// @ = a

// king= K, Queen = Q, Rook = R, Bishop = I, Knight= N, Pawn= a la letra de la casilla en el que esta

// x capture B x c6(pice, x(by), casilla a la que se mueve )

// para promotion $$ convierte e8 = Q (casilla del peon, = , letra en la que se convierte ), eXd1 = N

// + check , # checkmate, $$ jaque , jaque mate

// castling $$ enroque  0-0, 0-0-0

// Nce4 = N , c , e4 (pieza, lugar de origen, casilla a la que se mueve )

// 1 use file letter (if diferrent), 2 use rank number (if diferent), 3 use both(if still not identifiable )

//pieza jugable

// pieza no jugable

// fila a @ h = todas las casillas con la misma posision de fila son esa fila = x

// rango 1 @ 8 = todas las casillas con la misma posision de rango son ese rango = y

// casilla = cuadro con una posicion de fila y una de rango

// casilla pos = cuadro al donde se va a mover la pieza

// casilla rival = cuadro donde hay una pieza rival

// casilla aliado = cuadro donde hay una pieza aliada

// casilla de pieza = cuadro donde esta la pieza que vas a mover

//pasa = las casillas que pasa, para llegar @ casilla pos

// si jugador1 play , tiempo jugador1 = tiempo jugador1 -1

// si jugador1 movio, jugador1 pause, jugador2 play

//si jugador2 play , tiempo jugador2 = tiempo jugador2 -1

// si jugador2 movio, jugador2 pause, jugador1 play

// si el \_\_\_  movimiento es valido === true, movio , si no, no hacer movimiento y sonar alarma

// pawn movimieto valido  jugador1

// si casilla pos !== casilla rival || casilla aliada, true, sino false

// si pasa !== casilla rival || casilla aliada, true, si no false

// si casilla pieza !== rango2 && casilla pos es === casilla pieza.rango +1 , true,  si no false

//si casilla pieza === rango2 && casilla pos es === casilla pieza.rango +2 || casilla pieza.rango +1, true, si no false

// si  casilla pos = casilla pieza.fila + 1 || casilla pieza.fila -1  && casilla pieza.rango +1 && casilla rival, true, funcion comer, si no false

// falta peon al pazo

// pawn movimieto valido  jugador2

// si casilla pos !== casilla rival || casilla aliada, true, sino false

// si pasa !== casilla rival || casilla aliada, true, si no false

// si casilla pieza !== rango7 && casilla pos es === casilla pieza.rango -1 , true,  si no false

//si casilla pieza === rango7 && casilla pos es === casilla pieza.rango -2 || casilla pieza.rango -1, true, si no false

// si  casilla pos = casilla pieza.fila + 1 || casilla pieza.fila -1  && casilla pieza.rango -1 && casilla rival, true, funcion comer, si no false

// si funcion limites === true , true , sino false

// falta peon al pazo

//rook movimiento valido juador1 y jugador2

// si pasa !== casilla rival || casilla alida, true, sino false

// si casilla pos.fila === casilla pieza.fila || si casilla pos.rango === casilla pieza.rango, true, sino false

// si funcion limites === true , true , sino false

// bishop movimientos validod

// si pasa !== casilla rival || casilla aliada, true, si no false

// si la diferencia en casilla pos es  +n || -n fila === +n rango  && +n || -n rango , true  ,sino false

// si funcion limites === true, true, si no false

//Knight movimientos validos

// si 2 rango && 1 fila || 2 rango && -1 fila || -2 rango && 1 fila || -2 rango && -1 fila || 1 rango y 2 fila ||  -1 rango y 2 fila ||

//  1 rango y -2 fila ||  -1 rango y -2 fila, true , sino false

// si funcion limites === true , true , sino false

// king movimiento valido jugador1 y jugador2

// si casilla pos= casilla pieza.fila+1 || casilla pieza.fila-1 || casilla pieza.rango+1 || casilla pieza.rango-1 ||

// (casilla pieza.fila + 1 || casilla pieza.fila -1  && casilla pieza.rango +1)|| (casilla pieza.fila + 1 ||

// casilla pieza.fila -1  && casilla pieza.rango -1)

// si funcion limites === true , true , sino false

//Queen movimientos validos

//si casilla pos.fila === casilla pieza.fila || si casilla pos.rango === casilla pieza.rango ||

// la diferencia en casilla pos es  +n || -n fila === +n rango  && +n || -n rango, true, sino false

// si pasa !== casilla rival || casilla aliada, true, si no false

// si funcion limites === true , true , sino false

// funcion movio, pieza.posicion = casilla pos.posicion

// funcion comer, pieza rival === casilla pos , cambiar a una posision fuera del tablero , pieza no jugable

// funcion limites, si pos.fila =< a ||   pos.fila =>h || pos.rango =< 1 ||pos.rango =>8, false, si no true

// funcion cheeck, si pieza rival, puede moverse a la casilla donde esta el rey, play sound cheek

//funcion cheeck mate, si todas las pos, puede hacer la funcion cheeck, game state end.

// funcion brinco,  si casilla pasa === casilla, break, si casilla pasa === casilla aliada || casilla rival, pieza selec, z -10(// que suba en la altutud en esa casiilla)

// falta funcion enroque

namespace Chess.Elements

{

    /// <summary>

    /// </summary>

    public abstract class Piece\_Base : Sprite

    {

        #region Constructor

       \* public Piece\_Base(Image image, PiecePlayer player) : base(image, new Point())

        {

          \*  this.Player = player;

        }

        #endregion

        #region Properties

        /// <summary>

        /// </summary>

       \* public PiecePlayer Player { get; set; }

        /// <summary>

        /// </summary>

        public Point Location { get; set; }

        /// <summary>

        /// </summary>

        public bool Selected { get; set; }

        /// <summary>

        /// </summary>

       \* public PosMove[] Moves { get; set; }

        /// <summary>

        /// </summary>

        public Point[] EnabledMoves { get; set; }

        /// <summary>

        /// </summary>

        public Image SelectedImage { get; set; }

        #endregion

        #region Methods

        /// <summary>

        /// </summary>

        /// <param name="drawHandler">controlador de dibujado</param>

        public override void Draw(DrawHandler drawHandler)

        {

            if (this.Selected)

                drawHandler.Draw(this.SelectedImage, this.Position);

            base.Draw(drawHandler);

        }

        #endregion

    }

    /// <summary>

    /// </summary>

   \* public class PosMove

    {

      \*  public PosMove(int x, int y, bool isLinear = true, Move\_Type type = Move\_Type.General)

        {

            this.Direction = new Point(x, y);

            this.Type = type;

            this.IsLinear = isLinear;

        }

        /// <summary>

        /// </summary>

        public Point Direction { get; set; }

        /// <summary>

        /// </summary>

        public Move\_Type Type { get; set; }

        /// <summary>

        /// </summary>

        public bool IsLinear { get; set; }

    }

    /// <summary>

    /// </summary>

    public enum Move\_Type

    {

        General = 1,

        Special = 2

    }

    ///<summary>

    /// </summary>

  \*  public enum PiecePlayer

    {

   \*     Player1,

    \*    Player2

    }

}

namespace Chess.Elements

{

   \* public class Pawn : Piece\_Base

    {

        #region Constructor

       \* public Pawn(Image image, PiecePlayer player) : base(image, color)

        {

       \*     Moves = new PosMove[]

            {

         \*       new PosMove(0,-1, false, Move\_Type.Special),

          \*      new PosMove(0,-2, false, Move\_Type.Special),

           \*     new PosMove(-1,-1, false, Move\_Type.Special),

            \*    new PosMove(1,-1, false, Move\_Type.Special),

            };

        }

        #endregion

    }

}

namespace Chess.Elements

{

  \*  public class Rook : Piece\_Base

    {

        #region Constructor

      \*  public Rook(Image image, PiecePlayer player) : base(image, color)

        {

          \*  Moves = new PosMove[]

            {

           \*     new PosMove(0,1),

           \*     new PosMove(0,-1),

           \*     new PosMove(1,0),

           \*     new PosMove(-1,0),

            };

        }

        #endregion

    }

}

namespace Chess.Elements

{

  \*  public class Bishop : Piece\_Base

    {

        #region Constructor

       \* public Bishop(Image image, PiecePlayer player) : base(image, color)

        {

            Moves = new PosMove[]

            {

              \*  new PosMove(1,1),

               \* new PosMove(-1,1),

              \*  new PosMove(1,-1),

              \*  new PosMove(-1,-1),

            };

        }

        #endregion

    }

}

namespace Chess.Elements

{

  \*  public class Knight : Piece\_Base

    {

        #region Constructor

       \* public Knight(Image image, PiecePlayer player) : base(image, color)

        {

         \*   Moves = new PosMove[]

            {

           \*     new PosMove(-1,-2, false),

          \*      new PosMove(1,-2, false),

          \*      new PosMove(-1,2, false),

           \*     new PosMove(1,2, false),

            \*    new PosMove(2,-1, false),

          \*      new PosMove(2,1, false),

            \*    new PosMove(-2,-1, false),

           \*     new PosMove(-2,1, false),

            };

        }

        #endregion

    }

}

namespace Chess.Elements

{

 \*   public class Queen : Piece\_Base

    {

        #region Constructor

     \*   public Queen(Image image, PiecePlayer player) : base(image, color)

        {

         \*   Moves = new PosMove[]

            {

             \*   new PosMove(0,1),

             \*   new PosMove(0,-1),

              \*  new PosMove(1,0),

             \*   new PosMove(-1,0),

             \*   new PosMove(1,1),

              \*  new PosMove(-1,1),

              \*  new PosMove(1,-1),

             \*   new PosMove(-1,-1),

            };

        }

        #endregion

    }

}

namespace Chess.Elements

{

   \* public class King : Piece\_Base

    {

        #region Constructor

        \*public King(Image image, PiecePlayer player) : base(image, color)

        {

           \* Moves = new PosMove[]

            {

         \*       new PosMove(-1,-1, false),

         \*       new PosMove(-1,0, false),

         \*       new PosMove(-1,1, false),

          \*      new PosMove(0,-1, false),

          \*      new PosMove(0,1, false),

          \*      new PosMove(1,-1, false),

          \*      new PosMove(1,0, false),

          \*      new PosMove(1,1, false),

            };

        }

        #endregion

    }

}