

Folder src\ejerciciosUD9\ajedrez

4 printable files

(file list disabled)

src\ejerciciosUD9\ajedrez\JuegoTablero.java

```
1 package ejerciciosUD9.ajedrez;
2
3 public interface JuegoTablero {
4     // pasará un índice numérico a su letra correspondiente del tablero
5     char indiceAPosicion(int indice);
6
7     // pasará un índice numérico a su posición numérica del tablero
8     int indiceAPosicionN(int indice);
9
10    // pasará una letra que indica una posición en el tablero a su correspondiente
11    // índice
12    int posicionAIndice(char posicion);
13
14    // pasará una posición numérica en el tablero a su correspondiente índice
15    int posicionAIndice(int posicion);
16 }
17
```

src\ejerciciosUD9\ajedrez\peon.java

```
1 package ejerciciosUD9.ajedrez;
2 public class peon extends pieza implements JuegoTablero{
3     /**
4      * Constructor similar al de la clase torre:
5      */
6     public peon(int x, int y, ColorPieza color) {
7         super(x, y, color);
8     }
9
10    /**
11     * Metodo para mover la pieza:
12     */
13    @Override
14    void mover(int x, int y) {
15        if (this.getColorpieza() == ColorPieza.NEGRO) {
16            if (this.getPosicion().x == 1 && this.getPosicion().y == y) {
17                if (x == 1 || x == 2) {
18                    this.posicion.setLocation(x, y);
19                }
20            } else if (this.getPosicion().x-x == 1) {
21                this.posicion.setLocation(x+1, y);
22            }
23        } else {
24            if (this.getPosicion().x == 7 && this.getPosicion().y == y) {
25                if (x == 1 || x == 2) {
26                    this.posicion.setLocation(x-1, y);
27                }
28            } else if (this.getPosicion().x-x == 1) {
```

```

29         this.posicion.setLocation(x-1, y);
30     }
31 }
32 }
33
34 /**
35  * Procedemos a implementarle los interfaces:
36  */
37
38 // devolverá la letra correspondiente del tablero de ajedrez.
39 @Override
40 public char indiceAPosicion(int indice) {
41     String letraTablero = "ABCDEFGH";
42     return letraTablero.charAt(indice);
43 }
44
45 // devolverá su posición numérica
46 @Override
47 public int indiceAPosicionN(int indice) {
48     if (indice >= 0 && indice <= 7) {
49         return indice + 1;
50     }
51     return -1;
52 }
53
54 // Devolvera posicion numerica de la letra.
55 @Override
56 public int posicionAIndice(char posicion) {
57     String letraTablero = "ABCDEFGH";
58     if (letraTablero.contains(String.valueOf(posicion))) {
59         return (letraTablero.indexOf(posicion) + 1);
60     }
61     return -1;
62 }
63
64 // devolverá su correspondiente índice numérico.
65 @Override
66 public int posicionAIndice(int posicion) {
67     if (posicion >= 1 && posicion <= 8) {
68         return posicion-1;
69     }
70     return -1;
71 }
72 }
73

```

src\ejerciciosUD9\ajedrez\pieza.java

```

1  package ejerciciosUD9.ajedrez;
2  import java.awt.Point;
3  public abstract class pieza {
4      //Creamos los atributos e importamos java.awt.Point para poder usar la clase point:
5      public enum ColorPieza{BLANCO,NEGRO};
6      protected Point posicion = new Point();
7      private ColorPieza colorpieza;
8      private boolean comida;
9      /**
10     * -----Primer constructor-----
11     * Constructor para crear una pieza asignarla en una posicion del tablero.

```

```
12     * @param x coordenada x
13     * @param y coordenada y
14     * @param colorpieza color de la pieza
15     * @param comida si esa posicion ha sido comida.
16     */
17     public pieza(int x, int y, ColorPieza colorpieza) {
18         this.posicion=new Point(x, y);
19         this.colorpieza = colorpieza;
20         if (comprobarPosicion(x)&&comprobarPosicion(y)) {
21             this.comida=false;
22         }else{
23             this.comida=true;
24         }
25     }
26
27     /**
28     * -----Segundo constructor-----
29     * El segundo, que deberá usar el primero, no recibirá ningún parámetro
30     * y creará una nueva Pieza en la posición 10, 10 y de color BLANCO.
31     */
32     public pieza(){
33         this(10,10,ColorPieza.BLANCO);
34     }
35
36     /**
37     * -----Tercer constructor-----
38     * El tercero, que usará el primero,
39     * recibe como parámetro una Pieza, y copiará sus atributos a la nueva Pieza,
40     */
41     public pieza(pieza piezaTercerConst){
42         this(piezaTercerConst.posicion.x, piezaTercerConst.posicion.y,
43         piezaTercerConst.colorpieza);
44     }
45
46     //Getters y setters
47     public Point getPosicion() {
48         return posicion;
49     }
50     public void setPosicion(Point posicion) {
51         this.posicion = posicion;
52     }
53     public ColorPieza getColorpieza() {
54         return colorpieza;
55     }
56     public void setColorpieza(ColorPieza colorpieza) {
57         this.colorpieza = colorpieza;
58     }
59     public boolean isComida() {
60         return comida;
61     }
62
63     public void setComida(boolean comida) {
64         this.comida = comida;
65     }
66
67     //HashCode
68     @Override
69     public int hashCode() {
70         final int prime = 31;
71         int result = 1;
```

```

71         result = prime * result + ((posicion == null) ? 0 : posicion.hashCode());
72         result = prime * result + ((colorpieza == null) ? 0 : colorpieza.hashCode());
73         return result;
74     }
75
76     /**
77     * Para saber si una pieza es igual a otra se comprueba que:
78     * 1. Misma clase
79     * 2. Mismo color
80     * 3. Misma posicion
81     */
82     @Override
83     public boolean equals(Object obj) {
84         if (this == obj)
85             return true;
86         if (obj == null)
87             return false;
88         if (getClass() != obj.getClass())
89             return false;
90         pieza other = (pieza) obj;
91         if (posicion == null) {
92             if (other.posicion != null)
93                 return false;
94         } else if (!posicion.equals(other.posicion))
95             return false;
96         if (colorpieza != other.colorpieza)
97             return false;
98         return true;
99     }
100
101     //Metodos
102     /**
103     * comprobarPosicion comprobará si el número que se le pasa como parámetro
104     * está dentro de los límites establecidos, entre 0 y 7 y
105     * devolverá verdadero si está en los límites o falso en caso contrario.
106     * @param p
107     * @return true si esta en el rango o false si no lo esta.
108     */
109     static boolean comprobarPosicion(int p){
110         if (p <= 7 && p > 0) {
111             return true;
112         }
113         return false;
114     }
115
116     /**
117     * Crear el métodos públicos y abstracto:
118     */
119     abstract void mover(int x,int y);
120 }

```

src\ejerciciosUD9\ajedrez\torre.java

```

1 package ejerciciosUD9.ajedrez;
2
3
4 public class torre extends pieza implements JuegoTablero {
5     /**
6     * Implementamos el primer constructor de nuestra clase pieza.

```

```
7      */
8      public torre(int x, int y, ColorPieza colorPieza) {
9          super(x, y, colorPieza);
10     }
11
12     /**
13      * Moveremos las fichas mientras una coordenada se mantenga igual.
14      */
15     @Override
16     void mover(int x, int y) {
17         if (comprobarPosicion(x) && comprobarPosicion(y)) {
18             if (this.getPosicion().x == x || this.getPosicion().y == y) {
19                 this.posicion.setLocation(x, y);
20             }
21         }
22     }
23
24     /**
25      * Procedemos a implementarle los interfaces:
26      */
27
28     // devolverá la letra correspondiente del tablero de ajedrez.
29     @Override
30     public char indiceAPosicion(int indice) {
31         String letraTablero = "ABCDEFGH";
32         return letraTablero.charAt(indice);
33     }
34
35     // devolverá su posición numérica
36     @Override
37     public int indiceAPosicionN(int indice) {
38         if (indice >= 0 && indice <= 7) {
39             return indice + 1;
40         }
41         return -1;
42     }
43
44     // Devolvera posicion numerica de la letra.
45     @Override
46     public int posicionAIndice(char posicion) {
47         String letraTablero = "ABCDEFGH";
48         if (letraTablero.contains(String.valueOf(posicion))) {
49             return (letraTablero.indexOf(posicion) + 1);
50         }
51         return -1;
52     }
53
54     // devolverá su correspondiente índice numérico.
55     @Override
56     public int posicionAIndice(int posicion) {
57         if (posicion >= 1 && posicion <= 8) {
58             return posicion-1;
59         }
60         return -1;
61     }
62 }
```