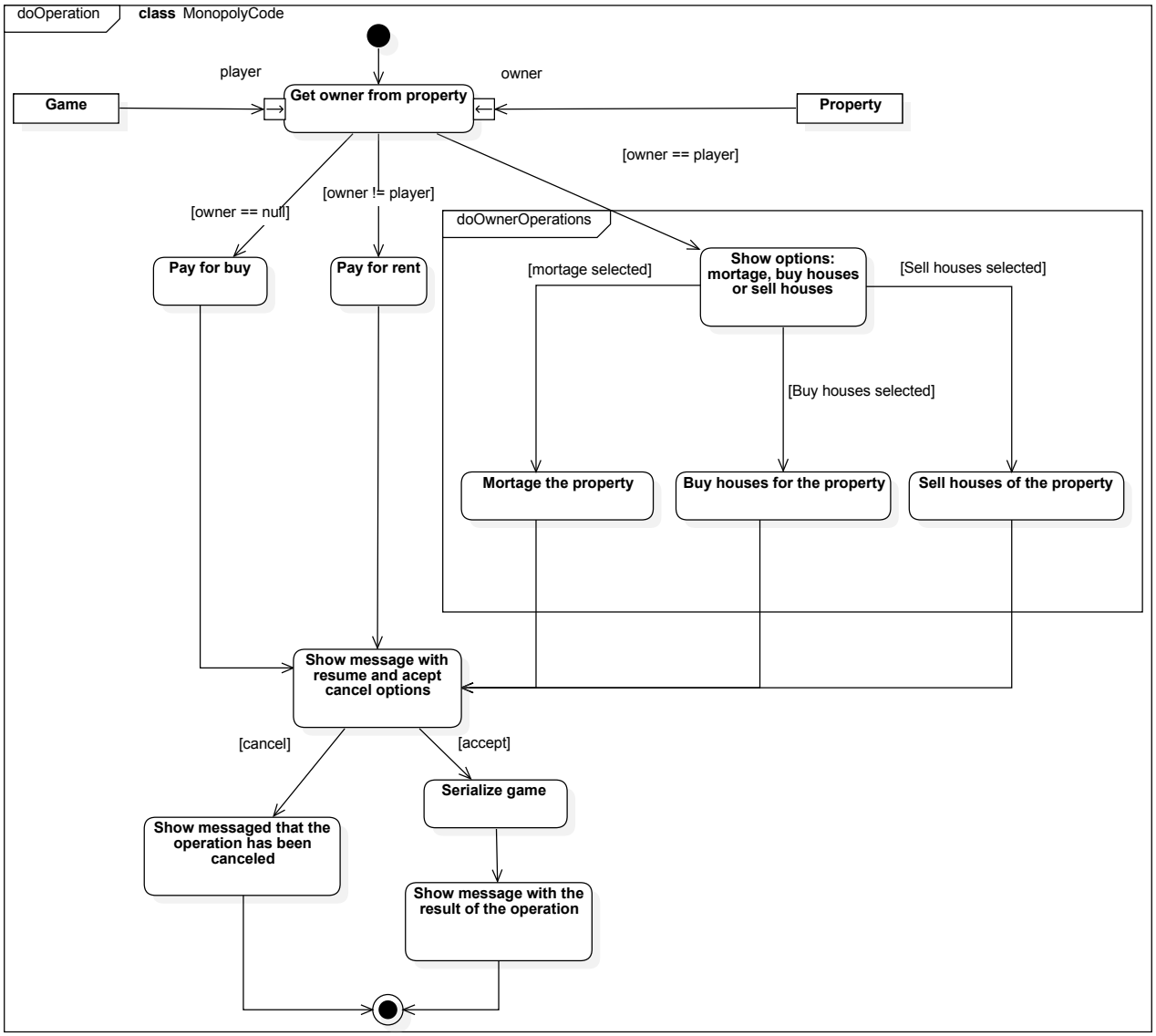

Figura 10.- Diagrama de secuencia que refleja lo que ocurre en doOperation de las propiedades.

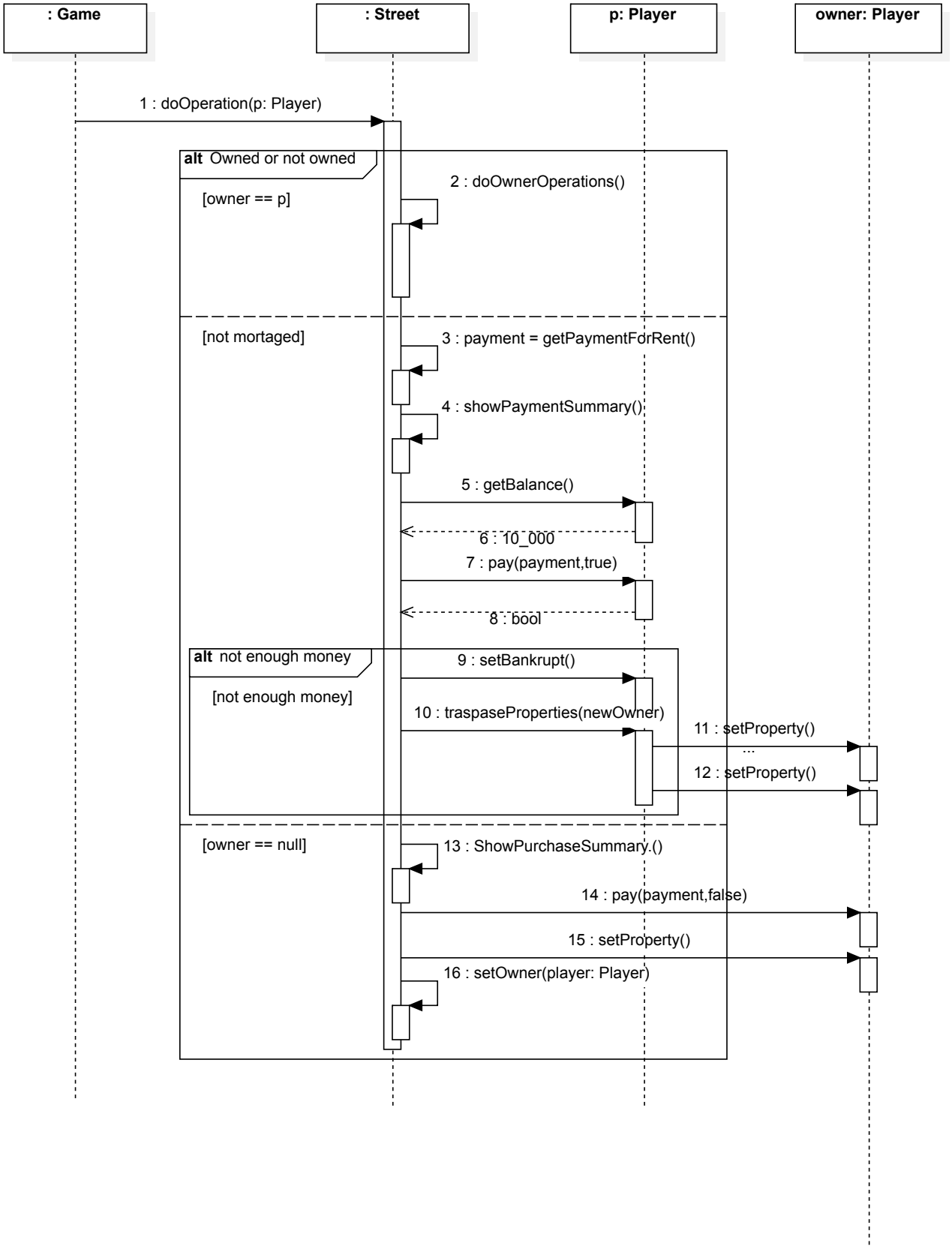

Figura 11.- Diagrama de secuencia que refleja las principales acciones que ocurren cuando se llama a doOperation sobre una propiedad de tipo Calle.

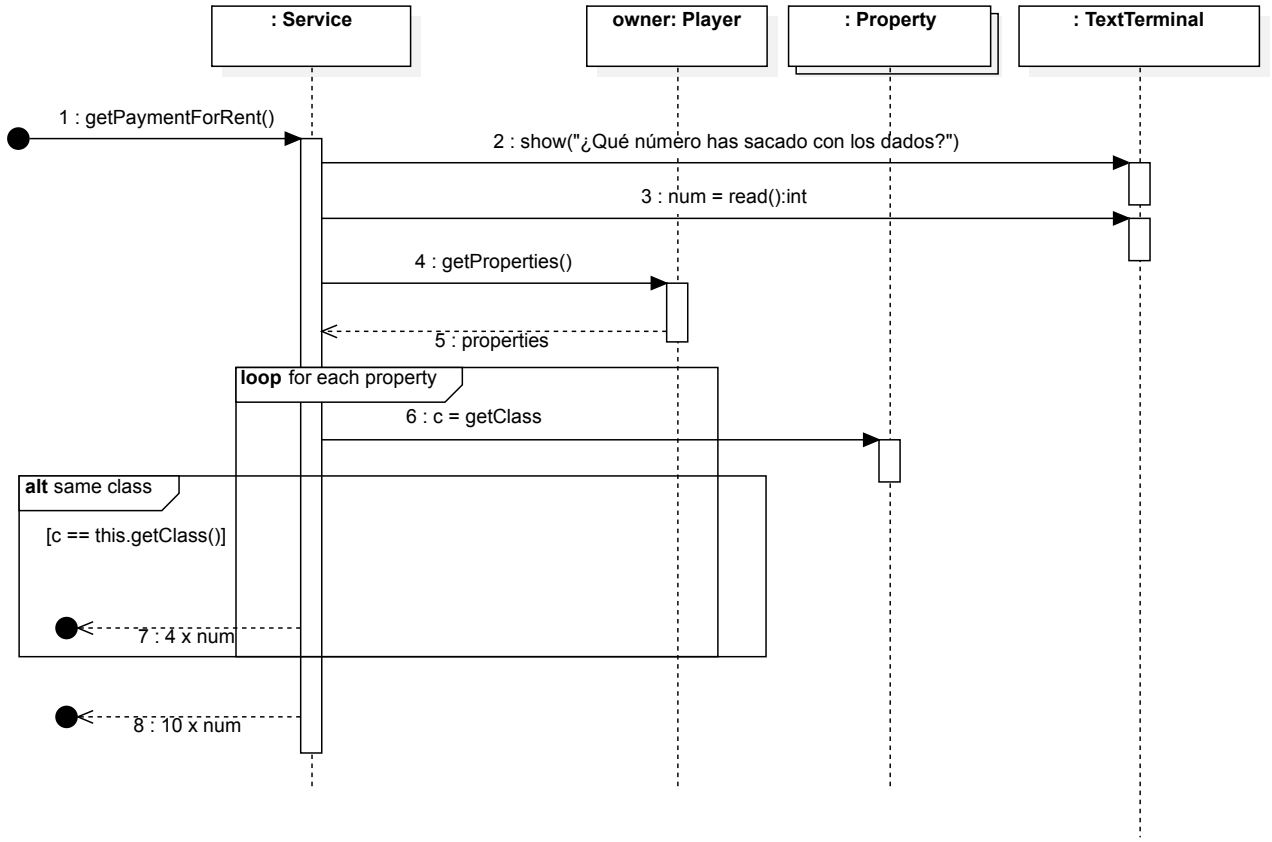

Figura 12.- Diagrama de secuencia que refleja el cálculo de lo que hay que pagar por caer en una propiedad de tipo servicio.

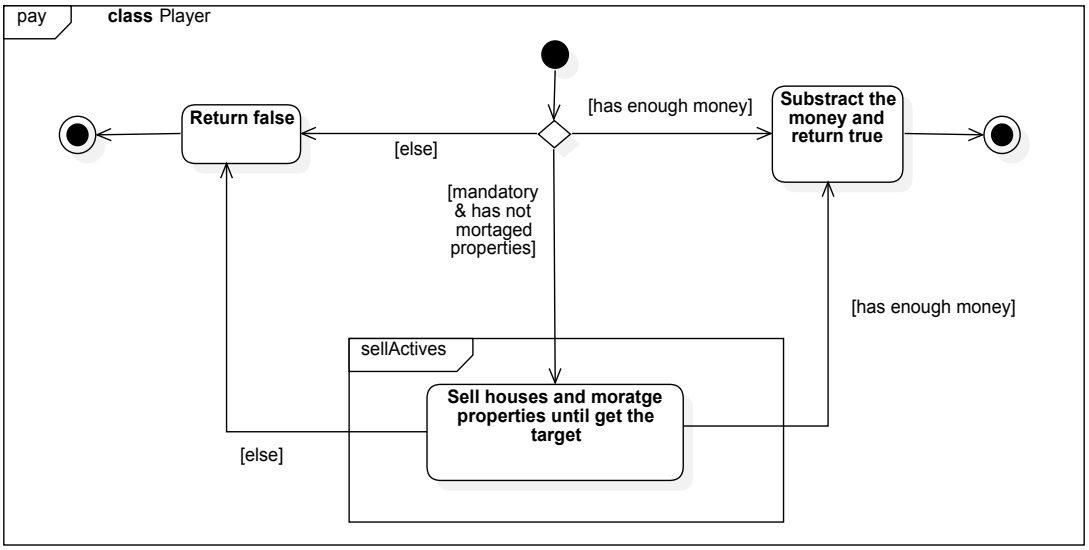

Figura 13.- Diagrama de actividad que refleja lo que ocurre dentro del método pay de la clase Player.

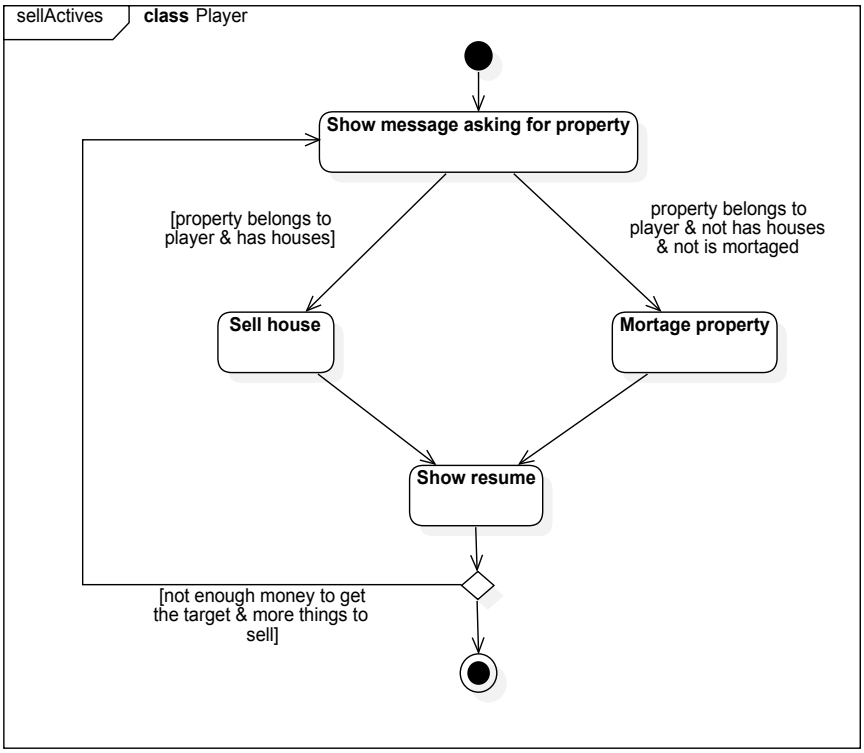

Figura 14.- Diagrama de actividad que refleja lo que ocurre dentro del método sellActives de la clase Player.

4 Conclusiones

Este documento presenta un diseño orientativo de la práctica del `MonopolyBank`. Para obtener una aplicación funcionando, hay muchos aspectos necesarios que quedan fuera del alcance de este documento. Dichos aspectos los debe resolver el estudiante en la implementación. Estos aspectos pueden incluir la creación de nuevos métodos e incluso clases, aunque se deben respetar las líneas maestras marcadas en este documento.