# RoadMap:

* Funciones básicas:
  + ~~Desplegar tablero gráfico.~~
  + ~~Desplegar información del juego en el panel inferior: Despliega información del estado del juego, del estado de la partida, y de los jugadores en los controles en el panel inferior.~~
  + Evaluar línea: algoritmo que evalúa si con la última ficha colocada se forma una línea que haga ganar al último jugador.
  + Resaltar línea ganadora creada.
  + Declarar ganador y reiniciar partida o salir del juego si los usuarios lo requieren.
  + Iniciar partida:
    - Solicitar nombre de los jugadores.
    - Inicializar las variables de juego.
* Funciones preferenciales:
  + .

# Diagrama de clases:

**Celda:**

* Rectangle celda
* Rectangle borde
* Ficha ficha
* boolean estáLlena = f
* int BORDE = 5
* int celdaID
* Celda(dbl, dbl, int, int)
* bool recibirFicha(Color)
* void unloadSlot()
* bool estaLlena()
* void highlightSlot()
* Color getTokenColor()
* void draw()
* String toString()

**Tablero:**

* Columna[] columnas
* boolean estaLleno = f
* Celda[] linea
* Tablero(int, int)
* bln recibirFicha(Color,int)
* void unloadBoard()
* int getQtyTokens()
* int getColumnsNumber()
* int getRowsNumber()
* int getSlotsNumber()
* int getBoardColumnWidth()
* int getBoardUpperMargin()
* int getBoardLeftMargin()
* int evaluateLine(pos,lineSz)
* void getEvalSlots()
* String toString()

**Columna:**

* ArrayList<Celda> celdas
* int qtyCeldas
* int columnaID
* int marcador
* int ANCHO = 60
* int MARGEN\_SUP = 30
* int MARGEN\_IZQ = 10
* Columna(int, int)
* boolean recibirFicha(Color)
* int getRowsNumber()
* int getQtyTokens()
* Celda getSlot(position)
* int getColumnWidth()
* int getUpperMargin()
* int getLeftMargin()
* boolean isFull()
* void unloadColumn()
* void draw()
* String toString()

**Jugador:**

* Nombre
* ColorFichas
* JugarTurno()
* Desplegar (toString())
* Desplegar (draw())

**Interface: Drawable**

* draw()

**Partida:**

* Tablero tablero
* Ellipse marcador
* boolean esTurnoAzul
* String nombreJugador1
* String nombreJugador2
* Canvas gameCanvas
* int tamLinea
* int columnaActiva = 0
* int juegosJugados = 1
* Color COLOR\_JUGADOR1
* Color COLOR\_JUGADOR2
* Color COLOR\_PANELES
* int ganadosJug1 = 0
* int ganadosJug2 = 0
* int ANCHO
* int MARGEN\_IZQ
* int MARGEN\_SUP
* Partida(Dificultad)
* JPanel crearPanelSuperior()
* JPanel crearPanelInferior()
* int ejecutarTurno()
* void reiniciarPartida()
* bln evaluarJugada(posicion, tamLinea)
* void informacionUsuarios()
* int[] parametrosJuego(Dific.)
* void leftButtonPressed()
* void rightButtonPressed()
* void playButtonPressed()
* void restartButtonPressed()
* IniciarPartida()
* CerrarPartida()
* String toString()
* void draw()

**Ficha:**

* int anchoLabio
* Ellipse cuerpo
* Ellipse borde
* Ellipse centro
* Ficha (int, int, int, Color)
* void setColor(Color)
* Color getColor()
* void highlightToken()
* void drawHole()
* String toString()
* void draw()

## Cont…

**Color (library):**

* Red
* green
* blue
* Color(int, int, int)
* getRed()
* getBlue()
* getGreen()

**Line (library) < Shape**

* X1
* Y1
* X2
* Y2
* Line(double, double double, double)
* getX()
* getY()
* getHeight()
* getWidth()
* setColor()
* translate()
* grow()
* toStirng()
* draw()
* paintShape()

**Rectangle (library) < Shape**

* Color
* filled
* x
* y
* width
* height
* Rectangle()
* Rectangle(dbl,dbl,dbl,dbl)
* getX()
* getY()
* getHeight()
* getWidth()
* translate()
* grow()
* setColor()
* draw()
* fill()
* toStirng()
* paintShape()

**Ellipse (library) < Shape**

* Color
* filled
* x
* y
* width
* height
* Ellipse()
* Ellipse(dbl,dbl,dbl,dbl)
* getX()
* getY()
* getHeight()
* getWidth()
* translate()
* grow()
* setColor()
* draw()
* fill()
* toString()
* paintShape()

**Interface: Shape (library)**

* getX()
* getY()
* getHeight()
* getWidth()
* paintShape()

**Picture (library) < Shape**

* image (BufferedImage)
* label (JLabel)
* source (String)
* x (double)
* y (double)
* xGrow (double)
* yGrow (double)
* Picture()
* Picture(double, double)
* Picture(String)
* load(String)
* getX()
* getY()
* getMaxX()
* getMaxY()
* getHeight()
* getWidth()
* pixels()
* getGrayLevels()
* Picture(int[][])
* toString()
* getColorAt(int)
* getColorAt(int, int)
* setColorAt(int, Color)
* setColorAt(int, int, Color)
* translate(double, double)
* grow(double, double)
* draw()
* paintShape()

**Canvas (library)**

* Canvas (static)
* shapes (ArrayList)
* background (buffImage)
* frame (JFrame)
* component (CanvasCmp)
* MIN\_SIZE (int)
* MARGIN (int)
* LOCATION\_OFFSET (int)
* canvasCmp (class)
* Canvas()
* getInstance()
* show()
* repaint()
* pause()
* snapshot()
* saveToDisk()

**CanvasCmp (library) << JComponent**

* paintComponent()
* getPreferredSize()

**Text (library) < Shape**

* color (Color)
* label (JLabel)
* x (double)
* y (double)
* xGrow (double)
* yGrow (double)
* Text (dbl, dbl, String)
* getX()
* getY()
* getHeight()
* getWidth()
* translate()
* grow()
* setColor()
* draw()
* fill()
* toStirng()
* paintShape()

**Evaluar Jugada:**

1. Lógica:
   1. Capturar las celdas adyacentes en un arreglo,
      1. Tomando como punto de partida la última ficha jugada (coordenadas),
      2. Capturar las celdas adyacentes en el arreglo de celdas
         1. Recorrer las celdas en la dirección adecuada:
            1. La captura de celdas se realiza en este orden de ángulos 0º, 45º, 90º, 135º, 225º, 270º, 315º.
            2. Al encontrarse la primera ‘coincidencia’ se abandona el ciclo.
            3. El recorrido se define por la dificultad del juego que define el tamaño de la línea a formar.
         2. Recuperar la celda del tablero por coordenada.
         3. Si la celda tiene ficha recuperar el color de la misma.
            1. Crear un ‘helper method en celda, getTokenColor(), retorne el color de la ficha.
   2. Evaluar el color de las fichas adyacentes,
      1. Verificar que sean del mismo color.
   3. Si todas son del mismo color retornar afirmativamente y salir de la función.
   4. Resaltar las celdas.
      1. Crear el método para resaltar ficha en la celda.
2. Parámetros:
   1. Posición de última ficha.
      1. Columna: Integer.
      2. Fila: Integer.
   2. Tamaño de la línea.
3. Retorna:
   1. Boolean;
      1. Negativo: no hay ganador y el juego continúa.
      2. Afirmativo: hay ganador.
4. Desarrollo:
   1. Programar el llenado del arreglo de celdas que sean adyacentes (y en línea), y resaltarlas.
   2. Programar el test case.

**Ganar Juego:**

1. Lógica:
   1. Resalta la línea ganadora.
   2. Otorga juego al ganador y actualiza marcador, juegos jugados, etc.
2. Parámetros:
   1. Color del jugador
3. Retorno:
   1. Nada.

**Ganar Partida:**

1. Lógica:
   1. Resalta la línea ganadora.
   2. Otorga juego al ganador y actualiza marcador, juegos jugados, etc.
2. Parámetros:
   1. Color del jugador
3. Retorno:
   1. Nada.