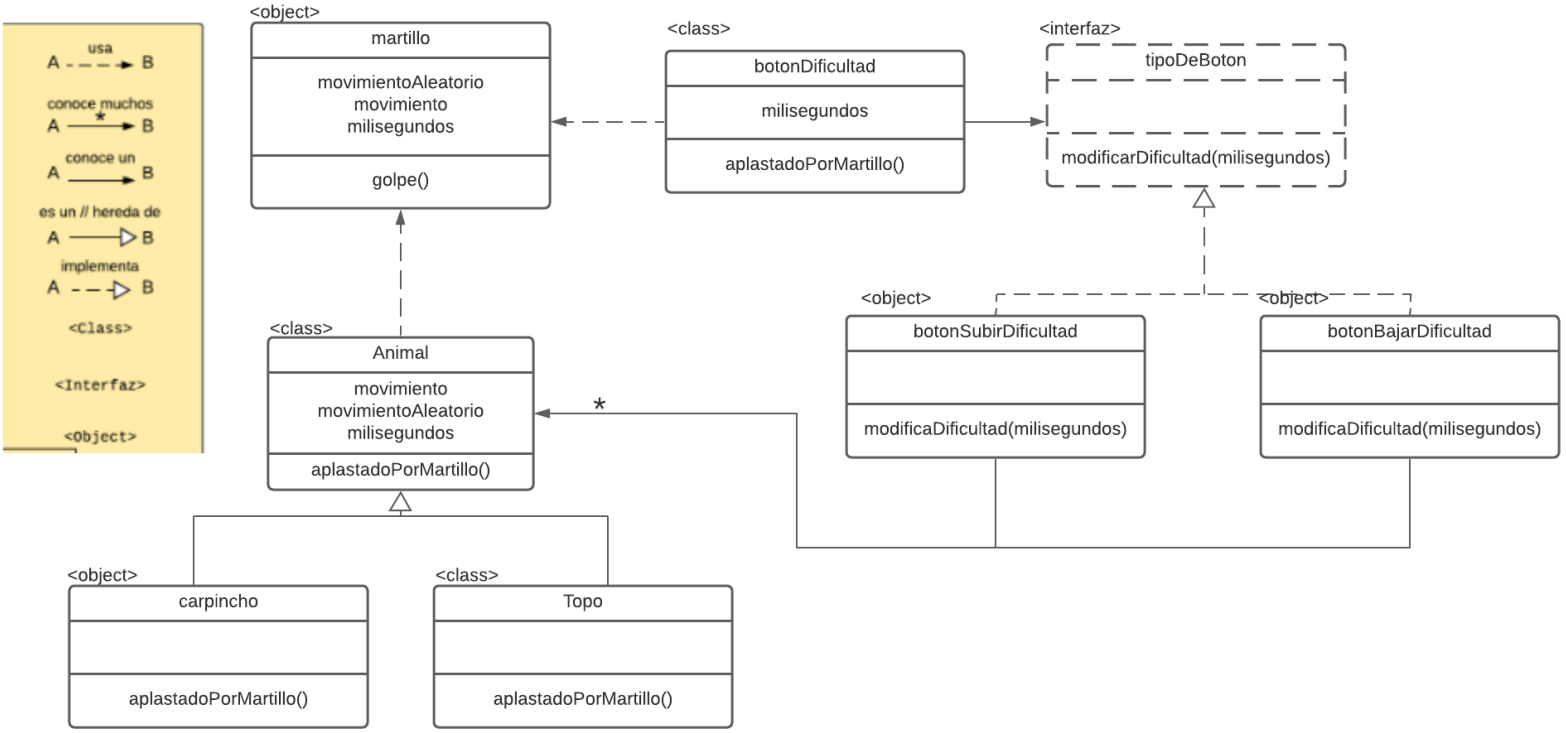
Entrega final TPI los4Fantásticos

1) Diagrama Estático



2) Polimorfismo: ¿Cuál es el mensaje polimórfico? ¿Qué objetos lo implementan? ¿Qué objeto se aprovecha de ello?

   El mensaje polimórfico es `aplastadoPorMartillo()`. Los objetos que lo implementan son los animales, los botones de dificultad y los botones de game over, donde cada uno de ellos implementan el mensaje de formas diferentes
   El objeto que se aprovecha de este mensaje es martillo cuando se envía el mensaje `game.colliders(self).first().aplastadoPorMartillo()` en donde no le interesa cómo se implementa este mensaje y tampoco cuál es el objeto al que se envía, lo que permite cambiar referencias (otros animales y/o botones) sin tener que realizar cambios de código y solo hacer que esos objetos entiendan el mensaje `aplastadoPorMartillo()`

3) Colecciones: ¿Qué operaciones de colecciones se utilizan? ¿Usaron mensajes con y sin efecto? ¿Para qué?

   Se utiliza una única colección, sin orden, en el programa: `const property topos`. Para esta colección sin orden se utiliza únicamente la operación `forEach` en varias ocasiones, más específicamente en: `topos.forEach({topo =>`

`game.addVisual(topo)})`, para enviar el mensaje para agregar la imagen asignada a cada topo dentro del conjunto, siendo este un mensaje con efecto. También se utiliza esa operación en `topos.forEach({topo => topo.agregarOnTick()})`, que le manda un mensaje con efecto a todos los topos para que comiencen a moverse aleatoriamente al iniciar el juego. Es usada también en los objetos `botonSubirDificultad` y `botonBajarDificultad`, en los mensajes con efecto `dontwhackthecapybara.topos().forEach({topo=>topo.subirVelocidad(milisegundosARestar)})` y `dontwhackthecapybara.topos().forEach({topo=>topo.bajarVelocidad(milisegundosASumar)})` respectivamente, para subir o bajar la velocidad a la que se mueven todos los topos de la colección cuando son golpeados por el objeto martillo los objetos "botones".
Finalmente, se vuelve a utilizar el `forEach` cuando se pierde en el juego, enviando el mensaje `dontwhackthecapybara.topos().forEach({topo=>game.remove Visual(topo)})` para quitar las visuales de todos los topos hasta que, viniendo al caso, se eligiera volver a jugar.
Usar esta colección con sus respectivas operaciones es de utilidad para no tener que enviarle cada mensaje a cada topo individualmente, pudiendo enviar el mensaje una única vez a todos.

4) Clases: ¿Usan clases? ¿Por qué? ¿Dónde o cuándo se instancian los objetos?

   Usamos clases para evitar la repetición de código ya que tenemos objetos que se comportan iguales y tienen la misma estructura, aunque estos pueden diferenciarse en algunas cosas. Por ejemplo, hicimos una clase "Animal" ya que dentro del juego tenemos al topo y al carpincho que tienen un comportamiento muy similar. De no usar clases estaríamos duplicando código. Instanciamos los objetos en el archivo donde se encuentra la configuración, llamado "`dontwhackthecapybara`" y los mismos se instancian antes de iniciar el juego.

5) Herencia: ¿Entre quiénes y por qué? ¿Qué comportamiento es común y cuál distinto?

   Siguiendo el ejemplo anterior, si bien el carpincho y el topo tienen comportamientos iguales, hay otros comportamientos que son distintos. Ambos objetos, heredan su comportamiento de la clase "Animal" pero observando el código podemos ver que al momento de declarar cada uno estamos redefiniendo el método `aplastadoPorMartillo()` ya que es en este método en el cual se diferencian.

6) Composición: ¿Qué objetos interactúan? ¿Dónde se delega? ¿Por qué no herencia?

   En nuestro juego los objetos que interactúan con composición son los objetos de `subirDificultad` y `bajarDificultad` los cuales son utilizados en la clase de `BotonDificultad`.
   El método `modificarDificultad(milisegundosARestar)` es delegado directamente a los objetos `subirDificultad` y `bajarDificultad` sin necesidad de que pasen por `BotonDificultad`.

En este caso no utilizamos herencia ya que de esta forma nos ahorramos tener que poner el método `modificarDificultad(milisegundosARestar)` dentro de `BotonDificultad` con llaves vacías el cual no haga nada, sino que se dirige directamente al método del objeto que le corresponda.