Práctica Programación 2 (Curso 2023/2024)
Parte 1: Hormiga de Langton.

- La práctica se debe entregar en **DeliverIt** donde se realizarán una serie de pruebas unitarias automáticas.
- Para que los archivos de código fuente compilen se debe respetar estrictamente los nombres de las clases y paquetes.
- Adicionalmente, el alumno puede comprobar el resultado del código fuente gráficamente completando los métodos de dibujado que sean necesarios.
- El dibujado **no** se comprueba en las pruebas automáticas.

1. Descripción general del problema

Se desea desarrollar varias clases necesarias para una aplicación que muestre la evolución de la *Hormiga de Langton*. La *Hormiga de Langton* es un autómata celular que sigue un conjunto simple de reglas en un mundo de casillas cuadradas bidimensionales[1].

Las reglas **originales** del juego indican que la hormiga se mueve de la siguiente manera:

1. Si está sobre un cuadrado blanco gira noventa grados a la izquierda, cambia el color del cuadrado y avanza un cuadrado.
2. Si está sobre un cuadrado negro gira noventa grados a la derecha, cambia el color del cuadrado y avanza un cuadrado.
3. La hormiga se mueve en una cuadrícula bidimensional de casillas cuadradas, cada una de las cuales puede ser de dos colores: blanco o negro, codificados por los números 0 y 1.
4. La hormiga siempre está orientada hacia arriba, abajo, izquierda o derecha.
5. La cuadrícula de casillas tiene un tamaño de 11x11. Si una hormiga se sale de la cuadrícula da media vuelta y avanza una casilla.

2. Modelo de Datos

Todas las clases, incluidas las clases que sirven para dibujar y hacer la animación, están en un **paquete** que se llama **`hl`** (hache y ele, de hormiga de Langton). El alumno debe escribir todas las clases en ese paquete.

Para realizar el programa se ha decidido escribir las siguientes clases que constituyen las correspondientes abstracciones de datos.

2.1. Clases e interfaces para la Hormiga de Langton original

Casilla. Los objetos de esta clase representan las casillas de la cuadrícula donde se mueven las hormigas. Todas las casillas empiezan siendo blancas. Tiene la siguiente especificación:

```
/** Cambiar el color de 'esta' casilla
 *
 * Alternando entre el blanco y el negro, codificados respectivamente por 0 y 1.
 */
public void cambiarColor ()  ;

/** El color de 'esta' casilla
 *
 * @return el numero de color de la casilla 0 para blanco, 1 para negro
 */
public int color ()  ;
```

[1] consultar: `https://es.wikipedia.org/wiki/Hormiga_de_Langton`

Orientación. Es la clase que representa las orientaciones que puede tener una hormiga (arriba, derecha, abajo, izquierda). Esta clase sirve para gestionar el movimiento de las hormigas porque almacenan los incrementos de casilla que están permitidos para mover una hormiga. Estos incrementos se corresponden con vectores unitarios en cada una de las direcciones en que puede estar una hormiga. Para guardar estas orientaciones permitidas se va a utilizar un *array* que guarda los vectores unitarios de *arriba*, *derecha*, *abajo* e *izquierda* y un índice, **rumbo**, para saber cuál es la orientación efectiva (un número del 0 al 3, 0 arriba, 1 derecha, 2 abajo y 3 para izquierda). Además, cuenta con varios métodos para girar la orientación. Su especificación es:

```
private int rumbo;
                                          /* Arriba     Derecha    Abajo     Izquierda */
private static final int[][] unitarios = { { 0, +1 }, { +1, 0 }, { 0, -1 }, { -1, 0 } };

private Orientacion (int rumbo)  ;

// Devuelve una nueva orientacion que resulta de girar 'esta' orientacion 180 grados
public Orientacion girarMediaVuelta ()  ;

// Devuelve una nueva orientacion que resulta de girar 'esta' orientacion 90 grados a la izquierda
public Orientacion girarIzquierda ()  ;

// Devuelve una nueva orientacion que resulta de girar 'esta' orientacion 90 grados a la derecha
public Orientacion girarDerecha ()  ;

// Mueve la hormiga, 'h', con un incremento de posicion dado por el vector
// unitario de 'esta' orientacion.
public void mover (IHormiga h)  ;

// Devuelve el angulo en grados de 'esta' orientacion respecto del eje vertical. Por ejemplo,
// arriba devuelve 0 grados mientras que izquierda devuelve 90 (o -270) y derecha son -90 (o 270).
public int getAnguloEnGrados ()  ;

public static final Orientacion IZQUIERDA = new Orientacion(3);
```

ICuadricula. Su declaración es:

```
public interface ICuadricula {
  // Devuelve la casilla situada en las coordenadas dadas por (x, y) en el plano 2D.
  Casilla casilla (int x, int y);
}
```

Cuadricula. La cuadrícula contiene un `array` de casillas. Tiene un constructor para crear todas las casillas y un método para devolver la casilla en unas coordenadas dadas que lanza una excepción si se intenta acceder a una casilla fuera de la cuadrícula (la hormiga se ha salido de la cuadrícula). Esta clase debe implementar la interfaz `ICuadricula`. La especificación es:

```
// Tamanho de la matriz que guarda las casillas, DIMxDIM
public static final int DIM = 11;

// Array bidimensional de tamanho DIMxDIM que guarda las casillas
private Casilla[][] casillas;

/** Construye e inicializa una nueva cuadricula
 */
public Cuadricula ()  ;

/** Devuelve la casilla  de 'esta' cuadricula en los indices de fila y columna dados por i y j.
 *  Se supone que los indices son las coordenadas (x, y) de la casilla en un plano 2D.
 * @throws IndexOutOfBoundsException cuando i o j no estan en el rango [0,Dim) */
@Override
public Casilla casilla (int i, int j)  ;
```

Hormiga. Cada hormiga guarda la posición en la que se encuentra en la cuadrícula y su orientación. Todas las hormigas empiezan orientadas hacia la izquierda. Las hormigas tienen un método que permite moverlas libremente mediante un incremento en sus coordenadas x e y. Además, tienen otros 2 métodos que permiten definir su comportamiento de acuerdo a las reglas del juego: `girar` y `cambiarColor`. A su vez, `avanzar` hace que la hormiga se mueva una única casilla en la dirección que tiene su orientación y `tick` permite hacer estas tres operaciones de forma sucesiva y recurrente, es decir se van haciendo de 1 en 1 y cuando se acaba se vuelve a empezar, el objetivo de esto es dibujar la animación del comportamiento de la hormiga lo más claramente posible. La hormiga debe implementar la interfaz `IHormiga`, el resto de funciones de esa interfaz permiten acceder a los datos necesarios para dibujarla. La interfaz es:

```
public interface IHormiga {

  // Los posibles giros que puede hacer una hormiga.
  static enum Giro { IZQUIERDA, DERECHA, MEDIA_VUELTA, SIN_GIRO }

  /** Mueve 'esta' hormiga el numero de casillas indicadas en los parametros.
   *
   * @param incX, incY los incrementos de posicion de la hormiga, es decir,
   * la hormiga se desplaza esa cantidad de casillas en cada eje independientemente
   * de su orientacion. */
  public void mover (int incX, int incY);

  /** 'Esta' hormiga gira su orientacion segun el color de la casilla que ocupa
   *
   *  Segun las reglas del juego gira en un sentido u otro dependiendo del
   *  color de la casilla,
   *  @param cuadricula la cuadricula donde se encuentra la hormiga y que
   *  se utiliza para obtener la casilla.
   *  @return eL giro que se ha realizado
   *  @throws IllegalStateException si el color de la casilla no es
   *  uno de lo que la hormiga reconoce en sus instrucciones de actuacion.
   */
  public Giro girar (ICuadricula cuadricula);

  /** 'Esta' hormiga cambia el color de la casilla que ocupa
  *
  *  Se busca la casilla en la 'cuadricula' y se cambia su color.
  *  @param cuadricula la cuadricula donde se encuentra la hormiga y que
  *  se utiliza para obtener la casilla.
  */
  public void cambiarColor (ICuadricula cuadricula);

  /** 'Esta' hormiga avanza una casilla en el sentido de su orientacion
   *
   *  Se utiliza la orientacion (sus vectores unitarios) para que la hormiga se
   *  mueva una casilla.
   * */
  public void avanzar ();

  /** Llama sucesivamente y recurrentemente a los metodos girar, cambiarColor y avanzar
  *
  *  La primera vez se llama a girar, la segunda a cambiarColor, la tercera a
  *  avanzar y vuelta a empezar.
  *  @param cuadricula la cuadricula donde se encuentra la hormiga y que
  *  se utiliza para obtener la casilla.
  * */
 public void tick (ICuadricula cuadricula);

  /**
   * Devuelve las coordenadas de la hormiga.
   * @return como un array { x, y } siendo x e y las coordenadas de la hormiga en el plano 2D
   */
  int[] coordenadas ();

  /** La ruta del archivo con la imagen de la hormiga.
```

```
     * Por ejemplo: "res/ant_1.png"
     * @return la ruta relativa al archivo
     */
    String rutaDeLaImagen ();

    /** El angulo en grados de 'esta' hormiga
     * que se debe girar en sentido anti-horario la imagen dada por el metodo anterior,
     * para que, al dibujar el dibujo de la hormiga quede orientado correctamente
     * @return el angulo en grados de la orientacion de 'esta' hormiga
     */
    int getAnguloEnGrados ();
}
```

2.2. Clases para dibujar la cuadrícula y la hormiga

Para dibujar la cuadrícula y la hormiga son necesarias las siguientes clases que se suministran junto con el enunciado:

- Una clase de utilidades, `Graficos`, con métodos estáticos para dibujar cuadrícula y hormigas
- Una clase `Principal` para hacer la animación, es decir, para hacer que las hormigas cambien y se dibujen periódicamente sus cambios.
- Se utiliza también la clase `StdDraw` directamente a partir de su código fuente.

IMPORTANTE: **Las pruebas en DeliverIt no comprueban la animación**.

3. Se pide

Escribe el código fuente de las clases:

- `Casilla`
- `Orientacion`
- `Cuadricula`
- `Hormiga`

Y entregalas en DeliverIt.

4. Registro de Cambios

Respecto de la primera versión se han introducido los siguientes cambios:

1. Se han quitado los nombres de giros en relación con el sentido horario o antihorario. Ahora se denominan como giros izquierda y derecha tal y como aparecen en la descripción de las reglas de movimiento de las hormigas.
2. Se ha corregido una errata en las pruebas en DeliverIt por el cuál el giro antihorario y horario estaban intercambiados.
3. El atributo que indica la orientación efectiva en la clase `Orientacion` se ha renombrado como `rumbo`.
4. El atributo que guarda la matriz de casillas en la cuadrícula se ha cambiado a `private`.