Práctica 3: Gravity

Fecha de entrega: 27 de Febrero, 16:00

Objetivo: Implementación y uso de la herencia, interfaces, excepciones, patrones de diseño

1. Introducción

En esta práctica extenderemos la práctica 2 añadiendo la posibilidad de jugar a un nuevo juego llamado *Gravity*. Igual que los anteriores, es un juego de tablero para dos jugadores en el que gana el primero que consigue colocar cuatro fichas en línea, ya sea vertical, horizontal o diagonal.

Las diferencias principales de Gravity con los anteriores son las siguientes:

- El tamaño del tablero no es fijo, sino que se establece al iniciar la partida. Aunque lo normal es jugar en tableros de 10x10, se pueden utilizar tamaños de tablero distintos (y no necesariamente cuadrados).
- Los jugadores colocan la ficha en *cualquier* casilla del tablero, siempre y cuando ésta esté vacía.
- Una vez colocada, la ficha es *atraída* por los bordes (de ahí el nombre *Gravity*), de forma que ésta se desplaza en línea recta hacia el borde (o bordes) más cercanos.

En la figura 1 pueden verse tres ejemplos de movimiento (el tablero izquierdo representa las fichas en las posiciones donde las coloca el jugador, y el tablero derecho las posiciones donde finalmente van a parar). La ficha 1 aparece colocada más cerca del borde superior, por lo que es atraída hacia él, mientras que la ficha 2 es atraída por el borde izquierdo. Por su parte, la ficha 3 es equidistante al borde derecho y al inferior, por lo que la dirección que toma en su "caída" es diagonal.

De manera intuitiva, puede verse el tablero como una pirámide de base cuadrangular. Al colocar una ficha, ésta se ve afectada por la inclinación del lado de la pirámide, y cae hasta que encuentra otra ficha o el lado.

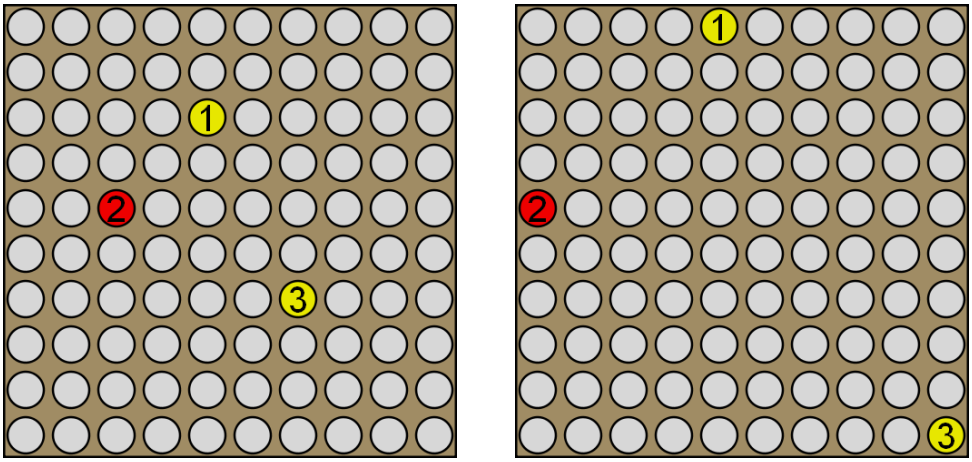

Figura 1: Ejemplo de movimiento en Gravity

Además, añadiremos la posibilidad de que los jugadores no sean controlados por el usuario de la aplicación sino que sean jugadores automáticos que colocan las fichas aleatoriamente.

2. Nueva funcionalidad

La aplicación a desarrollar tiene la misma funcionalidad que la creada en la práctica 2 añadiendo:

- Aparecen algunos comandos adicionales que el usuario podrá utilizar durante el juego.
- La ejecución podrá ser configurada utilizando los parámetros a la aplicación.
- Aunque no es funcionalidad propiamente dicha sino mejora, se retocarán los mensajes de error dados por la aplicación cuando el usuario realiza movimientos incorrectos. En esta nueva aplicación los mensajes concretos darán detalles específicos del tipo de error.

2.1. Nuevos comandos

Debe añadirse un nuevo comando que permite cambiar el tipo de jugador encargado de un color determinado. Este comando tendrá la forma:

```
Jugador <color> <tipo>
```

donde el color podrá ser `blancas` o `negras` y el tipo de jugador podrá ser `humano` o `aleatorio`.

Cuando se utiliza un jugador aleatorio, en el momento en el que el usuario ejecuta `poner`, la aplicación en lugar de preguntar la columna (y fila si se está jugando a Gravity), elegirá una posición aleatoria dentro del tablero y colocará ahí la ficha. El jugador aleatorio, eso sí, debe ser capaz de elegir una posición *correcta*: en el Conecta 4 elegirá una columna que no esté llena, y en Gravity una posición que no esté ocupada.

Dado que este comando no cambia el estado del tablero, la operación *no* forma parte de las acciones que pueden deshacerse por lo que tras el cambio en el tipo del jugador, la ejecución de `deshacer` eliminará el último movimiento realizado en lugar de restaurar el tipo de jugador anterior.

Por otro lado, el comando `Jugar` se extiende añadiendo la posibilidad de indicar que queremos jugar a Gravity. Dado que en Gravity el tamaño del tablero puede configurarse, se deberá especificar su número de columnas y filas. Por ejemplo: `Jugar gr 9 10`. Igual que ocurría en la práctica anterior, el comando no puede deshacerse y elimina el registro de acciones a deshacer.

Además, independientemente de a qué juego se solicite cambiar, los tipos de jugadores se reinician y, independientemente de la configuración que hubiera, vuelven a ser jugadores humanos.

Por último, y dado que el número de comandos posibles ha ido creciendo, añadimos un nuevo comando, `Ayuda`, que muestra por pantalla todos los comandos disponibles y una breve explicación de lo que hace cada uno.

2.2. Parámetros de la aplicación

El usuario podrá seleccionar el tipo de juego al lanzar la aplicación utilizando los argumentos pasados en la línea de órdenes. En concreto, el usuario podrá utilizar los siguientes parámetros (ver la sección 3 para ejemplos de uso)[1]:

- `-g` o `--game` seguido del tipo de juego: `c4` para Conecta 4, `co` para Complica y `gr` para Gravity (si no se indica nada, se jugará a Conecta 4). En caso de seleccionar Gravity, se utilizará el tamaño del tablero por defecto de 10x10, a no ser que se diga otra cosa con los parámetros siguientes.
- `-x` o `--tamX` seguido de un número: permite indicar el número de columnas que tendrá el tablero si se eligió jugar a Gravity.
- `-y` o `--tamY`: igual que el anterior pero para indicar el número de filas.
- `-h` o `--help`: solicita a la aplicación que muestre la *ayuda* sobre cómo utilizar los parámetros.

2.3. Mensajes de error personalizados

En la práctica anterior cuando el usuario quería poner en una columna incorrecta, la aplicación mostraba un simple `Movimiento incorrecto`. En esta nueva práctica mejoraremos el texto dependiendo del tipo de error:

- En Conecta 4 y Complica, si el usuario indica una columna que no existe se mostrará "`Columna incorrecta. Debe estar entre 1 y K.`" donde `K` será sustituido por `7` o `4` según corresponda.
- En Conecta 4 si se intenta colocar en una columna llena se escribirá "`Columna llena.`".

[1] Para el análisis de los parámetros se recomienda hacer uso de alguna biblioteca de las disponibles para Java.

- En Gravity, si se intenta colocar en una posición incorrecta (fuera de los límites del tablero), se pondrá "`Posición incorrecta.`".
- Por último, también en Gravity, si se intenta colocar en una posición que ya tiene ficha, se especificará "`Casilla ocupada.`".

3. Ejemplos de ejecución

A continuación mostramos un ejemplo de ejecución. Observa que el texto que aparece en **negrita** representa lo que escribe el usuario, mientras que aparece ***en rojo y cursiva*** todo aquello que la aplicación escribe por la salida de error.

3.1. Parámetros de la aplicación

El primer ejemplo muestra lo que ocurre si solicitamos a la aplicación que muestre la ayuda de uso:

```
%> java [detalles omitidos] tp.pr3.Main -h

usage: tp.pr3.Main [-g <game>] [-h] [-x <columnNumber>] [-y <rowNumber>]
 -g,--game <game>           Tipo de juego (c4, co, gr). Por defecto, c4.
 -h,--help                  Muestra esta ayuda.
 -x,--tamX <columnNumber>   Número de columnas del tablero (sólo para
                            Gravity). Por defecto, 10.
 -y,--tamY <rowNumber>      Número de filas del tablero (sólo para
                            Gravity). Por defecto, 10.
```

Si el usuario utiliza un parámetro no admitido, la aplicación lo indicará con un error y terminará con código de error 1:

```
%> java [detalles omitidos] tp.pr3.Main -z

Uso incorrecto: Unrecognized option: -z
Use -h|--help para más detalles.
```

Algo similar ocurrirá si se especifica un tipo de juego incorrecto:

```
%> java [detalles omitidos] tp.pr3.Main --game noExiste

Uso incorrecto: Juego 'noExiste' incorrecto.
Use -h|--help para más detalles.
```

Para terminar con el catálogo de errores, si se incluyen parámetros adicionales, también se dará un error:

```
%> java [detalles omitidos] tp.pr3.Main -g c4 y otras cosas

Uso incorrecto: Argumentos no entendidos: y otras cosas
Use -h|--help para más detalles.
```

Inicio con una partida de Gravity en un tablero de 10 filas y 8 columnas:

```
%> java [detalles omitidos] tp.pr3.Main -g gr -x 8 -y 10
```

```
|        |
|        |
|        |
|        |
|        |
|        |
|        |
|        |
|        |
|        |
+--------+
 12345678

Juegan blancas
Qué quieres hacer? salir
```

3.2. Ejemplo de salida de la orden Ayuda

```
%> java [detalles omitidos] tp.pr3.Main -g gr -x 13 -y 3

|             |
|             |
|             |
+-------------+
 1234567890123

Juegan blancas
Qué quieres hacer? Ayuda
Los comandos disponibles son:

PONER: utilízalo para poner la siguiente ficha.
DESHACER: deshace el último movimiento hecho en la partida.
REINICIAR: reinicia la partida.
JUGAR [c4|co|gr] [tamX tamY]: cambia el tipo de juego.
JUGADOR [blancas|negras] [humano|aleatorio]: cambia el tipo de jugador.
SALIR: termina la aplicación.
AYUDA: muestra esta ayuda.

|             |
|             |
|             |
+-------------+
 1234567890123

Juegan blancas
Qué quieres hacer? SALIR
```

3.3. Partida sencilla de Gravity

```
%> java [detalles omitidos] tp.pr3.Main -g co

|    |
|    |
|    |
|    |
|    |
```

```
|    |
|    |
+----+
 1234

Juegan blancas
Qué quieres hacer? jugar gr 10 10
Partida reiniciada.
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
+----------+
 1234567890

Juegan blancas
Qué quieres hacer? poner
Introduce la columna: 3
Introduce la fila: 2
|  O       |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
+----------+
 1234567890

Juegan negras
Qué quieres hacer? poner
Introduce la columna: 4
Introduce la fila: 4
|X O       |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
+----------+
 1234567890

Juegan blancas
```

```
Qué quieres hacer? poner
Introduce la columna: 4
Introduce la fila: 4
|X O       |
| O        |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
+----------+
 1234567890

Juegan negras
Qué quieres hacer? poner
Introduce la columna: 3
Introduce la fila: 4
|X O       |
| O        |
|          |
|X         |
|          |
|          |
|          |
|          |
|          |
|          |
+----------+
 1234567890

Juegan blancas
Qué quieres hacer? poner
Introduce la columna: 3
Introduce la fila: 3
|X O       |
| O        |
|  O       |
|X         |
|          |
|          |
|          |
|          |
|          |
|          |
+----------+
 1234567890

Juegan negras
Qué quieres hacer? poner
Introduce la columna: 3
Introduce la fila: 1
Casilla ocupada.
|X O       |
```

```
| O         |
|  O        |
|X          |
|           |
|           |
|           |
|           |
|           |
|           |
+-----------+
 1234567890

Juegan negras
Qué quieres hacer? poner
Introduce la columna: 33
Introduce la fila: 4
Posición incorrecta.
|X O        |
| O         |
|  O        |
|X          |
|           |
|           |
|           |
|           |
|           |
|           |
+-----------+
 1234567890

Juegan negras
Qué quieres hacer? poner
Introduce la columna: 3
Introduce la fila: 4
|X O        |
| O         |
|  O        |
|XX         |
|           |
|           |
|           |
|           |
|           |
|           |
+-----------+
 1234567890

Juegan blancas
Qué quieres hacer? poner
Introduce la columna: 3
Introduce la fila: 4
|X O        |
| O         |
|  O        |
|XXO        |
|           |
```

```
|           |
|           |
|           |
|           |
|           |
+-----------+
 1234567890

Juegan negras
Qué quieres hacer? poner
Introduce la columna: 4
Introduce la fila: 8
|X O        |
| O         |
|  O        |
|XXO        |
|           |
|           |
|           |
|           |
|           |
|   X       |
+-----------+
 1234567890

Juegan blancas
Qué quieres hacer? poner
Introduce la columna: 3
Introduce la fila: 2
|X O        |
| OO        |
|  O        |
|XXO        |
|           |
|           |
|           |
|           |
|           |
|   X       |
+-----------+
 1234567890

Ganan las blancas
```

4. Entrega de la práctica

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual, no más tarde de la fecha y hora indicada en la cabecera de la práctica.

El fichero debe tener al menos el siguiente contenido[2]:

- Directorio `src` con el código de todas las clases de la práctica.
- Fichero `alumnos.txt` donde se indicará el nombre de los componentes del grupo.

[2]Puedes incluir también opcionalmente los ficheros de información del proyecto de Eclipse

Anexo práctica 3: detalles de implementación

Para la implementación de la práctica vas a necesitar implementar nuevas clases e interfaces además de las que existían en la Práctica 3. En concreto utilizaremos una interfaz FactoriaJuego, encargada de crear los objetos que intervienen en el juego. Los métodos abstractos de los que dispone son los siguientes:

```
public abstract ReglasJuego creaReglas();
public abstract Movimiento creaMovimiento(int fila, int col, FICHA color);
public abstract Jugador creaJugadorAleatorio();
public abstract Jugador creaJugadorHumano(Scanner sc);
```

Como tenemos tres tipos distintos de juegos, tendremos tres clases que implementan esta interfaz, que serán FactoriaJuegoConecta4, FactoriaJuegoComplica y FactoriaJuegoGravity. Por ejemplo, la clase FactoriaJuegoConecta4 implementa el método creaReglas como:

```
public ReglasJuego creaReglas() {return new ReglasJuegoConecta4();}
```

De forma similar el método creaJugadorHumano tiene asociado el código:

```
public Jugador creaJugadorHumano(Scanner sc) {return new JugadorHumanoConecta4(sc);}
```

El parámetro Scanner se requiere para que el jugador humano pueda solicitar datos por la consola. Por otro lado ahora existe la posibilidad de tener jugadores de distintos tipos (humanos y aleatorios). Para ello, implementaremos una interfaz (puedes utilizar también una clase abstracta) Jugador, con un único método Movimiento getMovimiento(FactoriaJuego factoria, Tablero tab, FICHA color), que devuelve el movimiento a realizar por el jugador. Esta interfaz es implementada por las clases JugadorAleatorioConecta4, JugadorAleatorioComplica, JugadorAleatorioGravity, JugadorConsolaConecta4, JugadorConsolaComplica y JugadorConsolaGravity. Por ejemplo, la clase JugadorHumanoGravity, implementa el método getMovimiento como:

```
public Movimiento getMovimiento(FactoriaJuego factoria, Tablero tab, FICHA color) {
  System.out.print("Introduce la fila: ");
  int fila = sc.nextInt();
  System.out.print("Introduce la columna: ");
  int columna = sc.nextInt();
  sc.nextLine();
  return factoria.creaMovimiento(fila,columna,color);

}
```

donde sc es un atributo privado de tipo Scanner de la clase JugadorHumanoGravity.

La clase Controlador contiene, entre otros, atributos privados para almacenar la factoria de juego FactoriaJuego factoria, y para almacenar los dos jugadores que intervienen en el juego Jugador jugador1 y Jugador jugador2. Vamos a asumir que el jugador1 siempre juega con las fichas blancas, y el jugador2 juega con las negras. La constructora de Controlador que se invoca desde el método main, tiene tres argumentos: Controlador(FactoriaJuego factoria, Partida partida, Scanner in), que inicializan parte de los atributos de la clase. Los dos jugadores se inicializan en la constructora como jugadores humanos, que serán creados por la factoria. De igual forma la factoria nos permitirá crear las reglas de juego asociadas a la partida.

Con respecto al manejo de excepciones, ahora los métodos booleanos desaparecen y en su lugar devuelven excepciones cuando no puedan realizar su función. Por ejemplo, cuando se va a ejecutar un movimiento correspondiente a una casilla incorrecta, el método correspondiente lanzará una excepción que deberá tratarse de forma adecuada. Controla también, utilizando excepciones, que los datos numéricos son efectivamente numéricos, es decir, que cuando se solicita una columna al usuario, éste no introduzca caracteres no numéricos.