**Parcial Final. POOB-02**

Jefer Alexis González Romero  
Nota esperada:

**I. REFACTORIZANDO: BBD-MDD XP**

Seleccione un comportamiento del proyecto que requiera refactorización.

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

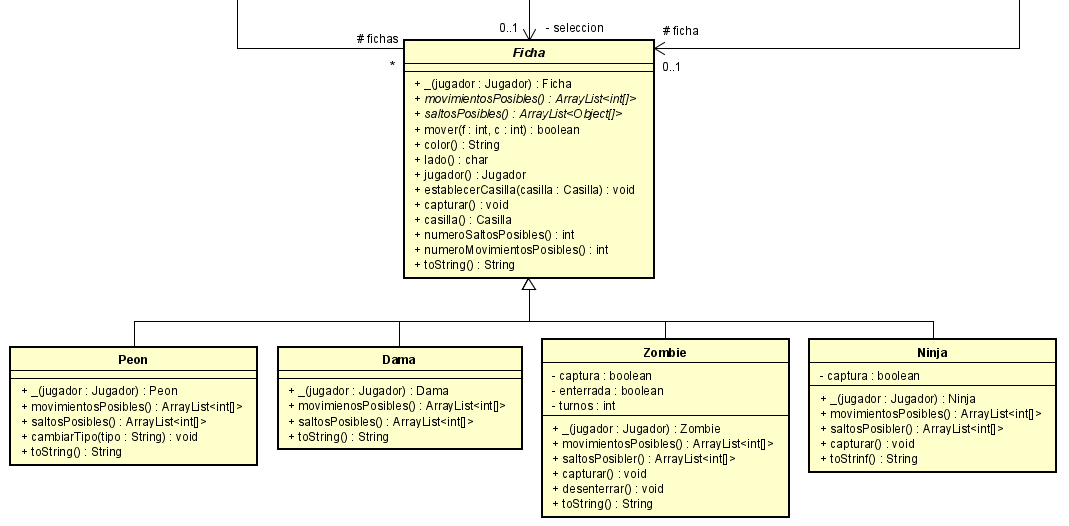
Se seleccionó las fichas *Ninja* y *Zombie* que extienden a Ficha, estas comparten el atributo captura y tienen los métodos *movimientosPosibles() y saltosPosibles()* muy similares, por lo tanto, se presenta duplicación de código, lo cual es redundante. Esto provoca que, si se desea cambiar el modo de calcular los movimientos o lo saltos posibles, se tenga que hacer en cada clase.

**2.** Incluya en el documento todos los componentes BDD-MDD del comportamiento seleccionado: requisitos, diseño (estructural y de comportamiento), código (presentación y aplicación) y pruebas (unidad y aceptación).

**Requisitos**

Hay redundancia en los tipos de fichas *Zombie y Ninja* que extienden a Ficha ya que tienen el atributo captura y tienen los métodos *movimientosPosibles() y saltosPosibles()* muy similares, así que le código está duplicado.

**Diseño**

**Estructural**

**Comportamiento**

Método *movimientosPosibles()* de la clase *Zombie.*

**Diagrama

Descripción generada automáticamenteTabla

Descripción generada automáticamente**

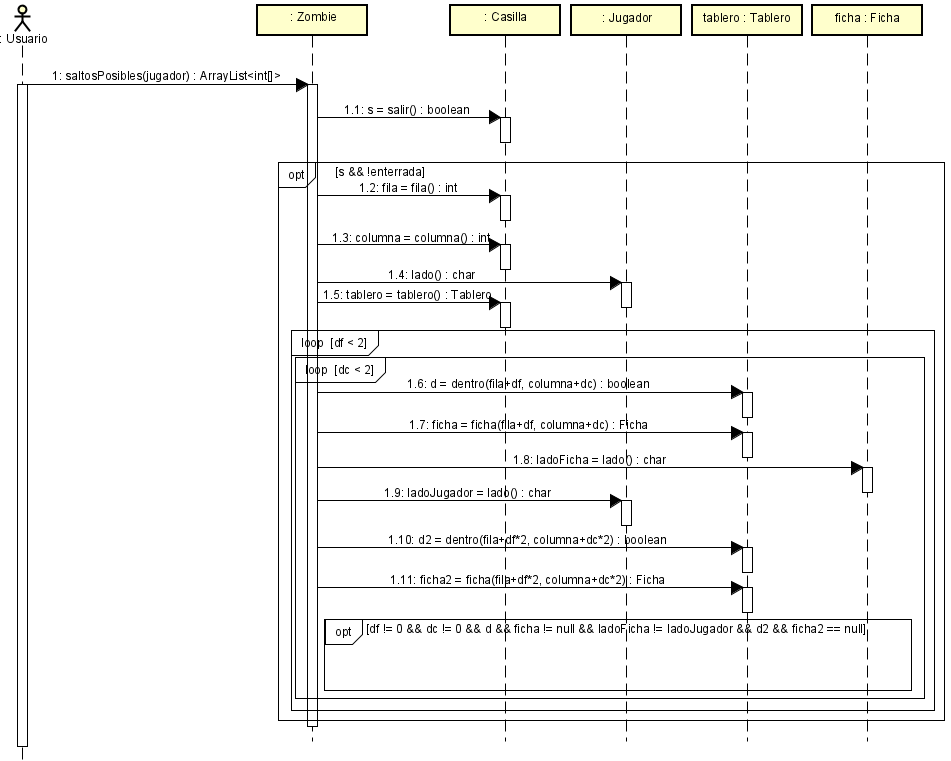
Método *movimientosPosibles()* de la clase *Ninja.*

**Diagrama

Descripción generada automáticamenteTabla

Descripción generada automáticamente con confianza media**

Método *saltosPosibles()* de la clase *Zombie.*

****

Método *saltosPosibles()* de la clase *Ninja.*

**Diagrama, Tabla

Descripción generada automáticamente**

**Código**

**Presentación**

****

****

**Aplicación**

Clase *Zombie.*

**Texto

Descripción generada automáticamente**

Clase *Ninja.*

**Texto

Descripción generada automáticamente**

**Pruebas**

**Pruebas Unitarias**

Pruebas Ficha *Zombie.*

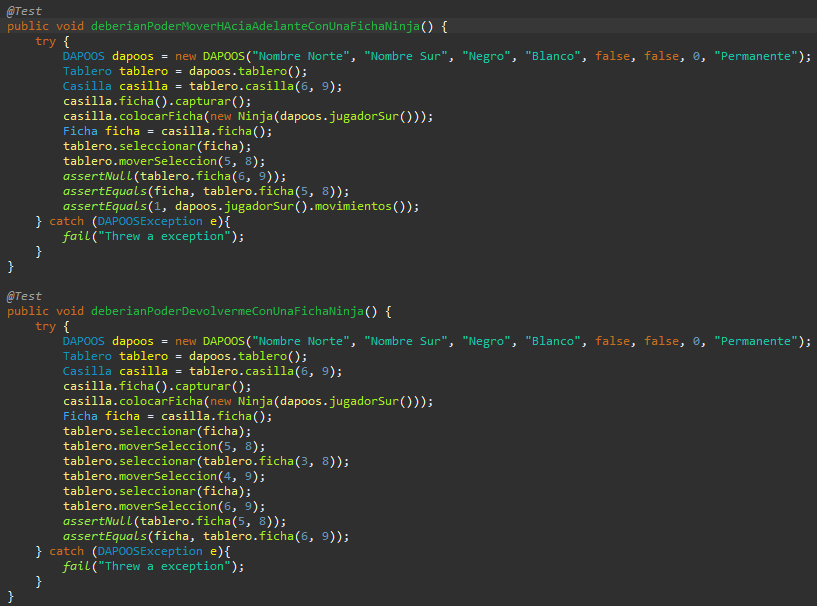
* deberiaPoderMoverHaciaAdelanteConUnaFichaZombie()
* deberiaPoderDevolvermeConUnaFichaZombie()
* deberiaPoderCapturarFichasConFichasZombies()

 Texto

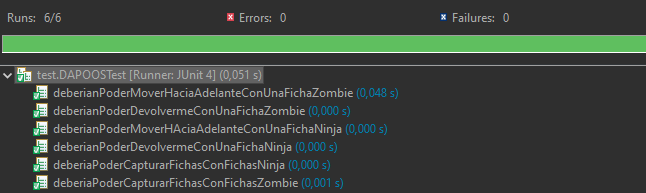
Descripción generada automáticamente

Pruebas Ninja.

* deberiaPoderMoverHaciaAdelanteConUnaFichaNinja()
* deberiaPoderDevolvermeConUnaFichaNinja()
* deberiaPoderCapturarFichasConFichasNinja()

 Texto

Descripción generada automáticamente

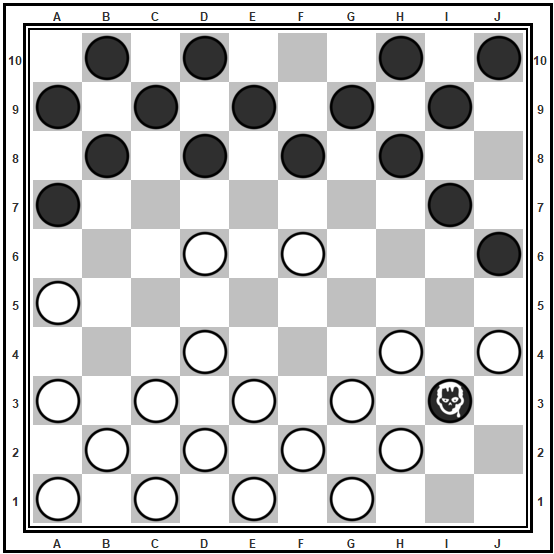
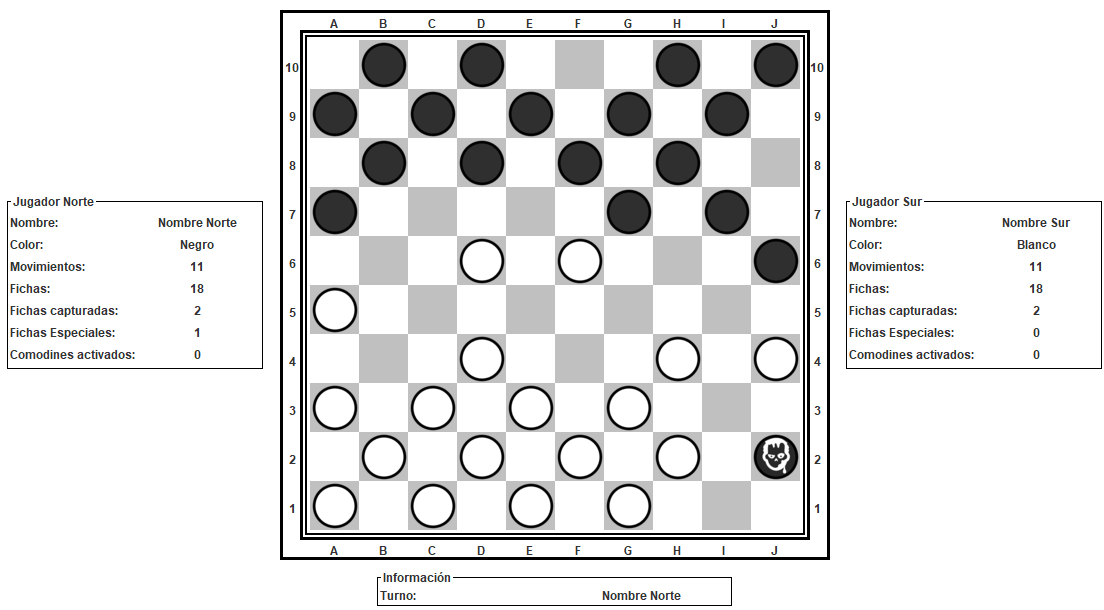


**Pruebas de aceptación**

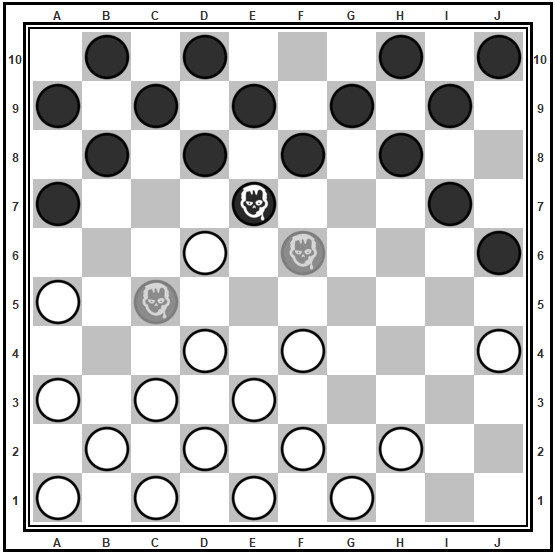
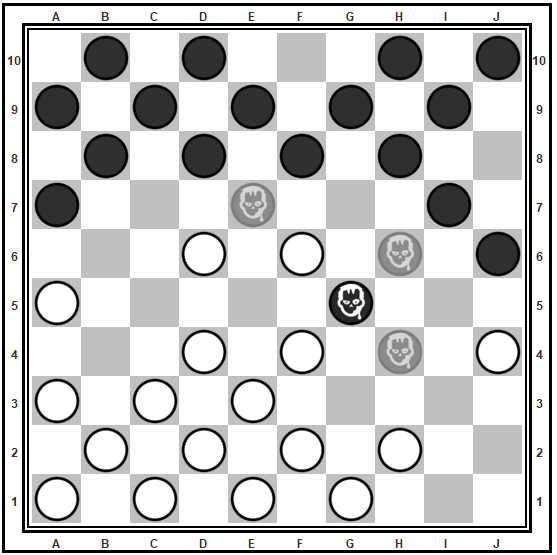
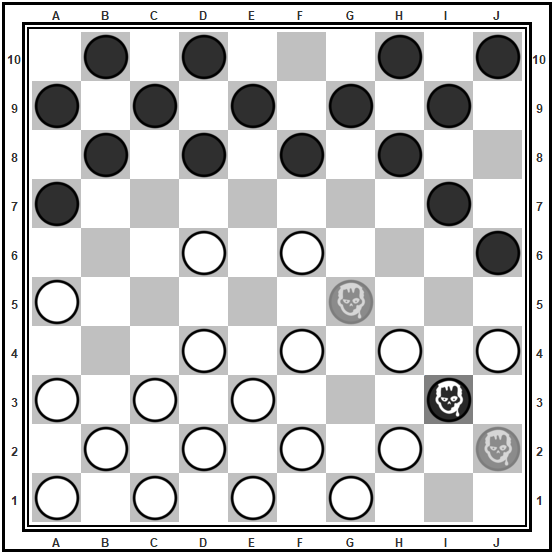
1. Se selecciona la opción *Abrir juego* y se abre el archivo *PruebaZombie.dat.*
2. Se realizan los siguientes movimientos:
   1. J2 → I3 (Se captura la ficha que estaba en G7, porque podía capturar)
3. G3 → F4
4. I3 → G5
5. G5 → E7
6. E7 → C5
   1. F4 → G5 (Se captura la ficha que estaba en D4, porque podía capturar)
7. C5 → D4

**Capturas**

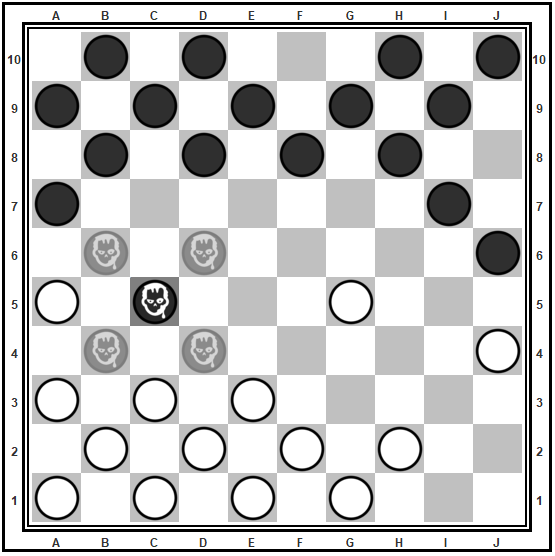
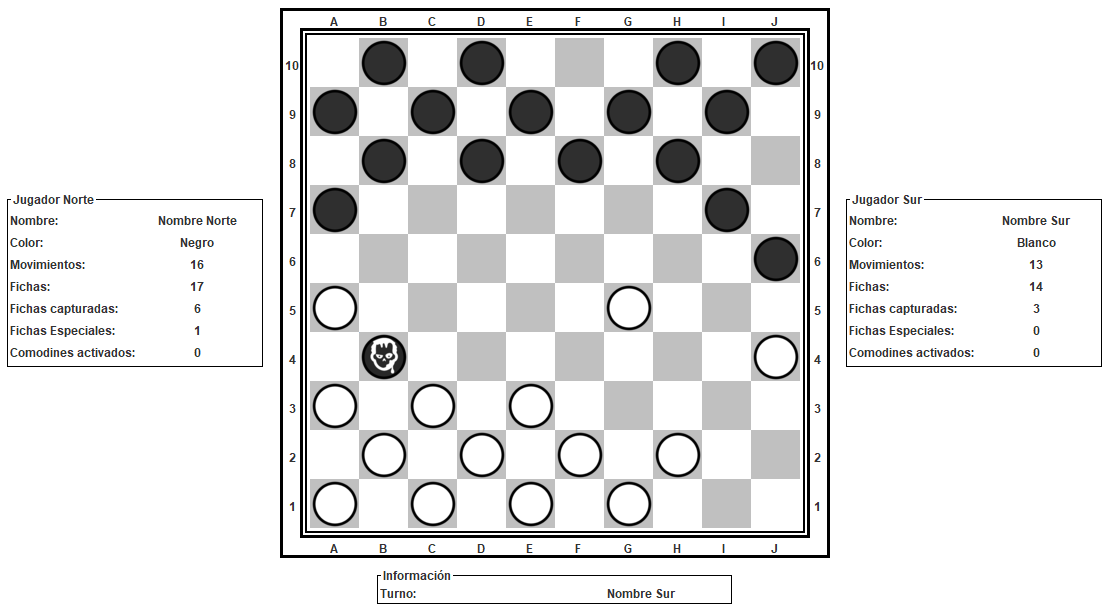
**Movimiento hacia adelante**

****

**Capturar Ficha**

****

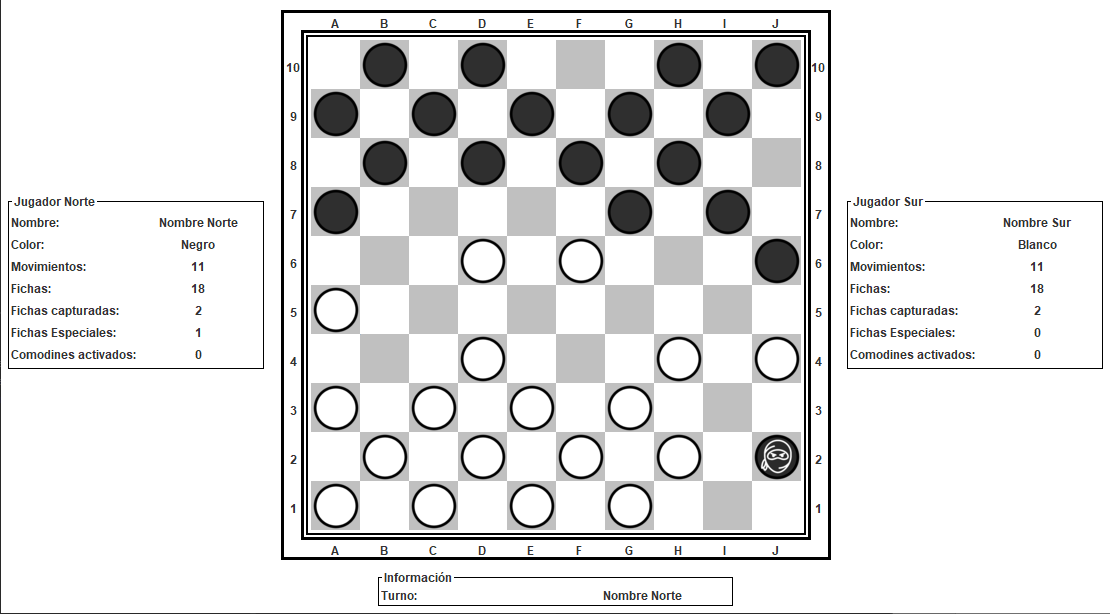
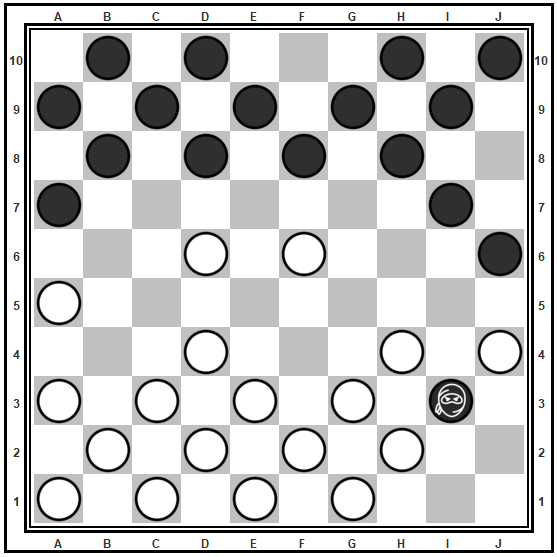
**Movimiento hacia atrás**

**** ****

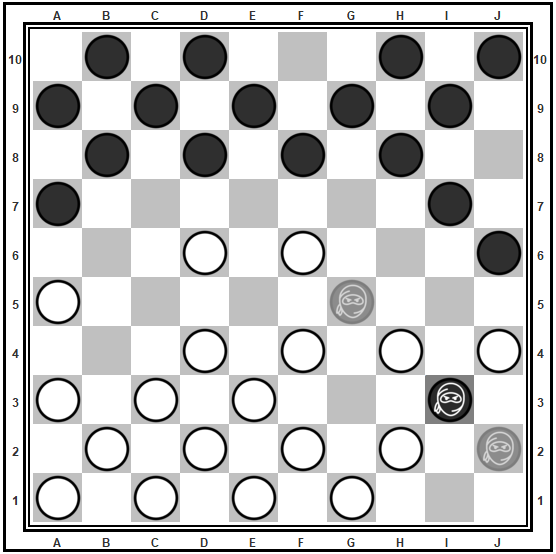
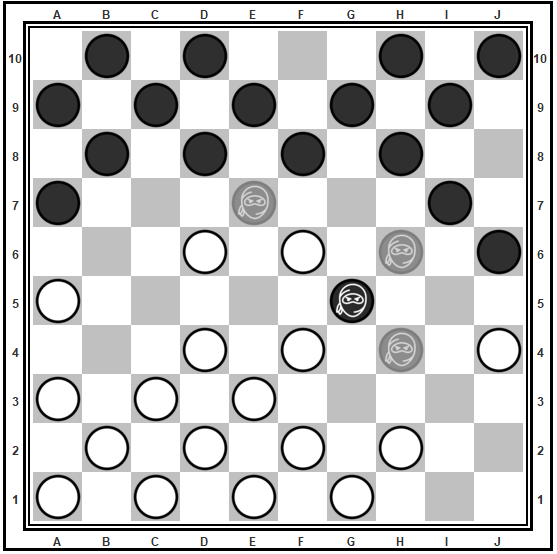
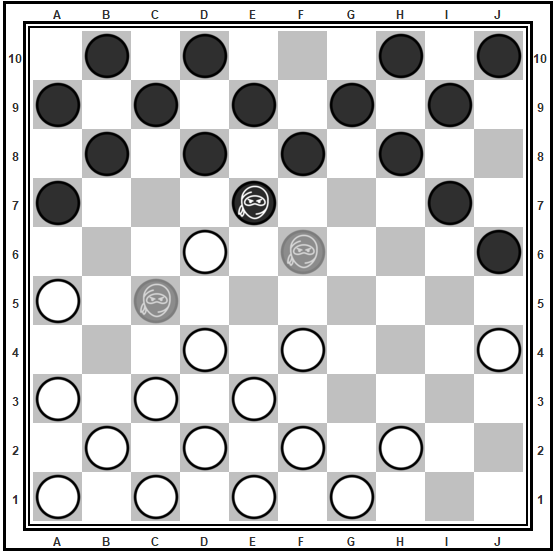
1. Se abre el menú, se oprime la opción *Abrir* y se abre el archivo *PruebaNinja.dat.*
2. Se realizan los siguientes mismos movimientos realizados en el punto 2.

**Capturas**

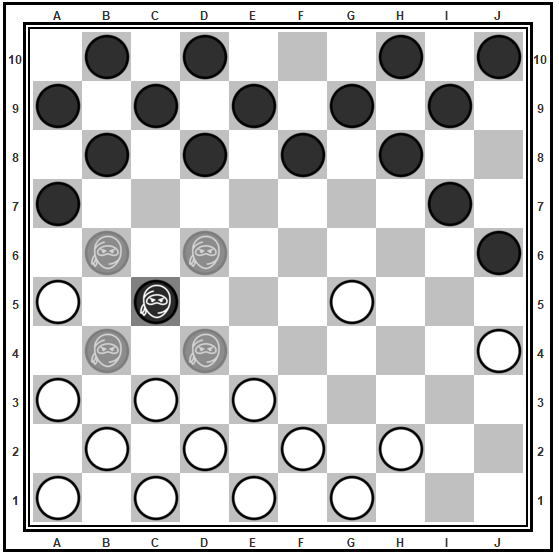
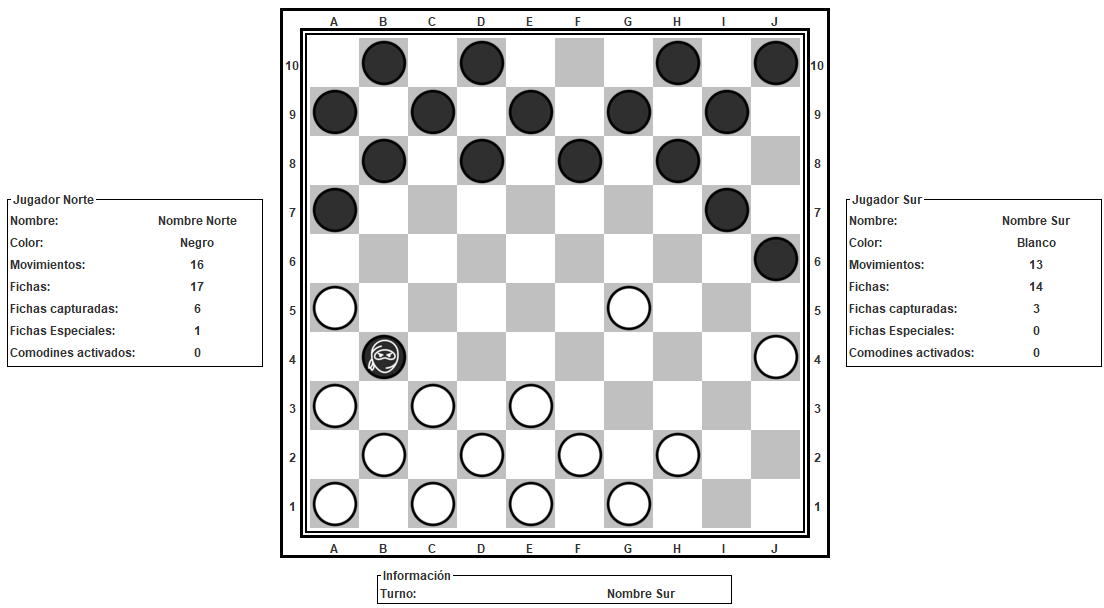
**Movimiento hacia adelante**

 ****

**Capturar Ficha**

**** **** 

**Movimiento hacia atrás**

**** 

**3.** ¿Cuál patrón podría aplicar para la refactorización? ¿por qué?

Se puede aplicar el patrón de reutilización plantilla, ya que este soluciona el problema de ¿Cómo implementar comportamientos comunes solo una vez y a la vez permitir especificar los comportamientos que pueden variar? Y lo soluciona a través de la creación de una clase abstracta, en la cual se implementan las conductas que no varían y las que varían se declaran en la clase abstracta (métodos abstractos) y se implementan en las subclases

**4.** Refactorizar este comportamiento. Explique e incluya los componentes BDD-MDD que cambiaron.

Se añade una clase abstracta llamada *Nueva* que extenderá a ficha y tendrá la implementación de los métodos movimientosPosibles() y saltosPosibles(), tendrá como método abstracto *toString().* Además, *Zombie y Ninja* extenderán a *Nueva,* donde *Zombie* sobrescribirá los métodos movimientosPosibles() y saltosPosibles(), ya que se le añaden un condicional, pero de igual forma llamará a los métodos de la super clase.

**Diseño**

**Estructural**

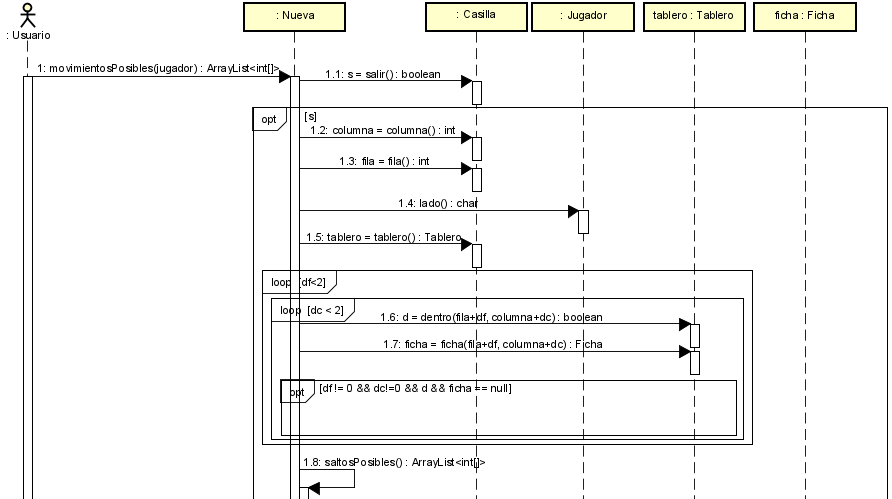
Diagrama

Descripción generada automáticamente

**Comportamiento**

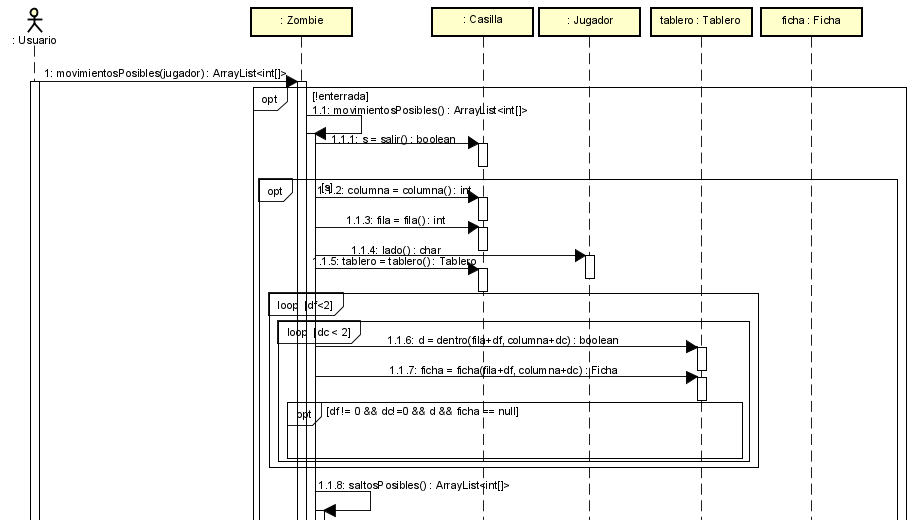
Se hacen los diagramas de secuencia para *Nueva* y *Zombie* (que llama a los métodos de la superclase), a *Ninja* no se le hace debido a que los métodos son exactamente igual al de Nueva.

Método *movimientosPosibles()* de la clase *Nueva.*

 Diagrama, Tabla

Descripción generada automáticamente

Método *movimientosPosibles()* de la clase *Ninja.*

 Diagrama, Tabla

Descripción generada automáticamente

Método *saltosPosibles()* de la clase *Nueva.*

**Tabla

Descripción generada automáticamente**

Método *saltosPosibles()* de la clase *Zombie.*

**Tabla

Descripción generada automáticamente**

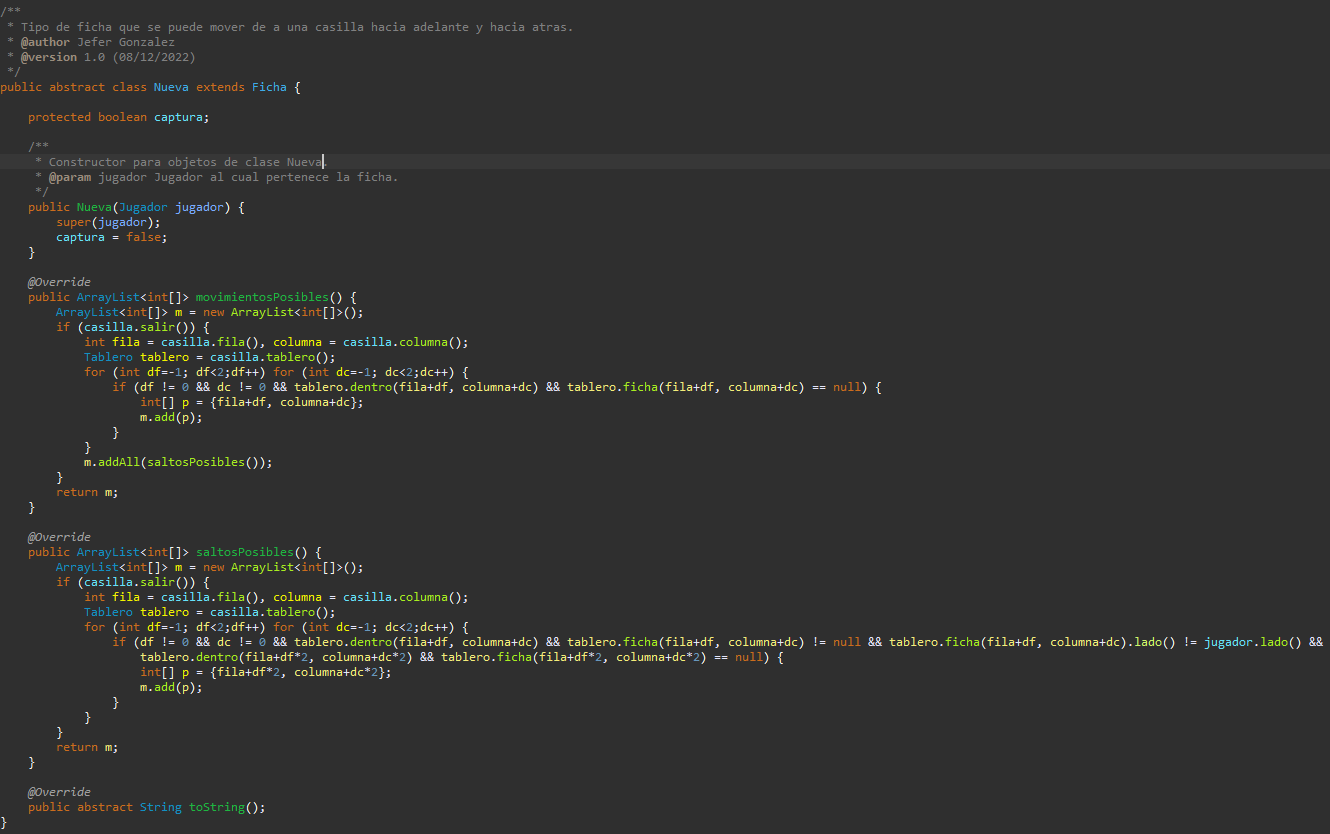
**Código**

**Presentación**

No se cambia nada referente a la capa de presentación.

**Aplicación**

Clase *Nueva*

****

Clase *Zombie*

**Texto

Descripción generada automáticamente**

Clase *Ninja*

Texto

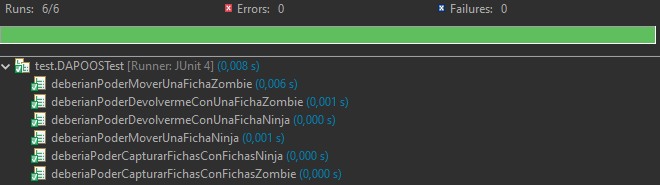
Descripción generada automáticamente

**Pruebas**

**Pruebas unitarias**

Se usaron las mismas pruebas que se tenían.

* deberiaPoderMoverHaciaAdelanteConUnaFichaZombie()
* deberiaPoderDevolvermeConUnaFichaZombie()
* deberiaPoderCapturarFichasConFichasZombies()
* deberiaPoderMoverHaciaAdelanteConUnaFichaNinja()
* deberiaPoderDevolvermeConUnaFichaNinja()
* deberiaPoderCapturarFichasConFichasNinja()

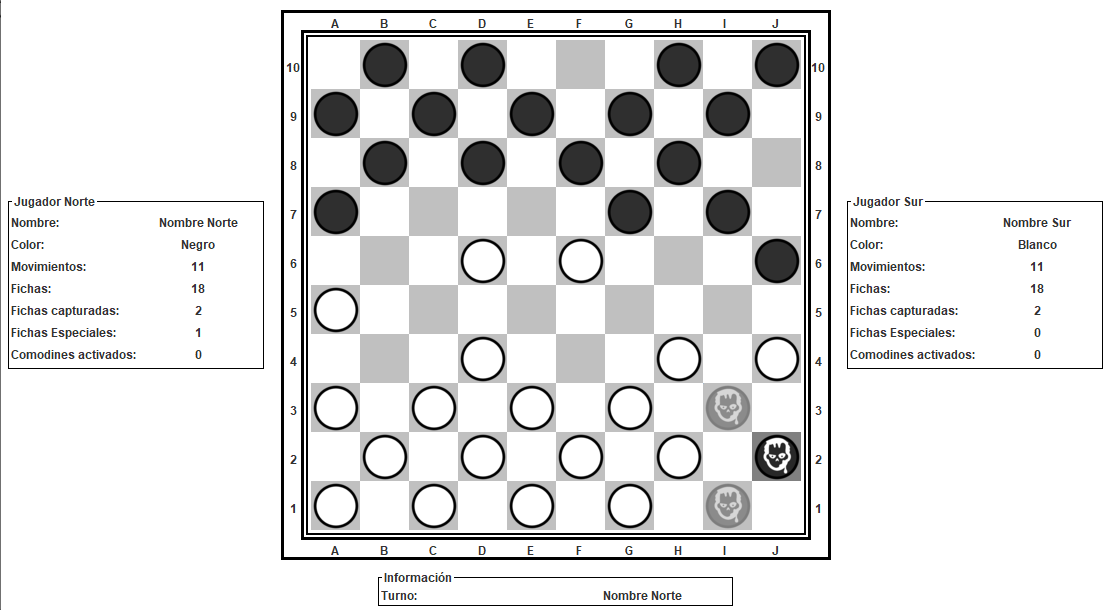
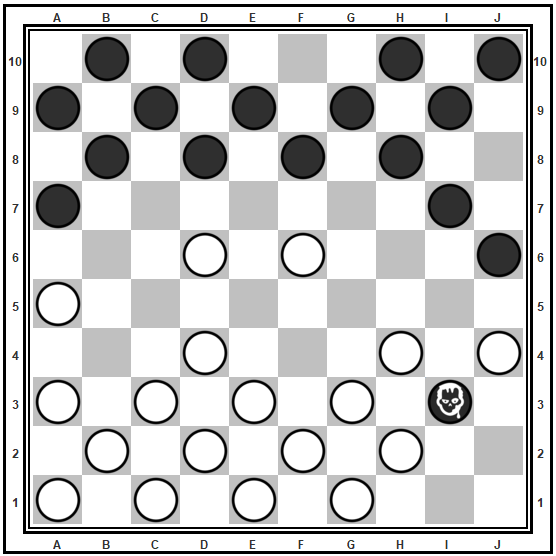


**Pruebas de aceptación**

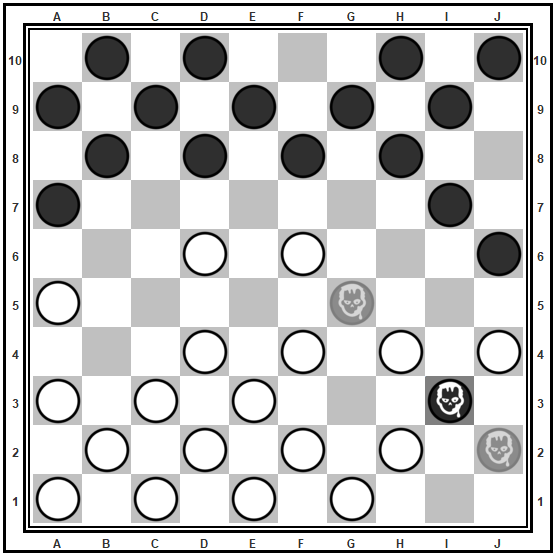
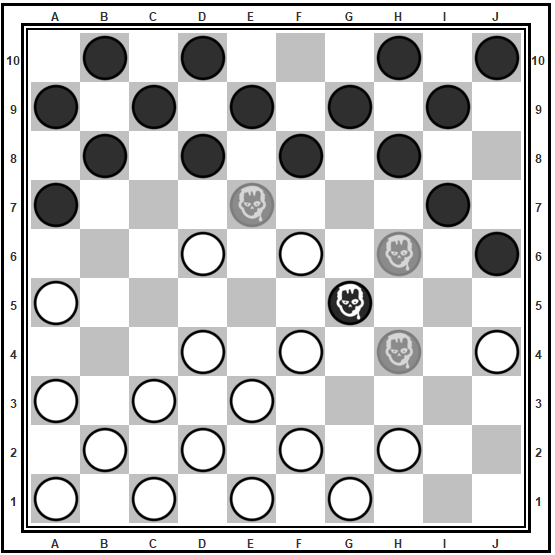
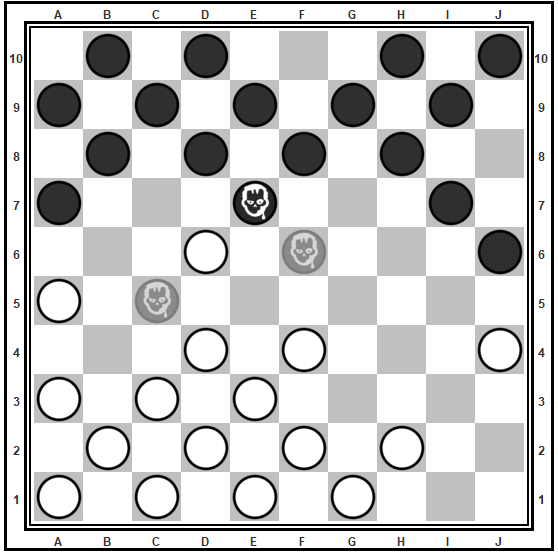
Se realizan la misma prueba de aceptación hechas anteriormente.

**Capturas**

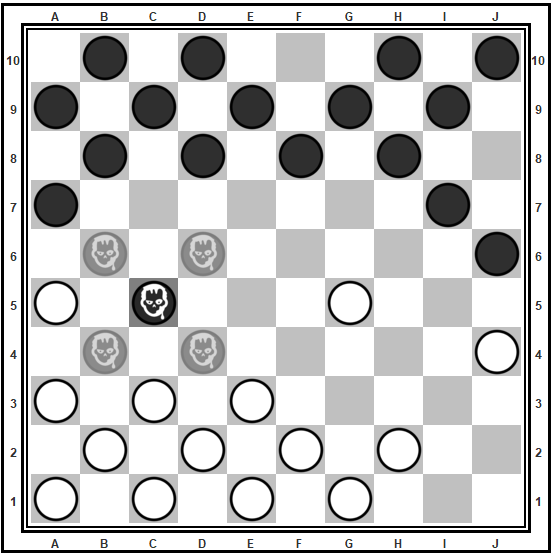
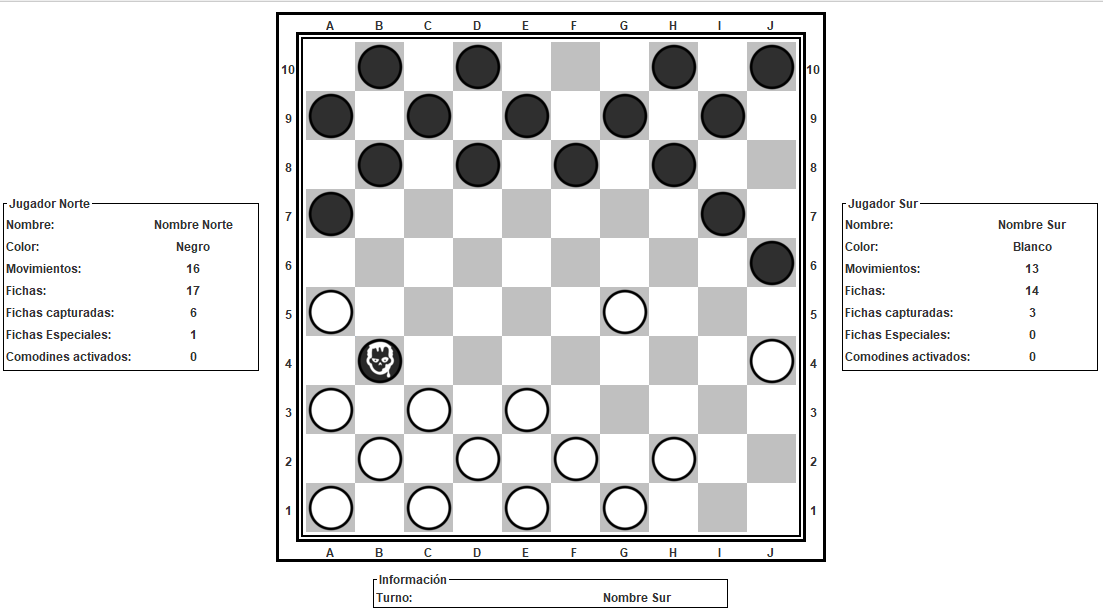
**Movimiento hacia adelante Zombie**

**** ****

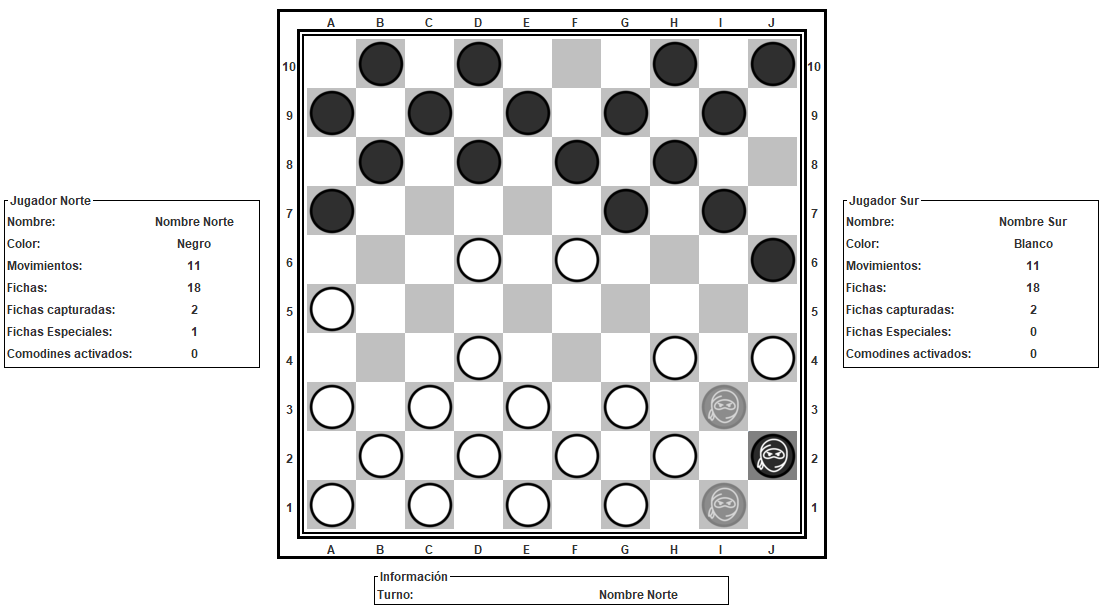
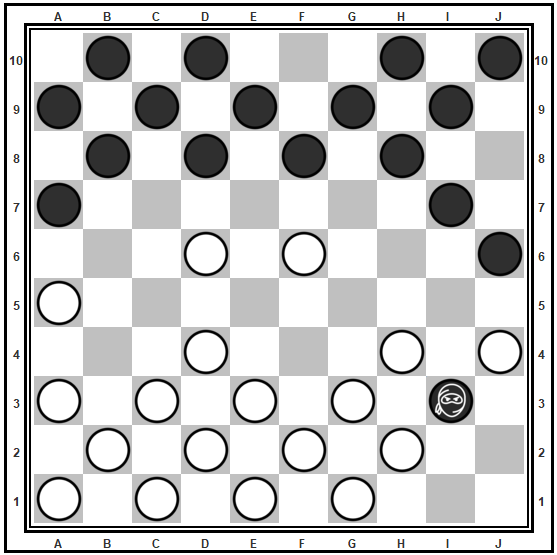
**Capturar Ficha Zombie**

**** **** 

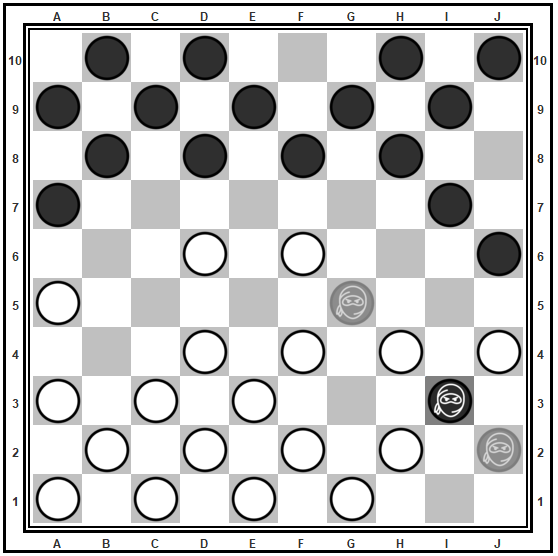
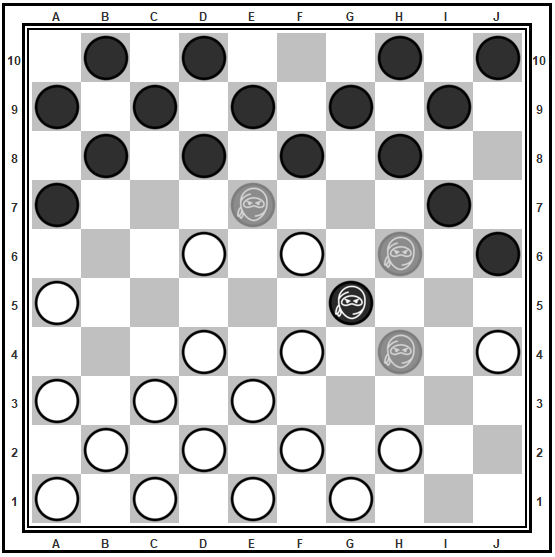
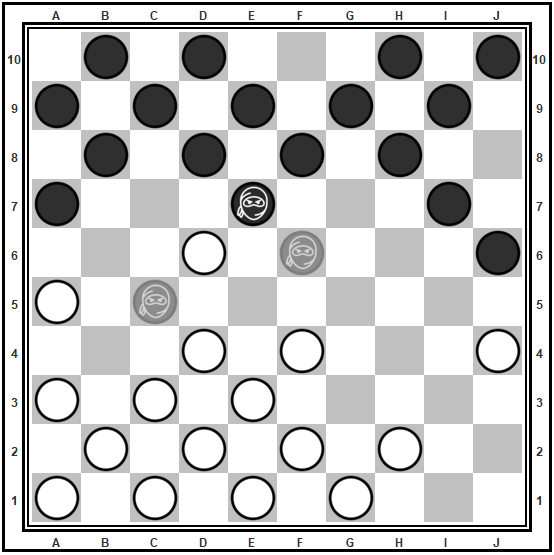
**Movimiento hacia atrás Zombie**

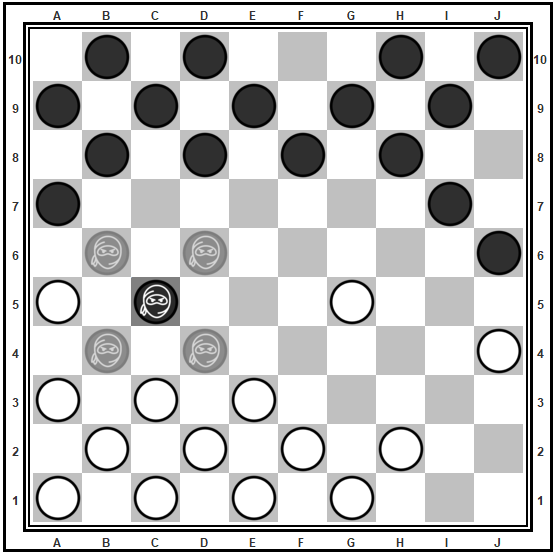
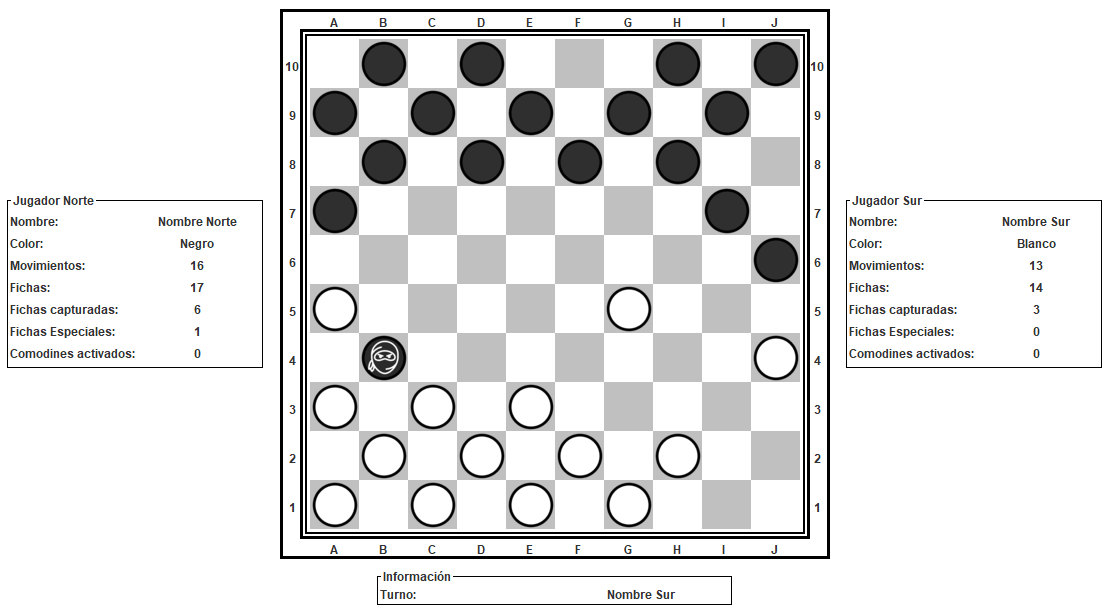
**Movimiento hacia adelante Ninja**

**Capturar Ficha Ninja**

**Movimiento hacia atrás Ninja**

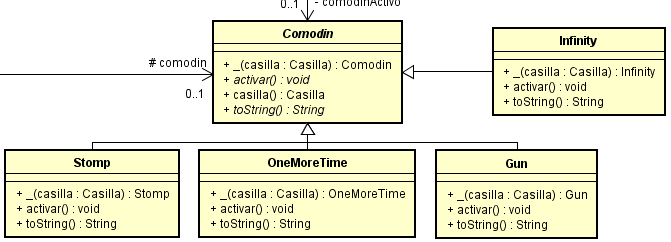
 

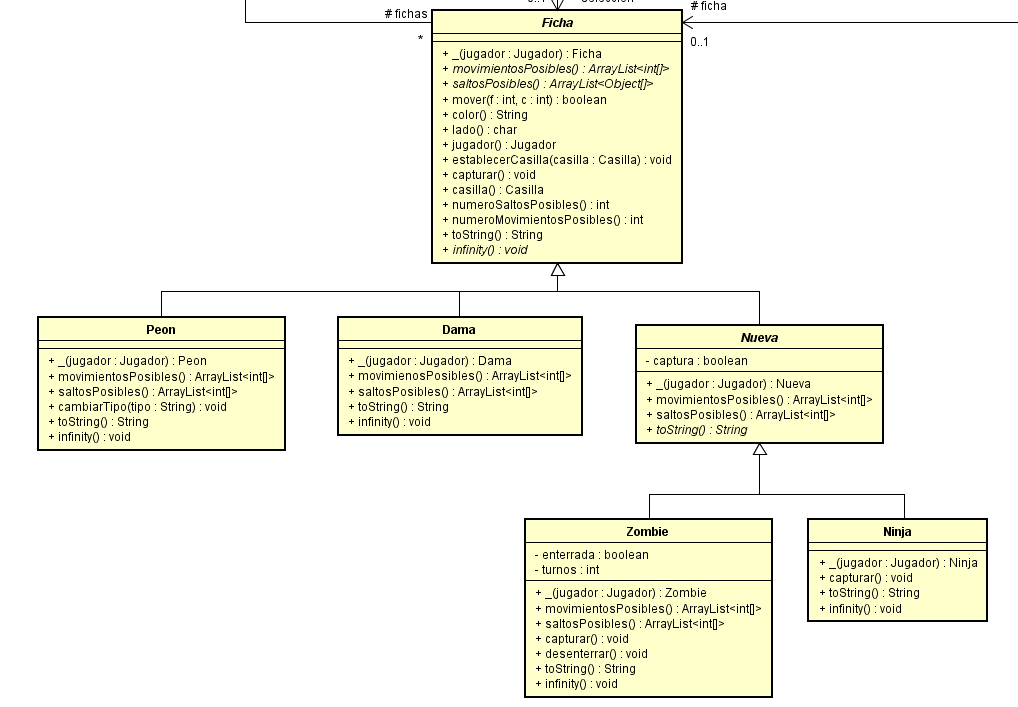
**II. EXTENDIENDO. Nuevo elemento**

**1.** Incluya en el documento los componentes BDD-MDD: requisitos, diseño (estructural y de comportamiento), código (presentación y aplicación) y pruebas (unidad y aceptación).

**Diseño**

**Estructural**



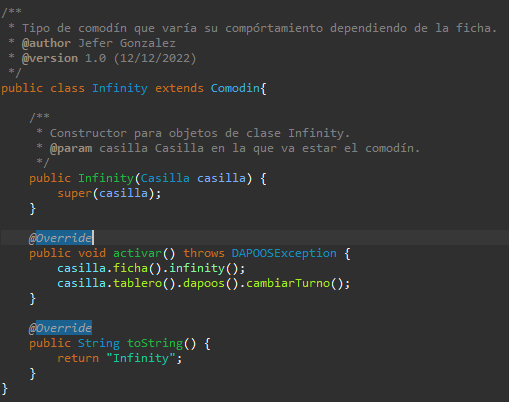
****

**Comportamiento**

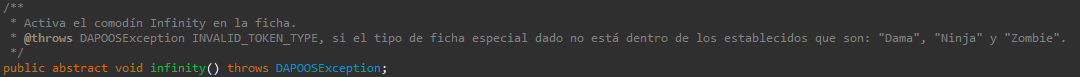
**Código**

**Presentación**

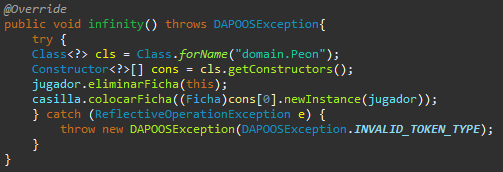
**Aplicación**

****

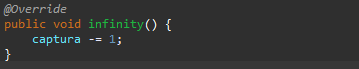
En Ficha:

****

En Zombie:



En Ninja:



**Pruebas**

**Pruebas unitarias**

****

**Pruebas de aceptación**

**2.** ¿Cuáles patrones usa en su implementación?

Variaciones protegidas, es el principio fundamental de protegerse frente a nuevos requisitos.