Programación I
Grado en Informática
Curso 2019 / 2020

Práctica obligatoria – 2ª Convocatoria Ordinaria

Tabla de contenidos

1. Descripción de la aplicación

CandyClean es un juego que parte de un tablero cuadrado con casillas de colores y tiene como objetivo llegar a vaciar el tablero. Para ello, se deberán ir seleccionando casillas del tablero perteneciente a una fila y una columna en la que haya más de una casilla contigua a la seleccionada del mismo color en horizontal o en vertical. La aplicación ofrecerá información sobre el estado del tablero en todo momento.

Utilizando una interfaz de tipo texto, un jugador intentará vaciar el tablero, lo que no siempre es posible. La aplicación permitirá ir eliminando casillas del tablero mediante su selección por parte del usuario. La eliminación de casillas implicará siempre la operación de compactar el tablero haciendo que las casillas superiores ocupen las posiciones de las casillas eliminadas.

La aplicación termina cuando el usuario gana, es decir, cuando consigue vaciar el tablero, o cuando el usuario pierde, es decir, cuando ya no quedan parejas contiguas de casillas del mismo color en la misma fila o en la misma columna.

2. Funcionamiento de la aplicación

2.1. Configuración inicial

La aplicación recibe como parámetros de entrada el tamaño del lado del tablero (número entero) y el número de colores de las casillas del tablero (número entero). El lado del tablero deber ser mayor o igual a 5 y el número de colores debe ser mayor o igual que 3 y menor o igual que 7. En caso de que el usuario no respete la sintaxis de la aplicación, ésta deberá emitir un mensaje de error lo más detallado posible respecto al error cometido y mostrará el siguiente mensaje antes de terminar la ejecución:

```
Sintaxis del programa:
MainCandyClean dimension numColors
```

2.2. Ejemplo de funcionamiento de la aplicación

Una vez arrancada la aplicación, se le muestra al usuario la situación inicial en la que aparece el estado inicial del tablero y se solicita introducir valores de fila y columna para seleccionar una casilla. Un ejemplo de ejecución del juego sería el siguiente:

```
java –cp classes es.unileon.prg1.candyClean.MainCandyClean 5 3
```

Ejecución 1: El usuario pierde

```
 |0|1|2|3|4|
0
1
2
3
4
Points: 0
Row:
0
Column:
```

```
4
 |0|1|2|3|4|
0
1 BBBBBBBB
2 BBBBBBBBBB
3 GGGGBBBBGG
4 BBBBGGGGGG
Points: 6
Row:
2
Column:
2
 |0|1|2|3|4|
0
1
2 BBBB  BB
3 GGGG  BBGG
4 BBBBGGGGGG
Points: 13
Row:
3
Column:
0
 |0|1|2|3|4|
0
1
2    BB
3 BBBB  BBGG
4 BBBBGGGGGG
Points: 15
Row:
4
Column:
0
 |0|1|2|3|4|
0
1
2    BB
3    BBGG
4  BBGGGGGG
Points: 18
Row:
4
Column:
4
 |0|1|2|3|4|
0
1
2
3    BB
4  BB  BB
Points: 22
Row:
4
Column:
3
 |0|1|2|3|4|
0
1
2
3
4  BB
Points: 24
You loose!
```

Ejecución 2: El usuario gana

```
  |0|1|2|3|4|
0
1
2
3
4
Points: 0
Row:
0
Column:
4
  |0|1|2|3|4|
0
1
2
3
4
Points: 6
Row:
2
Column:
3
  |0|1|2|3|4|
0
1
2
3
4
Points: 13
Row:
3
Column:
0
  |0|1|2|3|4|
0
1
2
3
4
Points: 15
Row:
4
Column:
0
  |0|1|2|3|4|
0
1
2
3
4
Points: 18
Row:
4
Column:
4
  |0|1|2|3|4|
0
1
2
3
4
Points: 22
Row:
4
Column:
```

```
2
 |0|1|2|3|4|
0 
1 
2 
3 
4 
Points: 25
You win!
```

2.3. Log de la aplicación

Durante cada ejecución de la aplicación se crea un fichero de registro con la información generada. El contenido de este fichero deberá ser tal que, con la menor cantidad de información posible, nos permita reproducir el desarrollo del juego fielmente.

Para generar los ficheros de registro se deberá utilizar obligatoriamente `log4j2`. No se admitirá ninguna otra solución. El uso directo de ficheros de texto queda excluido.

2.4. Tratamiento de errores

La aplicación deberá llevar a cabo la gestión de errores a través de excepciones. En concreto, se deberán tener en cuenta todos los casos en los que el usuario introduzca valores no válidos para la dimensión, el número de colores, la fila y la columna.

2.5. Condiciones de terminación

La aplicación deberá comprobar cada vez que el usuario introduce una fila y una columna si el juego ha llegado a su fin. Además, deberá emitir un mensaje al finalizar la partida indicando si el jugador ha ganado o ha perdido y la puntuación alcanzada. Por cada casilla eliminada se sumará un punto a la puntuación final.

2.6. Casos de prueba

Se deberán llevar a cabo tests para comprobar el correcto funcionamiento de todo el código desarrollado. Para ello, se deberá incluir en la aplicación el código necesario para generar una situación inicial para los test idéntica a la planteada en el apartado 2.2, y completarla con las situaciones necesarias para comprobar el correcto funcionamiento de la aplicación.

3. Ejecución de la aplicación

La ejecución del programa debe hacerse *obligatoriamente* de la siguiente forma:

```
java –cp classes es.unileon.prg1.candyClean.MainCandyClean dimension numColors
```

donde:

- `dimension` indica el número de casillas que tiene el tablero por filas y por columnas
- `numColors` es el número de colores de las casillas del tablero

La ejecución de la aplicación deberá generar un tablero de la dimensión especificada en la ejecución rellenado con casillas del número de colores especificados de manera aleatoria.

4. Material proporcionado

Se ofrece como material inicial la clase `Color` para el tratamiento de los caracteres gráficos con color, junto con el tipo enumerado `BackgroundColor` en el que se incluyen los ocho colores disponibles para el juego, de los cuales siete se pueden utilizar para las casillas, ya que el negro se asigna a las casillas vacías.

5. Normas de entrega

- La fecha límite de entrega será el día anterior al examen de la asignatura, al mediodía (30 de enero de 2020 a las 12:00 horas).
- La práctica se encontrará en el correspondiente repositorio de GitHub y contendrá la siguiente estructura de subdirectorios:
  - `src`: ficheros con el código fuente (ficheros `.java`)
  - `test`: ficheros con el código fuente de los tests de `JUnit`
  - `classes`: ficheros compilados (ficheros `.class`) – **No incluido en GitHub (generado por el `build.xml`)**
  - `lib`: librerías utilizadas
  - `doc`: documentación generada por `javadoc` (ficheros `.html`) – **No incluido en GitHub (generado por el `build.xml`)**
  - `etc`: `build.xml` y diagrama de clases UML (archivo generado con el dia http://projects.gnome.org/dia/ o con http://draw.io **en formato `.jpg`**)
  - `log`: Ficheros de registro – **No incluido en GitHub (generado por el `build.xml`)**
- El código fuente de todos los ficheros de la práctica deberá incluir los nombres de los autores, utilizando la etiqueta `@author` de `javadoc`.
- El código estará incluido en el paquete `es.unileon.prg1.candyClean`
- Es *condición imprescindible* para aprobar la práctica que los ficheros `.java` entregados compilen correctamente, es decir, sin generar errores.
- Es *condición imprescindible* para aprobar la práctica que los ficheros `.class` generados a partir de los fuentes entregados ejecuten correctamente, es decir, no den errores de ejecución y ofrezcan la funcionalidad requerida.
- Es *condición imprescindible* para aprobar la práctica la entrega de la documentación correctamente generada, es decir, habiéndola generado haciendo uso de las opciones necesarias para que se genere documentación de todas las clases, atributos y métodos.
- Es *condición imprescindible* para aprobar la práctica que se incluya el correspondiente fichero `build.xml` con los target necesarios para compilar, generar la documentación, probar (pasar los tests) y ejecutar la práctica, en ese orden. Cada uno de los targets deberá establecer las debidas dependencias respecto de los demás (incluido el target dedicado a limpiar).

- Es *condición imprescindible* para aprobar la práctica que genere ficheros de registro de cada ejecución que permitan reproducir cada una de las ejecuciones.
- Es *condición imprescindible* para aprobar la práctica que se incluyan tests de JUnit para cada una de las clases desarrolladas que comprueben la funcionalidad de cada clase desarrollada. **La cobertura del código debe ser del 100%. No se llevarán a cabo tests de las clases Teclado, MainCandyClean (clase que contiene el main), y de la clase encargada de la interfaz de usuario.**
- Es *condición imprescindible* realizar *commits* a medida que la práctica evoluciona. No se considerarán las prácticas con actividad en el repositorio solo en los últimos días previos a la entrega.