Taller 5

Integrantes:
Nicolas Merchan: 202112109
Julian Rivera: 202013338
Juan Felipe Serrano: 201921654

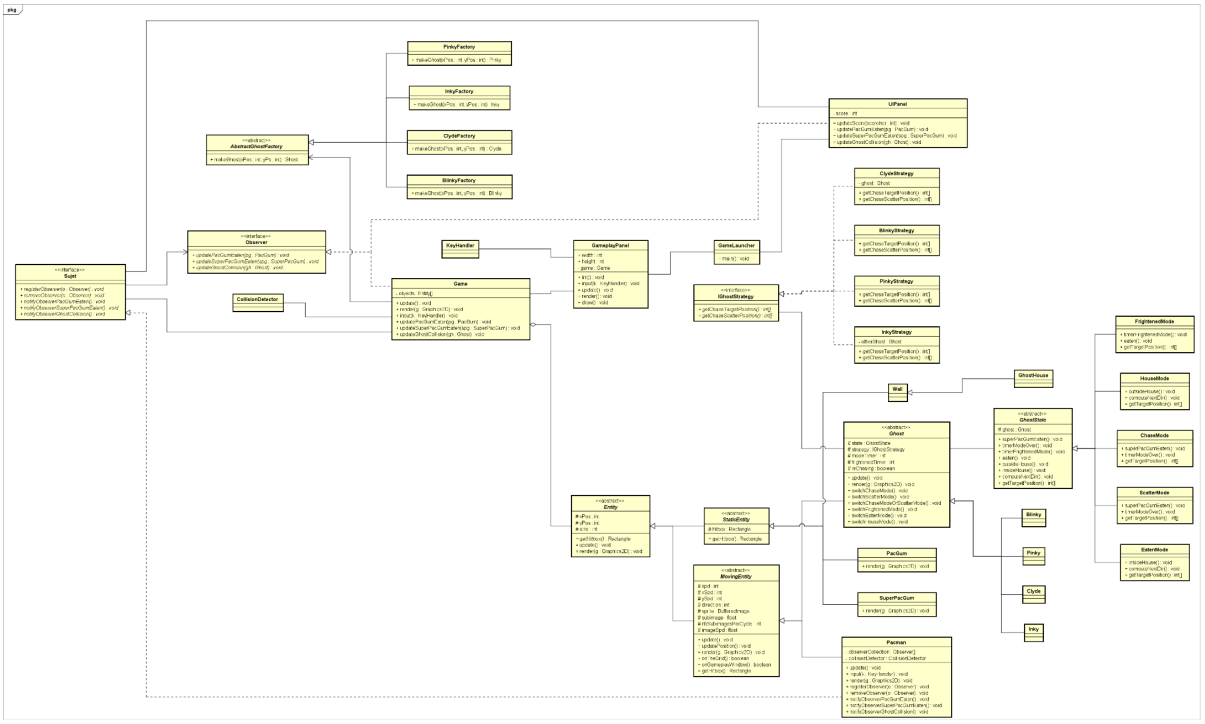
Deben entregar un documento que contenga al menos la siguiente información:

1. Información general del proyecto, deben incluir la URL para consultar el proyecto.

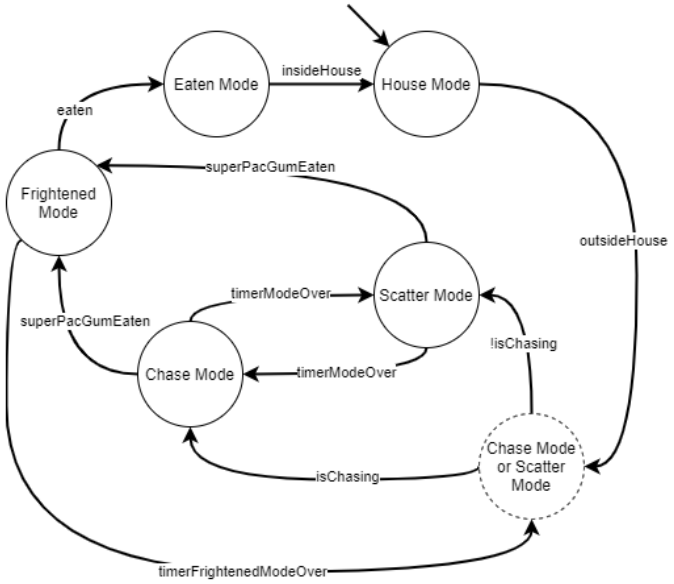
El proyecto que escogimos para analizar es una implementación del juego de PacMan, lo escogimos porque vimos que está muy bien documentado y utiliza claramente el patrón que buscábamos explicar: State.

Repositorio GITHUB a analizar: https://github.com/lucasvigier/pacman

Diagrama de Clases del proyecto:

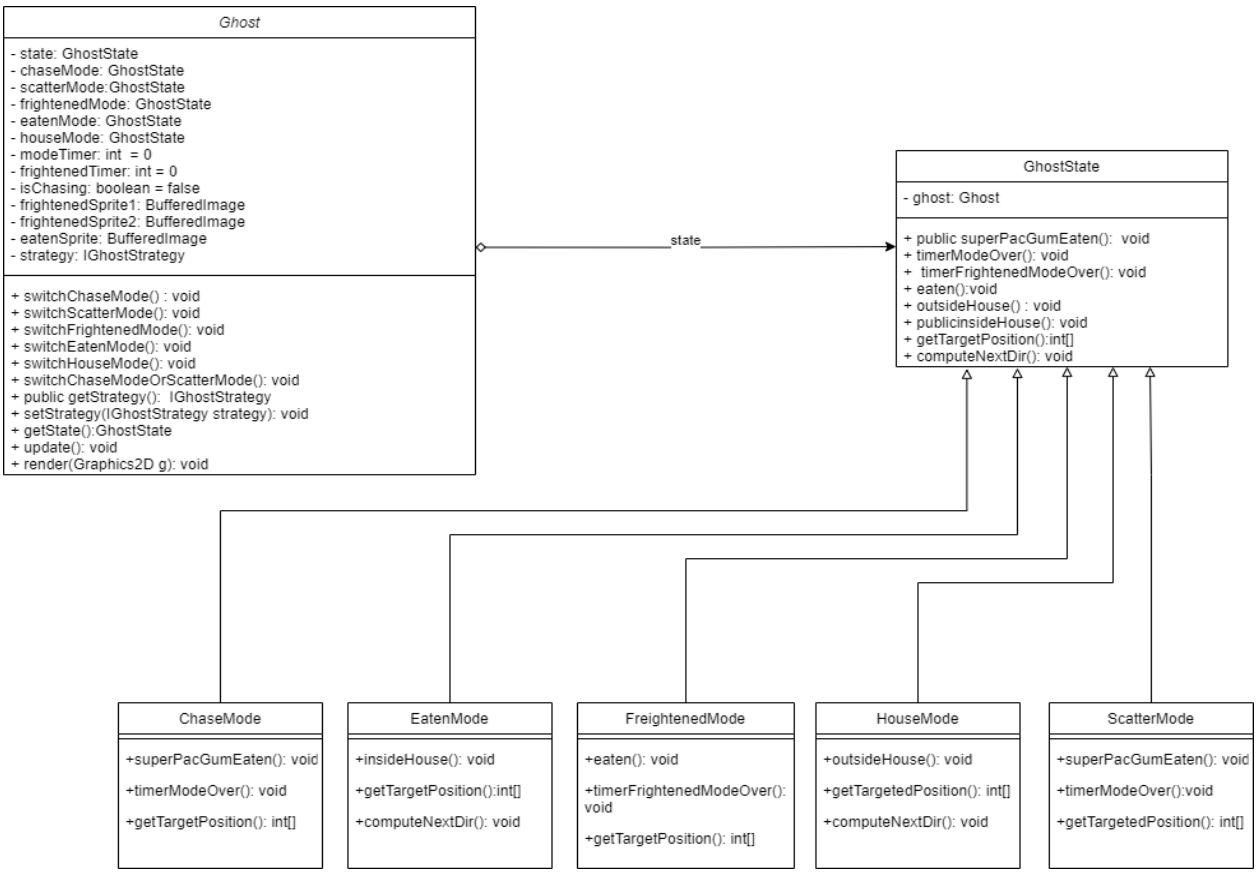


FlowChart de logica de estados de fantasmas del juego:



2. Información y estructura del fragmento del proyecto donde aparece el patrón. No se limiten únicamente a los elementos que hacen parte del patrón: para que tenga sentido su uso, probablemente van a tener que incluir elementos cercanos que sirvan para contextualizarlo.

La parte del proyecto que se identifico el patron de diseno de state es la parte de estados de los fantasmas, como su nombre indica estas clases estan reservadas para emular el estado de los fantasmas en un momento estecifico del juego. El uso del patron state en este lugar tiene sentido debido a como estan estricturadas las clases de estado, estas quieren represetnar un mismo tipo de objeto abstracto de las cuales todas heredan pero tienen una diferencia que las hace esencialmennte distintas, esta diferencia es tan notoria que crear una clase completa para representarlas a la hora de calcular el comportamiento de una fantasmas y su siguinte movimento tiene sentido e incluso fascilitan el manejo de los estados de los fantasmas. A continuacion se presenta un diagrama de clases de las conexiones habladas(lo hicimos nosotros porfa mirenlo):



3. Información general sobre el patrón: qué patrón es y para qué se usa usualmente
   Información del patrón aplicado al proyecto: explicar cómo se está utilizando el patrón dentro del proyecto

el patron de state es un patron de comportamiento y es una solucion al problema de finitos estados de un objeto, el objeto se comporta bastamente distinto en sus diferentes estados pero se quiere seguir manteninedo el mismo objeto o por lo menos su cadena de herencia, en estos casos es bueno usar el patron. En el proyecto como fue explicado en puntos anteriores se usa para modelar los comportamientos de los fantasmas, los fantasmas se comportan de diferentes maneras dependiendo del estado del juego por lo que usar el patron de state tiene sentido, los fantasmas tienen un objeto llamado state de clase GhostState que representa el estado del fantasma, dependiendo de que pasa este estado se combierte en ChaseMode,EatenMode, FreightenedMode, HouseMode o ScatterMode.

4. ¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene?

   Tiene sentido utilizar el patrón State en este punto del proyecto porque nos permite anidar los diferentes comportamientos que necesitamos de un fantasma en una sola clase abstracta, además nos da la ventaja de añadir un número n de comportamientos, tantos como se requiera, de manera sencilla, pues solo tenemos que añadir otra clase hija de GhostState.

5. ¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?

   Se pierde cohesión en el código, pues el mismo objetivo lo están realizando múltiples clases. Así mismo, en caso de tener múltiples subclases anidadas a la clase abstracta GhostState, si esta última se modifica, es extenso hacer los cambios en cada una de sus subclases.

6. ¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón?

   El patrón “State” busca solucionar problemas relacionados con el manejo de estados de objetos. En el caso del patrón de comportamiento en cuestión, se crean diferentes clases heredadas de la clase principal “GhostState” para representar los diferentes estados que puede tener el objeto “Ghost”. Una alternativa a dicha implementación puede ser la modificación y/o creación de métodos en las clases “MovingEntity” y “Ghost” y la creación de métodos en la “GhostState” para permitir la modificación directa de los atributos de dicha clase. Ello se llevaría a cabo con el fin de que los cambios de estado en “GhostState” se realicen sin el uso de las clases heredadas “FrightenedMode”, “HouseMode”, “ChaseMode”, “ScatterMode” y “EatenMode”.