Folder src\ejerciciosUD9\ajedrez

4 printable files

(file list disabled)

src\ejerciciosUD9\ajedrez\JuegoTablero.java

```java
package ejerciciosUD9.ajedrez;

public interface JuegoTablero {
    // pasará un índice numérico a su letra correspondiente del tablero
    char indiceAPosicion(int indice);

    // pasará un índice numérico a su posición numérica del tablero
    int indiceAPosicionN(int indice);

    // pasará una letra que indica una posición en el tablero a su correspondiente
    // índice
    int posicionAIndice(char posicion);

    // pasará una posición númerica en el tablero a su correspondiente índice
    int posicionAIndice(int posicion);
}

```

src\ejerciciosUD9\ajedrez\peon.java

```java
package ejerciciosUD9.ajedrez;
public class peon extends pieza implements JuegoTablero{
    /**
     * Constructor similar al de la clase torre:
     */
    public peon(int x, int y, ColorPieza color) {
        super(x, y, color);
    }

    /**
     * Metodo para mover la pieza:
     */
    @Override
    void mover(int x, int y) {
        if (this.getColorpieza() == ColorPieza.NEGRO) {
            if (this.getPosicion().x == 1 && this.getPosicion().y == y) {
                if (y == 1 || y == 2) {
                    this.posicion.setLocation(x, y);
                }
            } else if (this.getPosicion().x-x == 1) {
                this.posicion.setLocation(x+1, y);
            }
        }else{
            if (this.getPosicion().x == 7 && this.getPosicion().y == y) {
                if (y == 1 || y == 2) {
                    this.posicion.setLocation(x-1, y);
                }
            }else if (this.getPosicion().x-x == 1) {
```

```
                this.posicion.setLocation(x-1, y);
            }
        }
    }

    /**
    * Procedemos a implementarle los interfaces:
    */

    // devolverá la letra correspondiente del tablero de ajedrez.
    @Override
    public char indiceAPosicion(int indice) {
        String letraTablero = "ABCDEFGH";
        return letraTablero.charAt(indice);
    }

    // devolverá su posición numérica
    @Override
    public int indiceAPosicionN(int indice) {
        if (indice >= 0 && indice <= 7) {
            return indice + 1;
        }
        return -1;
    }

    // Devolvera posicion numerica de la letra.
    @Override
    public int posicionAIndice(char posicion) {
        String letraTablero = "ABCDEFGH";
        if (letraTablero.contains(String.valueOf(posicion))) {
            return (letraTablero.indexOf(posicion) + 1);
        }
        return -1;
    }

    // devolverá su correspondiente índice numérico.
    @Override
    public int posicionAIndice(int posicion) {
        if (posicion >= 1 && posicion <= 8) {
            return posicion-1;
        }
        return -1;
    }
}

```

src\ejerciciosUD9\ajedrez\pieza.java

```
package ejerciciosUD9.ajedrez;
import java.awt.Point;
public abstract class pieza {
    //Creamos los atributos e importamos java.awt.Point para poder usar la clase point:
    public enum ColorPieza{BLANCO,NEGRO};
    protected Point posicion = new Point();
    private ColorPieza colorpieza;
    private boolean comida;
    /**
     * ---------------Primer constructor--------------------
     * 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,
    piezaTercerConst.colorpieza);
43         }
44 
45          //Getters y setters
46         public Point getPosicion() {
47             return posicion;
48         }
49         public void setPosicion(Point posicion) {
50             this.posicion = posicion;
51         }
52         public ColorPieza getColorpieza() {
53             return colorpieza;
54         }
55         public void setColorpieza(ColorPieza colorpieza) {
56             this.colorpieza = colorpieza;
57         }
58         public boolean isComida() {
59             return comida;
60         }
61 
62         public void setComida(boolean comida) {
63             this.comida = comida;
64         }
65 
66         //Hashcode
67         @Override
68         public int hashCode() {
69             final int prime = 31;
70             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
121 }
122
```

src\ejerciciosUD9\ajedrez\torre.java

```
1  package ejerciciosUD9.ajedrez;
2
3  import java.awt.Point;
4
```

```java
public class torre extends pieza implements JuegoTablero {
    /**
     * Implementamos el primer constructor de nuesta clase pieza.
     */
    public torre(int x, int y, ColorPieza colorPieza) {
        super(x, y, colorPieza);
    }

    /**
     * Moveremos las fichas mientras una coordenada se mantenga igual.
     */
    @Override
    void mover(int x, int y) {
        if (comprobarPosicion(x) && comprobarPosicion(y)) {
            if (this.getPosicion().x == x || this.getPosicion().y == y) {
                setPosicion(new Point(x, y));
            }
        }
    }

    /**
     * Procedemos a implementarle los interfaces:
     */

    // devolverá la letra correspondiente del tablero de ajedrez.
    @Override
    public char indiceAPosicion(int indice) {
        String letraTablero = "ABCDEFGH";
        return letraTablero.charAt(indice);
    }

    // devolverá su posición numérica
    @Override
    public int indiceAPosicionN(int indice) {
        if (indice >= 0 && indice <= 7) {
            return indice + 1;
        }
        return -1;
    }

    // Devolvera posicion numerica de la letra.
    @Override
    public int posicionAIndice(char posicion) {
        String letraTablero = "ABCDEFGH";
        if (letraTablero.contains(String.valueOf(posicion))) {
            return (letraTablero.indexOf(posicion) + 1);
        }
        return -1;
    }

    // devolverá su correspondiente índice numérico.
    @Override
    public int posicionAIndice(int posicion) {
        if (posicion >= 1 && posicion <= 8) {
            return posicion-1;
        }
        return -1;
    }
}
```