EXAMEN DE PROGRAMACIÓN
GRADO EN INGENIERÍA INFORMÁTICA
Leganés, Junio de 2015

Universidad Carlos III de Madrid

LEA ATENTAMENTE ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- **Rellene todas las hojas a bolígrafo, no utilice lápiz ni bolígrafo rojo**
- **El tiempo máximo de realización es de 3 horas**
- **Se permiten apuntes y/o libros para la realización del examen. No se permiten dispositivos electrónicos de ningún tipo**

El objetivo del examen es crear un programa para jugar a un juego con cierto parecido al dominó. Tenemos una serie de fichas rectangulares que tienen un color en cada extremo, de entre cinco posibles (**R**ojo, **V**erde, **A**zul, **M**arrón o **B**lanco). El número de fichas se le pedirá al usuario y los colores de cada uno de los dos lados se generarán pseudo-aleatoriamente de forma que el problema tenga solución (en el enunciado se darán los detalles). Al iniciar la partida las fichas se colocan en línea aleatoriamente. El usuario debe colocar las fichas de forma que el color de la parte derecha de una ficha sea el mismo que el de la parte izquierda de la siguiente ficha y que la línea de fichas comience y termine por Blanco, es decir, la ficha al inicio de la línea tendrá la parte izquierda blanca, mientras la del final de la línea tendrá blanco en la parte derecha. De esta forma todas las fichas estarán conectadas por su color. A continuación se puede ver un ejemplo de partida con 5 fichas. Se muestra la inicial del color de cada ficha, si dos fichas están conectadas (coincide el color de la derecha de una ficha con el de la izquierda de la siguiente), la inicial se muestra en mayúsculas, si no están conectadas, la inicial se muestra en minúsculas:

```
¿Número de fichas?
5
Movimientos: 0
[B m][v b][m B][B v][m m]
¿Origen?
3
¿Destino?
0
Movimientos: 1
[B v][b m][v b][m b][m m]
¿Origen?
2
¿Destino?
1
Movimientos: 2
[B V][V B][B M][M b][m m]
¿Origen?
4
¿Destino?
5
¡Posición incorrecta!
Movimientos: 2
[B V][V B][B M][M b][m m]
¿Origen?
3
¿Destino?
4
Movimientos: 3
[B V][V B][B M][M M][M B]
¡Has ganado!
```

Pregunta 1 (0,3 puntos).- Crear un tipo enumerado, denominado `Color` para guardar los posibles colores de una ficha.

```
enum Color {ROJO, AZUL, VERDE, MARRON, BLANCO};
```

Pregunta 2 (2,5 puntos).- Crear la clase `Ficha` con las siguientes características:

a) **(0,2 puntos)** Atributos privados: dos arrays de dos posiciones, uno denominado `colores` para guardar los colores de cada parte de la `Ficha` y otro, denominado `conectada`, para guardar si la `Ficha` está conectada con la de la derecha o con la de la izquierda. Elegir el tipo más adecuado. Asumir que la posición 0 de cada array representa la izquierda de la `Ficha` y la 1 la derecha.

b) **(0,2 puntos)** Un método `setColor` que no recibe parámetros y que pone ambos lados de la `Ficha` en blanco (el color será `BLANCO` para ambos)

c) **(0,3 puntos)** Un método `setColor` que recibe un array de `Color` y pone la `Ficha` con esos colores. Si el array no tiene el tamaño adecuado se deberá imprimir un mensaje de error por pantalla y poner la `Ficha` en blanco.

d) **(0,3 puntos)** Un constructor que recibe un array de `Color` y crea la `Ficha` con esos colores. Si el array no tiene el tamaño adecuado se deberá imprimir un mensaje de error por pantalla y poner la `Ficha` en blanco.

e) **(0,2 puntos)** Un constructor sin parámetros que crea una `Ficha` en blanco.

f) **(0,2 puntos)** Un método `getColor` que recibe un entero y devuelve el `Color` de la parte izquierda o derecha de la `Ficha`, según el número recibido.

g) **(0,2 puntos)** Métodos `get` y `set` para el atributo `conectada`.

h) **(0,3 puntos)** Un método `getInicialColor`, que reciba un `Color` y devuelva su inicial.

i) **(0,6 puntos)** Un método `toString` que devuelva una cadena que representa a la `Ficha` tal y como aparece en el ejemplo de la primera página. Por ejemplo para una `Ficha` Roja-Verde que está conectada con la anterior pero no con la posterior, devolverá `[R v]`

```
public class Ficha {
	private Color [] colores;
	private boolean [] conectada;


	/** Este método pone la ficha en blanco (los 2 colores son BLANCO) */
	public void setColor() {
		for (int ii=0; ii<2; ii++){
			colores [ii] = Color.BLANCO;
			}
		//Dado que solo son dos colores también se podría haber hecho
		//colores[0] = Color.BLANCO;
		//colores[1] = Color.BLANCO;
	}

	public void setColor (Color [] c){
		if (c.length == 2) {
			colores[0] = c[0];
			colores[1] = c[1];
			//También podríamos haber hecho directamente colores = c
			//pero entonces ambos arrays serían el mismo
```

```
                }
            else {
                System.out.println("Número incorrecto de colores. Se creará una ficha en
blanco");
                setColor();
            }
        }

        public Ficha (Color [] c){
            //Creamos el array
            //Como sabemos que siempre tendrá dos elementos, lo podríamos
            //haber hecho también en la declaración
            colores = new Color[2];
            //Llamamos al método setColor que ya hace las comprobaciones
            setColor(c);
            conectada = new boolean[2];
        }


        public Ficha () {
            colores = new Color [2];
            //Llamamos al método setColor sin parámetros, ya que lo tenemos
            setColor();
            conectada  = new boolean[2];
            }

        public Color getColor (int pos){
            //Como no nos dicen que comprobemos si el número es correcto no lo hacemos
            return colores[pos];
            //Una implementación mejor: si el número no es correcto
            //devolvemos el color del lado derecho
            //if (pos == 0) return colors[pos];
            //else return colors[1];
        }

        public boolean [] getConectada(){
            return conectada;
        }
        public void setConectada (boolean [] c){
            //No nos piden comprobar el tamaño del array
            conectada [0]=c[0];
            conectada [1]=c[1];
        }

        private String getInicialColor (Color color){
            switch (color){
            case ROJO: return "R";
            case AZUL: return "A";
            case VERDE: return "V";
            case MARRON: return "M";
            //Si no incluimos el default, ¡no compilará!
            //Si no sabes por qué, pregunta en el foro
            default: return "B";
            }
        }

        public String toString (){
            String resultado="";
            String izquierda = getInicialColor(colores[0]);
            String derecha = getInicialColor(colores[1]);
            //Para ver si deben ir en mayúscula o minúscula, miramos el array conectada
            if (!conectada[0])
```

```
                izquierda = izquierda.toLowerCase();
            if (!conectada[1])
                derecha = derecha.toLowerCase();
            resultado = "["+izquierda+" "+derecha+"]";
            return resultado;
        }
}
```

Pregunta 3 (6,2 puntos).- Crear la clase `Tablero` con las siguientes características:

a) **(0,2 puntos)** Atributos privados: `fichas`, de tipo array de `Ficha`, y `movimientos`, cuyo tipo debe ser elegido por el alumno.

b) **(0,3 puntos)** Un constructor que reciba el número de `fichas` del `Tablero` y cree el array (pero no las fichas). Si el número es menor de 2, creará un `Tablero` de 10 fichas.

c) **(0,5 puntos)** Un método `obtenerColorAleatorio`, que devuelva un `Color` de entre los posibles, elegido aleatoriamente.

d) **(1 punto)** Un método `obtenerColorFichaAnterior` que recibe la posición de una `Ficha` y devuelve el `Color` de la derecha de la ficha anterior. Si la posición de la `Ficha` es la primera (y por lo tanto no tiene `Ficha` anterior) devolverá `BLANCO`. Hacer también un método `obtenerColorFichaSiguiente` que devuelva el color de la izquierda de la `Ficha` posterior. Si es la última, devolverá `BLANCO`

e) **(1 punto)** Un método `rellenarTablero`, que crea las `fichas` del array de forma que la partida sea resoluble. Para ello, la primera `Ficha` tendrá `BLANCO` en su izquierda y un color aleatorio en la derecha. La segunda, a su izquierda el mismo color que la primera tenía a la derecha y un color aleatorio a la derecha, y así sucesivamente hasta la última que tendrá `BLANCO` a su derecha. Ejemplo con 5 fichas:

`[B M][M R][R B][B V][V B]`

f) **(1 punto)** Un método `comprobarConexiones` que comprueba todas las `fichas` mirando para cada `Ficha` si está conectada con la de su izquierda y la de su derecha, rellenando el array `conectada` de esa `Ficha` convenientemente. Si todas las fichas están conectadas, devolverá `true`, si hay alguna sin conectar, devolverá `false`.

g) **(0,2 puntos)** Un método `comprobarPosicion` que recibe un entero y comprueba si es una posición válida del array de `fichas`. Devolverá `true` o `false`, según sea válida o no. Si no es válida imprimirá además por pantalla: `¡Posición incorrecta!`

h) **(1 punto)** Un método `moverFicha` que recibe la posición inicial de la `Ficha` y la posición de destino. Primero comprueba si ambas posiciones son correctas, si no lo son, no hace nada. Si lo son coloca la `Ficha` de la posición inicial en la posición de destino, **moviendo** el resto convenientemente. Ejemplo:

Si tenemos `[B M][M V][V R]` y aplicamos `moverFicha(0,2)` tendremos `[M V][V R][B M]`

Debe funcionar tanto si el origen es menor que el destino como si es mayor (tal y como se muestra en el ejemplo de la primera página).

Cada vez que se haga un movimiento se debe incrementar en 1 el atributo `movimientos`.

i) **(0,8 puntos)** Un método `barajarFichas` que **intercambie** aleatoriamente tantas `fichas` como longitud tiene el array `fichas`, de forma que lo desordene. Al contrario que el

método anterior este método tiene que intercambiar las fichas dos a dos, sin mover el resto de fichas no implicadas en el cambio. Ejemplo:

Si tenemos `[B M][M V][V R]` e intercambiamos la 0 y la 2 tendremos `[V R][M V][B M]`

En este caso, como tenemos 3 fichas habría que hacer tres intercambios aleatorios como el mostrado.

j) **(0,2 puntos)** Un método `toString` que devuelva el número de movimientos y el estado actual del `Tablero`, tal y como muestra el ejemplo de la primera página.

```
public class Tablero {
	private Ficha [] fichas;
	private int movimientos;

	public Tablero (int longitud){
		if (longitud <2)
			//Aquí estamos creando el array, pero no los objetos que contiene
			fichas = new Ficha[10];
		else
			fichas = new Ficha [longitud];
	}

	public Color obtenerColorAleatorio () {
		int color = (int) (Math.random()*5);
		switch(color){
		case 0: return Color.ROJO;
		case 1: return Color.AZUL;
		case 2: return Color.VERDE;
		case 3: return Color.MARRON;
		//Si no incluimos el default, ¡no va a compilar!
		//Si no sabes por qué, pregúntalo en el foro
		default: return Color.BLANCO;
		}
	}

	public Color obtenerColorFichaAnterior (int numero){
		Color anterior;
		//Si la ficha es la primera tiene que ser BLANCO
		//Con esta condición, de paso devolvemos BLANCO para cualquier
		//número de ficha no válido
		if (numero<1 || numero>=fichas.length)
			anterior = Color.BLANCO;
		else
			anterior = fichas[numero-1].getColor(1);
		return anterior;
	}

	public Color obtenerColorFichaSiguiente (int numero){
		Color siguiente;
		if (numero<0 || numero>=fichas.length-1)
			siguiente = Color.BLANCO;
		else
			siguiente = fichas[numero+1].getColor(0);
		return siguiente;
	}


	public void rellenarTablero (){
		for (int pos = 0; pos<fichas.length; pos++){
			//Como el constructor de Ficha recibe un array de color
```

```
                //creamos un array auxiliar
                Color [] aux = new Color [2];
                //Rellenamos el color de la izquierda con el de la anterior
                aux[0] = obtenerColorFichaAnterior(pos);
                //Para el de la derecha, miramos primero si es la última
                if (pos== fichas.length-1) {
                        aux[1] = Color.BLANCO;
                }
                //Si no lo es, cogemos un color aleatorio
                else {
                        aux[1] = obtenerColorAleatorio();
                }
                //Ahora creamos la ficha de esa posición
                fichas[pos] = new Ficha(aux);
        }
}

public boolean comprobarConexiones () {
        boolean todas = true;
        for (int ii=0; ii<fichas.length; ii++){
                boolean conectadaD=false, conectadaI=false;
                //Comprobamos izquierda
                if (fichas[ii].getColor(0)== obtenerColorFichaAnterior(ii))
                        conectadaI = true;
                //Comprobamos derecha
                if (fichas[ii].getColor(1) == obtenerColorFichaSiguiente(ii))
                        conectadaD = true;
                //Llamamos al método setConectada de Ficha para cambiar el array
                fichas[ii].setConectada(new boolean []{conectadaI,conectadaD});
                //Con que una ficha no esté conectada en alguna de sus caras
                //ya no estarán todas conectadas
                if (!(conectadaD && conectadaI))
                        todas = false;
        }
        return todas;
}

public boolean comprobarPosicion (int posicion){
        boolean res;
        res = posicion>=0 && posicion<fichas.length;
        if (!res)
                System.out.println("¡Posición incorrecta!");
        return res;
}

public void moverFicha (int origen, int destino){
        //Primero comprobamos que las posiciones son correctas
        if (comprobarPosicion(origen) && comprobarPosicion(destino)) {
                //Guardamos la ficha a mover en una variable auxiliar
                Ficha aux = fichas[origen];
                //Si la movemos hacia el final del array
                if (origen<destino) {
                        //Desplazamos todas las fichas hacia delante
                        for (int ii=origen; ii<destino; ii++){
                                fichas[ii]=fichas[ii+1];
                        }
                }
                //Movemos hacia el principio del array
                else {
                        for (int ii=origen; ii>destino; ii--){
                                //Desplazamos las fichas hacia atrás
                                fichas[ii]=fichas[ii-1];
```

```
                        }
                    }
                    //Ponemos la ficha en su sitio
                    fichas[destino] = aux;
                }
                movimientos++;
        }

        public void barajarFichas (){
                for (int ii=0; ii<fichas.length; ii++){
                    //No se pide comprobar que origen y destino sean distintos
                    //así que no lo hacemos
                    int origen = (int)(Math.random()*fichas.length);
                    int destino = (int)(Math.random()*fichas.length);
                    Ficha aux = fichas[origen];
                    fichas[origen] = fichas[destino];
                    fichas[destino] = aux;
                }
        }

        public String toString (){
                String res = "Movimientos: "+ movimientos+"\n";
                for (int ii=0; ii<fichas.length;ii++)
                    res += fichas[ii].toString();
                return res;
        }
}
```

Pregunta 4 (1 puntos).- Crear un método `main` dentro de una clase denominada `UsoTablero` que pida al usuario que introduzca por teclado el número de fichas del `Tablero` y lo cree. A continuación debe rellenarlo, barajar las fichas, comprobar las conexiones entre fichas e imprimirlo. Luego irá preguntando al usuario los movimientos que quiere realizar, comprobando cada vez si todas las fichas están conectadas. En caso afirmativo terminará, si no, volverá a pedir un movimiento (debe comportarse como el ejemplo de la primera página)

```
import java.util.Scanner;

public class UsoTablero {

        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("¿Número de fichas?");
                int fichas = sc.nextInt();
                Tablero tab = new Tablero(fichas);
                tab.rellenarTablero();
                tab.barajarFichas();
                boolean fin = tab.comprobarConexiones();
                while (!fin){
                    System.out.println(tab);
                    System.out.println("¿Origen?");
                    int origen = sc.nextInt();
                    System.out.println("¿Destino?");
                    int destino = sc.nextInt();
                    tab.moverFicha(origen, destino);
                    fin = tab.comprobarConexiones();
                }
                System.out.println(tab);
                System.out.println("¡Has ganado!");
                sc.close();
        }
}
```