Graphical User Interface 描画する

オブジェクト指向プログラミング特論

2020年度

只木進一：工学系研究科

本日の目的

- Javaの描画の基本を知る
  - 描画手順
- マウスイベントの基本を知る
  - マウスの動き
  - マウスボタンの操作

今日のサンプルプログラム

- https://github.com/oop-mc-saga/SampleGUI2

描画の基本
javax.swingにおける描画

- paint()がpaintComponent()、paintBorder()、およびpaintChildren()を順に呼び出す
  - java.awsではpaint()メソッドが描画
- paintComponent()メソッドを上書き
  - この中で描画対象を描く

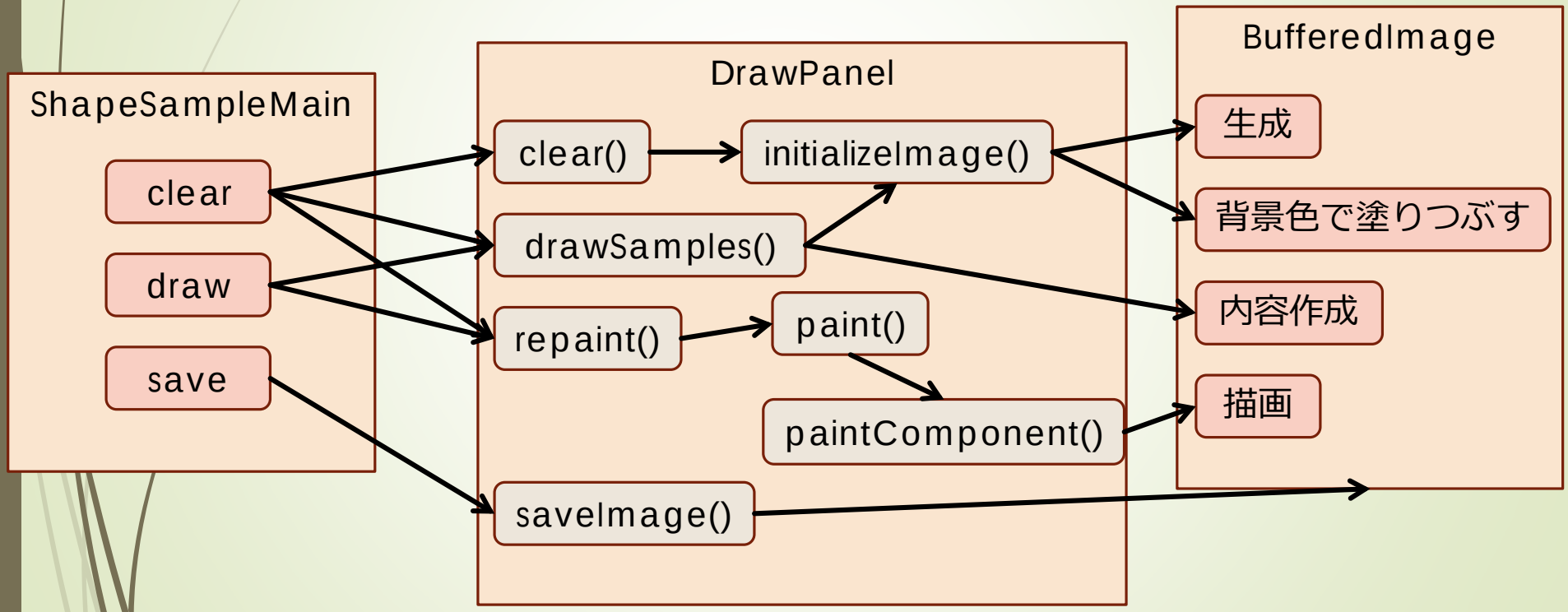
描画の基本
再描画に注意

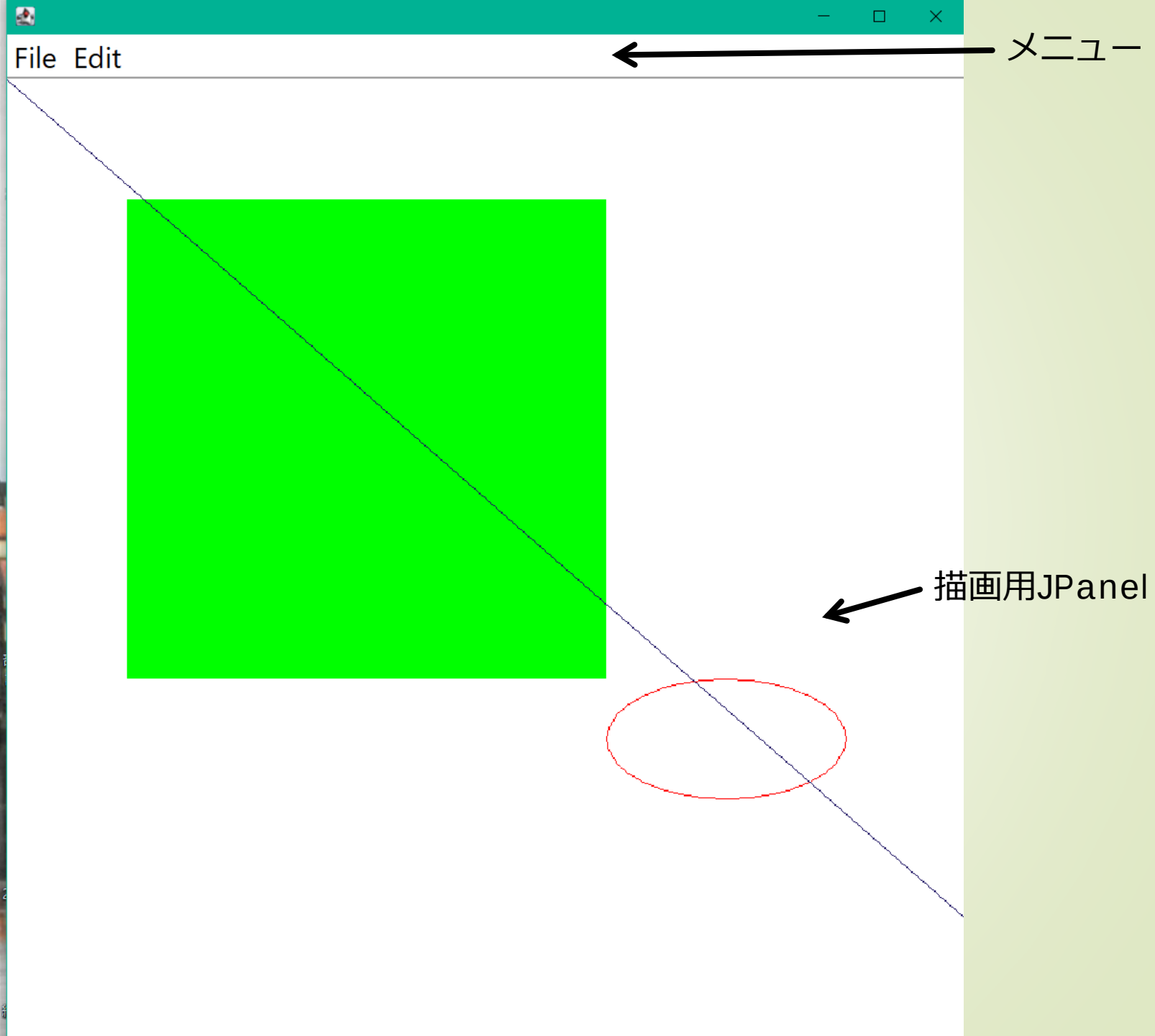
- ウィンドウの重なり
  - 順序が代わる度に再描画
- アイコン状態からの復帰
- 最初から書き直すと時間がかかる

描画の基本2

- ウィンドウの重なりで再描画が必要な場合
  - repaint() →paint()
- 再描画に備えて java.awt.image.BufferedImageとしてイメージを生成しておく
  - paintComponent()内ではimageを張り付けるだけ
  - imageをファイルに保存できる

shapeSampleの全体構成




File Edit
メニュー
描画用JPanel

イメージ初期化 initializeImage()メソッド

- BufferedImageのインスタンスを生成する
- 背景で全体を塗りつぶす

```
1.  public void initializeImage() {
2.      Dimension dimension = getPreferredSize();
3.      //空のイメージ生成
4.      image = new BufferedImage(dimension.width,
5.              dimension.height, BufferedImage.TYPE_INT_RGB);
6.      Graphics2D g = (Graphics2D) image.getGraphics();
7.      g.setColor(this.getBackground());//背景色で塗りつぶし
8.      g.fillRect(0, 0, dimension.width, dimension.height);
9.  }
```

イメージを作成する drawSamples()メソッド

```
1.      public void drawSamples() {
2.          initializeImage();
3.          Graphics2D g = (Graphics2D) image.getGraphics();
4.          //四角形
5.          g.setColor(Color.GREEN);
6.          g.fill(new Rectangle2D.Double(100., 100., 400., 400.));
7.          //楕円
8.          g.setColor(Color.RED);
9.          g.draw(new Ellipse2D.Double(500., 500., 200., 100.));
10.         //直線
11.         g.setColor(new Color(30, 20, 100));
12.         g.draw(new Line2D.Double(0., 0., 800., 700.));
13.     }
```

JPanelにイメージを表示するpaintComponent()メソッド

- ここで図を描いていては、時間がかかる
- あらかじめ描いてあるimageを表示するだけ

```
1.  public void paintComponent(java.awt.Graphics g) {
2.      if (image == null) {return;}
3.      //ここではimageを貼り付けるだけ
4.      g.drawImage(image,
5.          0, 0, image.getWidth(), image.getHeight(), this);
6.  }
```

イメージをファイルに保存する

- javax.imageio.ImageIOクラス
  - イメージファイルの操作のメソッドの集合
  - ファイルからの読み込み
    - BufferdImage read(File file)
  - ファイルへの書き出し
    - write(RenderedImage im, String formatName, File output)
    - RenderedImage：BufferedImageが実装しているインターフェース

```
1.      public void saveImage(File file) {
2.          if (!fileChooser.FileUtilGUI.checkWritable(file)) {
3.              return;
4.          }
5.          try ( FileOutputStream out = new FileOutputStream(file)) {
6.              String ext = FileIO.getExtention(file.getName());
7.              javax.imageio.ImageIO.write(image, ext, out);
8.              String message
9.                      = "イメージを" + file.getName() + "に保存しました。";
10.             fileChooser.FileUtilGUI.showMessage(message);
11.         } catch (IOException ex) {
12.             fileChooser.FileUtilGUI.showError(ex.getMessage());
13.         }
14.     }
```

前回のfileChooser.FileUtilの応用

簡単なドローツール

- マウスで線を描く
  - java.awt.event.MouseListener
  - java.awt.event.MouseMotionListener
- 線の太さ
  - java.awt.BasicStroke
- 消す
  - 背景色で太い線を引く


File Edit
ERASER
Line WIdth

マウスイベントを拾う

- インターフェイスの実装
  - java.awt.event.MouseListener
  - java.awt.event.MouseMotionListener
  - 対応するメソッド
- リスナの設定

```
addMouseListener(this);
addMouseMotionListener(this);
```

マウスでの描画の動作

- マウスボタンを押す
  - mousePressed(MouseEvent e)
- マウスをドラッグする
  - mouseDragged(MouseEvent e)
- マウスボタンを離す
  - mouseReleased(MouseEvent e)

マウス描画の基本的考え方

- マウスボタンを押す
  - マウス位置をpointへ保存
    - pointはjava.awt.Pointのインスタンス
- マウスをドラッグする
  - 直前の位置と現在の位置を結ぶ
  - 現在の位置をpointへ保存
- マウスボタンを離す
  - pointをクリアする

直前の点と現在の点を結ぶ

```
1.      private void lineSegment(int x, int y) {
2.          Graphics2D g = (Graphics2D) image.getGraphics();
3.          if (eraser) {//消しゴムの場合
4.              g.setColor(this.getBackground());
5.              g.setStroke(eraserStroke);
6.          } else {
7.              g.setColor(this.getForeground());
8.              g.setStroke(stroke);
9.          }
10.         //前の点から現在の点へ線を引く
11.         g.drawLine(point.x, point.y, x, y);
12.     }
```

マウスを押す

```
public void mousePressed(MouseEvent e) {
  //新たな位置の生成
  point = new Point(e.getPoint());
}
```

マウスドラッグ

```
public void mouseDragged(MouseEvent e) {
    if (point != null) {
        lineSegment(e.getX(), e.getY());
        point = new Point(e.getPoint());//現在の点を更新
    }
    repaint();
}
```

マウスボタンを離す

```
public void mouseReleased(MouseEvent e) {
    if (point != null) {
        lineSegment(e.getX(), e.getY());
        point = null;//現在の点を消去
    }
    repaint();
}
```

線幅の設定

- 線の設定クラス
  - java.awt.BasicStroke
  - 線幅、終端処理などを設定する
- Graphics2D.setStroke()メソッド

```
public void setLineWidth(int w) {
  if (w < 1) {
     w = 1;
  }
  stroke = new BasicStroke((float) w, BasicStroke.CAP_ROUND,
        BasicStroke.JOIN_ROUND);
}
```

色の設定

- javax.swing.JColorChooserを使う

```
private void selectColorActionPerformed(
                    java.awt.event.ActionEvent evt) {
    java.awt.Color newColor =
        JColorChooser.showDialog(
            this, "Choose Foreground Color", getBackground());
    drawPanel.setForeground(newColor);
}
```