Tarea 2 ILI-253
Lenguajes de Programación
Primer semestre, 2014

Francisco Bórquez
francisco.borquez@usm.cl

Álvaro Hernández
alvaro.hernandezi@alumnos.usm.cl

Teodoro Saavedra
teodoro.saavedra@alumnos.usm.cl

24 de abril de 2014

1. Objetivos

- *Utilizar conceptos de polimorfismo, herencia e interfaces en un lenguaje de programación orientado a objetos como Java.*
- *Analizar y crear reglas programáticas para la construcción de software.*
- *Introducir a los estudiantes al concepto de patrones de diseño.*

2. Tarea

Los alumnos deberán construir un juego siguiendo las reglas especificadas en este documento. El juego consta en destruir bloques de distintos colores que estén en fila de 3 o más, hasta alcanzar una cantidad determinada de bloques destruidos.

3. Reglas del Juego

El juego presentara una pantalla de presentación donde el jugador presionara alguna tecla e iniciara el juego. Luego se le pedirá al jugador la cantidad de bloques que deberá romper de cada color para poder finalizar el juego. Estos colores son el color rojo que estará representado por la letra R, el color azul que estará representado por la letra B, el color naranjo que estará representado por la letra O, el color verde que estará representado por la letra G y el color amarillo que estará representado por la letra Y. Además existirán bloques comodines, los cuales son de tipo 1 y de tipo 2, que estarán representados por $ y & respectivamente.

El tablero estará lleno de estos distintos bloques, el tamaño de dicho tablero será de 15x15 bloques. Para destruir bloques deberán estar 3 o más bloques del mismo color alineados, ya sea, horizontal o verticalmente. Esto se lograra por medio del intercambio temporal de la posición de dos bloques, donde, si es que se logra la destrucción de bloques, es decir, alinear los 3 o más bloques, este intercambio se hará permanente. Los bloques de comodín no tienen color pero permiten ser usados como relleno para poder destruir bloques y así al ser destruidos activar su habilidad.

La habilidad de los bloques al ser destruidos dependiendo de su tipo, los de tipo 1 al ser destruidos rompen toda la fila en la que se encuentran junto con él, en cambio los de tipo 2 rompen toda la columna.

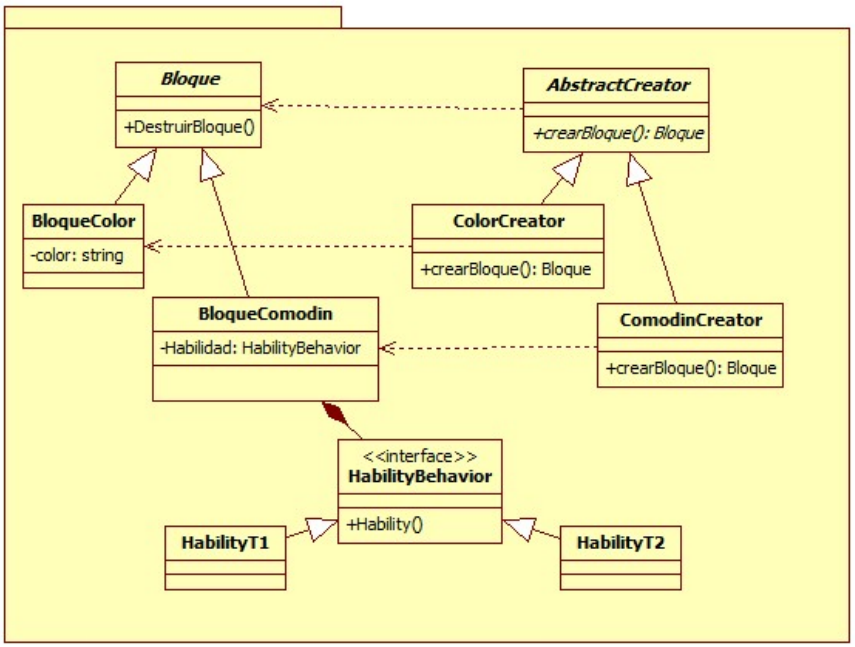
Cuando una columna de bloques no está llena debe ser rellenada con bloques ingresados desde la parte alta de

la columna, es decir, que estás tiene un tipo de funcionamiento parecido al de las pila. El llenado de las columnas no llenas, como también el del tablero al principio del juego, son completamente al azar, todos los colores tiene la misma posibilidad de aparecer, con excepción de un bloque comodín, que tiene 5 % de probabilidad de aparecer, pudiendo ser cualquiera de sus dos tipos. Por tanto, un bloque tiene 95 % de ser un bloque de color, pudiendo ser cualquiera de los distintos colores, y 5 % de ser un bloque comodín, pudiendo ser también de cualquiera de sus tipos.

Luego de romper la cantidad de bloques ingresada al principio del juego, este se da por finalizado. Por otro lado si es que el jugador no tiene más jugadas por hacer, lo que el juego lo debe verificar por sí solo, este remezclara los bloques que están en el tablero. Además se deberá mostrar la cantidad de bloques rotos de cada color como también la cantidad requerida para terminar el juego.

4. Desarrollo

- Deberán implementar el siguiente modelo:



- El modelo consta de dos patrones de diseño, los cuales son **Factory Method** y **Strategy**, y deberán seguir los conceptos detrás de estos.
- Todos los atributos de las clases deben tener sus *getters* y *setters* correspondientes.
- Toda función y clase en cursiva es *abstract*.
- Los tipos y atributos especificados son obligatorios. Se pueden crear y extender según sea necesario. El cambio de alguno de estos debe estar bien justificado en el README.
- Se deben usar las interfaces especificadas.
- El programa debe incluir un ANT para su compilado.

5. Archivos a entregar

- Bloque.java

- BloqueColor.java
- BloqueComodin.java
- AbtractCreator.java
- ColorCreator.java
- ComodinCreator.java
- HabilityBahavior.java
- HabilityT1.java
- HabilityT2.java
- build.xml
- README.txt

6. Sobre la entrega

- La revisión se realizara usando entorno Linux.
- El código debe venir identado y sin warnings.
- Cada función debe llevar una descripción según lo siguiente:

```
/******** Funcion: Nombre_Funcion **************
Descripcion: Funcion en cargada de...
Parametros:
n1 entero
n2 entero
Retorno: Retorna...
**********************************************/
```

- Debe estar presente y **funcionando** el archivo ANT para que se efectúe la revisión.
- No se aceptaran proyectos de Eclipse u otro entorno de desarrollo.
- Se debe trabajar en grupos de dos personas.
- La entrega debe realizarse en un archivo **.tar.gz** y debe llevar el nombre **Tarea2_Rol-1_Rol-2**.
- El archivo README.txt debe contener el nombre y rol de los integrantes del grupo, como también la forma en que se debe jugar el juego, como también observaciones y cambios que sean necesarios mencionar.
- La entrega se debe entregar antes de las **23.55 hrs**. del dia **Sábado 10 de Mayo del 2014**.
- Por cada dia de atraso se descontaran 20 puntos.
- Las copias serán evaluadas con nota 0.