# ESCUELA COLOMBIANA DE INGENIERÍA PROGRAMACIÓN ORIENTADA A OBJETOS

**Parcial tercer tercio. S17: 2020-1**

**Nombre: Iván Camilo Rincon Saavedra Nota esperada:** 37

**POOng**

## REFACTORIZANDO: BBD-MDD XP

Seleccione un comportamiento del proyecto que requiera refactorización.

El comportamiento del movimiento de la pelota, ya que esta rebota con los bordes de la cancha, siendo este no correcto del todo ya que no se comporta como una pelota de tenis verdadera

* 1. **Explique su selección, detallando las ventajas y desventajas de la implementación actual.**

Ventajas:

* Al siempre rebotar con los bordes garantizamos que no ocurra fallas o comportamientos no desados, como por ejemplo que se salga la pelota por las partes laterales de la cancha
* No genera puntos si la pelota sale por los laterales

Desventajas:

* El golpeo a las sorpresas actuales puede llegar a demorarse
  1. **Incluya en el documento los componentes BDD-MDD del comportamiento seleccionado: requisitos, diseño (estructural y de comportamiento), código (presentación y aplicación) y pruebas (unidad y aceptación). [NOTA: Consultar contenido al final del documento]**

El método con el diagrama , será el de POOng rebotaPersonaje.

* 1. **¿Cuáles patrones se usan en la implementación actual? ¿Cuál patrón podría aplicar para la refactorización? ¿por qué?**

Actualmente la clase pelota maneja herencia y polimorfismo, por lo que sería muy fácil hacer la refactorización del movimiento o Angulo en el que se mueve la pelota para sus clases hijas, por lo que lo mejor seria mantener el patrón actual.

* 1. Refactorice este comportamiento. Explique e incluya los componentes que cambiaron.

## EXTENDIENDO. Nuevo elemento.

Extienda el proyecto para poder incluir un nuevo elemento (sorpresa, blanco o pelota) con las siguientes características:

### Si es sorpresa o blanco:

* Otorga al jugador que la capture, la habilidad de no perder el 50% de fortaleza si es golpeado por un bloque. Aplica solo una vez.
* Algunas veces (1 de cada 2) estos elementos buscan a la pelota.

### Si es pelota:

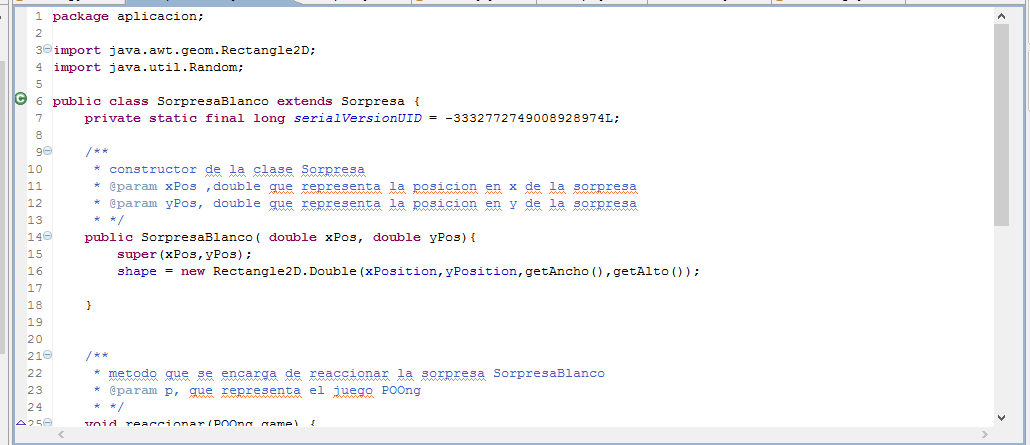
* Su velocidad aumenta en un 10% y rompe los bloque.
* El efecto dura 10s o hasta impactar un bloque (lo que primero ocurra).
* Si aparece un bloque lo busca para romperlo.
  1. **Incluya en el documento los siguientes componentes BDD-MDD: requisitos, diseño (estructural y de comportamiento), código (presentación y aplicación) y pruebas (unidad y aceptación).**
  2. **¿Cuáles patrones usa la implementación actual?**

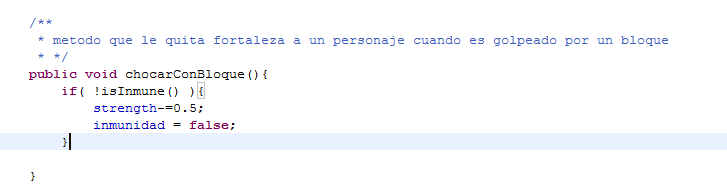
En la implementación actual se usa herencia como se puede ver en la super clase **Elemento** que contiene atributos y métodos comunes de los elementos del juego, polimorfismo con las diferentes sorpresas ya que su forma de reaccionar siempre será diferente según su definición en el pdf

**Cambios Realizados al Proyecto**

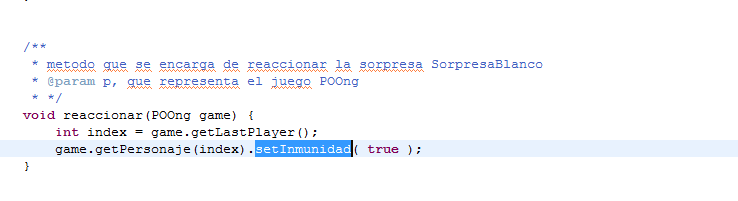
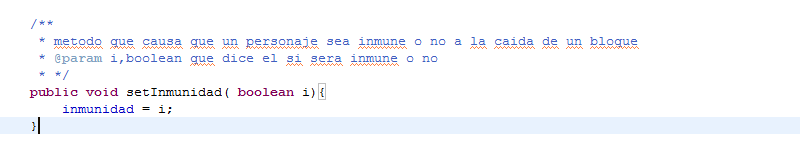
**Segundo Punto**

**Creacion de una nueva clase llamada SorpresaBlanco**

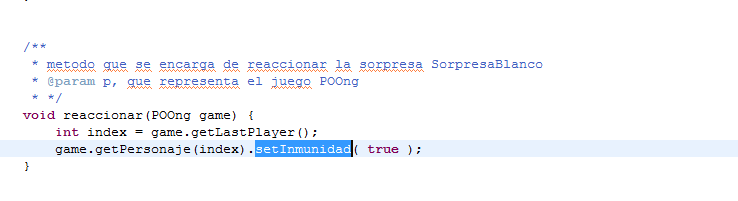
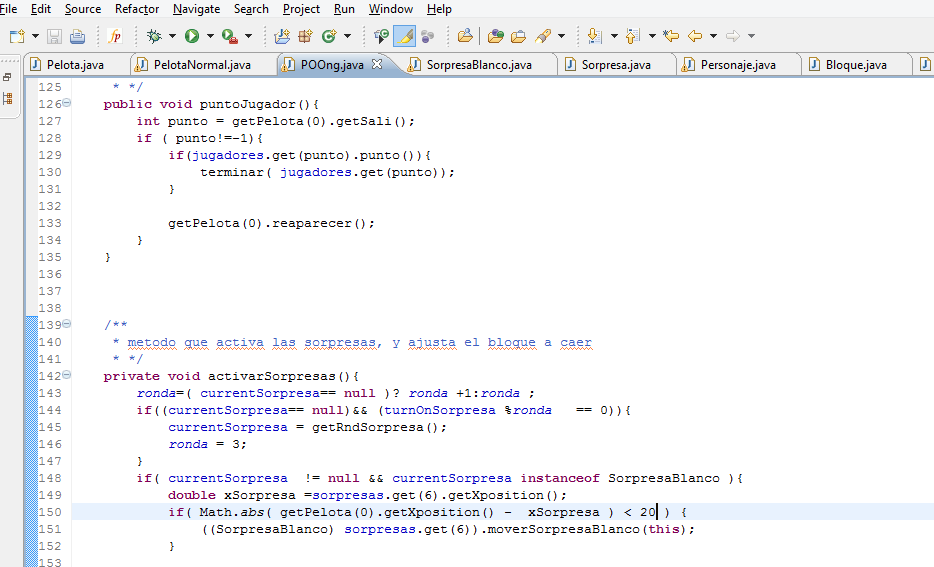
**en esta clase se encuentra dos métodos**

* Reaccionar: el cual activa un poder de inmunidad al personaje, esta inmunidad se ve representada como un booleano que dice si podemos bajarle la resistencia o no al personaje

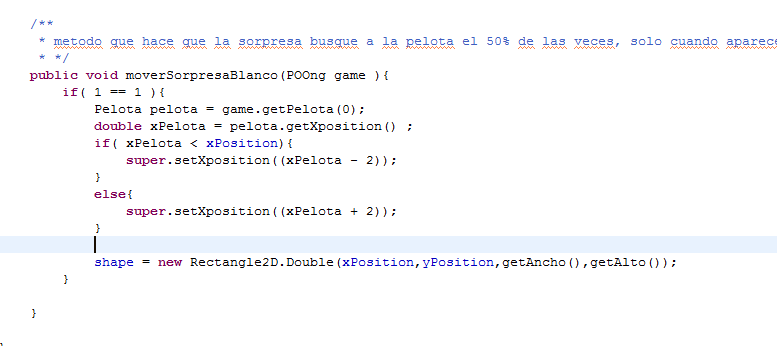
Como podemos apreciar si inmune es falso entrara al if y afectara la fortaleza de lo contrario no lo afctara

Para definir esta inmunidad se le creo un método a Personaje setInmunidad:

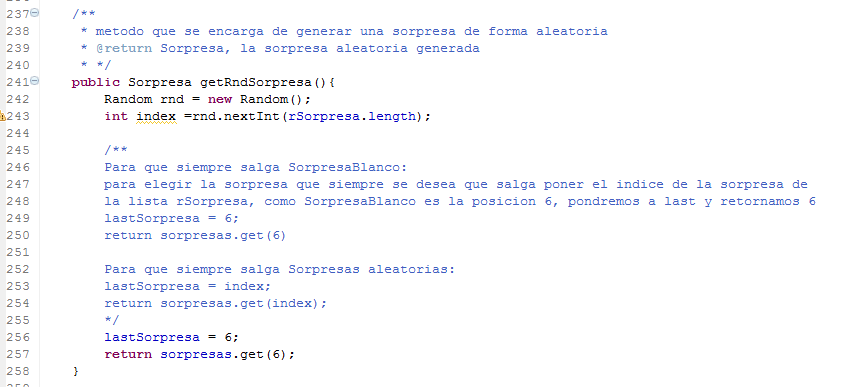
Para mover el bloque



En la línea 151 de la clase Poong, se revisa si la ultima sorpresa fue la implementada y si la diferencia de las posiciones de x son menores a 20, el bloque empezara a seguir la pelota

para garantizar que solo se haga el 50% de las veces, se usa el atributo seguir de la clase nueva

La sorpresa siempre será la nueva si se desea de forma aleatoria revisar la clase POOng método:



IMPORTANTE:

El desarrollo de los puntos anteriores deben estar en el documento entregable del parcial. (Sólo se revisará en el proyecto lo que relacionen en el documento).

### COMPONENTES BDD-MDD

|  |  |
| --- | --- |
| Requisitos | Descripción del comportamiento |
| Diseño estructural | Zona(s) del diagrama de clase relevantes |
| Diseño de comportamiento | Diagrama(s) de secuencias relevantes |
| Código presentación | Código de los método(s) de la capa de presentación que atiende(n) el comportamiento |
| Código aplicación | Código de los método(s) de la capa de aplicación que atiende(n) el comportamiento |
| Pruebas de unidad | Código de las pruebas y evidencia de su ejecución |
| Prueba aceptación | Descripción y dos pantallas significativas |