

# Estanciero

Jacobo y su hermano Samuel son unos enfermitos del estanciero y para mejorar sus tácticas y estudiar nuevas estrategias deciden modelar su juego preferido en Smalltalk (que capos!).

De cada **jugador nos va a interesar el dinero** que posee (por el hecho de cobrar y pagar) **y las propiedades** que tiene (por el hecho de comprarlas). Se dispone de un tablero con casilleros.

En su mayoría los casilleros son propiedades, donde **de cada propiedad se conoce su precio de compra inicial y su dueño**. Los otros casilleros son premios y el casillero de salida, explicados más adelante.

## Caída en una propiedad

Cuando un jugador J cae en una propiedad puede ocurrir alguna de estas tres cosas

- Que la propiedad no pertenezca a nadie (es decir, **que el dueño sea el Banco**):  
En este caso el jugador que cayó ahí compra la propiedad pagándole al Banco
- Que el **dueño** de la propiedad sea un **jugador rival** (esto es  $J \neq \text{dueño}$ )  
El jugador debe abonarle al dueño el valor de la renta de esa propiedad.
- Que el **dueño** sea uno mismo (esto es  $J = \text{dueño}$ )  
No pasa nada

## Tipos de Propiedades

Hay 2 tipos de propiedades

- Campos
- Empresas

### I. Campos

Algunas propiedades son "Campos" y están agrupadas en grupos llamados "Provincias".

**Una provincia conoce los campos que tiene y cada campo conoce a que provincia pertenece.**

De cada campo se **conoce su valor de renta fijo** (o sea, el valor de la renta sin estancias) y también **conoce el costo de construcción de cada estancia** (en ese campo).

Además, cada campo sabe decirnos el valor de renta para los casos en que la propiedad tenga construida estancias (**cada campo conoce la cantidad de estancias que tiene construidas en él**) que se calcula con la siguiente fórmula:

$$\text{Renta para } N \text{ estancias} = 2^N * \text{valor de renta fijo}$$

Recordamos del ingreso que  $2^0 = 1$

Ejemplo, la renta del campo "Campo Bravo" que tiene un valor de renta fijo de \$3000 y tiene contruidas 3 estancias va a ser \$24.000 porque  $2^3 * 3000 = 24000$

**Construcción de estancias:** Un jugador puede construir una estancia en un campo si

- tiene toda la provincia (es el dueño de todos los campos de ella) y además
- si hace una construcción de forma pareja. Por ejemplo, no puedo construir 3 estancias en un campo y ninguna en otro de la misma provincia. La máxima diferencia aceptada es de 1 estancia

### II. Empresas

Otras propiedades son empresas, son tres en todo el juego que conviene acumularlas ya que la renta aumenta al tener mayor cantidad de las mismas.

Si un jugador cae en una empresa ese jugador debe

1. tirar los dados nuevamente (asumamos que saco X)
2. pagarle al dueño de dicha empresa un monto igual al resultado de hacer la cuenta:

$$X * \$30.000 * \text{cantidad de empresas que tiene el dueño}$$

*Tienen que hacer que las propiedades entiendan el mensaje #sosEmpresa (todas devuelven false excepto las empresas)*

**NOTA IMPORTANTE:** Está prohibido usar ifTrue:/ifFalse: y similares con mensajes como #sosEmpresa, #sosCampo, #sosBanco, #sosCampoLoco o similares (ejemplo, self tipo = 'empresa') – Única excepción **punto 1.a**



## Parcial de Objetos (modificado)

### Paradigmas de Programación

Ya se cuenta con las clases indicadas

Asumimos que todas las propiedades al momento de ser creadas tienen como **dueño** al Banco. Ustedes no tienen que crear los casilleros asuman que el tablero ya está completa y correctamente armado.  
Los métodos que están en el diagrama ya están implementados y hacen lo que su nombre indica  
(los pueden usar si lo creen necesario y también pueden agregar otros)

```
ParDeDados class >> valorDados  
  ^ (1 to: 6) atRandom + (1 to: 6) atRandom
```

```
Juego >> empezar  
  [ self estaTerminado ]  
  whileFalse:  
    [ jugadores do:  
      [ :j | self queJuegue: j ] ]
```

El método Tablero >> casillerosDesde: casilleroInicial hasta: unNumero

- **Recibe** un casillero (un objeto instancia de alguna clase creada por vos) y un número
- **Devuelve** una colección de casilleros (la colección es instancia de OrderedCollection y es un subconjunto de los casilleros que conoce el tablero) donde el primer elemento es el casillero siguiente a casilleroInicial, el segundo es el siguiente al siguiente a casilleroInicial y así; siendo el último elemento de la colección el casillero que está a unNumero de posiciones de casilleroInicial

### Requerimientos

1.

- Conocer la cantidad de empresas que tiene un jugador (solo en este punto se puede usar #sosEmpresa).

Saber la **renta** que tiene que pagar un **jugador** con respecto a una **propiedad**

- Para los **campos** tener en cuenta la cantidad de estancias que hay construidas
- Para las **empresas** tener en cuenta
  - la cantidad que saco en los dados (recuerden que el **jugador** tira los dados nuevamente)
  - la cantidad de empresas que tiene el dueño

2.

- Hacer que un jugador pague una suma de dinero a un acreedor; si no tiene dinero suficiente se debe lanzar un error. Cuando un jugador paga a otro, su dinero disminuye y el del acreedor aumenta (*tirar un error significa enviarle el mensaje #error: al objeto receptor del mensaje*)
- Hacer que cualquier casillero entienda el mensaje `pasó: unJugador`
  - Por defecto cuando un jugador pasa por un **casillero no sucede nada**
  - Si un jugador pasa por la **Salida cobra \$5000**<sup>(\*)</sup>
- Saber para una provincia qué jugadores tienen campos en ella (se espera como respuesta una colección de "dueños" sin repetidos)

3. Hacer que cualquier casillero entienda el mensaje `cayó: unJugador`

- Cuando un jugador cae en una propiedad lo que pasa depende del dueño, si es el **banco** le indica al jugador que compre la propiedad, si es un **jugador rival** éste le dice al que cayó que pague la renta de la propiedad
- Cuando cae en **Premio Ganadero** el jugador cobra \$2500<sup>(\*)</sup>
- Cuando cae en **Salida** no pasa nada

4.

- Hacer que un jugador se **mueva sobre** una colección de casilleros que le llega por parámetro. Esto hace que:
  - El jugador **pase por todos** los casilleros de esa colección
  - El jugador **caiga en el último** casillero de esa colección
  - El **casillero actual** del jugador sea el **último casillero** de esa colección
- Hacer el método `Juego >> queJuegue: unJugador` en donde unJugador tiene que
  - tirar los dados (digamos que sacó X) y,
  - moverse sobre los casilleros correspondientes(OJO! primero tienen que saber en donde está el jugador para que se mueva sobre los casilleros desde ahí hasta X)

5. Construir **una estancia en un campo** validando que se pueda construir en la provincia de dicho campo (si no se puede no se hace nada). Ver **Construcción de estancias** en la página 1.  
No olvidarse de que el jugador pague lo correspondiente por dicha estancia (solo nos interesa que decremente su dinero no aumenta el de nadie).

6. Hay unos **"campos locos"** en los que la **renta** es 20% más que en un campo normal, y que cuando **pasa** un jugador (distinto del dueño) que tiene más de 5 empresas, el dueño le cobra \$500.

<sup>(\*)</sup> esta es plata espontánea, o sea, no se le decrementa a nadie

06/07/2010

Martes Mañana - 1er cuat.

