2016/03/11

NetBeansではじめるJava

第二回　画像処理ソフトウェアの開発

ArkOak 代表　加納 徹

Java講習会の流れ

5. 画像の入出力

6. マウスによる画像情報の取得

7. 画像の上からお絵描き

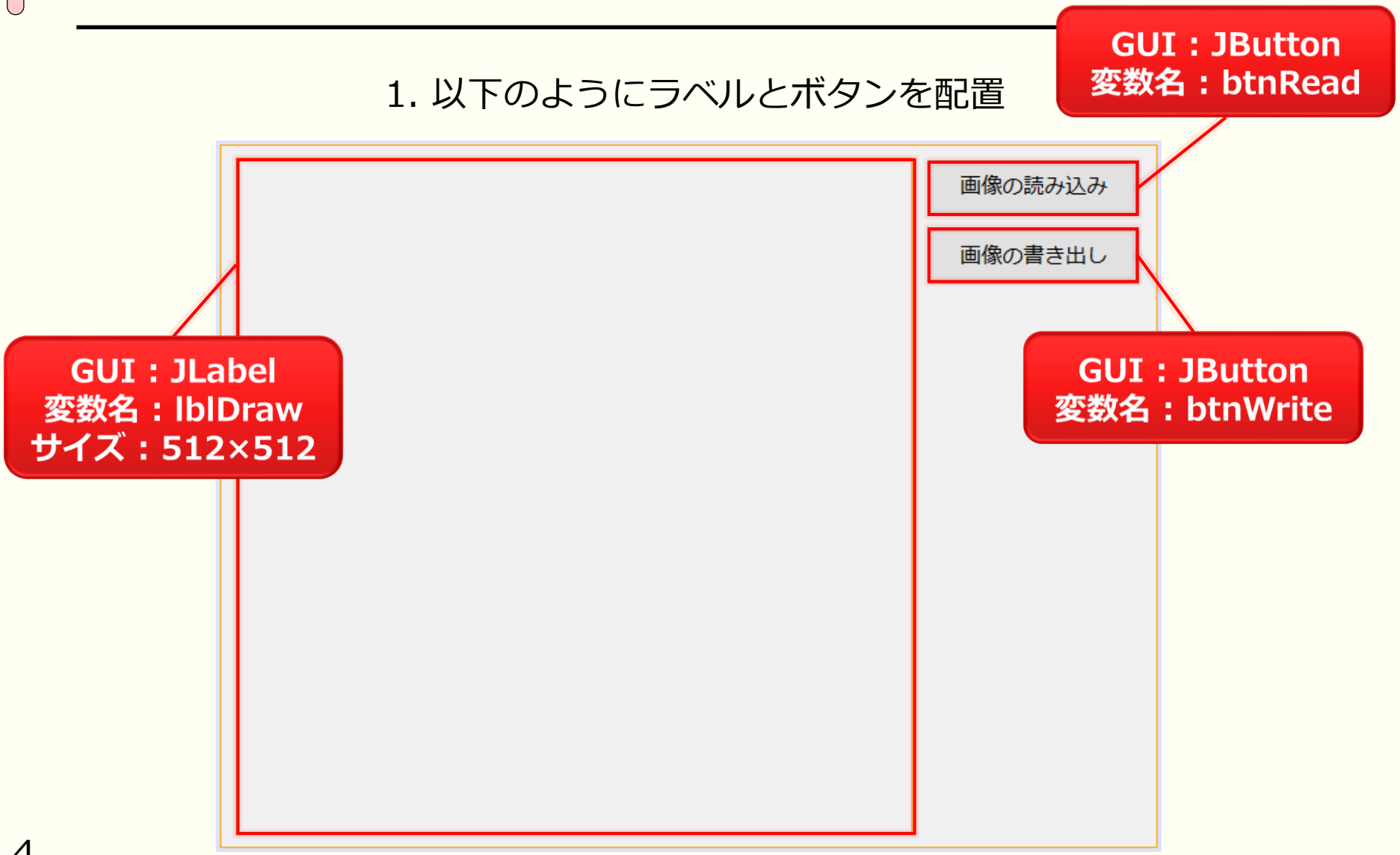
8. 画像処理ソフトウェアの開発

5. 画像の入出力

新規プロジェクト
「**ImageProcessing**」
を作ろう

画像の入出力

1. 以下のようにラベルとボタンを配置

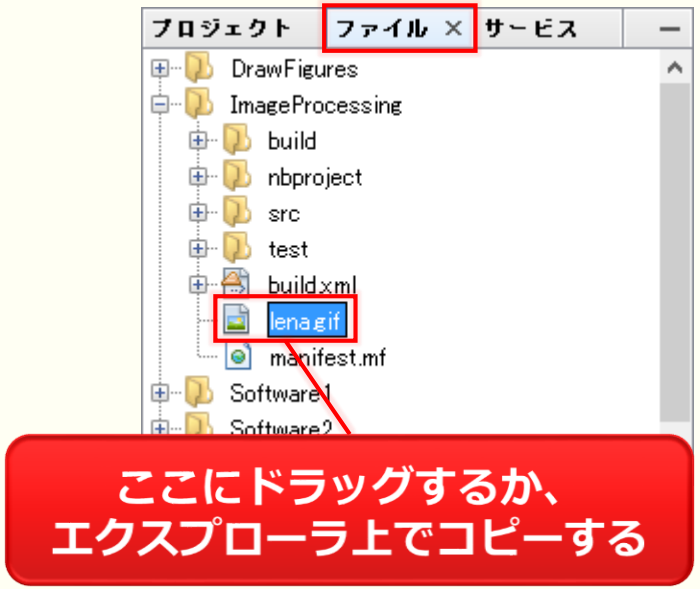


画像ファイルの準備

2. 画像ファイルをダウンロード

http://arkoak.com/material/lena.gif

3. 「ファイル」タブから「ImageProcessing」の中に格納



画像ファイルの読み込み

4.「画像の読み込み」ボタンをダブルクリックして以下を処理を記述

```
BufferedImage bufImage;   // イメージパネル

private void btnReadActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // 画像ファイルの読み込み
        bufImage = ImageIO.read(new File("lena.gif"));
        // 画像の表示
        lblDraw.setIcon(new ImageIcon(bufImage));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

画像情報を記憶する領域

画像ファイルのオープン

例外処理

画像ファイルの読み込み

5. 実行結果

画像ファイルの書き出し

6.「画像の書き込み」ボタンをダブルクリックして以下を処理を記述

```
private void btnWriteImgActionPerformed(java.awt.event.ActionEvent evt) {
        // 画像パネルが空の場合、終了
        if (bufImage == null)  return;
        try {
                // 画像ファイルの書き出し
                ImageIO.write(bufImage, "bmp", new File("out.bmp"));
                JOptionPane.showMessageDialog(this, "保存しました");
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

画像の形式を指定して出力

メッセージダイアログの表示

bmp/jpg/png/gif などの画像形式に対応しているよ

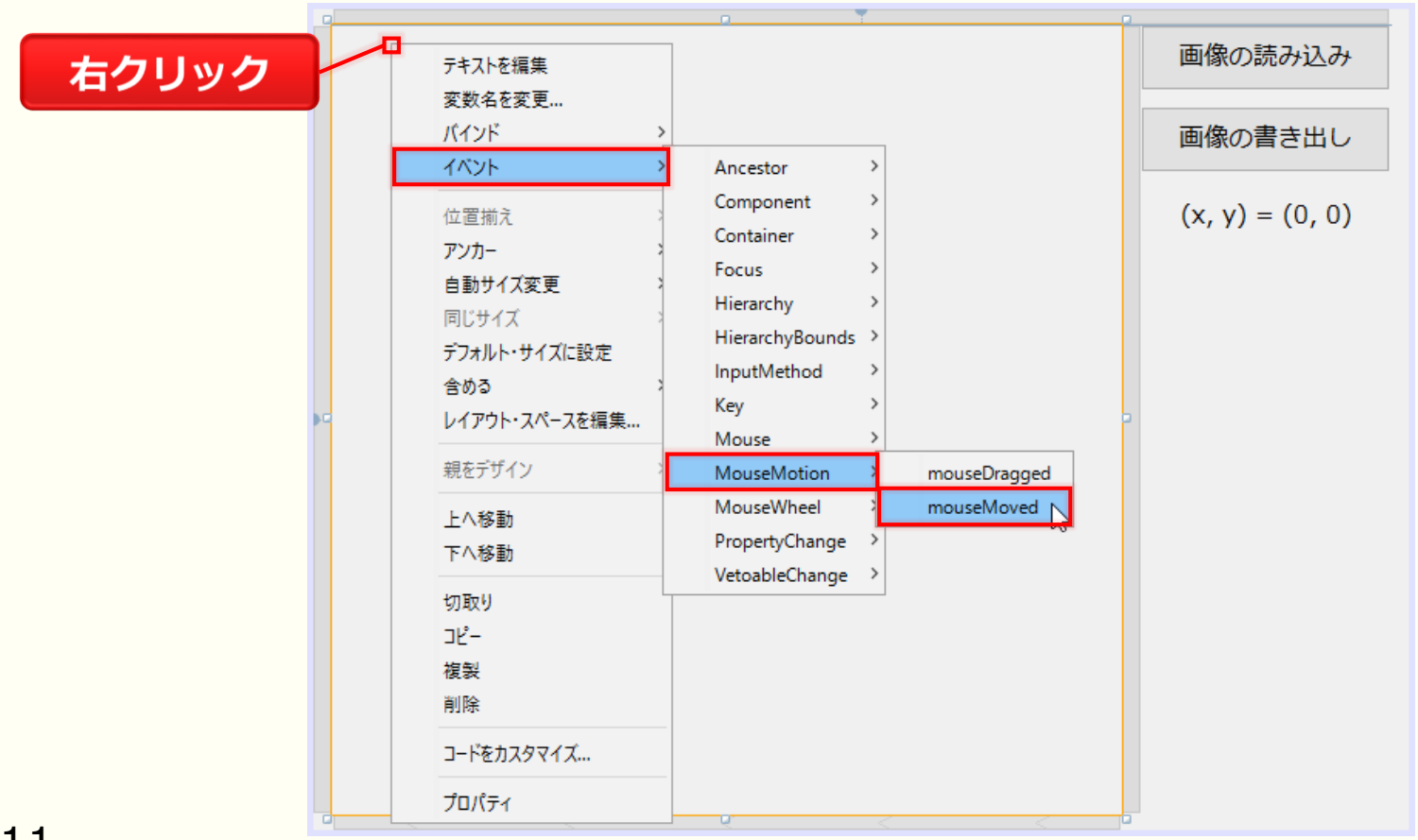
6. マウスによる画像情報の取得

マウス座標の取得

1. 以下のようにラベルを配置

マウス座標の取得

2. ラベル (lblDraw) 上で右クリックし、
[イベント] ⇨ [MouseMotion] ⇨ [mouseMoved] と進む



マウス座標の取得

3. 以下のソースコードを記述

```
private void lblDrawMouseMoved(java.awt.event.MouseEvent evt) {
    // マウスの座標取得
    int mouseX = evt.getX();
    int mouseY = evt.getY();
    // マウス座標の表示
    lblMousePos.setText(" (x, y) = (" + mouseX + ", " + mouseY + ")");
}
```

マウスの座標を整数値で取得

ラベル上でマウスに動きがあるたびに呼び出されるメソッド
マウスの動きは MouseMotionListener によって監視されている

マウス座標の取得

4. 実行結果

画像情報の取得

5. 以下のようにラベルとテキストフィールドを配置

画像情報の取得

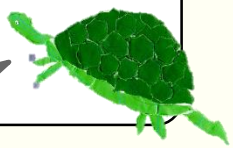
6. 以下のソースコードを **lblDrawMouseMoved** に追記

```
private void lblDrawMouseMoved(java.awt.event.MouseEvent evt) {
        // マウスの座標取得
        int mouseX = evt.getX();
        int mouseY = evt.getY();
        // マウス座標の表示
        lblMousePos.setText(" (x, y) = (" + mouseX + ", " + mouseY + ")");
        // 画像パネルが空の場合、終了
        if (bufImage == null)  return;
        // 色情報の取得
        Color c = new Color(bufImage.getRGB(mouseX, mouseY));
        // RGB情報の表示
        txtRed.setText("" + c.getRed());
        txtGreen.setText("" + c.getGreen());
        txtBlue.setText("" + c.getBlue());
}
```

座標 (mosueX, mouseY) における色情報を取得し、Color 変数に格納

Color 変数から色成分の抽出

RGB（Red, Green, Blue）

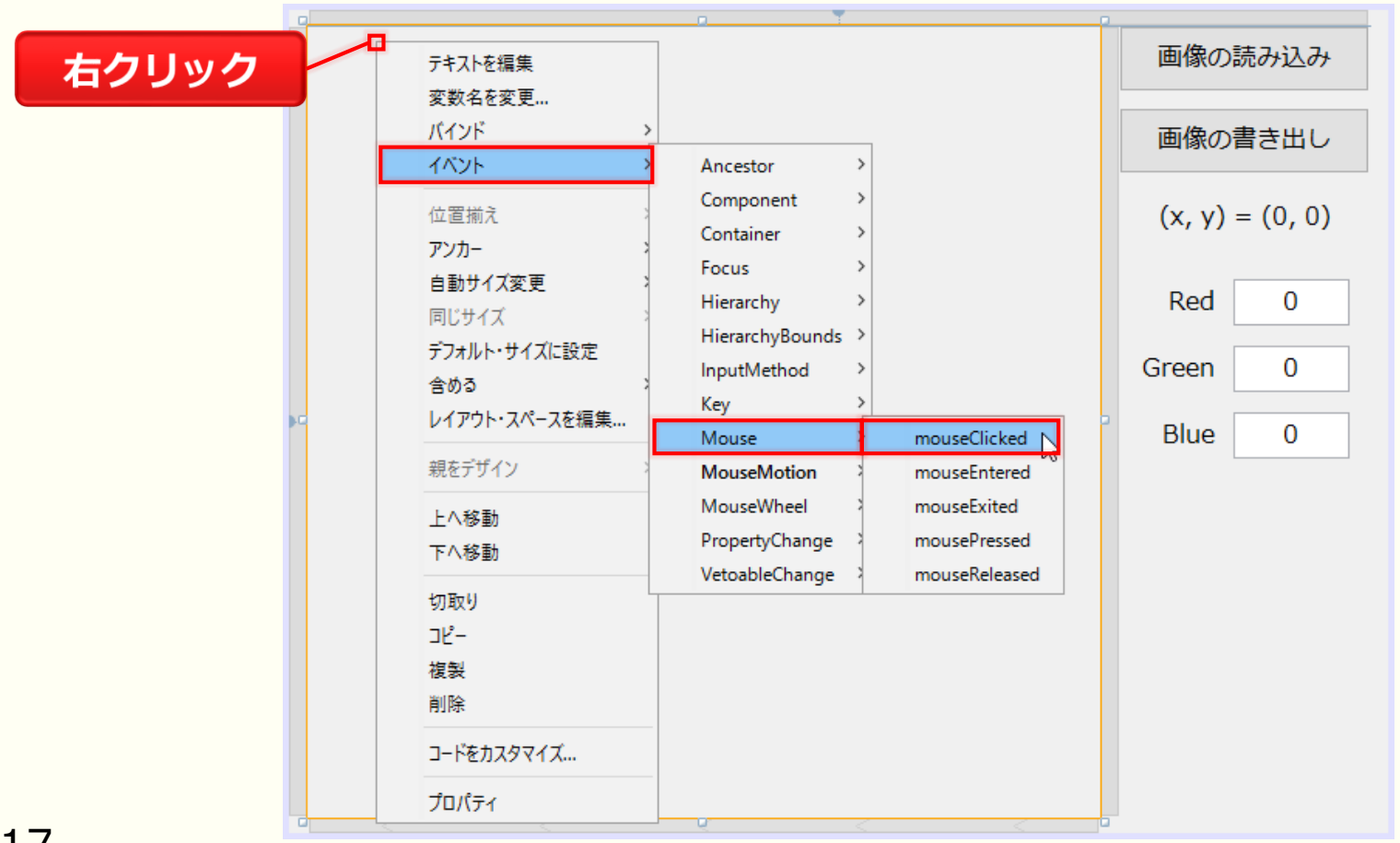


画像情報の取得

7. 実行結果

クリックした位置の情報を固定

8. ラベル (lblDraw) 上で右クリックし、
[イベント] ⇨ [Mouse] ⇨ [mouseClicked] と進む



クリックした位置の情報を固定

9. 以下のソースコードを記述

```
boolean clicked = false;   //クリックされた状態か否かを保存するboolean型変数
private void lblDrawMouseClicked(java.awt.event.MouseEvent evt) {
        // 画像パネルが空の場合、終了
        if (bufImage == null)  return;
        // クリックする度に状態を切り替える
        if (clicked)  {
                clicked = false;
        } else {
                clicked = true;
        }
}
```

ラベル上でクリックする度に、
clicked の true/false を切り替える

boolean型変数は、そのまま if文の評価式に使うことができる

クリックした位置の情報を固定

10. 以下のソースコードを **lblDrawMouseMoved** に追記

```
private void lblDrawMouseMoved(java.awt.event.MouseEvent evt) {
        // クリックされた状態の場合、終了
        if (clicked)  return;
        // マウスの座標取得
        int mouseX = evt.getX();
        int mouseY = evt.getY();
        // マウス座標の表示
        lblMousePos.setText(" (x, y) = (" + mouseX + ", " + mouseY + ")");
        // 画像パネルが空の場合、終了
        if (bufImage == null)  return;
        // 色情報の取得
        Color c = new Color(bufImage.getRGB(mouseX, mouseY));
        // RGB情報の表示
        lblRed.setText("   R = " + c.getRed());
        lblGreen.setText("   G = " + c.getGreen());
        lblBlue.setText("   B = " + c.getBlue());
}
```

clicked が true のとき、処理をここで終了させる

7. 画像の上からお絵描き

クリックした位置にお絵描き

1. 以下のソースコードを **lblDrawMouseClicked** に追記

```
boolean clicked = false;   //クリックされた状態か否かを保存するboolean型変数
private void lblDrawMouseClicked(java.awt.event.MouseEvent evt) {
        // 画像パネルが空の場合、終了
        if (bufImage == null)  return;
        // クリックする度に状態を切り替える
        clicked = !clicked;
        // クリックされた位置に色を塗る
        if (clicked) {
                Graphics g = bufImage.createGraphics();
                g.setColor(Color.red);
                g.fillOval(evt.getX() - 5, evt.getY() - 5, 10, 10);
                lblDraw.setIcon(new ImageIcon(bufImage));
        }
}
```

if文を使わない シンプルな形に書き換え可能

クリックされた状態のときのみ実行

赤色で直径10ピクセルの円を描画

Graphicsクラスを使って画像の上からお絵描きができる

クリックした位置にお絵描き

2. 実行結果

色が赤くない・・？　それに点が消えてない・・・

クリックした位置にお絵描き

3. 以下のように **btnReadActionPerformed** の処理を変更

```
BufferedImage bufImage, bufImageShow;   // イメージパネル
private void btnReadActionPerformed(java
        try {
                // 画像ファイルの読み込み
                bufImage = ImageIO.read(new File("lena.gif"));
                bufImageShow = new BufferedImage(512, 512, BufferedImage.TYPE_INT_RGB);
                Graphics g = bufImageShow.createGraphics();
                g.drawImage(bufImage, 0, 0, this);
                // 画像の表示
                lblDraw.setIcon(new ImageIcon(bufImageShow));
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

表示用のイメージパネル

手動でパネルを初期化

表示用パネルに画像データを描画

表示用パネルにをラベルに貼り付け

画像記憶のパネルと表示用のパネルを別々に用意しよう

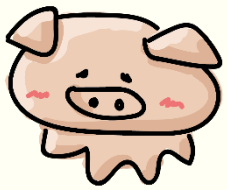
クリックした位置にお絵描き

4. 以下のように **btnWriteActionPerformed** の処理を変更

```
private void btnWriteActionPerformed(java.awt.event.ActionEvent evt) {
        // 画像パネルが空の場合、終了
        if (bufImage == null)  return;
        try {
                // 画像ファイルの書き出し
                ImageIO.write(bufImageShow, "bmp", new File("out.bmp"));
                JOptionPane.showMessageDialog(this, "保存しました");
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

表示用パネルを出力する

表示されている結果を出力するためには、ここも変更



クリックした位置にお絵描き

5. 以下のように **lblDrawMouseClicked** の処理を変更

```
boolean clicked = false;   //クリックされた状態か否かを保存するboolean型変数
private void lblDrawMouseClicked(java.awt.event.MouseEvent evt) {
        // 画像パネルが空の場合、終了
        if (bufImage == null)  return;
        // クリックする度に状態を切り替える
        clicked = !clicked;
        // クリックされた位置に色を塗る
        if (clicked) {
                Graphics g = bufImageShow.createGraphics();
                g.setColor(Color.red);
                g.fillOval(evt.getX() - 5, evt.getY() - 5, 10, 10);
                lblDraw.setIcon(new ImageIcon(bufImageShow));
        } else {
                Graphics g = bufImageShow.createGraphics();
                g.drawImage(bufImage, 0, 0, this);
                lblDraw.setIcon(new ImageIcon(bufImageShow));
        }
}
```

表示用パネル上に円を描画

表示用パネルの上から画像を再描画

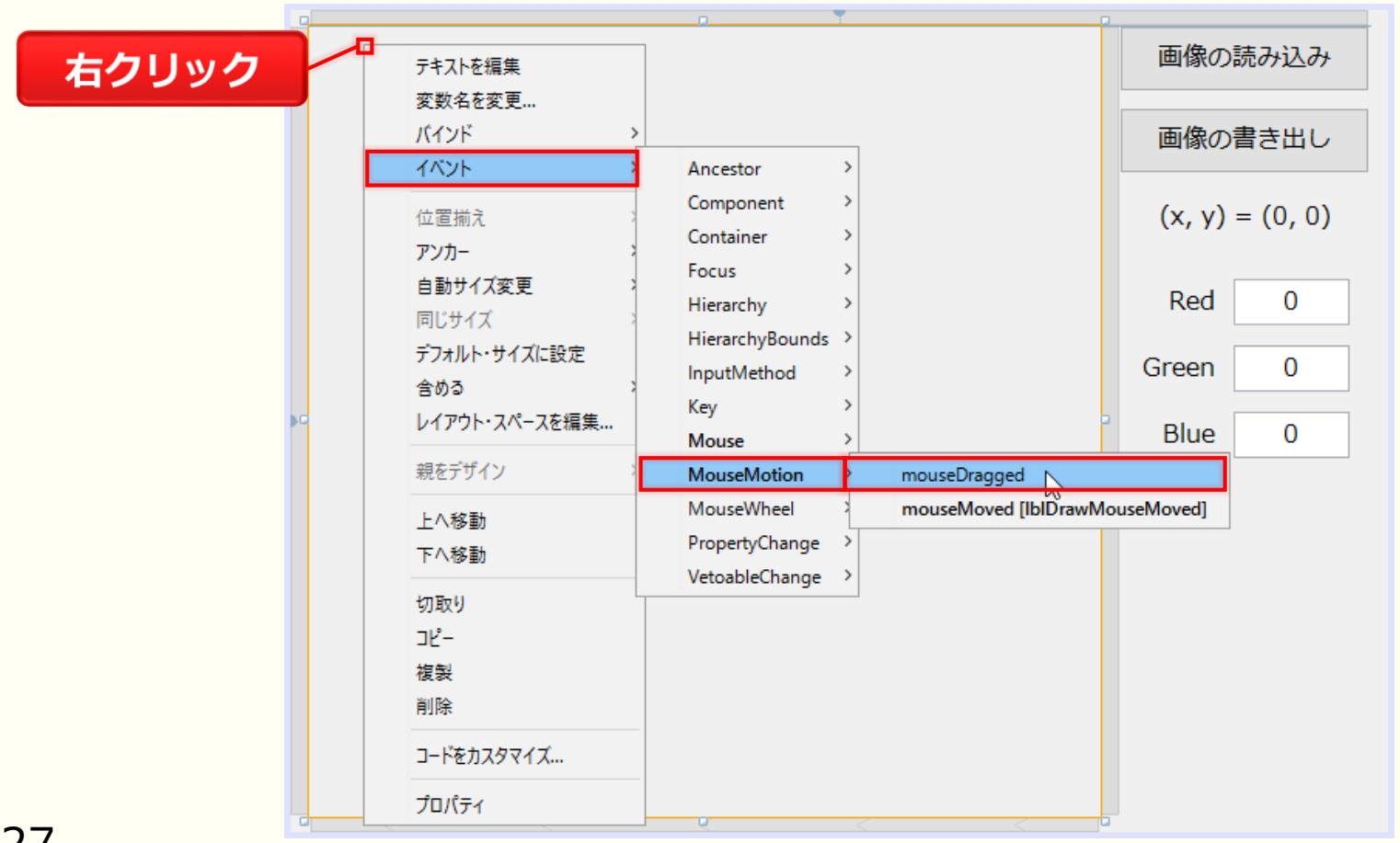
クリックした位置にお絵描き

6. 実行結果

赤くなった！さらに点も一つだけになった

ドラッグした位置にお絵描き

5. ラベル (lblDraw) 上で右クリックし、
[イベント] ⇨ [MouseMotion] ⇨ [mouseDragged] と進む



ドラッグした位置にお絵描き

6. 以下のソースコードを記述

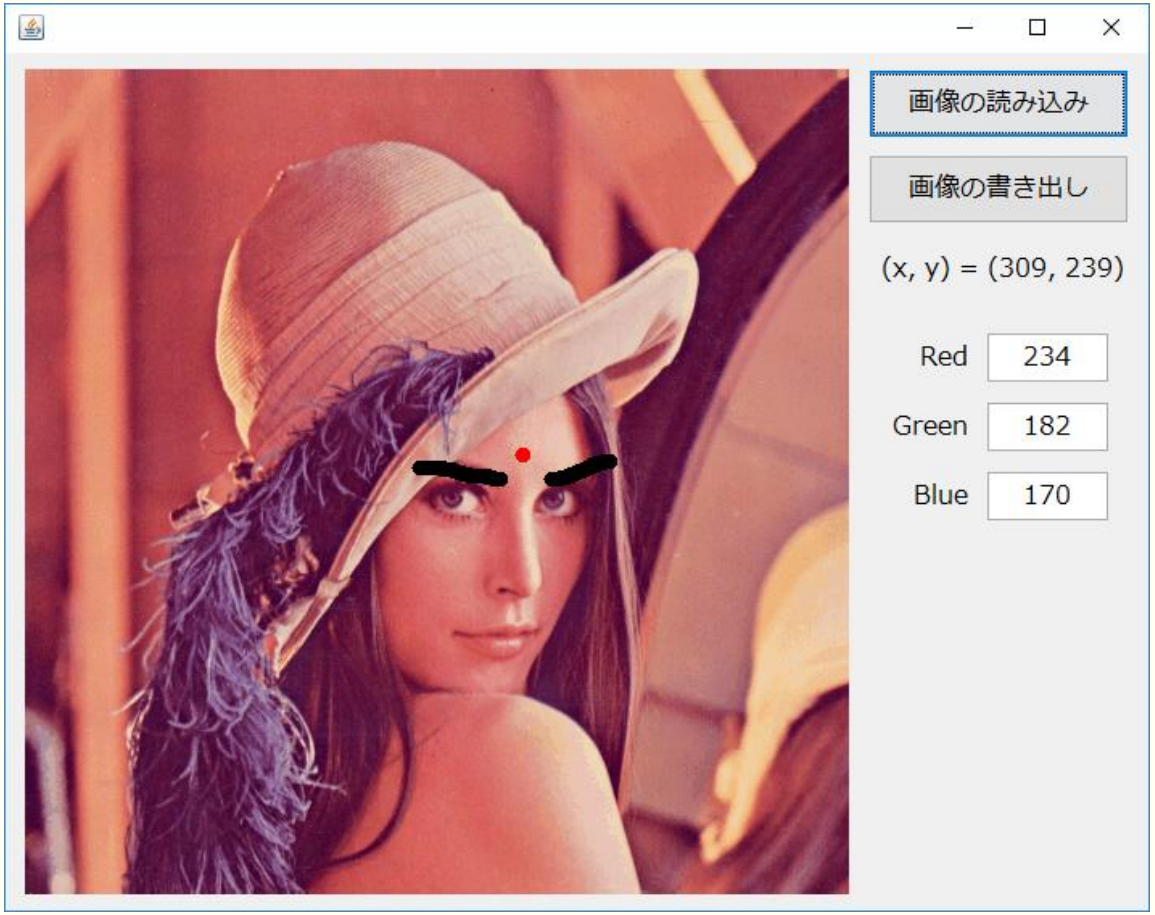
```
private void lblDrawMouseDragged(java.awt.event.MouseEvent evt) {
    // 画像パネルが空の場合、終了
    if (bufImg == null)  return;
    // ドラッグしている間、円を描き続ける
    Graphics g = bufImageShow.createGraphics();
    g.setColor(Color.black);
    g.fillOval(evt.getX() - 5, evt.getY() - 5, 10, 10);
    lblDraw.setIcon(new ImageIcon(bufImageShow));
}
```

黒い円をカーソルの位置に描く

ドラッグしている間、カーソルの位置に円が描かれる！

ドラッグした位置にお絵描き

7. 実行結果



8. 画像処理ソフトウェアの開発

8.1 グレースケール化

グレースケール化

1. 以下のようにボタンを配置

グレースケール化

5. btnGrayScale をダブルクリックし、以下のソースコードを記述

```
private void btnGrayScaleActionPerformed(java.awt.event.MouseEvent evt) {
        if (bufImage == null) return;    // 画像パネルが空の場合、終了
        Color c;
        int red, green, blue, gray;
        // 全てのピクセルに対して処理
        for (int y = 0; y < bufImage.getHeight(); y++) {
                for (int x = 0; x < bufImage.getWidth(); x++) {
                        c = new Color(bufImage.getRGB(x, y));
                        red = c.getRed();
                        green = c.getGreen();
                        blue = c.getBlue();
                        gray = (red + green + blue) / 3;   // 単純平均法
                        bufImageShow.setRGB(x, y, new Color(gray, gray, gray).getRGB());
                }
        }
        lblDraw.setIcon(new ImageIcon(bufImageShow));
}
```

画像の全ピクセルに対する処理

RGB情報の抽出

RGB値を平均化し、色を置き換える

グレースケール化

3. 実行結果

グレースケール化

- **単純平均化**
  - RGB値を単純に平均化するグレースケール化手法

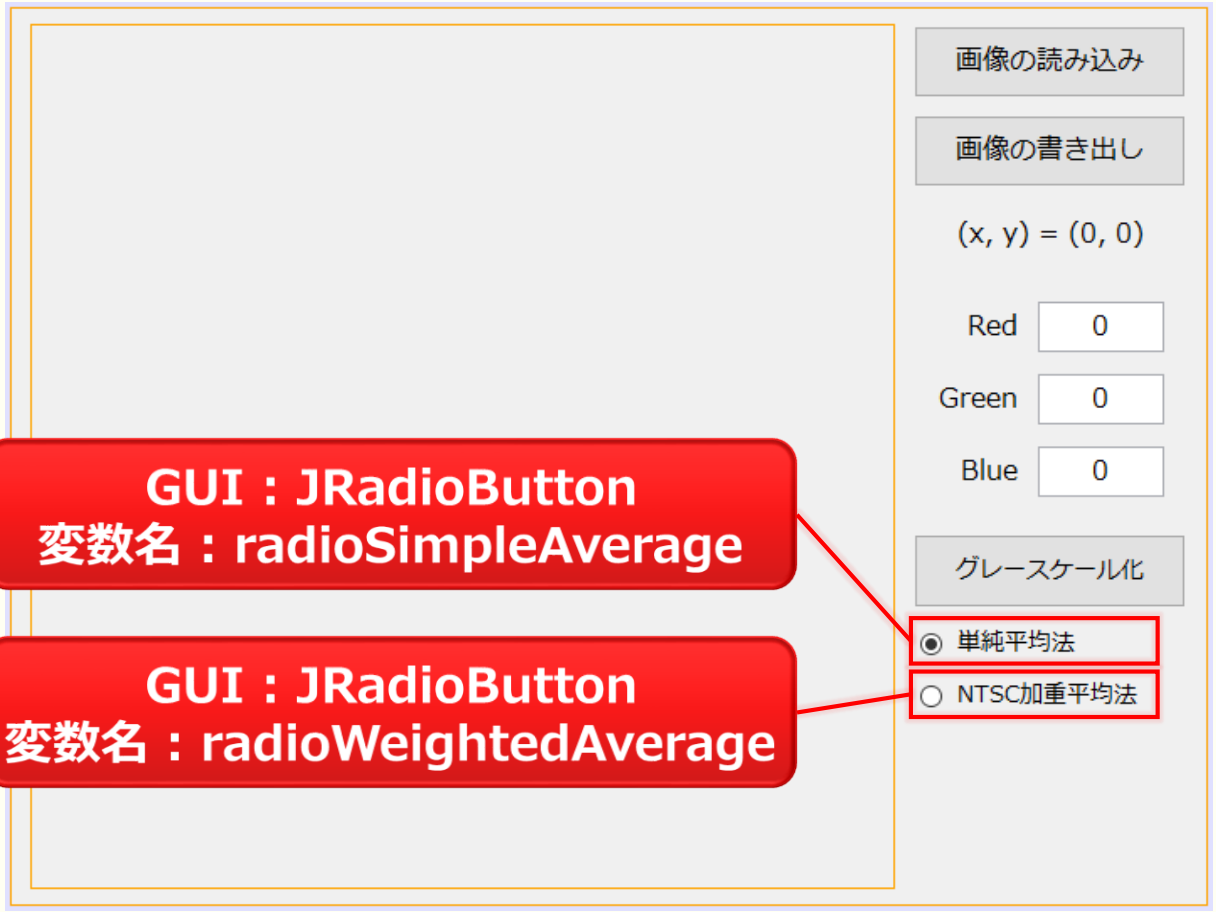
$$\mathrm{Gray} = \frac{\mathrm{Red} + \mathrm{Green} + \mathrm{Blue}}{3}$$

- **NTSC加重平均化**
  - NTSC (National Television System Committee) が規格したグレースケール化手法
  - 人間の視覚が緑に敏感で青に鈍感であることを考慮

$$\mathrm{Gray} = 0.298912 \times \mathrm{Red} + 0.586611 \times \mathrm{Green} + 0.114478 \times \mathrm{Blue}$$

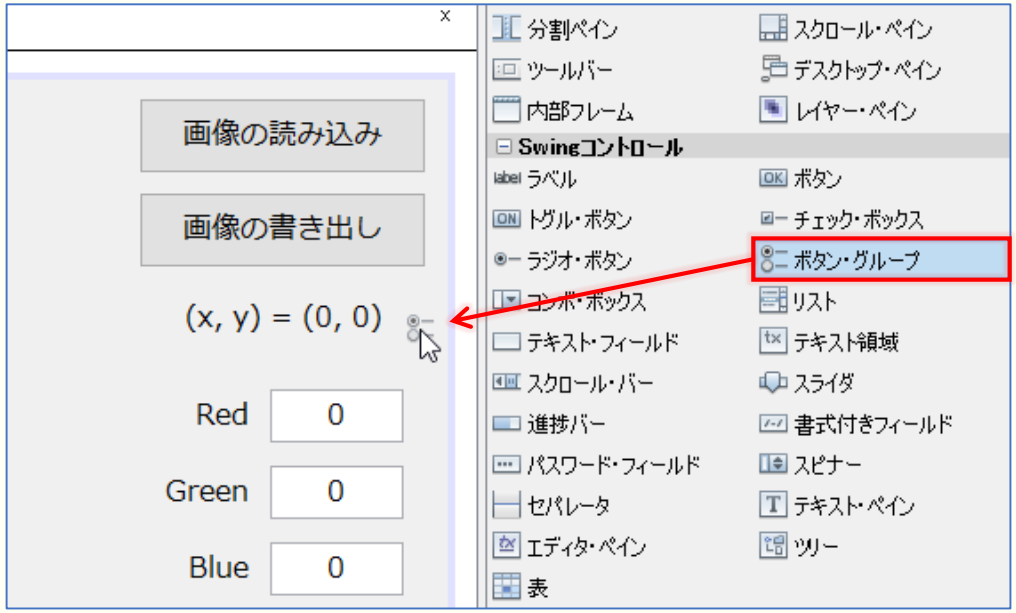
ラジオボタンとボタングループ

1. 以下のようにラジオボタンを配置

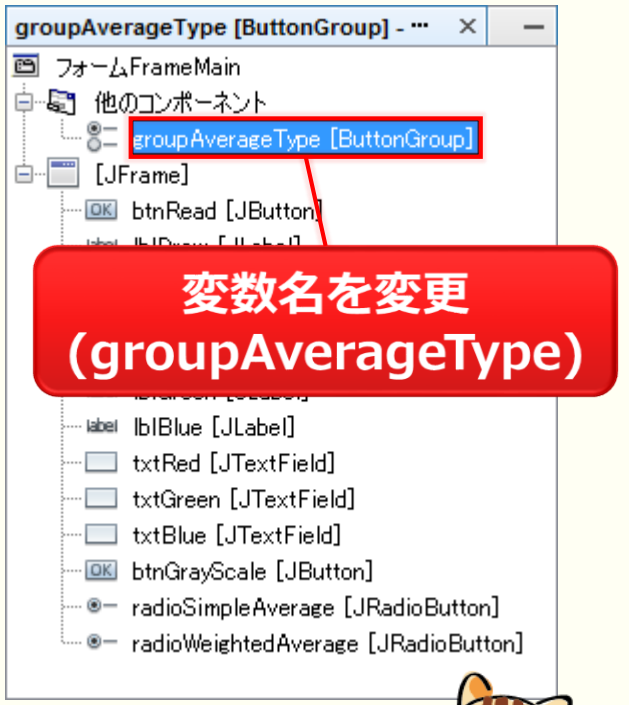


ラジオボタンとボタングループ

2. ボタングループをGUIデザイナ上に
ドラッグ&ドロップ



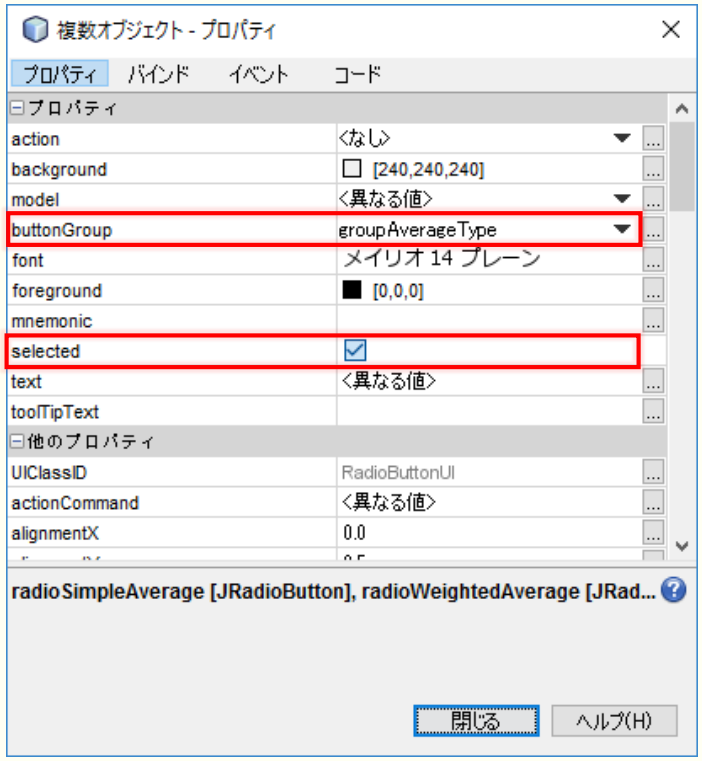
3. 画面左下のナビゲータから
ボタングループの変数名を変更



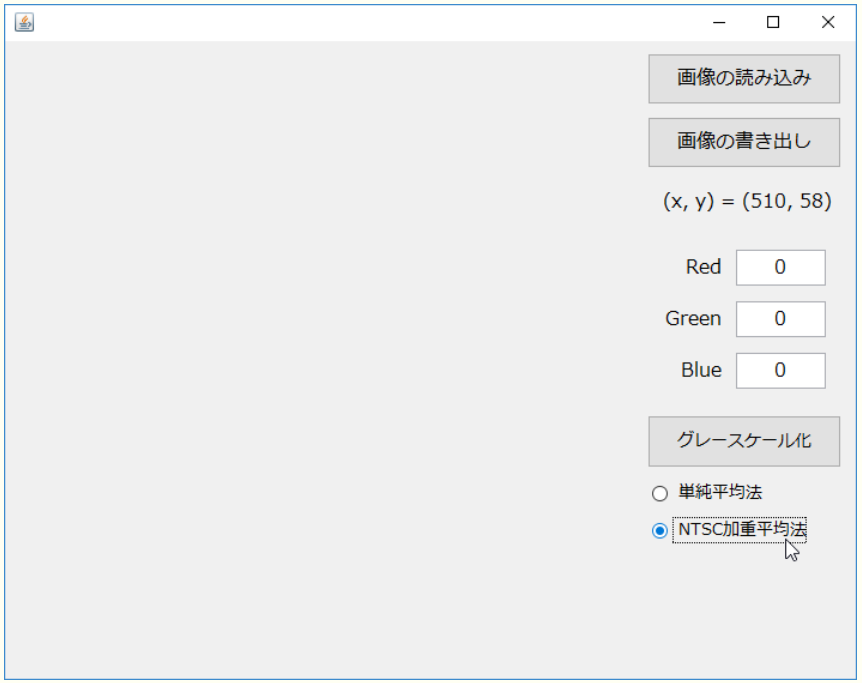
ボタングループは目に見えない Swingコントロール

ラジオボタンとボタングループ

4. ラジオボタンの buttonGroup を groupAverageType に設定



5. 同じボタングループに登録されたボタンは、同時に選択状態にならない



一つはあらかじめ [selected] 状態にしておこう

グレースケール化

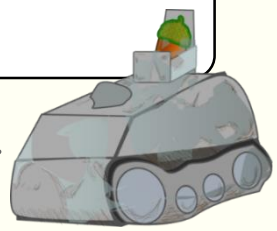
6. 以下のように **btnGrayScaleActionPerformed** の処理を変更

```
// 全てのピクセルに対して処理
for (int y = 0; y < bufImage.getHeight(); y++) {
        for (int x = 0; x < bufImage.getWidth(); x++) {
                c = new Color(bufImage.getRGB(x, y));
                red = c.getRed();
                green = c.getGreen();
                blue = c.getBlue();
                if (radioSimpleAverage.isSelected())
                        gray = (red + green + blue) / 3;
                else
                        gray = (int) (0.298912 * red + 0.586611 * green + 0.114478 * blue);
                bufImageShow.setRGB(x, y, new Color(gray, gray, gray).getRGB());
        }
}
lblDraw.setIcon(new ImageIcon(bufImageShow));
```

単純平均化

NTSC加重平均化

実数の前に (int) をつけると、**キャスト**（型変換）できる



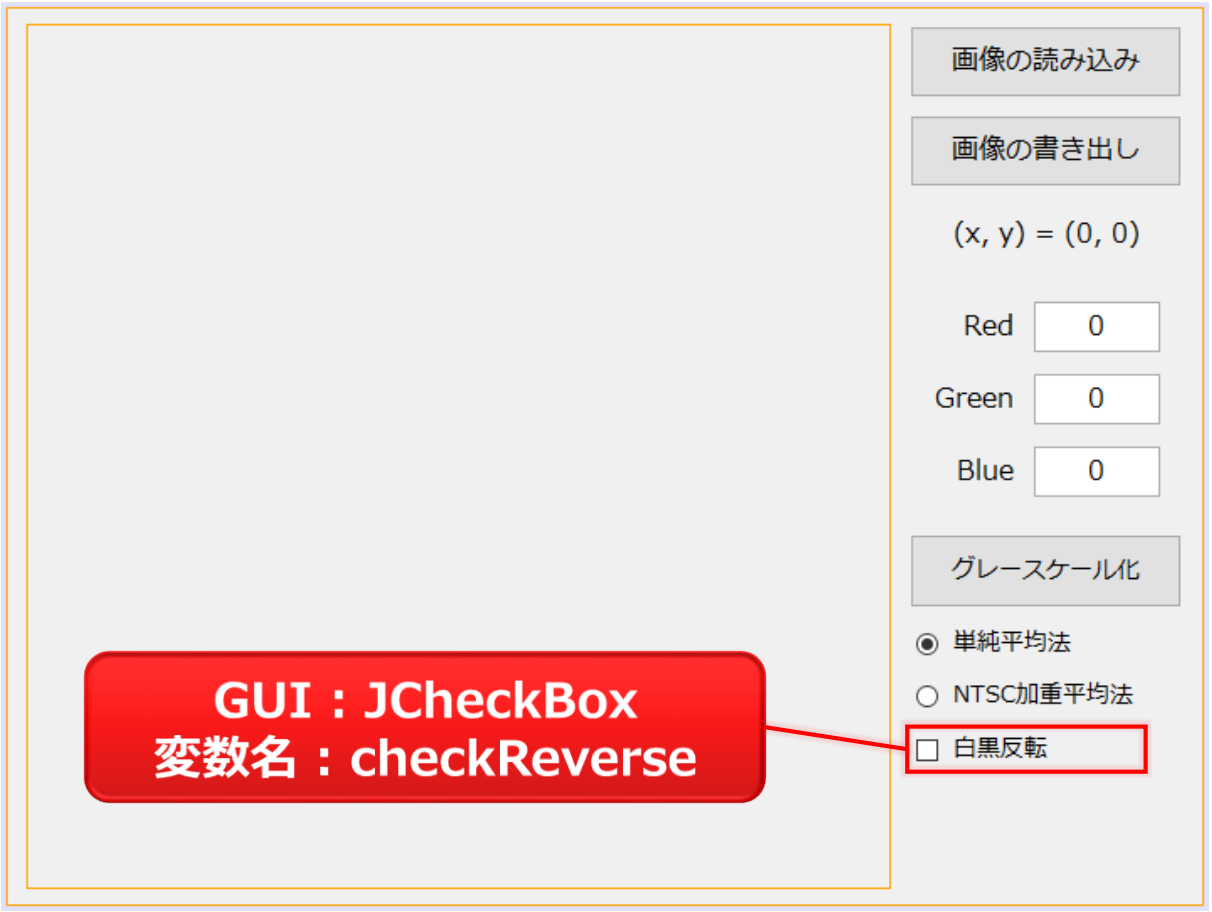
グレースケール化

7. 実行結果

8.2 色の反転

色の反転

1. 以下のようにチェックボックスを配置



色の反転

2. 以下のように **btnGrayScaleActionPerformed** の処理を変更

```
// 全てのピクセルに対して処理
for (int y = 0; y < bufImage.getHeight(); y++) {
        for (int x = 0; x < bufImage.getWidth(); x++) {
                c = new Color(bufImage.getRGB(x, y));
                red = c.getRed();
                green = c.getGreen();
                blue = c.getBlue();
                if (radioSimpleAverage.isSelected())
                        gray = (red + green + blue) / 3;
                else
                        gray = (int) (0.298912 * red + 0.586611 * green + 0.114478 * blue);
                if (checkReverse.isSelected())  gray = 255 - gray;
                bufImageShow.setRGB(x, y, new Color(gray, gray, gray).getRGB());
        }
}
lblDraw.setIcon(new ImageIcon(bufImageShow));
```

グレーの値を反転

グレーの値の範囲は 0～255

色の反転

3. 実行結果


画像の読み込み
画像の書き出し
(x, y) = (473, 0)
Red
198
Green
97
Blue
101
グレースケール化
単純平均法
NTSC加重平均法
白黒反転

8.3　二値化処理

二値化処理

1. 以下のようにチェックボックスを配置

二値化処理

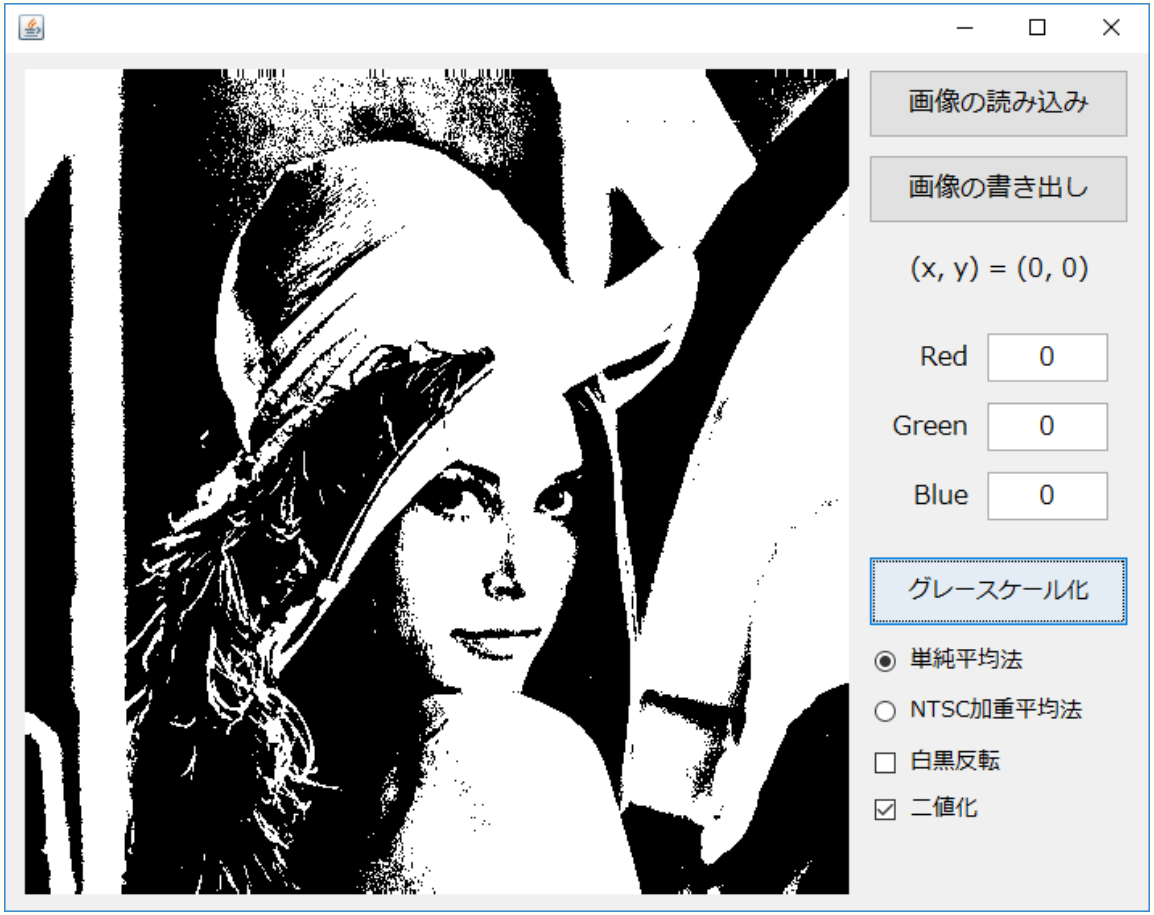
2. 以下のように **btnGrayScaleActionPerformed** の処理を変更

```
for (int y = 0; y < bufImage.getHeight(); y++) {
        for (int x = 0; x < bufImage.getWidth(); x++) {
                c = new Color(bufImage.getRGB(x, y));
                red = c.getRed();
                green = c.getGreen();
                blue = c.getBlue();
                if (radioSimpleAverage.isSelected()) gray = (red + green + blue) / 3;
                else    gray = (int) (0.298912 * red + 0.586611 * green + 0.114478 * blue);
                if (checkReverse.isSelected())  gray = 255 - gray;
                if (checkBinalize.isSelected()) {
                        if (gray < 128)     gray = 0;
                        else                gray = 255;
                }
                bufImageShow.setRGB(x, y, new Color(gray, gray, gray).getRGB());
        }
}
lblDraw.setIcon(new ImageIcon(bufImageShow));
```

黒 (0) か白 (255) に分類

二値化処理

3. 実行結果


画像の読み込み
画像の書き出し
(x, y) = (0, 0)
Red
0
Green
0
Blue
0
グレースケール化
単純平均法
NTSC加重平均法
白黒反転
二値化

次回予告

9. X線CTの原理

10. 画像再構成アルゴリズムの実装

11. X線CTのシミュレーション

ちょっとマニアックな内容

お疲れ様でした

アンケートにご協力下さい

http://goo.gl/forms/YYf7mtdDuQ