Ingeniería de Software
Trabajo Práctico 2

Documento de Diseño v1.1.0

Alumnos:
Blanco, Lucas
Murillo, Sebastian

Docente:
Miceli, Martin

FCEFyN – UNC
2019

Introducción:

El presente documento cubre el diseño del software. Se presentarán distintos diagramas mostrando diversas vistas.

Se utilizarán los patrones singleton, observer y strategy.

- Singleton:

  Se utilizará simplemente para correr una sola instancia del programa.

- Observer:

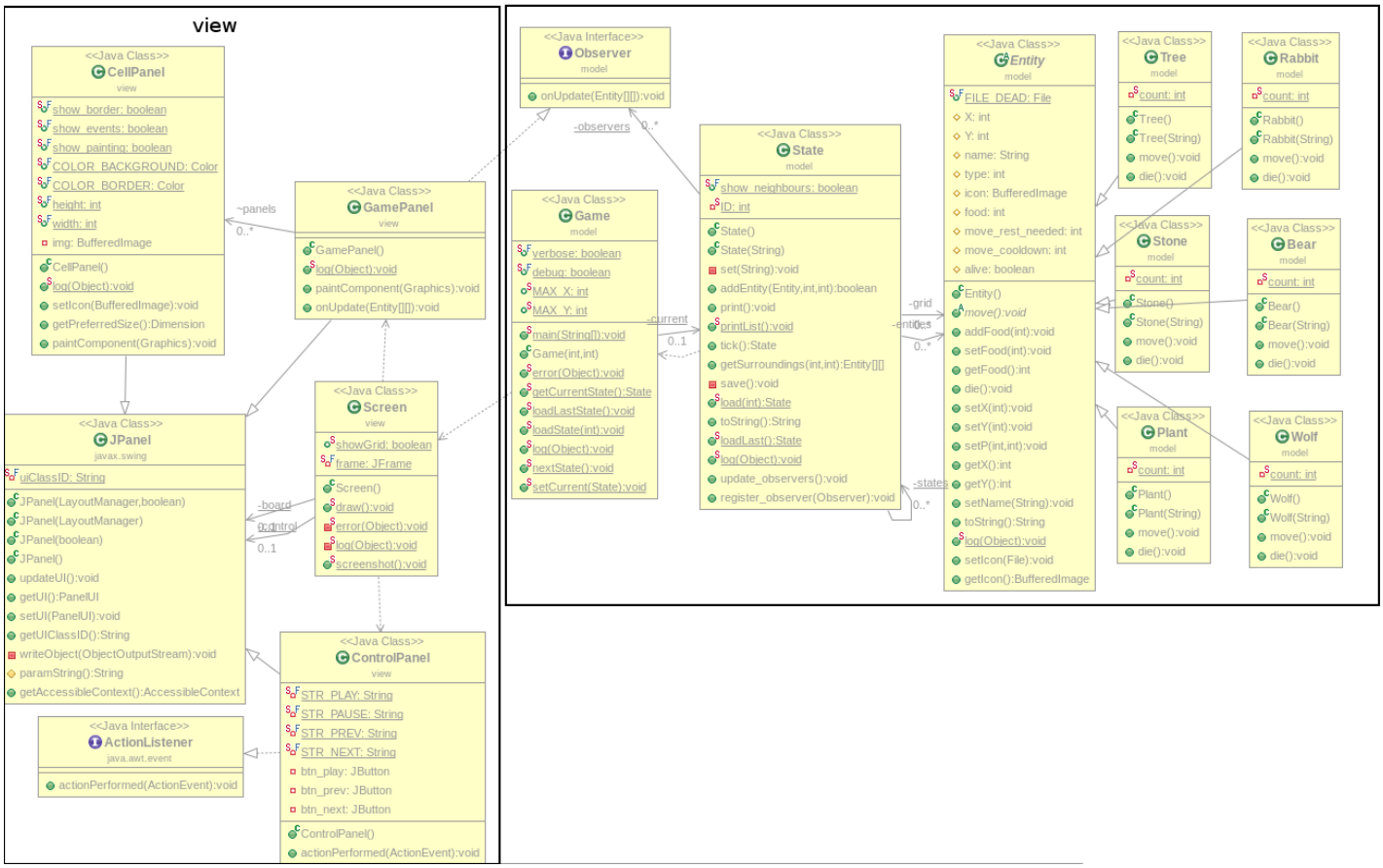
  Permitirá que distintas clases se suscriban a la actualización de los datos del modelo, evitando tener que modificar código cada vez que se quiera crear una nueva clase que deba estar al corriente de los datos.

- Strategy:

  Simplifica el mantenimiento y adicion de nuevas clases al codigo. Encapsula código propio de las clases dentro de las mismas, evitando el uso de condicionales.

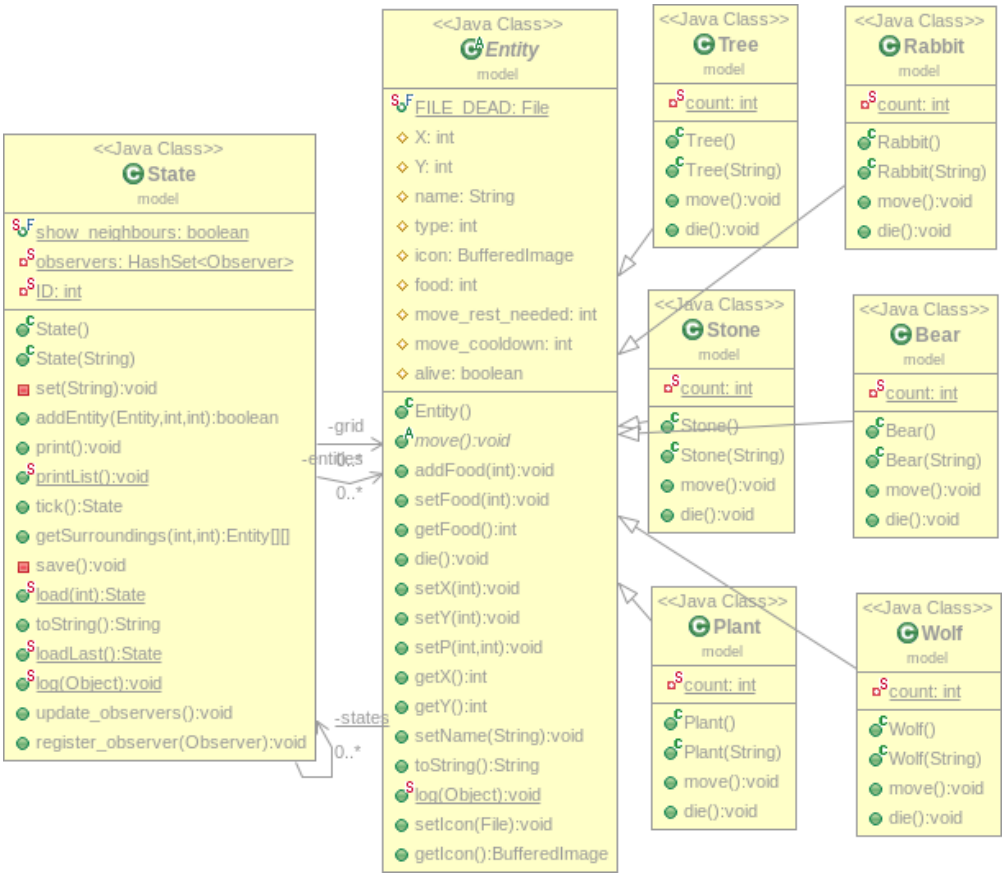
Diagramas de Clase:

Diagrama Completo:



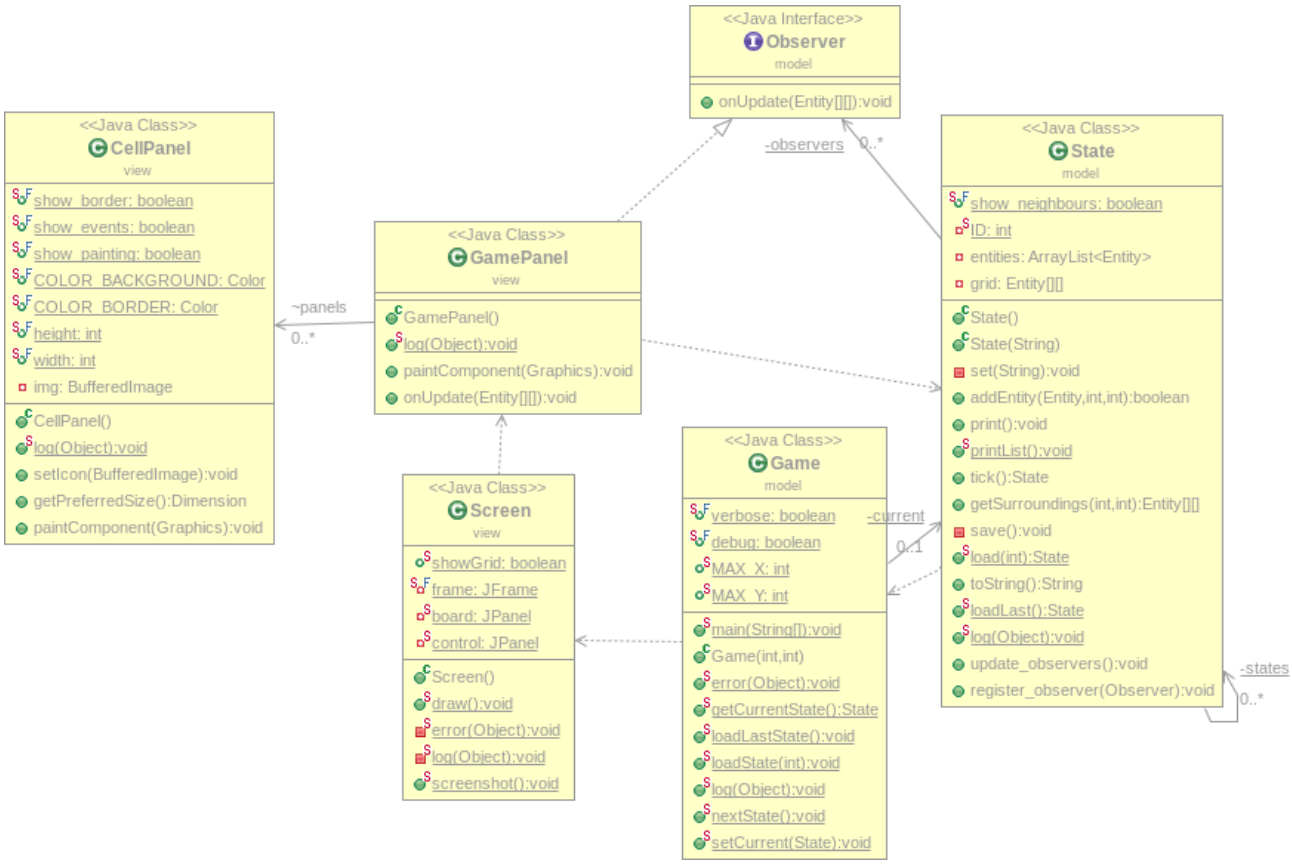
Muestra toda la estructura del software. Del lado izquierdo se encuentran las cases del paquete view, del derecho, las clases de model.

Diagrama del Patron Strategy:



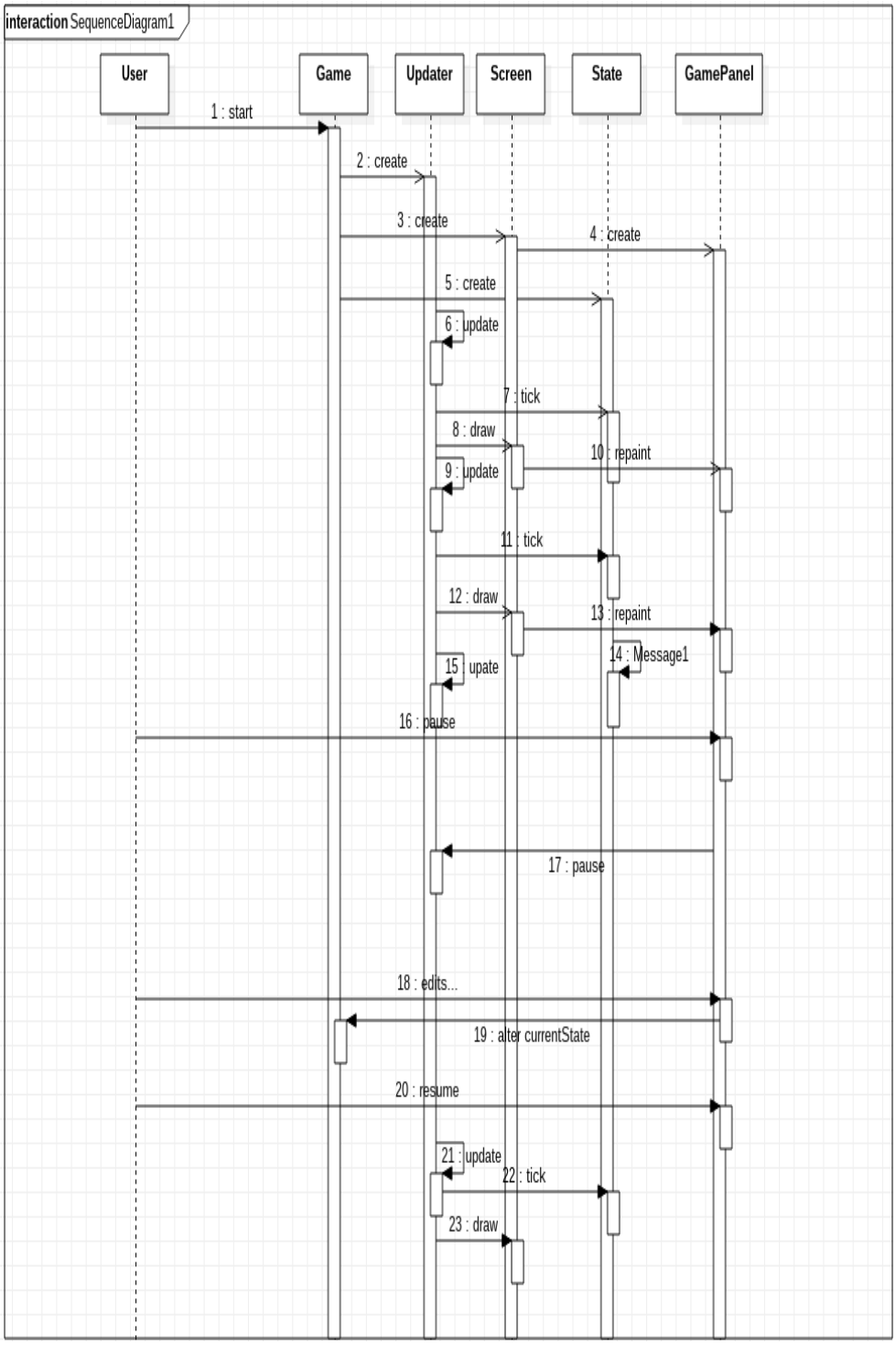
A cada actualizacion de ‘state’, este recorre una lista de ‘entity’ llamando al metodo ‘move’ de cada una. Cada tipo de entidad (conejo, planta, oso, etc) se mueve (o no) de distinta forma (implementado en cada una de estas clases).

Diagrama del Patrón Observer:



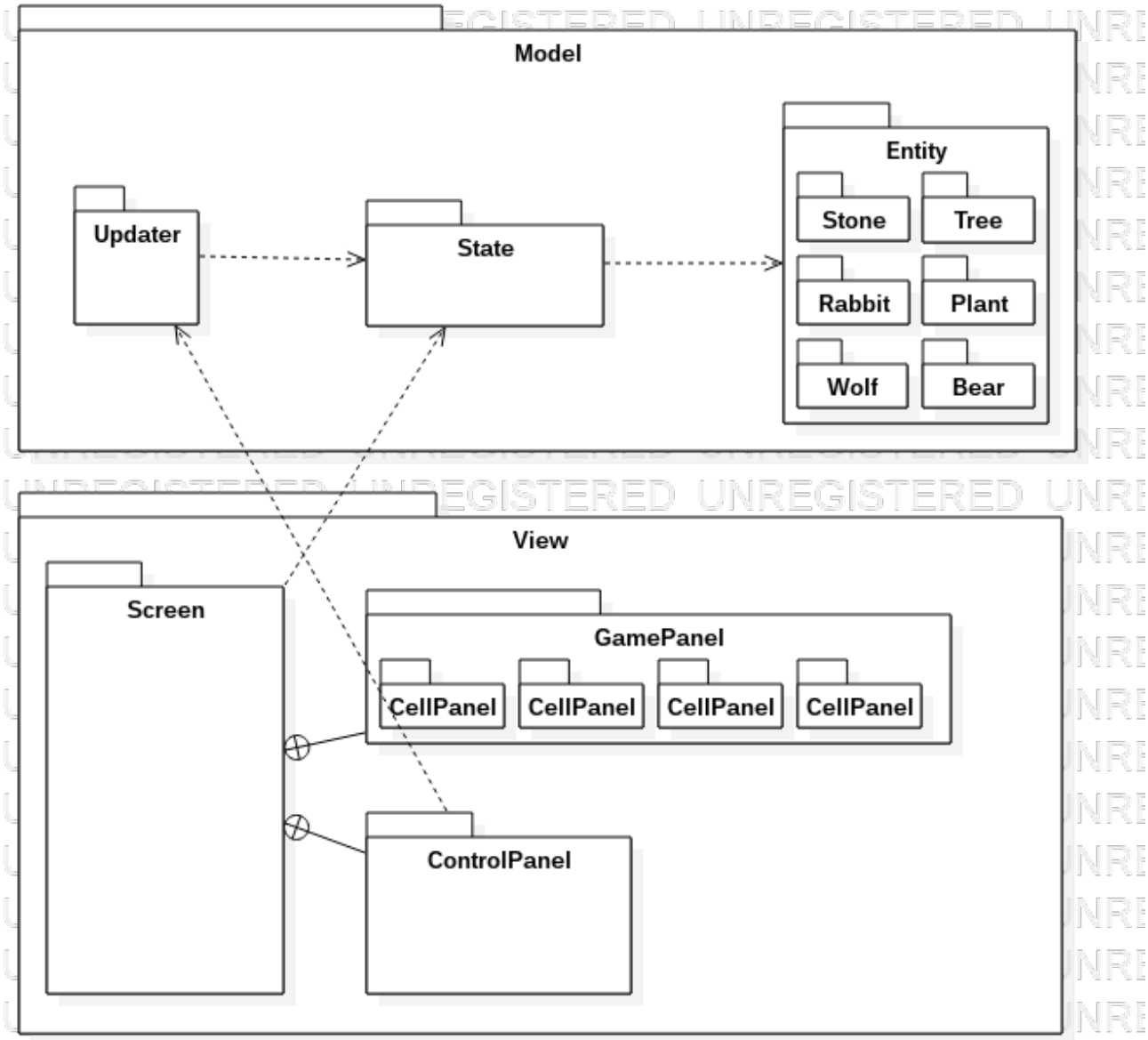
Al crearse, GamePanel se registra como observador de ‘state’, cuando ocurra un cambio en ‘state’, este llamará al metodo update_observers(), lo que cual hará que se actualice GamePanel (parte de ‘view’).

Diagrama de Secuencia:



En este diagrama se ejemplifica el uso del software. Un usuario inicia el programa, que automaticamente empieza a simular movimientos de fauna. Luego, el usuario pausa el programa, edita informacion sobre el estado de la simulacion, y resume su auto-actualizacion.

Diagrama de Paquetes:



En el diagrama se muestra la organización general de los paquetes de clases

Unit Tests:

Unit Tests se han pseudo-generado automaticamente para las clases del software utilizando eclipse IDE. Aun asi, deben de implementarse para asegurar el correcto funcionamiento del programa.

Los tests correrán automaticamente al subir código al repositorio, por medio del sistema de integracion continua (Travis CI, ver CMP).

Para correr estos tests, se deben incluir en el archivo '.travis.yml' en la sección de 'scripts'

Historial del Documento:

v1.0.0

- version inicial

v1.1.0

- añadidas dependencias en diagrama de paquetes
- borrado diagrama de clases, reemplazado por 3 diagramas de clases distintos
- añadido como correr test