Práctica 4

Herencia, interfaces y excepciones

Inicio: A partir del lunes 21 de marzo
Duración: 3 semanas
Entrega: Semana del 18 de abril
Peso de la práctica: 30%

El objetivo de esta práctica es el diseño de un pequeño juego utilizando técnicas de programación orientada a objetos más avanzadas que en las prácticas anteriores. En concreto, se hará especial énfasis en los siguientes conceptos:

- *interfaces*
- *herencia*
- *excepciones*
- *colecciones*

Juego

La práctica consiste en la implementación de un juego de dos jugadores, que consta de un tablero, celdas y fichas. Los jugadores podrán ir colocando las fichas en las distintas celdas en función de ciertas condiciones. La interfaz de Tablero (`IBoard`) sería la siguiente:

```
public interface IBoard {
	public ICell getCell(int row, int column);
	public int getRows();
	public int getColumns();
	public String toString();
	public void addToken(int row, int column, IToken f) throws ForbiddenToken;
	public List<ICell> getNeighbors(int row, int column);
	public ICell getSymmetric(ICell c);
}
```

Donde el método de *getCell* devolverá la celda situada en la posición recibida por argumento, los métodos *getRows* y *getColumns* devuelven el tamaño del tablero (filas y columnas). El método `addToken` añadirá una ficha en la celda correspondiente, si se puede, y en caso contrario lanzará una excepción de tipo `ForbiddenToken`. El método `getNeighbors` devolverá las celdas vecinas a una posición y el método `getSymmetric` devuelve la celda simétrica de una celda particular. La posición de una celda simétrica a una celda en una posición *f, c* queda determinada por: *TotalFilas-1-f*, *TotalColumnas-1-c*.

Las celdas sólo pueden contener una ficha, que será propiedad de uno de los dos jugadores o de ninguno (en caso de fichas "*muralla*" o "*potenciadoras*", que definiremos posteriormente). Los jugadores pueden añadir a las celdas distintos tipos de fichas pero la ficha siempre deberá ser vecina a la última ficha añadida por dicho jugador (una celda de una esquina tiene 3 vecinas y una del centro del tablero, 8). Para añadir la ficha debe emplearse el método de `addToken` de la interfaz `IBoard`, de forma que si el jugador coloca una ficha en una casilla donde ya había una ficha del otro jugador, la "sobrescribe", es decir, la ficha del jugador original desaparece y se pone una ficha del nuevo jugador. Hay fichas muralla (WA) "neutrales" que no pueden ser sobrescritas por ningún jugador. En el caso de que un jugador intente añadir una ficha en una celda que tenga este tipo de ficha, se deberá lanzar una excepción del tipo `ForbiddenToken`.

A continuación se indican otras dos interfaces que deben implementarse, la de celda (`ICell`) y ficha (`IToken`):

```
public interface ICell {
	public int getRow();
	public int getColumn();
	public IToken getToken();
	public void setToken(IToken f);
	public boolean isNeighbor(ICell c);
}
```

```
public interface IToken {
	public String toString();
	public boolean canBeOverwritten();
}
```

Las fichas tendrán distintos comportamientos, definiendo así la lógica del juego. Por un lado, tenemos las fichas que pertenecen a un jugador, que son: fichas normales (N), fijas (F), y multiplicadoras (M). Las fichas que no pertenecen a ningún jugador serían las potenciadoras (EN) y las fichas muralla (WA) definidas anteriormente.

Las fichas normales, son las que añaden los jugadores de manera general, tienen un valor de 1 y pueden ser sobrescritas por otro jugador. Las fichas fijas, no pueden ser sobrescritas y siguen teniendo un valor de 1. Las fichas multiplicadoras tienen un valor de 3, pero pueden ser sobrescritas. En cuanto a las fichas que no pertenecen a un jugador, están las potenciadoras, que son fichas que cuando un jugador accede a la celda que la contiene, pueden tener tres comportamientos distintos (los tres efectos con la misma probabilidad): 1) Añadir fichas normales del jugador en toda la columna. 2) Añadir fichas normales en toda la fila. 3) Añadir fichas normales en todas las celdas vecinas de la ficha. Una vez que el jugador ha puesto una ficha en la misma celda que una potenciadora, esta ficha potenciadora desaparece y vuelve a aparecer en cualquier celda de forma aleatoria (siempre y cuando no haya una ficha de ningún tipo en dicha posición), desapareciendo de su posición anterior. Como hay siempre dos fichas potenciadoras, cuando el jugador haya puesto una ficha en una celda que contenga una potenciadora y dicha potenciadora se haya comportado de alguna de las formas descritas, ambas desaparecen y vuelven a aparecer en otras dos posiciones aleatorias vacías. Las fichas "muralla", por otro lado, no pueden ser sobrescritas ni tampoco pertenecen a ningún jugador.

Desarrollo del Juego

El juego comienza preparando un tablero cuadrado de tamaño N (mismo número de columnas que de filas) en los que se genera un número de fichas muralla (WA) , de forma simétrica. Es decir, si la celda donde se va a poner la ficha muralla es la (f, c), se colocará otra en la posición (N-1-f, N-1-c). El número de fichas muralla es un parámetro que debe indicar el usuario. De la misma manera, se seleccionará una posición aleatoria donde poner una potenciadora (EN) junto con su simétrica, en dos celdas donde no haya fichas. Es posible que dependiendo de las dimensiones del tablero, la celda de una posición determinada coincida con su simétrica. En ese caso, solo se generaría una potenciadora. La posición inicial de los jugadores será en la (0,0) y en la (N-1, N-1). En este sentido, se asume que la celda (0,0) hace referencia a la esquina superior izquierda y la (N-1, N-1) en la esquina inferior derecha. Indicaremos mediante una excepción llamada `InvalidGame` la posibilidad de que la inicialización del juego sea incorrecta. Dicha excepción saltará si el tamaño del tablero es menor que 5 o si el número de fichas muralla por jugador es mayor o igual que el tamaño del tablero menos el número de fichas muralla por jugador más uno.

El juego funciona por turnos, de forma que cada turno el jugador pone una ficha que obligatoriamente tiene que ser vecina a la última ficha añadida por dicho jugador (en caso contrario saltará una excepción y pedirá al jugador repetir el movimiento). Después de un número de turnos determinado (parámetro del programa), el juego finalizará, dejando como ganador al jugador cuyo valor de fichas sea mayor. Adicionalmente, también finalizará si en lugar de introducir unas coordenadas, se le introduce la cadena "END".

En cada turno, una vez que cada jugador elige una coordenada válida, se genera un número aleatorio, de forma que de manera aproximada un 70% de las veces el usuario disponga de una ficha normal, un 20% de una ficha fija (F) y el 10% restante de una ficha multiplicadora (M). Por otro lado, cada 2 turnos la posición de las fichas "muralla" cambiará de manera aleatoria. Para ello, deberán eliminarse de las anteriores posiciones y añadirse a otras celdas que estén vacías.

Tipos de Fichas (Resumen)

- Normales (N): fichas de jugador que tienen un valor de 1. Pueden ser sobrescritas.
- Fijas (F): fichas de un jugador con valor 1. No pueden ser sobrescritas.
- Multiplicadoras (M): fichas de un jugador que tienen un valor de 3. Pueden ser sobrescritas.
- Potenciadoras (EN): pueden tener los 3 efectos descritos anteriormente cuando un jugador accede a una ficha de este tipo. Una vez que un jugador ha puesto una ficha en una celda que contiene una potenciadora, éstas desaparecen y vuelven a aparecer en una posición aleatoria del tablero que no contenga ninguna ficha de ningún tipo (ella, y su simétrica).
- Muralla (WA): fichas que se generan al inicio del juego y no pueden ser sobrescritas

Excepciones

Habrá una excepción obligatoria (llamada `ForbiddenToken`) que puede saltar a la hora de introducir las fichas. Una excepción de este tipo, o compatible, se lanzará en alguno de estos casos: cuando un jugador quiera colocar una ficha en una celda que contenga alguna ficha que no pueda ser sobrescrita (fijas o muralla), cuando un jugador coloque una ficha en una celda no vecina a la posición de la última ficha que ha añadido o cuando quiera colocar una ficha en una celda que se sale de los márgenes del tablero.

Recuerda añadir excepciones adicionales, si resulta conveniente para obtener un buen diseño.

Apartado 1. Fichas y jugadores: (2 puntos)

En este apartado se definirán de manera correcta las interfaces, las clases y los métodos necesarios para el funcionamiento correcto de las fichas y los jugadores. El siguiente main de prueba contiene dos jugadores con algunas fichas:

```
public class TokensAndPlayersTest {
	public static void main(String args[]) {
		Player fstPlayer = new Player();
		Player sndPlayer = new Player();

		List<IToken> tokensPlayerOne = List.of(new NormalToken(fstPlayer),
							new FixedToken(fstPlayer));
		List<IToken> tokensPlayerTwo = List.of(new MultiplierToken(sndPlayer));

		for (IToken t: tokensPlayerOne)
			System.out.println(t + "  " + t.canBeOverwritten());

		IToken enhancer = new EnhancerToken();
		System.out.println(enhancer + "  " + enhancer.canBeOverwritten());

		IToken wall = new WallToken();
		System.out.println(wall + "  " + wall.canBeOverwritten());


		for (IToken t: tokensPlayerTwo)
			System.out.println(t + " " + t.canBeOverwritten());
	}

}
```

Cuya salida deberá ser la siguiente:

```
N1 true
F1 false
EN true
WA false
M2 true
```

Los números indicados en las fichas anteriores indican el jugador al que pertenece esa ficha. Las fichas potenciadoras (EN) y muralla (WA), al no pertenecer a ningún jugador, no imprimen ningún número asociado.

Apartado 2. Tablero y Celda (2 puntos)

En este apartado se van a definir las clases que implementen la interfaz de tablero y de celda. El tablero deberá recibir por parámetro al menos su tamaño y los dos jugadores. Recuerda que siempre debe haber fichas potenciadoras (si se puede) en posiciones simétricas. Es importante señalar que en cada turno se debe imprimir el tablero con todas las celdas indicando las que están vacías y las que tiene una ficha. Si dicha ficha es de jugador, deberá concatenar el número del jugador al lado. Para imprimir el tablero es obligatorio emplear las mismas cadenas para las fichas que las mostradas en este documento (WA para las "muralla", F para las fijas, N para las normales, M para las multiplicadoras y EN para las potenciadoras). Los guiones "--" se emplearán para las celdas vacías. En el caso de las fichas que pertenezcan a un jugador, será obligatorio mostrar el jugador al que pertenece dicha ficha.

Puedes emplear esta clase de prueba para comprobar que la inicialización del tablero es correcta, que las fichas potenciadoras se colocan en posiciones simétricas y qué ocurre cuando se introducen fichas en celdas ya ocupadas.

```
public class CellAndBoardTest {
	public static void main(String args[])  {
		Player p1 = new Player();
		Player p2 = new Player();
		IBoard board = new BoardGame(10, p1, p2);
		System.out.println(board);
		try {
			board.addToken(0, 0, new FixedToken(p1));
			System.out.println(board);
			board.addToken(0, 0, new FixedToken(p2));
		} catch (ForbiddenToken e) {
			System.out.println(e);
		}
		System.out.println(board);

	}
}
```

Ejemplo de salida esperada: ten en cuenta el mensaje de la excepción, que salta cuando se intenta poner una ficha en la posición 0,0 después de que se haya puesto una ficha fija. Las coordenadas de las fichas potenciadoras (EN) pueden cambiar.

```
N1  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  EN  --  --
--  --  EN  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  N2

F1  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  EN  --  --
--  --  EN  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  N2

Error.  the token F2 can not be placed in position 0,0 because: the cell contains a token that cannot be overwritten
F1  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  EN  --  --
--  --  EN  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  --
--  --  --  --  --  --  --  --  --  N2
```

Apartado 3. Juego (5 puntos)

En este apartado debes crear la clase `Game`, que tendrá (al menos) un método llamado `play` que controlará el juego por turnos. La entrada de las coordenadas donde colocar las fichas se hará por teclado, siendo obligatorio emplear la notación f,c para indicar la fila y la columna donde colocar la ficha. Por ejemplo, si el jugador introduce 0,4, se estará indicando que el jugador ha introducido una ficha en la fila 0, columna 4.

El final de juego llegará cuando se llegue al máximo de turnos o se introduzca la cadena de "END". Si el jugador no puede colocar ninguna ficha, deberá pasar turno al otro jugador y en caso de que ninguno de los dos pueda poner ninguna ficha, entonces el juego terminará (el juego se acaba por imposibilidad de poder seguir colocando fichas). Adicionalmente, cada turno, se indicará la puntuación de cada jugador y la posición de la última ficha añadida.

El siguiente listado contiene un programa de prueba. El método `play` de la clase `Game` será el encargado de procesar el turno de cada jugador:

```
 public class GameMain {
   public static void main(String args[])  {
               try {
                      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

                       System.out.println("Indicate the size of the board");
                       int size = Integer.parseInt(br.readLine());

                       System.out.println("Indicate the maximum turns");
                       int turns = Integer.parseInt(br.readLine());

                       System.out.println("Indicate the number of wall tokens per user");
                       int wallTokens = Integer.parseInt(br.readLine());

                       Game j = new Game(turns, size, wallTokens);
                       j.play();
               } catch (NumberFormatException nfe) {
                         System.out.println("Incorrect format");
               } catch (InvalidGame e) {
                         System.out.println(e);
               } catch (IOException e) {
                         System.out.println("Error reading the values");
               }

    }
}
```

Y un posible resultado:

```
Indicate the size of the board
5
Indicate the maximum turns
2
Indicate the number of wall tokens per user
1
Starting turn 1
N1  EN  --  --  WA
--  --  --  --  --
--  --  --  --  --
--  --  --  --  --
WA  --  --  EN  N2

1: points 1.0 last cell: 0,0
2: points 1.0 last cell: 4,4
Turn 1
Player 1 enter the coordinates of the new token:
0,1
N1  N1  N1  N1  WA
--  --  --  --  EN
--  --  --  --  --
EN  --  --  --  --
WA  --  --  --  N2

1: points 4.0 last cell: 0,1
2: points 1.0 last cell: 4,4
Turn 1
Player 2 enter the coordinates of the new token:
3,4
N1  N1  N1  N1  WA
--  --  --  --  EN
--  --  --  --  --
```

```
EN  --  --  --  M2
WA  --  --  --  N2

1: points 4.0 last cell: 0,1
2: points 4.0 last cell: 3,4
Starting turn 2
N1  N1  N1  N1  --
--  --  --  --  EN
--  WA  --  WA  --
EN  --  --  --  M2
--  --  --  --  N2

1: points 4.0 last cell: 0,1
2: points 4.0 last cell: 3,4
Turn 2
Player 1 enter the coordinates of the new token:
1,1
N1  N1  N1  N1  --
--  N1  --  --  EN
--  WA  --  WA  --
EN  --  --  --  M2
--  --  --  --  N2

1: points 5.0 last cell: 1,1
2: points 4.0 last cell: 3,4
Turn 2
Player 2 enter the coordinates of the new token:
3,3
N1  N1  N1  N1  --
--  N1  --  --  EN
--  WA  --  WA  --
EN  --  --  M2  M2
--  --  --  --  N2

1: points 5.0 last cell: 1,1
2: points 7.0 last cell: 3,3
```

Como se puede observar, a la hora de comenzar el turno 2, la posición de las fichas muralla ha cambiado. Por otro lado, si por ejemplo en lugar de una ficha muralla por jugador introdujeramos un valor de dos 2 fichas muralla por jugador, saltará la excepción anteriormente definida, ya que el número de fichas muralla por jugador (2) es mayor o igual al tamaño del tablero (5) - 3 (número de fichas muralla + 1).

```
Indicate the size of the board
5
Indicate the maximum turns
2
Indicate the number of wall tokens per user
2
Error.  Invalid board. Number of rows and columns must be 5. Number of wall tokens must not exceed 1
```

Describe en la memoria algunas pruebas adicionales que has ejecutado para comprobar que el programa cumple con la funcionalidad requerida.

Apartado 4. Lectura de ficheros: (0,5 puntos)
En este apartado vamos a habilitar la inicialización del juego leyendo los datos a partir de un fichero de texto. Para ello, será necesario hacer funcionar el siguiente main:

```
public class GameMainFromFile {
	public static void main(String args[]) {
		Game g = GameLoader.load(args[0]);
		if (g != null)
			g.play();
	}
}
```

El método `load` devolverá null en caso de que falle la inicialización del tablero. Este sería un ejemplo de fichero que se podría enviar por argumento (si se quiere conseguir la misma inicialización del apartado anterior):

```
5
2
1
```

El primer número indica el tamaño del tablero (5x5 en este caso), la segunda línea el número de turnos (2) y la tercera línea indica el número de fichas "muralla" por cada jugador (1). La salida sería la siguiente (solo se indica la inicialización del tablero):

```
Starting turn 1
N1  --  EN  --  --
--  --  --  --  WA
--  --  --  --  --
WA  --  --  --  --
--  --  EN  --  N2

1: points 1.0 last cell: 0,0
2: points 1.0 last cell: 4,4
Turn 1
Player 1 enter the coordinates of the new token:
```

Además de este ejemplo básico, será necesario hacer pruebas adicionales leyendo diferentes valores de inicialización del tablero, para comprobar que el programa procesa todas las posibles excepciones de manera correcta. Será necesario describir las pruebas realizadas.

Apartado 5. Cuestiones (0,5 puntos)
Responde a las siguientes preguntas:

1) Imagina que te piden realizar otro juego de tablero (por ejemplo, ajedrez o damas). ¿Qué clases o interfaces podrías reutilizar? ¿Habría que añadir algún método adicional?

2) Supón que creamos una ficha adicional cuyo valor es temporal. Es decir, cuando se coloca por primera vez tiene un valor de 3 puntos pero después de 3 turnos, su valor vuelve a ser 1. ¿Cómo incluirías esta nueva funcionalidad? No hace falta programar nada, solo indicar cómo hacerlo..

Notas:

- La justificación de esas cadenas para identificar las fichas es debido a que las fichas "muralla" y las potenciadoras no tienen jugador asociado (por esa razón se identifican con dos caracteres). Las fichas de jugador se identifican con una letra ya que el segundo carácter debe ser el identificador del jugador (1 o 2 en su caso).
- En esta práctica resulta imprescindible realizar un buen diseño. En la memoria será importante explicar las decisiones de diseño tomadas y las clases/interfaces auxiliares que se han creado.
- La estructuración de paquetes de la práctica se deja a criterio del estudiante, pero como mínimo, se debe seguir esta jerarquía: **ads.p4**. Dentro del directorio p4 se deja libertad para organizar el código, pero se penalizará disponer de todas las clases e interfaces en el mismo paquete.

Normas de Entrega:

- Se deberá entregar
  - un directorio **src** con todo el código Java,
  - un directorio doc con la documentación **Javadoc** generada para todas las clases e interfaces.
  - un directorio **txt** con los ficheros de pruebas empleados
  - un documento en **PDF** con una breve justificación de las decisiones que se hayan tomado en el desarrollo de la práctica, los problemas principales que se han abordado y cómo se han resuelto, así como los problemas pendientes de resolver. El documento pdf deberá incluir la descripción de las pruebas realizadas
  - el documento PDF incluirá el **diagrama de clases** del sistema y las respuestas a las cuestiones planteadas.
- Se debe entregar un único fichero ZIP con todo lo solicitado, que deberá llamarse de la siguiente manera: P4_GR<numero_grupo>_<nombre_estudiantes>.zip. Por ejemplo Marisa y Pedro, del grupo 2261, entregarán el fichero: GR2261_MarisaPedro.zip.