J3 課題
Java プログラミングレポート

2010494 中原良典

1　課題概要

図形を描画するプログラムを作った.
実装した機能

- 色は赤, 青, 黄, 緑, 黒, オレンジ, ピンクの 6 色で図形を描画できる. currentColor は, setDrawColor を使って DrawFrame 内で変更することにより，ボタンによる色の変更を実現した.
- 図形は長方形, 楕円, 直線を描画できる. DrawModel クラス内の createFigure の中で, ccurrentFigure という文字列の値によって RectangleFigure，OvalFigure，LineFigure に切り替えられるように実装した.

2　設計方針

全体のクラス構成
Figure クラス
図形を描画するクラス.
RectangleFigure，OvalFigure，LineFigure：それぞれ，長方形，楕円，直線を表すサブクラス.
DrawModel：描画したデータを記録しておくクラス.
このクラスは Observable のクラスなので，Observable クラスを継承しており，描かれた図形を ArrayList を使って記録している．これにより次々に生成される図形を記録することができる．Figure クラスで定義したメソッドを使用して描画する.
view クラス
DrawModel が記録している図形を画面に表示する ViewPanel クラスとして定義した．これは JPanel を継承している．また，ViewPanel は Observer のクラスなので，Observer インタフェースも implements している．このクラスでは DrawModel と Drawcontroller を受け取って，マウスからの操作に対応できるように addMouseListener メソッドと addMouseMotionListener メソッドを関連づけている.
DrawController クラス：リスナークラス
マウス操作によって新しく図形を描画できるために, MouseListener と MouseMotionListener を implements している.
最後にメインクラスを DrawFrame の中で作った．DrawModel，DrawControlloer，ViewPanel オブジェ

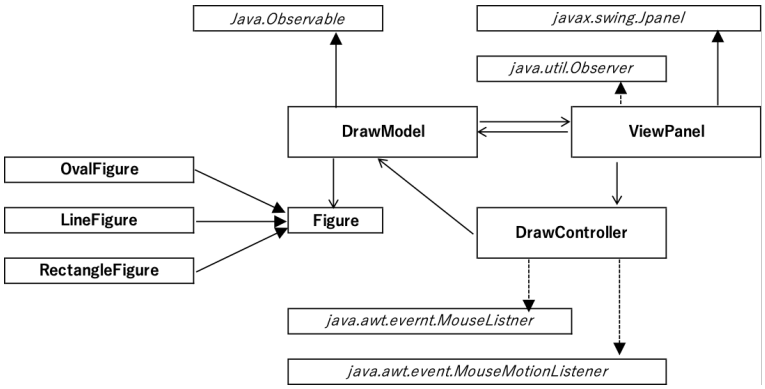

図 1　クラス図

クトを順に生成し，main メソッドで DrawFrame オブジェクトを生成している．

3　プログラムの説明

MVC モデルの GUI プログラムを作成した．図形を表すクラスとして Figure クラスを作り，そのサブクラスとして長方形，楕円，直線を表す RectangleFigure，OvalFigure，LineFigure を定義する．位置と大きさと色を x，y，w，h，c の引数で指定するコンストラクタを作る．

```
class Figure {
    protected int x, y, width, height;
    protected Color color;

    //位置と大きさと色をx，y，w，h，cの引数で指定するコンストラクタ
    public Figure(int x, int y, int w, int h, Color c) {
        this.x = x; this.y = y;
        width = w; height = h;
        color = c;
    }

    //位置x，yと大きさw，hを変更するメソッド
    public void setSize(int w, int h) {
        width = w; height = h;
    }
    public void setLocation(int x, int y) {
        this.x = x; this.y = y;
    }

    //2点の座標によって位置と大きさを設定するメソッド
    public void reshape(int x1, int y1, int x2, int y2) {
        int newx = Math.min(x1, x2);
        int newy = Math.min(y1, y2);
        int neww = Math.abs(x1 - x2);
```

```
        int newh = Math.abs(y1 - y2);
        setLocation(newx, newy);
        setSize(neww, newh);
    }
    public void draw(Graphics g) {}
}
```

長方形，円，直線を表すサブクラス RectangleFigure，OvalFigure，LineFigure をそれぞれ定義する．それぞれの図形はグラフィッククラスのメソッドを使って定義できる．長方形は drawRect，楕円は drawOval，直線は drawLine でそれぞれ定義される．

```
class RectangleFigure extends Figure {
    public RectangleFigure(int x, int y, int w, int h, Color c) {
        super(x, y, w, h, c);
    }
    public void draw(Graphics g) {
        g.setColor(color);
        g.drawRect(x, y, width, height);
    }
}

class OvalFigure extends Figure {
        public OvalFigure(int x, int y, int w, int h, Color c) {
                super(x, y, w, h, c);
        }
        public void draw(Graphics g) {
                g.setColor(color);
                g.drawOval(x, y, width, height);
    }
}

class LineFigure extends Figure {
        public LineFigure(int x, int y, int w, int h, Color c) {
                super(x, y, w, h, c);
    }
        public void draw(Graphics g) {
                g.setColor(color);
                g.drawLine(x, y, width, height);
    }
}
```

描画したデータを記録しておく，プログラムの中心となるクラス DrawModel を作る．Observer パターンにおける Observable なので，Observable クラスを継承している．描かれた図形は全て ArrayList を使って記録している．操作対象になっている図形，色を記録，つまりボタン操作によって設定されている図形を記録している．

```
class DrawModel extends Observable {
```

```
protected ArrayList<Figure> fig;
protected Figure drawingFigure;
protected Color currentColor;
protected String currentFigure; //現在の図形の形を入れる
public DrawModel() {
    fig = new ArrayList<Figure>();
    drawingFigure = null;
    currentColor = Color.red;
    currentFigure = "Rectangle";
}

//ArrayListを返すメソッドとArrayListの図形を取り出すメソッド
public ArrayList<Figure> getFigures() {
    return fig;
}
public Figure getFigure(int idx) {
    return fig.get(idx);
}

//形を変更するメソッド
//ここで入れられた文字列によって後で形を変更できるようにする
void setFigure(String figure) {
        currentFigure = figure;
}

//新たに図形を作って追加するメソッドと操作中の図形の形を変更するメソッド
//図形の情報を変更したらsetChangedメソッド，notifyObserversメソッドを実行して
//modelが変化したことをObserverであるviewに通知する
public void createFigure(int x, int y) {
    Figure f = new RectangleFigure(x, y, 0, 0, currentColor);
        if(currentFigure == "Oval") {
        f = new OvalFigure(x, y, 0, 0, currentColor);
    } else if(currentFigure == "Line") {
        f = new LineFigure(x, y, x, y, currentColor);
    }
        fig.add(f);
        drawingFigure = f;
        setChanged();
        notifyObservers();
}

public void reshapeFigure(int x1, int y1, int x2, int y2) {
    if (drawingFigure != null) {
        drawingFigure.reshape(x1, y1, x2, y2);
        setChanged();
        notifyObservers();
```

```
        }
    }
    void setDrawColor(Color c) {
            currentColor = c;
    }
}
```

MVC の view を作る．DrawModel が記録している図形を画面に表示するクラス ViewPanel を view のクラスとする．JPanel クラスを継承してレイアウトを設定できるようにする．ViewPanel は Observer パターンにおける Observer でもあるので，Observer インタフェースも同時に implements する．コンストラクタで DrawModel と DraeController を受け取り，addMouseListener メソッド，addMouseMotionListener メソッドを呼び出してリスナーオブジェクトを関連づけておく．

```
class ViewPanel extends JPanel implements Observer {
    protected DrawModel model;
    public ViewPanel(DrawModel m, DrawController c) {
        this.setBackground(Color.white);
        this.addMouseListener(c);
        this.addMouseMotionListener(c);
        model = m;
        model.addObserver(this);
    }

    //paintComponentメソッド
    //getFiguresメソッドでDrawModelが記録している全てのFigureを描画
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        ArrayList<Figure> fig = model.getFigures();
        for(int i = 0; i < fig.size(); i++) {
            Figure f = fig.get(i);
            f.draw(g);
        }
    }

    //Updateメソッド
    //repaintを呼び出して再描画
    public void update(Observable o, Object arg) {
        repaint();
    }
}
```

MVC の controller は MouseListener，MouseMotionListener を implements したリスナークラスになる．機能としては，マウスが押されたらその座標を開始点として記録し新たに図形を作成する．ドラッグするとその図形の形を変化させる．マウスが押されたかどうかは MouseListener インタフェース，ドラッグは MouseMotionListener インタフェースを実装することで機能する．

```
class DrawController implements MouseListener, MouseMotionListener {
    protected DrawModel model;
    DrawFrame frame;
    protected int dragStartX, dragStartY;
    public DrawController(DrawModel a) {
        model = a;
    }
    public void mouseClicked(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {
        dragStartX = e.getX(); dragStartY = e.getY();
        model.createFigure(dragStartX, dragStartY);
    }
    public void mouseDragged(MouseEvent e) {
        model.reshapeFigure(dragStartX, dragStartY, e.getX(), e.getY());
    }
    public void mouseReleased(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mouseMoved(MouseEvent e) {}
}
```

最後にメインクラスを作成する．JFrame クラスを継承した DrawFrame のクラスのコンストラクタの中で，機能実現のためのオブジェクトであるボタンをを生成しパネルに貼り付け，それらの部品を this に貼り付ける．DrawModel，DrawController，ViewPanel オブジェクトを順に生成し，main メソッドで DrawFrame オブジェクトを一つ生成する．

```
class DrawFrame extends JFrame implements ActionListener{
    DrawModel model;
    ViewPanel view;
    DrawController cont;
    private JPanel p1, p2;
    private JButton b1, b2, b3, b4, b5, b6, b7, b8, b9, b10;
    public DrawFrame() {
        model = new DrawModel();
        cont = new DrawController(model);
        view = new ViewPanel(model,cont);
        p1 = new JPanel();
        p2 = new JPanel();
        b1 = new JButton("Red");
        b2 = new JButton("Brue");
            b3 = new JButton("Yellow");
            b4 = new JButton("Green");
            b5 = new JButton("Black");
            b6 = new JButton("Orange");
            b7 = new JButton("Pink");
            b8 = new JButton("Rectangle");
```

```
        b9 = new JButton("Oval");
        b10 = new JButton("Line");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        b4.addActionListener(this);
        b5.addActionListener(this);
        b6.addActionListener(this);
        b7.addActionListener(this);
        b8.addActionListener(this);
        b9.addActionListener(this);
        b10.addActionListener(this);
        p1.setLayout(new GridLayout(7, 1));
        p1.add(b1); p1.add(b2); p1.add(b3); p1.add(b4);
        p1.add(b5); p1.add(b6); p1.add(b7);
        p2.setLayout(new GridLayout(3, 1));
        p2.add(b8); p2.add(b9); p2.add(b10);
        this.add(p1, BorderLayout.WEST);
        this.add(p2, BorderLayout.EAST);
    this.setBackground(Color.black);
    this.setTitle("Draw Editor");
    this.setSize(500, 500);
    this.add(view);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(true);
}

public void actionPerformed(ActionEvent e) {
        if(e.getSource() == b1) {
                model.setDrawColor(Color.RED);
        } else if(e.getSource() == b2) {
                model.setDrawColor(Color.BLUE);
        } else if(e.getSource() == b3) {
                model.setDrawColor(Color.YELLOW);
        } else if(e.getSource() == b4) {
                model.setDrawColor(Color.GREEN);
        } else if(e.getSource() == b5) {
                model.setDrawColor(Color.BLACK);
        } else if(e.getSource() == b6) {
                model.setDrawColor(Color.ORANGE);
        } else if(e.getSource() == b7) {
                model.setDrawColor(Color.PINK);
        } else if(e.getSource() == b8) {
                model.setFigure("Rectagnle");
        } else if(e.getSource() == b9) {
                model.setFigure("Oval");
```

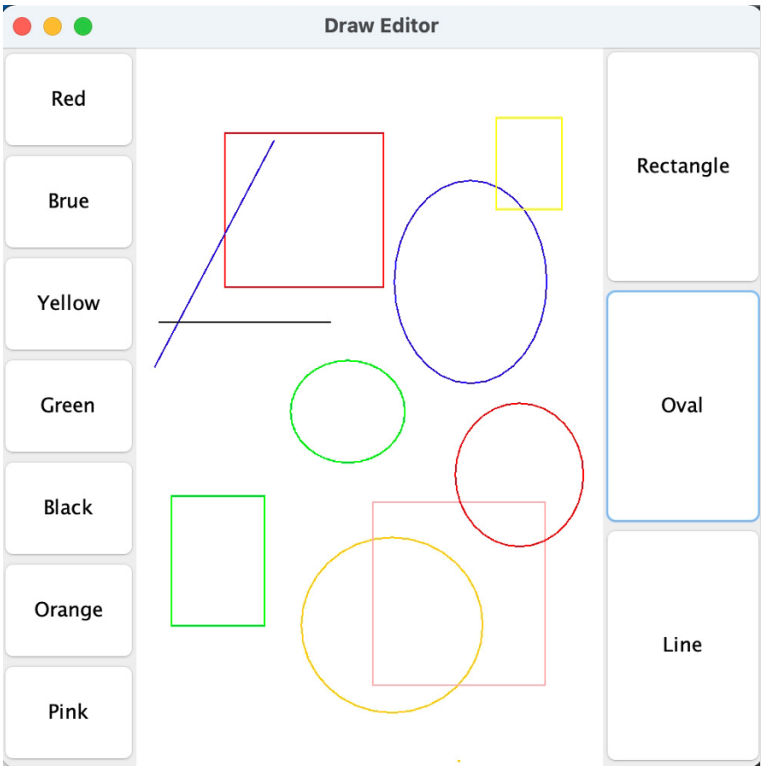

図 2　使用例

```
            } else if(e.getSource() == b10) {
                    model.setFigure("Line");
            }
    }

    public static void main(String[] args) {
        new DrawFrame();
    }
}
```

4　実行例

デフォルトで描くことができる図形は赤色の長方形である．そこからフレーム右側に並んだ，色が表示されているボタンを押すことで描画できる図形の色が変更され，右側に並んだ図形の形が表示されているボタンを押すことで描画する図形の形を変更することができる．以下に実行した際の例を示す．