importjava.awt.\*;

importjavax.swing.\*;

importjava.lang.\*;

/\*\*

 \*  A class that displays a map based on data held in a MapGrid object.

 \*

 \*  @authors Christina He and Audrey Dawson

 \*  @version Wednesday 20, 2017

 \*/

publicclassMapViewerextendsJComponent {

    /\*\* field to keep track of the MapGrid to be displayed \*/

    privateMapGridmap;

    /\*\* field to specifiy magnification of view \*/

    privateintmagnification;

    /\*\*field for starting coordinates/offset \*/

    privatePointcoordinate;

    /\*\* field for sixe of viewing window \*/

    privateintpixels;

    /\*\* default constructor that reads in MapGrid object and initializes all fields

     \*

     \* @param map mapGrid object

     \*/

    publicMapViewer(MapGridmap) {

        this.map = map;

        magnification = 10;

        coordinate = newPoint(0,0);

        setPreferredSize(newDimension(380,380));

        setMinimumSize(newDimension(380,380));

        pixels = 380;

    }

    /\*\* constructor that reads in MapGrid object and initializes all fields

     \*

     \* @param map mapGrid object

     \* @param x x value of offset point

     \* @param y y value of offset point

     \* @param magnification value of magnification

     \*/

    publicMapViewer(MapGridmap, intx, inty, intmagnification, intpixels) {

        this.map = map;

        this.magnification = magnification;

        coordinate = newPoint(x,y);

        setPreferredSize(newDimension(pixels,pixels));

        setMinimumSize(newDimension(pixels,pixels));

        this.pixels = pixels;

    }

    /\*\* accessor for magnification

     \*

     \* @return magnification value of magnification

     \*/

    publicintgetMag() {

        return magnification;

    }

    /\*\* modifier for magnification

     \*

     \* @param magnification value of magnification

     \*/

    publicintgetMag() {

        return magnification;

    }

    /\*\* modifier for magnification

     \*

     \* @param magnification value of magnification

     \*/

    publicvoidsetMag(intmagnification) {

        this.magnification = magnification;

    }

    /\*\* accessor for starting coordinate/offset

     \*

     \* @return coordinate offset point

     \*/

    publicPointgetCoordinate() {

        return coordinate;

    }

    /\*\* modifier for starting coordinate/offset

     \*

     \* @param x x value of offset point

     \* @param y y value of offset point

     \*/

    publicvoidsetCoordinate(double x, double y) {

        this.coordinate = newPoint(x,y);

    }

    /\*\* accessor for size of window

     \*

     \* @returnpixels size of viewing window

     \*/

    publicintgetPixels(){

        return pixels;

    }

    /\*\* modifer for size of window

     \*

     \* @param pixels size of viewing window

     \*/

    publicvoidsetPixels(intpixels) {

        this.pixels = pixels;

    }

    /\*\*

     \* overrides JComponentpaintComponent method and draws the MapGrid object

     \*

     \* @param g AWT Graphics object

     \*/

    publicvoidpaintComponent(Graphicsg) {

        // variable that holds the offset point  y value

        intyCoord = (int)coordinate.getY();

        // variable that holds the offset point x value

        intxCoord = (int)coordinate.getX();

        /\* sets color outside of map \*/

        g.setColor(Color.BLACK);

        g.fillRect(0,0,pixels, pixels);

        /\* loops through the MapGrid colors array to draw the map at the given magnification from the given offset\*/

        for (inti = 0; i<map.getGridHeight(); i++) {

            for (intj = 0; j<map.getGridWidth(); j++) {

                // sets the color based on the color data in MapGrid object

                g.setColor(map.getCellColor(i,j));

                // fills a rectangle starting from x and y coordinates based on offset and magnification

                g.fillRect((i-xCoord)\*magnification, (j-yCoord)\*magnification, magnification, magnification);

            }

        }

    }

    publicvoidNorth(){

        intx=(int)coordinate.getX();

        inty=(int)coordinate.getY();

        y=y-map.getGridHeight()/4;

        this.setCoordinate(x,y);

        repaint();

    }

    publicvoidSouth(){

        intx=(int)coordinate.getX();

        inty=(int)coordinate.getY();

        y=y+map.getGridHeight()/4;

        this.setCoordinate(x,y);

        repaint();

    }

    publicvoidWest(){

        intx=(int)coordinate.getX();

        inty=(int)coordinate.getY();

        x=x-map.getGridWidth()/4;

        this.setCoordinate(x,y);

        repaint();

    }

    publicvoidEast(){

        intx=(int)coordinate.getX();

        inty=(int)coordinate.getY();

        x=x+map.getGridWidth()/4;

        this.setCoordinate(x,y);

        repaint();

    }

    publicvoidZoomIn(){

        intx=(int)coordinate.getX();

        inty=(int)coordinate.getY();

        magnification=(int)(1.5\*magnification);

        this.setMag(magnification);

        x=x+(int)(((190-magnification\*map.getGridWidth())/2)/magnification);

        y=y+(int)(((190-magnification\*map.getGridHeight())/2)/magnification);

        this.setCoordinate(x,y);

        repaint();

    }

    publicvoidZoomOut(){

            intx=(int)coordinate.getX();

            inty=(int)coordinate.getY();

            magnification=(int)(magnification/1.5);

            this.setMag(magnification);

            x=x-(int)(((190-magnification\*map.getGridWidth())/2)/magnification);

            y=y-(int)(((190-magnification\*map.getGridHeight())/2)/magnification);

            this.setCoordinate(x,y);

            repaint();

    }

    /\*\* Mouse event handler\*/

    publicvoidMouse(){

        repaint();

    }