情報処理技法
(Javaプログラミング)2

第13回
操作に対して処理が行われるGUI(3),
絵を描いてみよう!

人間科学科コミュニケーション専攻
白銀 純子

第13回の内容

- コピー&ペースト
- プログラムでのお絵描きの基礎
  - 準備編
  - お絵描き編

前回の復習問題の解答

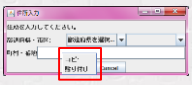
- GUIにおいて、ボタンが押されたときに何らかの処理がされるとき、人間がボタンを押すときから処理が開始されるまでの仕組みについて、簡単に説明しなさい。

解答例:
「リスナ」という機能が、人間がボタンを押すのを待っている。人間がボタンを押すと、リスナがそれを感知し、マウスでのボタンクリックなど、どのようなユーザイベントが発生したかを特定する。そして、そのユーザイベントに対応した処理を開始する。

ポップアップメニュー

ポップアップメニューの作成(1)

- ポップアップメニュー: その場その場で表示するメニュー
  - 多くの場合、マウスの右クリックで表示
  - Javaでの部品名: JPopupMenu



- ポップアップメニューの作成と表示
  - JPopupMenuのオブジェクトを作成
  - JPopupMenuのオブジェクトにJMenuItemのオブジェクトを登録
    - JMenuItemの扱い方は、メニューバーを作るときと全く同じ
  - JPopupMenuのオブジェクトを「show(...)」メソッドで画面上に表示
    - どの部品上のどの座標に表示するかを、showメソッドの引数(表示する部品, x座標, y座標の順で)として指定

ポップアップメニューの作成(2)

- マウス操作に応じて表示する場合
  - MouseListenerのメソッドのMouseEvent引数のメソッドで、必要な情報を取得
    - getComponent()メソッド: どの部品上でマウス操作が行われたかを取得
    - getX()メソッド: マウス操作が行われた場所のx座標
    - getY()メソッド: マウス操作が行われた場所のy座標

この3つの情報を、JPopupMenuのオブジェクトを表示するときに使用

ポップアップメニューの作成(3)

- マウスの右クリックでポップアップメニューを表示する例

```
import java.awt.event.*;
import javax.swing.*;

public class MouseSample extends JFrame implements ActionListener, MouseListener {
  JPopupMenu popup;
  JMenuItem copyItem, pasteItem;

  public MouseSample() {
    ……
  }
```

部品のオブジェクトの作成や表示はリスナのメソッド内で行うが、変数宣言はクラスのフィールドとして行う必要

ポップアップメニューの作成(4)

- マウスの右クリックでポップアップメニューを表示する例(続き)

```
public void mousePressed(MouseEvent e) {
  if (e.getButton() == MouseEvent.BUTTON3) {
    popup = new JPopupMenu();

    copyItem = new JMenuItem("コピー");
    copyItem.addActionListener(this);
    popup.add(copyItem);

    pasteItem = new JMenuItem("貼り付け");
    pasteItem.addActionListener(this);
    popup.add(pasteItem);

    popup.show(e.getComponent(), e.getX(), e.getY());
  }
}
```

マウスの右ボタンを押されたときの処理として定義

JPopupMenuの作成
- JMenuItemはオブジェクトを作成して、JPopupMenuのオブジェクトに登録

JPopupMenuオブジェクトの表示
- 「e.getComponent()」でマウス操作が行われた部品を取得
  - ポップアップメニューの表示先
- 「e.getX()」と「e.getX()」でマウス操作が行われたx座標・y座標を取得
  - ポップアップメニューを表示するx座標・y座標

ポップアップメニューの作成(5)

- マウスの右クリックでポップアップメニューを表示する例(続き)

```
public void actionPerformed(ActionEvent e) {
  if (e.getSource() == copyItem) {
    /* ポップアップメニューのcopyItemが押されたときの処理 */
  } else if (e.getSource() == pasteItem) {
    /* ポップアップメニューのpasteItemが押されたときの処理 */
  } else if (e.getSource() == okBut) {
    /* OKボタンが押されたときの処理 */
  }
 }
}
```

ポップアップメニューの項目が選択されたときの処理
- ウィンドウ上の他の部品(JButtonなど)とあわせて、if文で条件分岐

ポップアップメニューの作成(6)

- ちなみに…JTextFieldとJTextAreaで切り取り(カット)・コピー・貼り付け(ペースト)を行うには…
  - cut()メソッド: 選択された文字列を切り取り(カット)するメソッド
  - copy()メソッド: 選択された文字列をコピーするメソッド
  - paste()メソッド: 選択された文字列を貼り付け(ペースト)するメソッド

ポップアップメニューの作成(7)

- コピー&ペースト処理の記述例(JTextFieldがウィンドウ内に1つの場合)

```
import java.awt.event.*;
import javax.swing.*;

public class MouseSample extends JFrame implements ActionListener, MouseListener {
  JPopupMenu popup;
  JMenuItem copyItem, pasteItem;
  JTextField addressText;

  ……

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == copyItem) { /* ポップアップメニューのcopyItemが押されたときの処理 */
      addressText.copy();
    } else if (e.getSource() == pasteItem) { /* ポップアップメニューのpasteItemが押されたときの処理 */
      addressText.paste();
    }
  }
}
```


入力フィールドがウィンドウ内に複数ある場合は?(1)

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;

public class MouseSample extends JFrame
        implements ActionListener, MouseListener {
  JPopupMenu popup;
  JMenuItem copyItem, pasteItem;
  JTextField addressText;
  JTextComponent textComp;

  ……
  public void mousePressed(MouseEvent e) {
    if (e.getButton() == MouseEvent.BUTTON3) {
      ……
      popup.show(e.getComponent(), e.getX(), e.getY());
      textComp = (JTextComponent) e.getSource();
    }
  }
```

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == copyItem) {
  /* ポップアップメニューのcopyItemが押されたときの処理 */
      textComp.copy();
    } else if (e.getSource() == pasteItem) {
  /* ポップアップメニューのpasteItemが押されたときの処理 */
      textComp.paste();
    }
  }
}
```

入力フィールドがウィンドウ内に複数ある場合は?(1)

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;

public class MouseSample extends JFrame
        implements ActionListener, MouseListener {
  JPopupMenu popup;
  JMenuItem copyItem, pasteItem;
  JTextField addressText;
  JTextComponent textComp;

  .....
  public void mousePressed(MouseEvent e
    if (e.getButton() == MouseEvent.BUTTO
      .....
      popup.show(e.getComponent(), e.ge
      textComp = (JTextComponent) e.getSource();
    }
  }
```

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == copyItem) {
/* ポップアップメニューのcopyItemが押されたときの処理 */
      textComp.copy();
    } else if (e.getSource() == pasteItem) {
/* ポップアップメニューのpasteItemが押されたときの処理 */
      textComp.paste();
    }
  }
}
```

- JTextComponent: JTextFieldとJTextAreaの親クラス
  - javax.swing.textというパッケージに分類
  - どのJTextField/JTextAreaでポップアップメニューが表示されたかを記憶しておくための変数

入力フィールドがウィンドウ内に複数ある場合は?(2)

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;

public class MouseSample extends JFrame
        implements ActionListener,
  JPopupMenu popup;
  JMenuItem copyItem, pasteItem;
  JTextField addressText;
  JTextComponent textComp;

  .....
  public void mousePressed(MouseEvent
    if (e.getButton() == MouseEvent.BUTTON3) {
      .....
      popup.show(e.getComponent(), e.getX(), e.getY());
      textComp = (JTextComponent) e.getSource();
    }
  }
```

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == copyItem) {
/* ポップアップメニューのcopyItemが押されたときの処理 */
      textComp.copy();
    } else if (e.getSource() == pasteItem) {
```

入力フィールド上でマウスの右ボタンが押されたときの処理
- その1: ポップアップメニューを表示する
- その2: textComp変数に、どの入力フィールド上でマウスの右ボタンが押されたかを設定
  - ✓「e.getSource()」で、どの部品で操作が行われたかを取得
  - ✓ 操作された部品を「JTextComponent」でキャストしてtextComp変数に代入

入力フィールドがウィンドウ内に複数ある場合は?(3)

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;

public class MouseSample extends JFrame
        implements ActionListener, MouseListener {
  JPopupMenu popup;
  JMenuItem copyItem, pasteItem;
  JTextField addressText;
  JTextComponent textComp;
```

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == copyItem) {
/* ポップアップメニューのcopyItemが押されたときの処理 */
      textComp.copy();
    } else if (e.getSource() == pasteItem) {
/* ポップアップメニューのpasteItemが押されたときの処理 */
      textComp.paste();
    }
  }
}
```

textCompの部品に対し、コピーやペーストの処理を行う
- この処理が行われる前に、ポップアップメニューが表示されている
- ポップアップメニューが表示された部品がtextComp変数に設定されている

簡易メッセージ・入力欄の表示

簡易メッセージ・入力欄

- ちょっとしたメッセージや入力をしたい!
  - 確認メッセージ
  - 警告メッセージ
  - エラーメッセージ
  - 1つだけ入力
  - etc.

いちいちウィンドウを作るのは面倒!

JOptionPane

JOptionPane(概要)

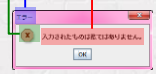
- 1行だけのメッセージや入力欄を簡易表示するためのウィンドウ
  - OK/CancelボタンやYes/Noボタンつき
  - 表示内容によって異なるメソッド利用
  - メソッドにメッセージ内容などを引数として設定
  - JOptionPaneが表示されているときは、他のウィンドウの操作不可

JOptionPane(showMessageDialog)

- 1行のメッセージを表示
  - OK/CancelやYes/Noなどの利用者の判断を求めないメッセージ(戻り値なし)
- 引数は4つ
  - 1つ目の引数: 常に「null」(ウィンドウの表示先を表すが、通常はnullにしておく)
  - 2つ目の引数: 表示するメッセージ(String型)
  - 3つ目の引数: ウィンドウのタイトル(String型)
  - 4つ目の引数: メッセージのタイプ(タイプによって表示されるアイコンが違う)
    - JOptionPane. ERROR_MESSAGE: エラーメッセージ
    - JOptionPane.INFORMATION_MESSAGE: 情報を提示
    - JOptionPane.WARNING_MESSAGE: ワーニングメッセージ
    - JOptionPane.QUESTION_MESSAGE: 質問メッセージ
    - JOptionPane.PLAIN_MESSAGE: アイコンなしでメッセージを表示

JOptionPane(showMessageDialog)(例)

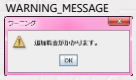
```
JOptionPane.showMessageDialog(null, "入力されたものは数ではありません。", "エラー", JOptionPane.ERROR_MESSAGE);
```



INFORMATION_MESSAGE

QUESTION_MESSAGE

WARNING_MESSAGE



PLAIN_MESSAGE

JOptionPane(showConfirmDialog)(1)

- 1行のメッセージを表示
  - OK/CancelやYes/Noなどの利用者の判断を求めるメッセージ
- 戻り値(int型)によって、どのボタンが押されたかを取得
  - JOptionPane.OK_OPTION: OKボタンが押されたときの戻り値
  - JOptionPane.CANCEL_OPTION: Cancelボタンが押されたときの戻り値
  - JOptionPane.YES_OPTION: Yesボタンが押されたときの戻り値
  - JOptionPane.NO_OPTION: Noボタンが押されたときの戻り値

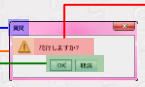
int型の変数を用意して戻り値を受け取り、if文で条件分岐して処理

JOptionPane(showConfirmDialog)(2)

- 引数は5つ
  - 1つ目の引数: 常に「null」(ウィンドウの表示先を表すが、通常はnullにしておく)
  - 2つ目の引数: 表示するメッセージ(String型)
  - 3つ目の引数: ウィンドウのタイトル(String型)
  - 4つ目の引数: OK/CancelやYes/Noのボタンのタイプ
    - JOptionPane.OK_CANCEL_OPTION: OK/Cancelボタンを表示
    - JOptionPane.YES_NO_OPTION: Yes/Noボタンを表示
    - JOptionPane.YES_NO_CANCEL_OPTION: Yes/No/Cancelボタンを表示
  - 5つ目の引数: メッセージのタイプ(タイプによって表示されるアイコンが違う)
    - 種類はshowMessageDialogメソッドと同じ

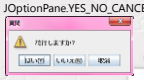
JOptionPane(showConfirmDialog)(例)

```
int code = JOptionPane.showConfirmDialog(null, "続行しますか?", "質問",
                JOptionPane.OK_CANCEL_OPTION, JOptionPane.WARNING_MESSAGE);
```



JOptionPane.YES_NO_OPTION

JOptionPane.YES_NO_CANCEL_OPTION

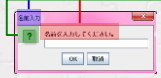


JOptionPane(showInputDialog)

- 1行の入力欄を表示
- 戻り値は入力された文字列(String型)
  - String型の変数を用意して戻り値を受け取り
- 引数は4つ
  - 1つ目の引数: 常に「null」(ウィンドウの表示先を表すが、通常はnullにしておく)
  - 2つ目の引数: 表示するメッセージ(String型)
  - 3つ目の引数: ウィンドウのタイトル(String型)
  - 4つ目の引数: メッセージのタイプ(タイプによって表示されるアイコンが違う)
    - 種類はshowMessageDialogメソッドと同じ

JOptionPane(showInputDialog)(例)

```
String name =
    JOptionPane.showInputDialog(null, "名前を入力してください。", "名前入力", JOptionPane.QUESTION_MESSAGE);
```



お絵描き～準備編～

お絵描きをするには?

- プログラムでの「紙」にあたるもの: JFrameまたはJPanel
- JFrameやJPanelの上に、座標を指定して、絵を描いていく
  - 楕円
  - 四角形
  - 線
  - etc.
- 絵を描くには、JPanelの方がオススメ

「紙」の準備(1)

- 「ただの」JPanelの上に絵を描くと、ウィンドウを操作すると、絵が消えてしまう
  - ウィンドウの最小/最大化、別のウィンドウを重ねる, etc.
- JPanelは画面に表示された瞬間に、JPanelの「paintComponent」メソッドで、絵が描かれる
  - 何も設定をしていなければ、JPanelをグレーに塗りつぶすだけ

※これらは、JFrameも同様

「紙」の準備(2)

- ウィンドウの操作がされるたびに「paintComponent」が呼び出される
  - グレーに塗りつぶされる = 描いた絵が消える
- 描いた絵が消されないようにするには?
  - 「paintComponent」を定義しなおす
    =ウィンドウの操作がされると、定義しなおされた「paintComponent」が呼び出される
    =ウィンドウの操作がされると、もう一度同じ絵が描かれる
    =絵が消えない
  - JPanelを継承して「paintComponent」をオーバーライド

JPanelを継承して...?(1)

- ファイルを1つ作成する

JPanelを継承したクラス

```
import java.awt.*;
import java.io.*;
import java.lang.*;
import javax.swing.*;

public class クラス名 extends JPanel{
  public void paintComponent(Graphics g){
    
  }
}
```

JPanelの継承

JPanel上に絵を描くメソッド

JPanelに描く絵の内容

JPanelを継承して...?(2)

JPanelを継承したクラス

```
import java.awt.*;
import java.io.*;
import java.lang.*;
import javax.swing.*;

public class クラス名 extends JPanel {
    public void paintComponent( Graphics g ) {
```

- 「g」は、「Graphics」というクラスのオブジェクトで、JPanel専用のペンの役割をする
- 絵を描くときは、「ペン」に対して、どのような絵を描くか命令をする

プログラム本体は?

- 書き方は通常のGUIプログラムと同じ
- 絵を描くために使うJPanelのみ、自分で作成したクラスを利用すること

```
public 本体のクラス名 extends JFrame {
  JFrame frame;
  JButton but1, but2;
  描画用パネルクラス名 canv;
  public 本体のクラス名 {
    ......
    canv = new 描画用パネルクラス名();
    ......
  }
  public static void main(String[] args) {
    new 本体のクラス名();
  }
}
```


お絵描き～本編～

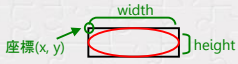
お絵描きのためのメソッド

- 絵を描くときは、JPanel専用の「ペン」に対する命令を記述
  →「ペン」の役割をする「Graphics」クラスに、絵を描くためのメソッドが用意されている
  - 「draw」で始まるメソッド: 四角形や円などを塗りつぶさずに描く
  - 「fill」で始まるメソッド: 四角形や円などを塗りつぶして描く
- 絵を描くときは、各部品の左上隅の点の座標と縦横の大きさを指定
- X座標は右方向、Y座標は下方向

メソッド例(1)

- drawRect(int x, int y, int width, int height)
  - 四角形を描く(fillRectも同様)
  - x, y: 左上の点のx座標とy座標
  - width, height: 四角形の幅と高さ
- drawOval(int x, int y, int width, int height)
  - 楕円を描く(fillOvalも同様)
  - x, y: 左上の点のx座標とy座標
  - width, height: 楕円の幅と高さ

ただし、楕円を四角形で囲んだときの、四角形の座標と幅・高さ



メソッド例(2)

- drawString(String str, int x, int y)
  - 文字列の描画
  - str: 描画する文字列
  - x, y: 文字列の左上の点のx座標とy座標
- draw3DRect(int x, int y, int width, int height, boolean raised)
  - 3Dで強調表示される四角形の描画(fill3DRectも同様)
  - x, y: 左上の点のx座標とy座標
  - width, height: 四角形の幅と高さ
  - raised: 「true」で浮き出たように見え、「false」で彫り込まれたような感じ

メソッド例(3)

- drawLine(int x1, int y1, int x2, int y2)
  - 線の描画
  - x1, y1: 線の開始点のx座標とy座標
  - x2, y2: 線の終了点のx座標とy座標
- drawPolyline(int[] xPoints, int[] yPoints, int nPoints)
  - 折れ線の描画
  - xPoints, yPoints: 折れ線の開始点, 折れる点, 終了点を配列に格納したもの
  - nPoints: 配列の中からいくつの点を使って折れ線を描くか

メソッド例(4)

- drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
  - 多角形の描画(fillPolygonも同様)
  - xPoints, yPoints: 多角形の折れ線の開始点, 折れる点, 終了点を配列に格納したもの
  - nPoints: 配列の中からいくつの点を使って多角形を描くか

メソッド例(5)

- drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
  - 楕円弧の描画(fillArcも同様)
  - x, y: 描画される孤の左上の点のx座標とy座標
  - width, height: 描画される孤の幅と高さ
  - startAngle: 開始角度
  - arcAngle: 開始角度に対する弧の展開角度の大きさ

メソッド例(6)

- drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
  - 角の丸い四角形の描画(fillRoundRectも同様)
  - x, y: 描画される四角形のx座標とy座標
  - width, height: 描画される四角形の幅と高さ
  - arcWidth: 4 隅の弧の水平方向の直径
  - arcHeight: 4 隅の弧の垂直方向の直径

メソッド使用例

```
public void CanvasPanel extends JPanel {
  public void paintComponent(Graphics g) {
    g.fillRect(5, 5, 390, 290); /* 四角形の描画 */

    g.fillOval(5, 5, 390, 290); /* 楕円の描画 */

    int[] x1 = {200, 5, 390};  /* 三角形の頂点のx座標の配列 */
    int[] y1 = {5, 150, 150};  /* 三角形の頂点のy座標の配列 */
    g.fillPolygon(x1, y1, 3); /* 三角形の描画 */

    int[] x2 = {200, 5, 390}; /* 三角形の頂点のx座標の配列 */
    int[] y2 = {290, 150, 150}; /* 三角形の頂点のy座標の配列 */
    g.fillPolygon(x2, y2, 3);  /* 三角形の描画 */
  }
}
```


色をつける

- 「setColor」というメソッドで、ペンの色を変更
- ペンの色は1度変えると、その後もずっと同じ色
  - ある部品を赤で描き、その次の部品を元の色で描きたい場合には、赤で部品を描いた後に、「setColor」でペンの色を黒に戻すことが必要
- 「setColor」の引数は「Color」というクラスのオブジェクト

色を使う

- 色を扱うクラス: Color
- 色の使い方その1: Javaで指定されている色を使う
  - Color.*色の名前*
- 色の使い方その2: RGBカラーを使う
  - 「new Color(int r, int g, int b)」で色を作成する
    - r: 赤成分(0～255)
    - g: 緑成分(0～255)
    - b: 青成分(0～255)

Javaでの色の名前

- Color.black
- Color.blue
- Color.cyan
- Color.darkGray
- Color.gray
- Color.green
- Color.lightGray
- Color.magenta
- Color.orange
- Color.pink
- Color.red
- Color.white
- Color.yellow

RGBカラー

- コンピュータの色: 赤(Red), 緑(Green), 青(Blue)の光から全ての色を表現する
  - それぞれ256段階の濃淡があり、それを混ぜ合わせて色を表現する
  - 混ぜ合わせる赤, 緑, 青の濃さの段階を数値で表す
  - 255が最も濃く、0が最も淡い

RGBカラー

例えば...
赤: 255, 緑: 0, 青: 0 →
赤: 204, 緑: 153, 青: 255→

コンピュータでは
- 赤, 緑, 青の256段階の濃淡の数値を16進数で表す
- 16進数で表した数値を赤・緑・青の順に2桁ずつでならべ、「#」を先頭に付けて色の名前として使う

RGBカラーを使うには?

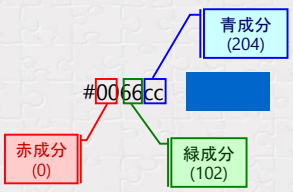
- 「色見本」などのキーワードで、色のサンプルリストを探す
- 色のサンプルリストから、好きな色を選択する
  - 色のサンプルリストは、「#」つきの色の名前が示されていることが多い
- 「#」つきの色の名前を、10進数の赤・緑・青の数値に直して、Colorクラスのコンストラクタに設定する

16進数(1)

- 数を16個の文字で表す方法
  - 普段は10進数(数を10個の文字で表す方法)
  - 10進数の「16」を、16進数では「10」と表記
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, fの16個の文字で表現
  - 「a」が10進数の10, 「b」が10進数の11, 「c」が10進数の12, 「d」が10進数の13, 「e」が10進数の14, 「f」が10進数の15

16進数での色の表現

- 「#」を最初につけ、R(赤)、G(緑)、B(青)の順に16進数で2桁ずつで表現



16進数の色の名前を10進数に直す

1. 「#」の後の6桁の16進数を2桁ずつに分解する
   - 左側から赤・緑・青の数値になる

16進数の色の名前を10進数に直す

2. アルファベットを10進数に直す
   - A → 10
   - B → 11
   - C → 12
   - D → 13
   - E → 14
   - F → 15

   ※アルファベットの大文字・小文字は関係なし

00 → 0 0
66 → 6 6
CC → 12 12

16進数を10進数に変換

3. 2. の数の1桁目の上に「16^0」、2桁目の上に「16^1」と書く

16^1 16^0
0 0
16^1 16^0
6 6
16^1 16^0
12 12

16進数を10進数に変換

4. 各桁の上の「16^n」と、それぞれの桁の数をかけあわせる

16^1 16^0
× × → 0 0
0 0

16^1 16^0
× × → 6×16 6×1
6 6

16^1 16^0
× × → 12×16 12×1
12 12

16進数を10進数に変換

5. 4. の各桁の数を足し合わせる

0 0 → 0 + 0 = 0 ← 赤成分の10進数
6×16 6×1 → 6×16+6×1 = 102 ← 緑成分の10進数
12×16 12×1 → 12×16+12×1 = 204 ← 青成分の10進数

6. 赤・緑・青成分の10進数をColorクラスのコンストラクタに設定

new Color(0 , 102 , 204)

色を使う(例)

キャンバスの「paintComponent」メソッド

```
public void paintComponent(Graphics g) {
   g.setColor(new Color(0, 0, 255)); /* 青色の四角形を描く */
   g.fillRect(5, 5, 390, 290);

   g.setColor(new Color(0, 255, 0)); /* 緑色の楕円を描く */
   g.fillOval(5, 5, 390, 290);

   int[] x1 = {200, 5, 390};
   int[] y1 = {5, 150, 150};
   g.setColor(Color.yellow); /* 黄色の三角形を描く */
   g.fillPolygon(x1, y1, 3);

   int[] x2 = {200, 5, 390};
   int[] y2 = {290, 150, 150};
   g.setColor(Color.cyan); /* シアン色の三角形を描く */
   g.fillPolygon(x2, y2, 3);
}
```

例～写して実行してみよう!～(1)

メインのプログラム

```
public class DrawPicture extends JFrame{
  CanvasPanel canv;

  public DrawPicture() {
    canv = new CanvasPanel();

    getContentPane().add(canv);
    setTitle("お絵かき");
    setSize(410, 340);
    setDefaultCloseOperation(Jframe.EXIT_ON_CLOSE);
    setVisible(true);
  }
  public static void main(String[] args) {
    new DrawPicture();
  }
}
```


例～写して実行してみよう!～(2)

描画領域のクラス

```
public class CanvasPanel extends JPanel {
  public void paintComponent(Graphics g) {
    g.setColor(new Color(0, 0, 255));
    g.fillRect(5, 5, 390, 290);

    g.setColor(new Color(0, 255, 0));
    g.fillOval(5, 5, 390, 290);

    int[] x1 = {200, 5, 390};
    int[] y1 = {5, 150, 150};
    g.setColor(Color.yellow);
    g.fillPolygon(x1, y1, 3);

    int[] x2 = {200, 5, 390};
    int[] y2 = {290, 150, 150};
    g.setColor(Color.cyan);
    g.fillPolygon(x2, y2, 3);
  }
}
```


期末試験のお知らせ

- 期末試験: 1月24日(火) 2限 9301教室
  - 時間: 90分
  - 持ち込み: 全て可
  - 内容: 後期の講義内容すべて
    - 用語の意味の選択・説明
    - 概念に関する説明
    - 実技