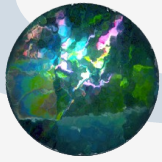
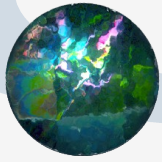


Graphical User Interface 描画する



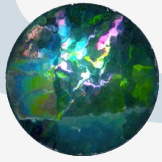
描画の基本

- javax.swing.JPanelに描画する
- paint()メソッドによって描画する
 - この中で描画対象を描く
- しかし
 - 画面の重なりによる再描画の場合
 - 最初から書き直すと時間がかかる

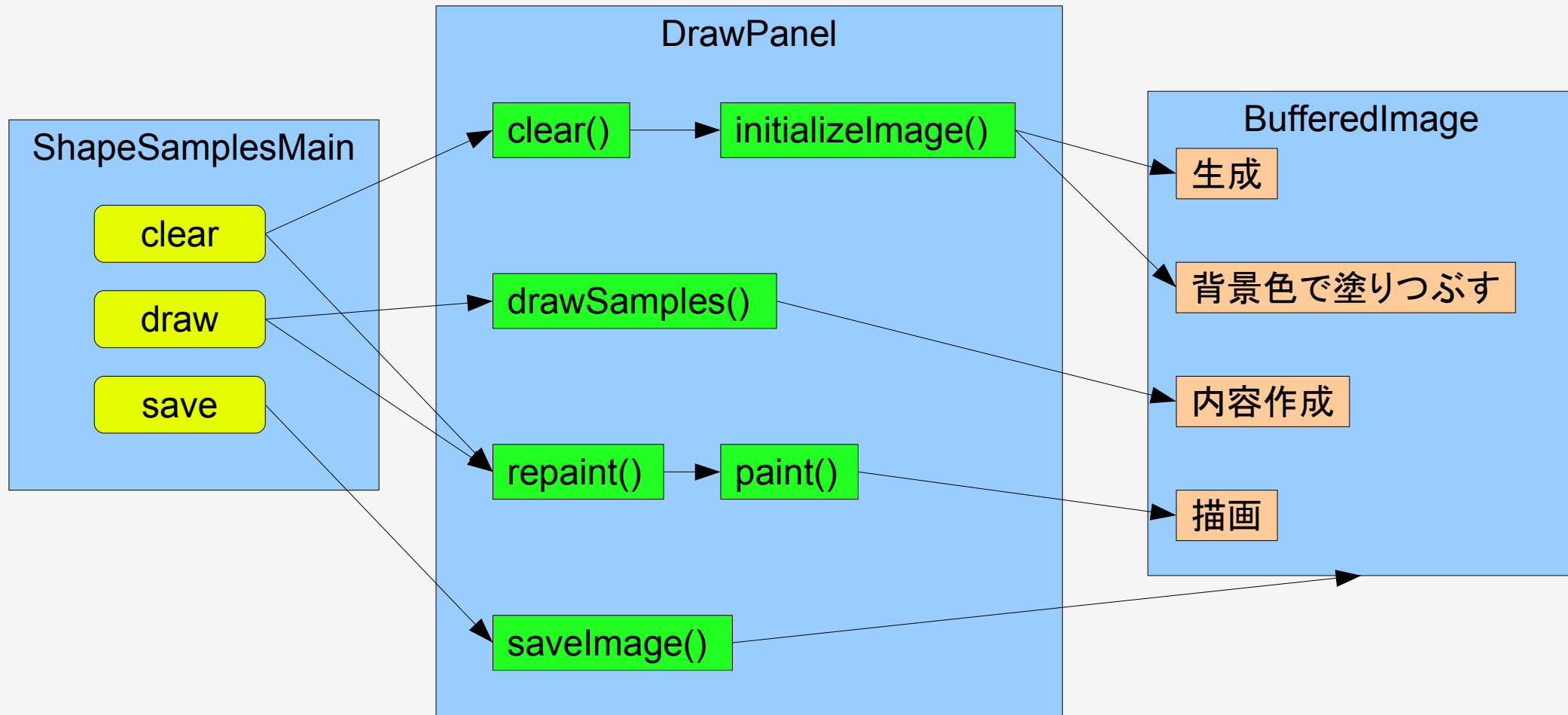


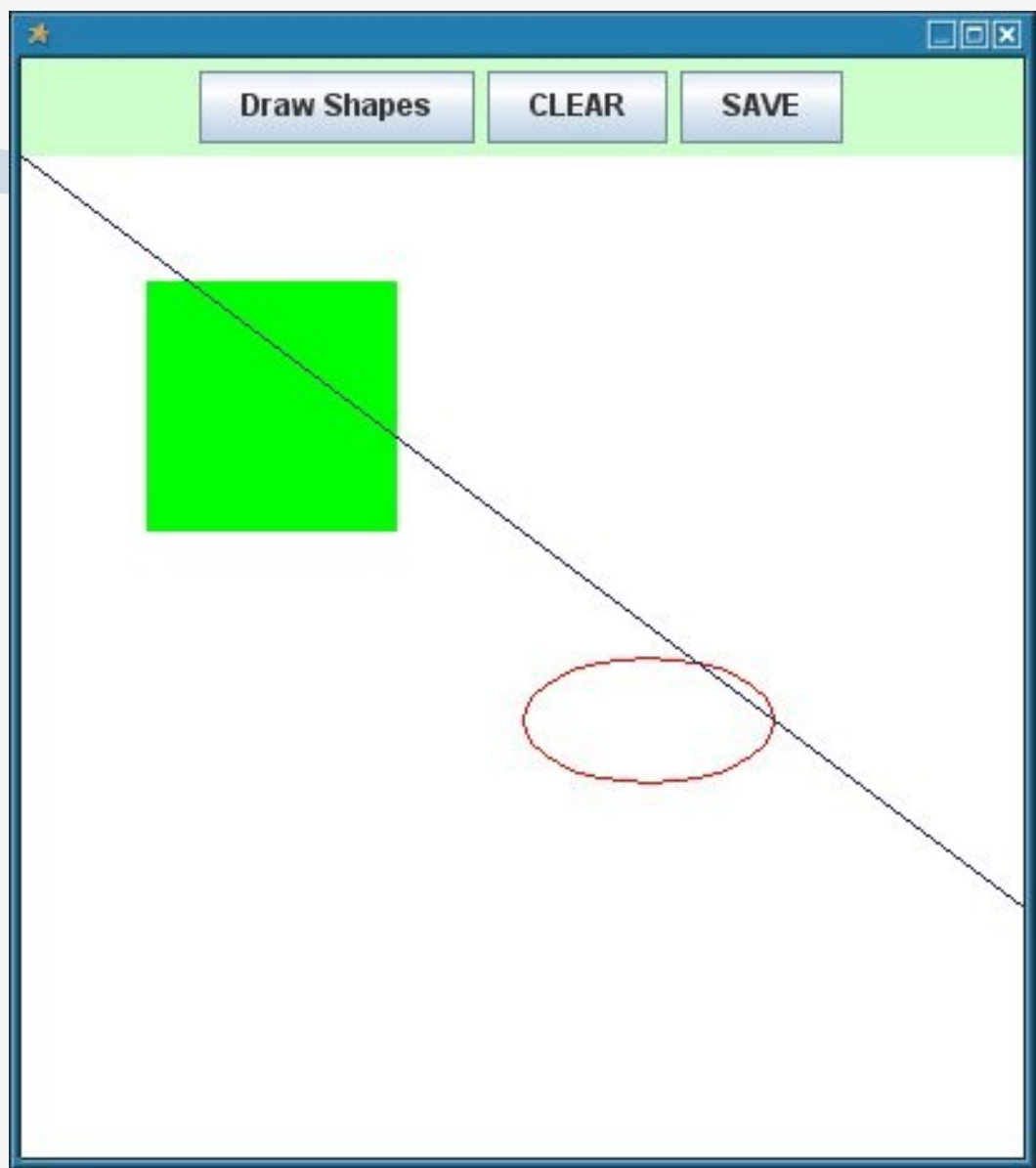
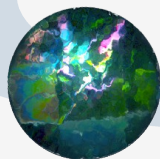
描画の基本2

- ウィンドウの重なりで再描画が必要な場合
 - `repaint()`を呼ぶ→`paint()`が呼ばれる
- 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
 - `paint()`内では`image`の描画のみ
 - `image`をファイルに保存できる



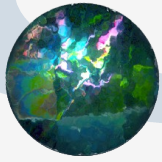
全体構成





ボタン用JPanel

描画用JPanel



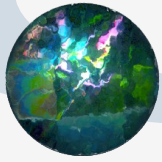
イメージ初期化

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

```
public void initializeImage() {  
    Dimension dimension = getPreferredSize();  
    image = new BufferedImage(dimension.width, dimension.height,  
        BufferedImage.TYPE_INT_RGB);  
    Graphics2D g = (Graphics2D) image.getGraphics();  
    g.setColor(this.getBackground());  
    g.fillRect(0, 0, dimension.width, dimension.height);  
}
```

JPanelのサイズで
イメージ生成

背景色で塗りつぶす



イメージを作成する

```
public void drawSamples() {  
    initializeImage();  
    Graphics2D g = (Graphics2D) image.getGraphics();
```

```
    Rectangle2D.Double rect = new Rectangle2D.Double(50., 50., 100., 100.);
```

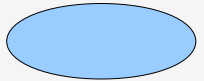


```
    g.setColor(Color.GREEN);
```

```
    g.fill(rect);
```

```
    Ellipse2D.Double ellipse = new Ellipse2D.Double(200., 200., 100., 50.);
```

```
    g.setColor(Color.RED);
```



```
    g.draw(ellipse);
```

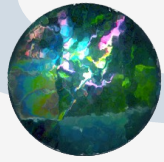
```
    g.setColor(new Color(30, 20, 100));
```

```
    Line2D.Double line = new Line2D.Double(0., 0., 400., 300.);
```

```
    g.draw(line);
```

```
}
```

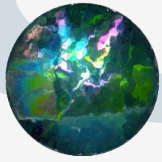




JPanelにイメージを表示する

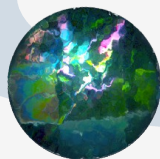
- JPanelを描画するのは、`paint()`メソッド
- ここで、図を描いていては、時間がかかる
- あらかじめ描いてある`image`を表示する

```
public void paint(java.awt.Graphics g) {  
    if (image == null) {  
        return;  
    }  
    g.drawImage(image, 0, 0, image.getWidth(), image.getHeight(), this);  
}
```

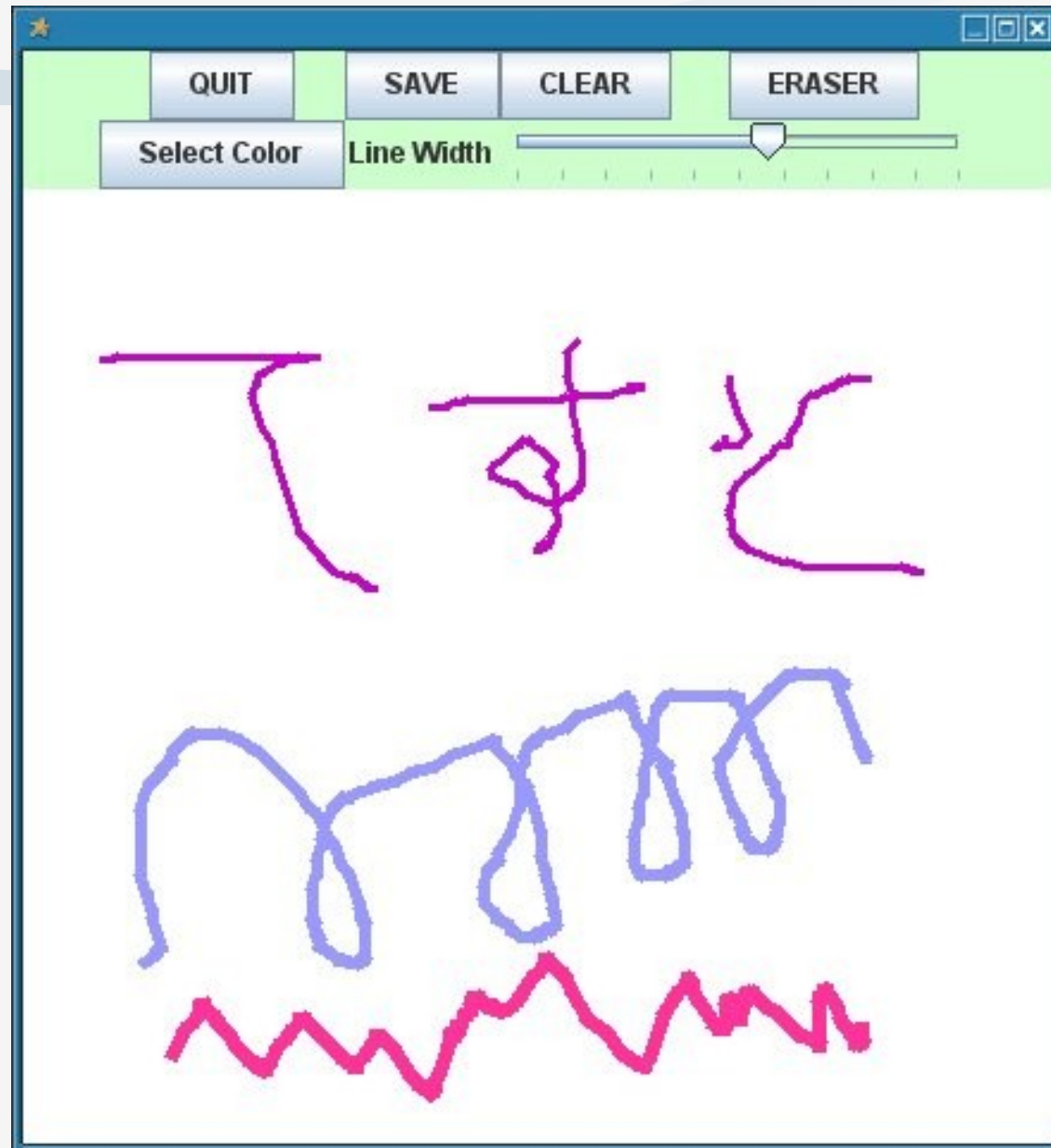
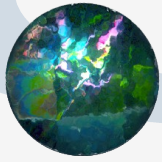
イメージをファイルに保存する

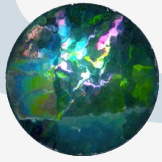
```
public void saveImage(File file) {
    if (!fileChooser.FileUtil.checkWritable(file)) {
        return;
    }
    FileOutputStream out = null;
    try {
        out = new FileOutputStream(file);
    } catch (FileNotFoundException ex) {
        fileChooser.FileUtil.showError(ex.getMessage());
    }
    if (out != null) {
        String ext = fileChooser.FileUtil.getExtention(file.getName());
        try {
            javax.imageio.ImageIO.write(image, ext, out);
            String message = "イメージを" + file.getName() + "に保存しました。";
            fileChooser.FileUtil.showMessage(message);
        } catch (IOException ex) {
            fileChooser.FileUtil.showError(ex.getMessage());
        }
    }
}
```



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

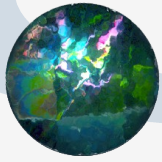




マウスイベントを拾う

- 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

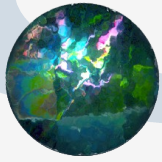
```
addMouseListener(this);  
addMouseMotionListener(this);
```



マウスでの描画の動作

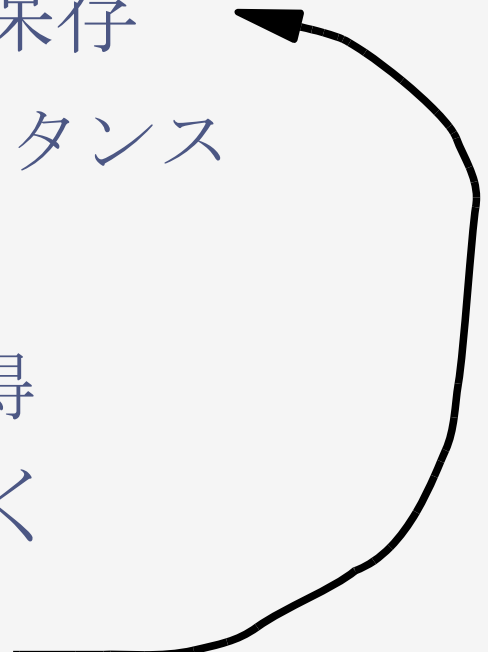
```
public void mousePressed(MouseEvent e) {  
    point = new Point(e.getPoint());  
}
```

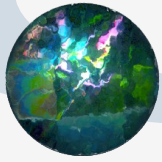
- マウスボタンを押す
 - mousePressed(MouseEvent e)
- マウスをドラッグする
 - mouseDragged(MouseEvent e)
- マウスボタンを離す
 - mouseReleased(MouseEvent e)



マウス描画の基本的考え方

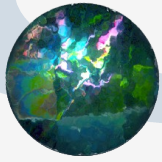
- 直前のマウスの位置を変数pointに保存
 - 変数pointはjava.awt.Pointのインスタンス
- マウสดラッグのイベントevtから
 - evt.getX()、evt.getY()で座標を取得
- Pointとマウス位置を結ぶ直線を引く
- マウス位置をpointに保存する





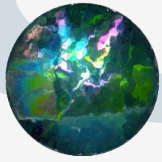
マウスドラッグ

```
public void mouseDragged(MouseEvent e) {  
    if (point != null) {  
        Graphics2D g = (Graphics2D) image.getGraphics();  
        if (eraser) { // 消しゴムの場合  
            g.setColor(this.getBackground());  
            g.setStroke(eraserStroke);  
        } else {  
            g.setColor(this.getForeground());  
            g.setStroke(stroke);  
        }  
        g.drawLine(point.x, point.y, e.getX(), e.getY());  
        point = new Point(e.getX(), e.getY());  
    }  
    repaint();  
}
```



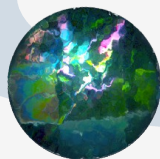
マウスボタンを離す

```
public void mouseReleased(MouseEvent e) {  
    if (point != null) {  
        Graphics2D g = (Graphics2D) image.getGraphics();  
        if (eraser) { // 消しゴムの場合  
            g.setColor(this.getBackground());  
            g.setStroke(eraserStroke);  
        } else {  
            g.setColor(this.getForeground());  
            g.setStroke(stroke);  
        }  
        g.drawLine(point.x, point.y, e.getX(), e.getY());  
        point = null;  
    }  
    repaint();  
}
```

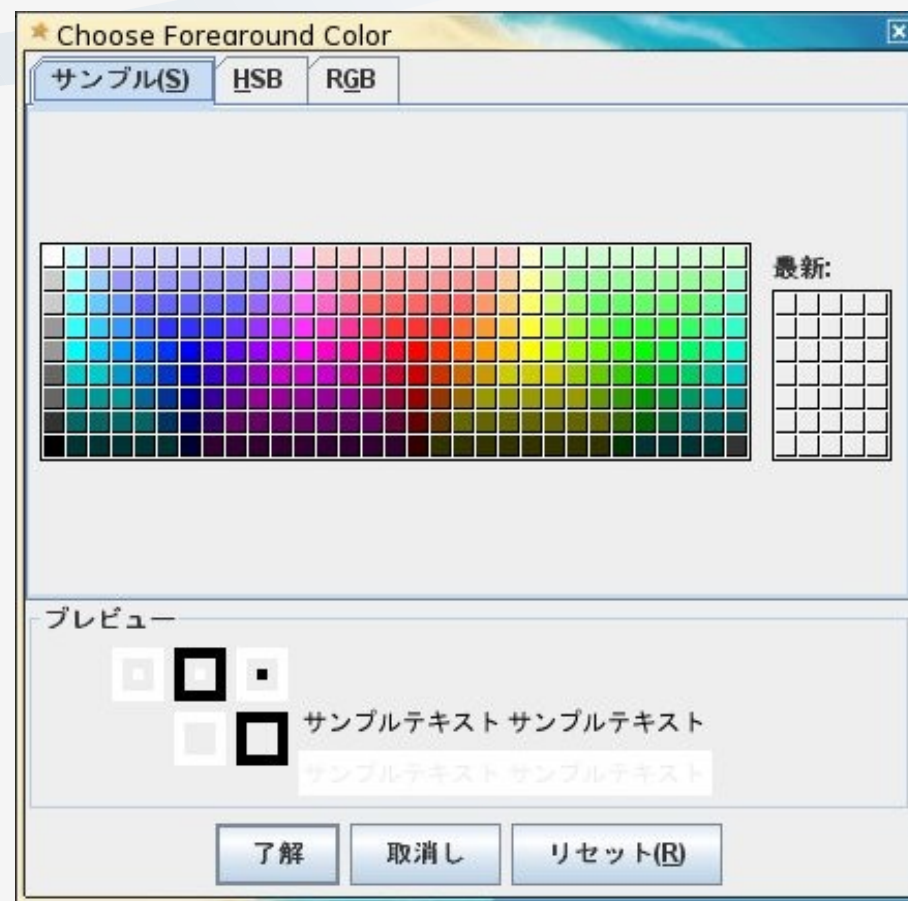
線幅の設定

- 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



色の設定

- javax.swing.JColorChooserを使う



```
java.awt.Color newColor = JColorChooser.showDialog(  
    this, "Choose Foreground Color", getBackground());  
drawPanel.setForeground(newColor);
```