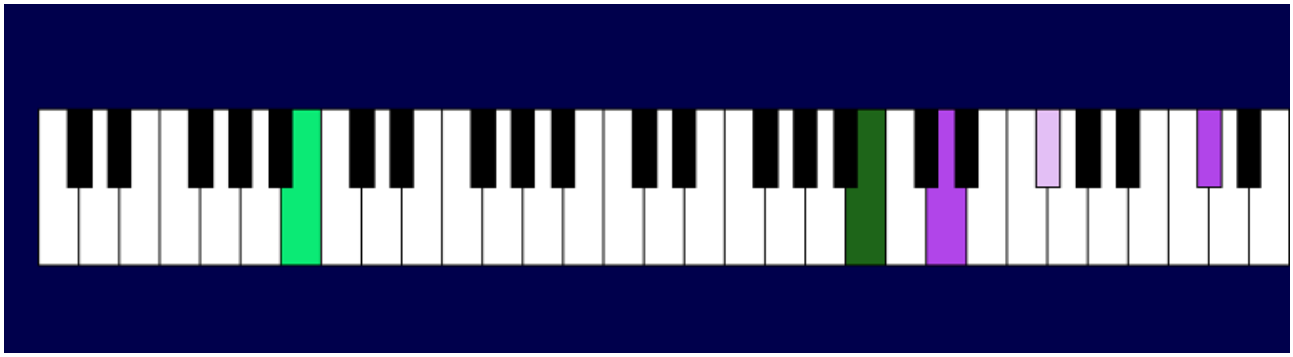
PROYECTO 4

PIANO MIDI



ElementoVisual

Esta interfaz representa un elemento gráfico (piano, teclas negras, blancas, etc) que se ve en la pantalla.

Métodos:

- **setPosicion**: Es un setter que pone al objeto las coordenadas en las que se mostrará en la pantalla el elemento visual.
- **setGraphics:** Es un setter que pone al objeto el graphics que se usará para dibujar en la pantalla
- **dibujar**: Es un método que hace que se dibuje en la pantalla el elemento visual

Tests:

No son necesarios, ya que al programar las clases se probarán los métodos de la interfaz.

Medible

Esta interfaz representa un elemento gráfico con anchura y altura

Métodos:

- **getAnchura**: Devuelve la anchura (en píxeles) del elemento visual
- **getAltura**: Devuelve la altura (en píxeles) del elemento visual

Tests:

No son necesarios, ya que al programar las clases se probarán los métodos de la interfaz.

Pulsable

Esta interfaz representa un elemento visual que se puede pulsar y soltar

Métodos:

- **pulsar**: Es un método que hace que se pulse el elemento visual
- **soltar**: Es un método que hace que se suelte el elemento visual
- **estaPulsado:** Devuelve true si el elemento visual está pulsado y false si no.
- **setColorPulsado:** Es un setter que permite indicar cuál será el color del elemento cuando esté pulsado.
- **getColorPulsado**: Devuelve el color del elemento visual cuando está pulsado
- **getColorNoPulsado**: Devuelve el color del elemento visual cuando está pulsado
- **getColor:** Devuelve el color del objeto, teniendo en cuenta si está pulsado o suelto.

Tests:

No son necesarios, ya que al programar las clases se probarán los métodos de la interfaz.

ElementoVisualTester

Este es un programa que sirve para hacer los tests de las clases que implementen la interfaz **ElementoVisual** (o sea, todos los elementos gráficos).

Propiedades:

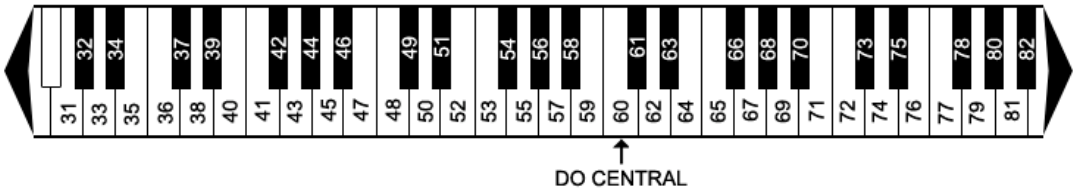
- **elemento**: Es el elemento visual que se quiere probar. Cada vez que un miembro del equipo programe un **ElementoVisual**, se usará esta clase para probarlo.
- **graphics:** Es el Graphics que se usará para dibujar el elemento visual.

Métodos:

- **primer Constructor**: inicializa la propiedad **graphics** y el **ElementoVisual** con los parámetros recibido
- **Segundo constructor:** Crea una Consola DAW, le pone de color de fondo gris y después rellena la propiedad **graphics** con el objeto Graphics que se obtiene de la Capa Canvas. Se inicializará también el elemento visual recibido como parámetro.
- **hacerPrueba**: hace estos pasos:
    1. Configura el elemento visual de la propiedad **elemento** de forma que:
        1. Sus coordenadas sean el punto (120,90)
        2. Su graphics sea el de la propiedad **graphics**
    2. Llama al método **dibujar** y si el elemento visual está correctamente programado, deberá verse en pantalla.
    3. Si el elemento visual es pulsable, se harán adicionalmente estos pasos:
        1. Se llamará al método **setColorPulsado** pasándole el color azul.
        2. Se hace una pausa de 2 segundos
        3. Se llama al método para pulsar el elemento y después al método dibujar. Si este está bien programado, deberá verse de color azul.
        4. Haz una pausa de 2 segundos
        5. Llama al método para soltar el elemento y después al método dibujar. El elemento deberá verse dibujado como al principio
    4. Haz una pausa de 2 segundos
- **main:** Crea un **ElementoVisualTester** usando su segundo constructor y pasando un objeto del **ElementoVisual** que se quiere probar y a continuación llama a su método **hacerPrueba**

Tecla

Esta clase representa una tecla cualquiera del piano. Cada tecla tiene un número que identifica la nota musical que suena al pulsarla. Aquí vemos los números de una parte del piano:



Por otra parte, las teclas pueden ser de color blanco o negro, lo que da lugar a dos tipos principales de tecla. A su vez, las teclas blancas pueden tener tres formas geométricas, lo que nos da tres tipos de teclas blancas. Las clases hijas de Tecla tienen los detalles para dibujar correctamente cada tipo de tecla.

Propiedades:

- **posicion:** Es un Point donde se guardan las coordenadas donde se va a dibujar la tecla
- **nota:** Es el número de la nota musical que suena cuando se pulsa la tecla, según los números de la imagen anterior.
- **pulsada:** Es un boolean que nos dice si la tecla está pulsada o no
- **colorPulsada:** Es el color con el que se pintará la tecla cuando está pulsada
- **graphics:** Es el graphics que se usará para dibujar la tecla en la pantalla.

Métodos:

- **Constructor:** Crea una tecla sin pulsar, asociada a la nota musical cuyo número se pasa como primer parámetro. Sus propiedades **posición** y **colorPulsada** se inicializa con **null**
- **getNumeroNota:** Devuelve el número de nota musical asociada a la tecla
- **setPosicion:** Asigna a la propiedad **posicion** un Point con las coordenadas donde se situará la tecla en la pantalla.
- **setGraphics:** Asigna la propiedad **graphics** con el parámetro recibido
- **pulsar:** Modifica la propiedad **pulsada** para que la tecla se considere pulsada
- **soltar:** Modifica la propiedad **pulsada** para que la tecla deje de estar pulsada
- **dibujar:** Hace estos pasos:
    - Si alguna de las propiedades **posicion** o **graphics** es null, lanza una **IllegalStateException** con el mensaje “hay que llamar a setPosición y setGraphics antes de llamar al método dibujar”.

    - Toma la propiedad Graphics y le pone el color que devuelva el método **getColor**
    - Llama al método **fillPolygon** para dibujar un polígono cuyas coordenadas X nos las da el método **getVerticesX** y sus coordenadas Y nos las da el método **getVerticesY**. Estos dos métodos son abstractos, pero se pueden llamar aquí sin ningún problema. En las clases hijas estos métodos serán programados para retornar la lista correcta de coordenadas según el tipo de tecla.
    - Ahora cambia el color del Graphics y se le pone color negro.
    - Se llama al método **drawPolygon** pasando los arrays de vértices anteriores. Con esto lo que hacemos es pintar un borde de color negro a la tecla.

- **setColorPulsado:** Cambia la propiedad **colorPulsada** y le pone el parámetro recibido.
- **estaPulsado:** Devuelve true si la tecla está pulsada y false si no.
- **getColorPulsado:** Devuelve el color que hay en la propiedad **colorPulsada**
- **getColorNoPulsado:** Es abstracto, y en las clases hijas se programará para devolver el color que tiene la tecla cuando no está pulsada
- **getAnchura:** Es abstracto, y en las clases hijas se programará para devolver la anchura de la tecla (en píxeles)
- **getAltura:** Es abstracto, y en las clases hijas se programará para devolver la altura de la tecla (en píxeles)
- **getVerticesX:** Es abstracto, y en las clases hijas se programará para devolver una lista con las coordenadas X de todos los vértices del polígono con el que se dibuja la tecla.
- **getVerticesY:** Es abstracto, y en las clases hijas se programará para devolver una lista con las coordenadas Y de todos los vértices del polígono con el que se dibuja la tecla.

Tests:

No son necesarios, ya que al programar las clases hijas se probarán los tests de esta clase padre.

TeclaNegra

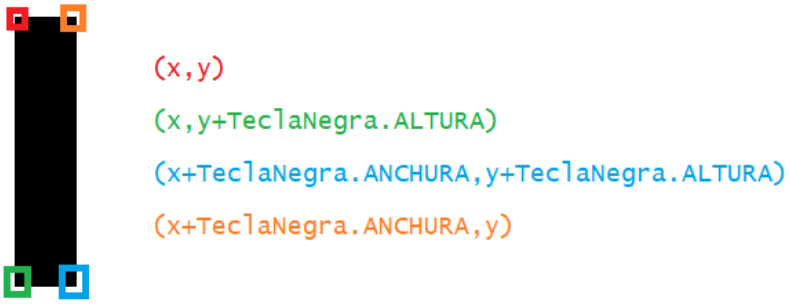
Esta clase es una tecla de color negro del piano.

Propiedades:

- **ANCHURA:** Es una constante que deberá guardar **15**
- **ALTURA:** Es una constante que deberá guardar **50**

Métodos:

- **Constructor:** Llama al constructor de la clase padre para inicializar las propiedades de la clase padre recibidas como parámetros.
- **getColorNoPulsado:** Deberá devolver un objeto de la clase Color con el color negro.
- **getAnchura:** Devuelve el valor de la constante ANCHURA
- **getAltura:** Devuelve el valor de la constante ALTURA
- **getVerticesX, getVerticesY:** Deberán examinar la propiedad heredada de la clase Tecla en la que se guarda la posición (x,y) donde se debe dibujar la tecla y devolver arrays con los vértices que están marcados con colores en la siguiente imagen:



En la imagen se ve cada vértice de la tecla pintado con un color diferente. En la derecha, tenemos las coordenadas correspondientes a cada vértice, pintadas con sus mismos colores. El método getVerticesX debe devolver un array con las coordenadas X de todos los vértices y el método getVerticesY un array con las coordenadas Y de todos los vértices

Test:

Modifica el método **main** de **ElementoVisualTester** para que el **ElementoVisual** que se pasa al constructor sea un objeto de este tipo de tecla. Puedes pasar como número de tecla el que quieras.

TeclaBlanca

Esta clase es una tecla de color blanco. Es una clase abstracta, porque hay tres tipos de teclas blancas según su forma geométrica. No obstante, en esta clase se recoge toda la funcionalidad común a los tres tipos de teclas blancas.

Propiedades:

- **ANCHURA:** Es una constante que deberá guardar **25**
- **ALTURA:** Es una constante que deberá guardar **100**

Métodos:

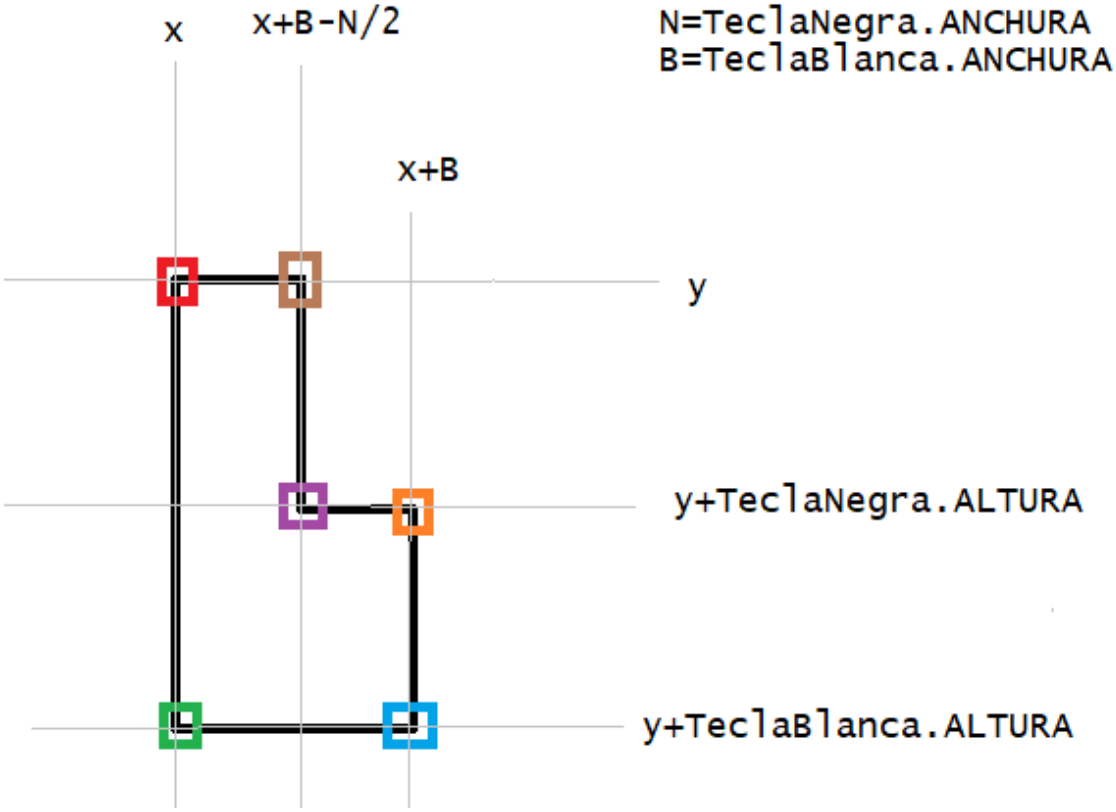
- **Constructor:** Llama al constructor de la clase padre para inicializar las propiedades de la clase padre recibidas como parámetros.
- **getColorNoPulsado:** Deberá devolver un objeto de la clase Color con el color blanco.
- **getAnchura:** Devuelve el valor de la constante ANCHURA
- **getAltura:** Devuelve el valor de la constante ALTURA
- **getVerticesX, getVerticesY:** Se mantienen abstractos para que los sobreescriban las clases hijas dando los vértices adecuados para cada tipo de tecla blanca.

Test:

No son necesarios, porque al programar las clases hijas se harán test que servirán para probar esta clase.

TeclaBlanca1

Esta es una tecla blanca cuyo polígono tiene esta forma:



Métodos:

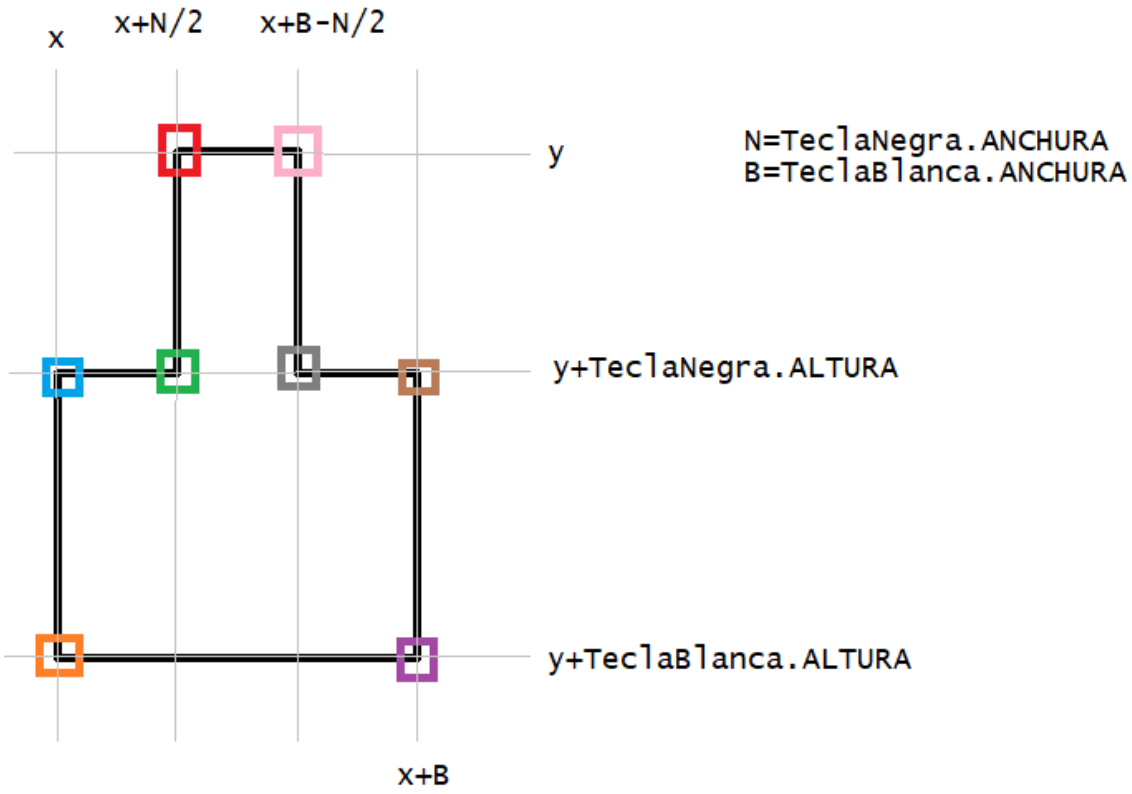
- **Constructor:** Llama al constructor de la clase padre para inicializar las propiedades de la clase padre recibidas como parámetros.
- **getVerticesX:** Devuelve un array con las coordenadas X de los vértices del polígono. Deberás obtener las coordenadas a partir de lo que se ve en la imagen
- **getVerticesY:** Devuelve un array con las coordenadas Y de los vértices del polígono. Deberás obtener las coordenadas a partir de lo que se ve en la imagen

Test:

Modifica el método **main** de **ElementoVisualTester** para que el **ElementoVisual** que se pasa al constructor sea un objeto de este tipo de tecla. Puedes pasar como número de tecla el que quieras.

TeclaBlanca2

Esta es una tecla blanca cuyo polígono tiene esta forma:



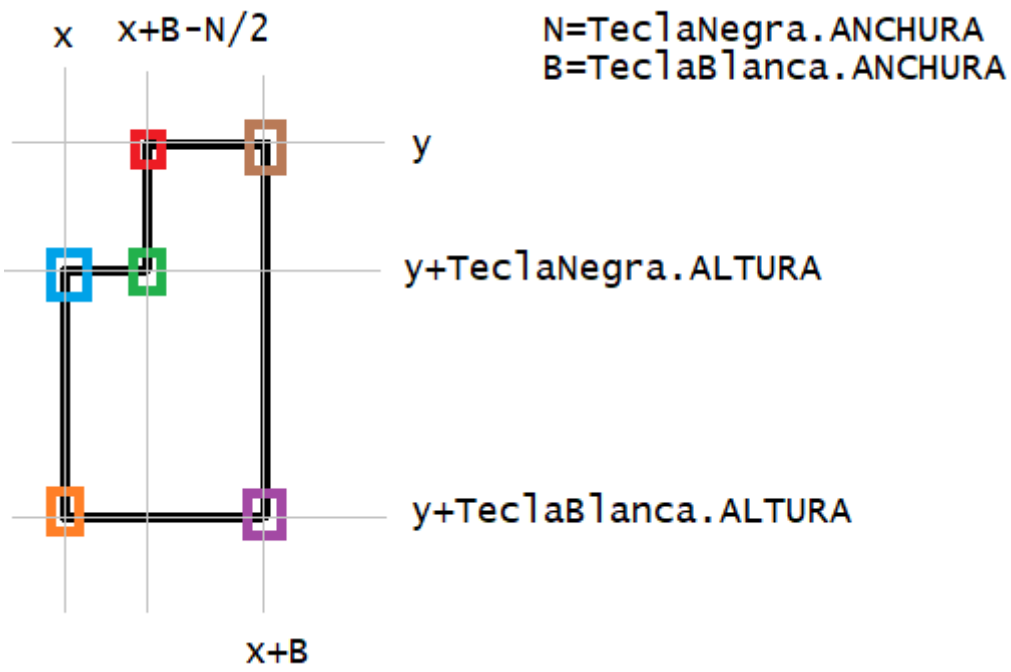
- **Constructor:** Llama al constructor de la clase padre para inicializar las propiedades de la clase padre recibidas como parámetros.
- **getVerticesX:** Devuelve un array con las coordenadas X de los vértices del polígono. Deberás obtener las coordenadas a partir de lo que se ve en la imagen
- **getVerticesY:** Devuelve un array con las coordenadas Y de los vértices del polígono. Deberás obtener las coordenadas a partir de lo que se ve en la imagen

Test:

- Modifica el método **main** de **ElementoVisualTester** para que el **ElementoVisual** que se pasa al constructor sea un objeto de este tipo de tecla. Puedes pasar como número de tecla el que quieras.

TeclaBlanca3

Esta es una tecla blanca cuyo polígono tiene esta forma:



- **Constructor:**Llama al constructor de la clase padre para inicializar las propiedades de la clase padre recibidas como parámetros.
- **getVerticesX:** Devuelve un array con las coordenadas X de los vértices del polígono. Deberás obtener las coordenadas a partir de lo que se ve en la imagen
- **getVerticesY:** Devuelve un array con las coordenadas Y de los vértices del polígono. Deberás obtener las coordenadas a partir de lo que se ve en la imagen

Test:

- Modifica el método **main** de **ElementoVisualTester** para que el **ElementoVisual** que se pasa al constructor sea un objeto de este tipo de tecla. Puedes pasar como número de tecla el que quieras.

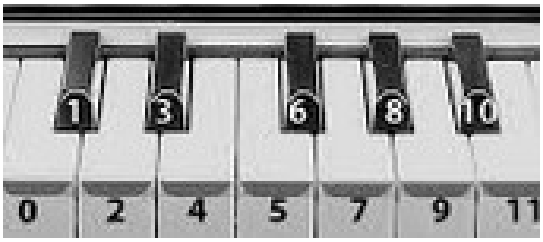
TeclaFactory

Esta clase facilita la obtención del tipo correcto de tecla a partir de un número de nota musical del piano. Sus métodos son:

- **Constructor:** No hace nada. Lo único importante de él es que es privado, para así forzar a que las personas que quieran obtener teclas deban usar el método **crearTecla**.
- **crearTecla:** Este método tiene que crear una tecla a partir número de nota musical recibido como parámetro. El problema es que se admiten hasta 128 notas musicales posibles y cada una tiene su propia tecla, tal y como se indica en la siguiente tabla. Las notas musicales se repiten en grupos de 12, teniendo cada vez un sonido más agudo.

Octave	Note numbers											
	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

El método crearTecla tiene que crear un objeto Tecla con el tipo correcto para cada una de las 128 notas posibles. Para evitar tener que hacer un switch gigante que tenga 128 cases, lo que haremos es calcular el **resto de dividir el número de nota musical entre 12**, para de esa forma, obtener una nota musical equivalente a la nuestra, pero dentro de una escala simple de 12 notas que podemos analizar fácilmente:



Observa que:

- Las notas 1,3,6,8 y 10 son de tipo TeclaNegra
- Las notas 0,5 son de tipo TeclaBlanca1
- Las notas 2,7 y 9 son de tipo TeclaBlanca2
- Las notas 4 y 11 son de tipo TeclaBlanca3

Por tanto el método crearTecla hará esto:

- Calcula el resto de dividir la nota musical que hemos recibido como parámetro entre 12 y obtendremos una nota con un número entre 0 y 11, que tiene la misma forma de tecla que la original.
- Mira en la imagen cuál es la forma correspondiente a la tecla que buscamos
- Crea una tecla con el número de la nota musical que recibimos como parámetro
- Devuelve dicha tecla

Test:

Haz un programa de pruebas que haga estos pasos:

- Crea una Consola DAW y ponle de color de fondo gris
- Haz un for que genere los números de 60 a 64 (ambos incluidos)
- Por cada número recorrido:
  - Usa **TeclaFactory** para obtener un objeto **Tecla** correspondiente a ese número
  - Crea un **ElementoVisualTester** usando el constructor que recibe como parámetro el graphics de la capa canvas de la consola creada.
  - Pon la tecla obtenida en el tester y llama a su método hacerPrueba

Si se hace bien el test, se deberán ver teclas diferentes que se van pulsando correctamente.

Piano

Esta clase es un piano cuyas teclas son pulsadas automáticamente conforme se reproduce una canción en formato midi. La canción puede tener varios instrumentos que suenan a la vez, y cada uno de ellos se identifica por un número. Puede haber varios tipos de piano, que se diferencian en cómo gestionan los distintos instrumentos que tiene la canción.

Propiedades:

- **teclaInicial:** Es el número de la primera tecla del piano
- **teclaFinal:** Es el número de la última tecla del piano
- **posicion:** Guarda las coordenadas donde se dibuja el piano. Dichas coordenadas corresponden a la posición del vértice superior izquierdo del piano.
- **graphics:** Es el **Graphics** que se utiliza para dibujar el piano.

Métodos:

- **Constructor:** Su misión es crear un piano cuya primera nota musical es el parámetro “teclaInicial” y su última nota musical es el parámetro “tecla Final”
- **getTeclaInicial:** Devuelve el número de la tecla inicial del piano
- **getTeclaFinal:** Devuelve el número de tecla final del piano
- **getTecla:** Devuelve la tecla que corresponde al instrumento cuyo número se pasa como el parámetro llamado **canal** y su número de tecla (ver la imagen del piano de la clase **Tecla**) es el segundo parámetro.
- **setGraphics:** Es un setter para la propiedad **graphics**.
- **setPosicion:** Es un setter para la propiedad **posicion**

PianoSencillo

Esta clase es un piano formado por un único teclado. Todos los instrumentos de la canción se pulsan y sueltan sobre el mismo teclado.

Propiedades:

- **teclas:** Es un Map donde se van a guardar las teclas del piano. Como clave se usará el número de nota musical, y como valor, la tecla correspondiente del piano, según el dibujo que hay en la página de la clase **Tecla**

Métodos:

- **Constructor:** Su misión es crear un piano cuya primera nota musical es el parámetro “teclaInicial” y su última nota musical es el parámetro “tecla Final”. Para ello, tendrás que programarlo siguiendo estos pasos:
    - Inicializa las propiedades **teclaInicial** y **teclaFinal** con los parámetros
    - Inicializa el Map de la propiedad **teclas** con un **HashMap**
    - Recorre todos los números que hay entre la primera y la última nota musical que va a tener el piano, y por cada uno de esos números:
        - Crea tecla correspondiente al número que estamos recorriendo.
        - Guarda la tecla creada en el Map, usando como clave su número de nota musical.
- **dibujar:** Este método hace esto:
    - recorre las teclas del piano
    - Llama al método **setGraphics** de la tecla recorrida y le pone el **graphics** del piano
    - Llama al método **setPosicion** de la tecla recorrida y la sitúa en las coordenadas apropiadas, teniendo en cuenta la anchura que aparece en los dibujos de las páginas de las teclas. La primera tecla se colocará en la posición indicada en la propiedad **posicion** y las demás, a partir de ella, sumando el ancho correspondiente a cada tipo de tecla.
    - Llama al método **dibujar** de cada una de ellas.
- **getTecla:** Este método localiza en el Map la tecla pasada como parámetro y la devuelve. Se ignora el parámetro **canal**.

Test:

Haz un programa de pruebas que haga estos pasos:

- Crea un **Piano** con las teclas entre 36 y 72
- Crea un **ElementoVisualTester** y pásale el piano
- Llama al método **hacerPrueba** del tester y comprueba que el piano aparece

ReproductorMidi

Esta clase es un reproductor que hace que comience a sonar un archivo MIDI y se encargará de pulsar y soltar las teclas de un piano que tiene conectado conforme suena la música, al ritmo de la reproducción.

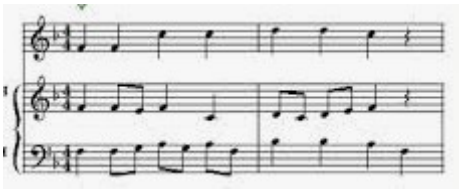
Para saber cuáles son las teclas que se pulsan y se sueltan mientras se reproduce la canción, el reproductor implementa la interfaz **Receiver**, (paquete **javax.sound.midi)**. De esta forma, el piano adquiere la habilidad de ser avisado cada vez que una nota musical se pulsa o se suelta durante la reproducción.

Propiedades:

- **COLORES:** Es un array que tiene que guardar 16 colores. Rellenalo con los 16 colores que más te gusten. Usaremos este array para que las notas musicales emitidas por cada instrumento tengan un color diferente.
- **piano:** Es un piano que está conectado al reproductor, de forma que el reproductor pueda controlarlo mientras suena la música.

Métodos:

- **constructor:** No hace nada y deja a **null** el piano conectado
- **conectar:** Conecta al reproductor el piano recibido como parámetro.
- **reproducir:** Este método recibe como parámetro la ruta de un archivo Midi que queremos reproducir, y utiliza las clases del paquete **javax.sound.midi** para:
    - Carga la canción MIDI cuya ruta se pasa como parámetro y un obtén un objeto **Sequence** con ella.



*Un **Sequence** es una partitura en formato MIDI*

    - Obtén un **Sequencer** con el secuenciador por defecto que usa Java

*El **Sequencer** es el encargado de reproducir la canción y enviar comandos musicales a los instrumentos y dispositivos que tiene conectados a él.*

- Abre el secuenciador
- Pide un **Transmitter** al **Sequencer**

*El **Transmitter** es un cable que permite conectar la salida de un dispositivo MIDI con la entrada de otro, llamado **Receiver**. Como la clase **ReproductorMidi** implementa la interfaz **Receiver**, podemos conectar el **Transmitter** que nos da el **Sequencer** con el objeto **ReproductorMidi** que estamos programando. De esa forma, recibiremos todos los comandos musicales que emita el **Sequencer** y podremos saber cuándo se pulsan y se sueltan las teclas durante la canción*

  - Conecta al **Transmitter** con el objeto **ReproductorMidi** que está siendo programado, de forma que este pueda recibir los comandos que envíe el secuenciador.
  - A continuación, carga en el **Sequencer** la canción MIDI que vamos a reproducir
  - Haz que el **Sequencer** comience a reproducir la canción.
  - Consulta al objeto **Sequence** para obtener la duración de la canción y haz una pausa con dicha duración.
  - Cuando termine la pausa, cierra **Transmitter** y el **Sequencer** para liberar los recursos del sistema que hayan utilizado.

- **send:** Es el método que hay que programar por implementar **Receiver**. Este método se activa automáticamente cada vez que el **Sequencer** genera un comando musical durante la reproducción de la canción. Deberás programar este método así:
  - Coge el **MidiMessage** recibido y usa **instanceof** para comprobar si su clase es **ShortMessage**
  - En caso de que no sea así, el método terminará sin hacer nada más.
  - En caso de que nuestro **MidiMessage** sea un **ShortMessage**, hazle un casting para obtener un **ShortMessage** a partir de él.
    - *Para todo lo que viene en adelante, usa el objeto ShortMessage*

- Obtén el “número de canal” del mensaje.
    - *El número de canal es el instrumento musical que emite el mensaje*
- Si el “número de canal” es 9, el método termina sin hacer nada más.
    - *Cuando el número de canal es 9, el instrumento es de percusión y no tiene sentido mostrarlo en el piano.*
- Si el número de canal es distinto de 9, obtén el número de la nota musical que sale en el **ShortMessage**
- Llama al método **getTecla** del piano conectado al reproductor, pasando como parámetro el número de la nota musical obtenida.
    - *De esta forma, obtienes la tecla que hay que pulsar o soltar*
- Si no existe esa tecla para esa nota (por ejemplo, porque el piano no ha sido creado con tantas teclas), el método terminará sin hacer nada más.
- Si la tecla existe, usa el método **getCommand** del **ShortMessage** para obtener un “número de comando” que nos dice si la tecla ha sido pulsada o soltada.
- Si el “número de comando” es igual a la constante **ShortMessage.NOTE_ON**, la tecla ha sido pulsada, y entonces hay que hacer esto:
    - Obtén el volumen con el que se ha pulsado la tecla:
        - Si el volumen es positivo:
            - Llama al método **setColorPulsado** de la tecla y pásale el color que corresponde al número de canal dentro del array **COLORES**
            - Llama al método **pulsar** de la tecla
        - Si el volumen es 0, llama al método **soltar** de la tecla.
- Si el “número de comando” es igual a la constante **ShortMessage.NOTE_OFF**, entonces es otro caso en el que hay que llamar al método **soltar**
- En ambos casos, para terminar hay que llamar al método **dibujar** de la tecla, de forma que la tecla se redibuje con su nuevo estado.

- **close:** No hace nada. Puedes dejarlo vacío.

Programa

Esta clase es el programa que crea un piano, lo conecta a un reproductor y por último, lo pone en marcha.

- **main:** Hace esto:
  - Iniciar una ConsolaDAW y ponerle un fondo con color sólido RGB(0,0,70)
  - Preguntar la ruta de un archivo
  - Crear un objeto PianoSencillo con las teclas comprendidas entre las notas musicales con números 24 y 108
  - Poner al piano:
    - El graphics de la CapaCanvas
    - Las coordenadas (120,90). Si por tu pantalla no se ve bien, puedes cambiarlas.
  - Tras poner las coordenadas, el piano deberá verse dibujado en la pantalla.
  - Ahora crea un ReproductorMidi y conéctale el piano.
  - Por último, haz que el reproductor reproduzca la canción cuya ruta se obtuvo al principio.