**Swingのまとめ**

1. **Swingとは**

SwingとはGUIを使ったアプリケーションを作成できる機能をまとめたものです。

GUIとはGraphical User Interfaceの略で、Window上にアイコンやメニューや図を表示し、それをマウスなどで動かすことができる仕組みです。

　　大きなメリット：　 EclipseやWindowsのコンソール画面を使わなくても、他のWindowsのプログラム、ゲームのように、プログラムを動かすことができます。

**２．プログラムを書く時の定石とTemplate（その１　JFarme）**

JFrameというクラスをextendして、フレーム（Windowの枠に相当）のクラスを作成します。

（１）画面のサイズ、タイトルバーの文字、クローズボタンの処理等を用意したコンストラクタを

　　　　　　　　　作成します。

（２）上記コンストラクタの中で、下記（２）で述べる、パネルを張るためのコンテナを用意する。

a．コンテナの作成は、Conteiner cont = getContendPane();

b.（２）で定義するパネルのインスタンスを生成し

c..　cont.add(panel);でコンテナにパネルを張ります

d.　複数のパネルを場所を指定してコンテナに張り付けることもできます。

パネルの位置の決め方は３種類

cont.add(panel, BorderLayout.CENTER);　NORTH,WEST等

他にBoxLayout, GridLayout Swing

（３）public static void main(String[] args)｛｝を作り、

インスタンス名を用意して、自身のコンストラクタの呼び出し生成する

そのインスタンスを見えるようにする。

**JFrameのTemplete　（JFrameMatome.java）**

**public** **class** JFrameMatome **extends** JFrame {

**public** JFrameMatome() { //コンストラクタ

setSize(400, 300); //Window サイズ設定（せってい）

setTitle("マイプログラム");

setDefaultCloseOperation(**EXIT\_ON\_CLOSE**);

//パネルを用意(ようい)

Container **cont** = getContentPane();

ActionPanel1 **panel1** = **new** ActionPanel1();

cont.add(panel1);

}

**public** **static** **void** main(String[] args) {

JFrame **frame** = **new** JFrameMatome();//フレーム生成

frame.setVisible(**true**); //見えるようにする

}

/\*\* 以下に、JPanelのクラスを記述します（内部クラスで書く場合）\*/

/\*\* JPanelは別クラスで記述してもOKです \*/

/\*\* ３．項のJPanelを張り付け、挿入してください。 \*/

}

**３. プログラムを書く時の定石とTemplate（その２　JPanel）**

JPanelというクラスをextendして、パネル（フレームに張り付ける画面）のクラスを作成します。

1. コンストラクタを作成します。
   * 1. setBackground(Color.WHITE);等の設定をします。
     2. public void paintComponent(Graphics g)｛　… ｝でパネルに描画する

　メソッドを記述します。

　ここに記述されたg.drawSting("ABC", x, y);が、パネルに描画されます。

　この中に記述できる　Graphics gの持つメソッドを、以下にリストアップします。

**JPanelのTemplete**

**class** ActionPanel1 **extends** JPanel {

**int** clickX, clickY; // クリック位置（いち）座標（ざひょう）

**public** ActionPanel1() {

setBackground(Color.**PINK**);

// イベントを有効にするActionListnerをこの後に追記します

/\*\* Listenerの記述　６項のイベントを挿入してください\*/

}

@Override

**public** **void** paintComponent(Graphics g) {

**super**.paintComponent(g); // 必ず記述、描画の前準備に相当

//描画する本体を記述する　以下は例

g.drawString("マイゲーム", 20, 20);

g.drawOval(clickX, clickY, 80,80); // 図形（ずけい）の描画（びょうが）

}

}

**４．Graphics gの持つ種々のメソッド**

g.setColor(Color.BLUE）; //色をセットする

g.drawString("ABC", x, y); //x,yの座標に"　"の文字を表示する

g.drawLine(sx, sy, ex, ey); // (sx,sy)と(ex,ey)を結ぶ直線を描画する

　 g.draqRect(sx, sy, ex, ey); // (sx,sy)と(ex,ey)が対角となる四角形を描画する

g,fillRect(sx, sy, ex, ey); // 同上の塗りつぶし

g.drawOval(左上のx座標、y座標、 幅(はば), 高(たか)さ);　// 楕円の描画

g.fillOval(左上のx座標、y座標、 幅(はば), 高(たか)さ);　// 塗りつぶし楕円の描画

**５．画像の読み込みと画像を表示するメソッド**

画像ファイルをimage1として読み込む方法

ImageIcon icon1= new ImageIcon("src/airplane.gif”);

Image image1 = icon1.getImage();

画像の表示

g.drawImage(image1, x, y, sx, sy, this);

左上(x,y)座標にサイズsx,syで画像を描画する。

**６．イベントの取り込み（リスナーの）**

（１）パネルのクラスを作るとき、必要なListener（インターフェース）をimplementsします。

イベントごとに、必要とするListenerが異なるので注意が必要です

例えば、マウス動作を捉えるには、MouseListenerをimplementsします。

各種のイベントに対応するため、下記の５つのメソッドを実装する必要があり、動作をさせたい

　　　　イベントの中で、処理（例えば、Iconを表示する、文字を出す）を記述します。

public void mousePressed(MouseEvent e) {}　public void mouseReleased(MouseEvent e) {}

public void mouseClicked(MouseEvent e) {}　public void mouseEntered(MouseEvent e) {}

public void mouseExited(MouseEvent e) {}。

（２）上記（１）では記述量が増えるため、自分で必要なイベントを捉えるアダプタと呼ぶ

クラスを定義しJPanelでそのクラスをインスタンス化して利用することができます。

Mouseの場合では、クラスMouseAdapterをextendしたクラスを作ります。

**イベントを捉えるクラス定義のテンプレート(Mouseクリックのケース)**

**class** MyMouseAdapter **extends** MouseAdapter { // JPanelのインナークラス

/\*\* マウスが押（お）されたら　\*/

@Override

**public** **void** mousePressed(MouseEvent e) {

clickX = e.getX();// クリック位置（いち）の取得

clickY = e.getY();

repaint(); // 再描画要求（さいびょうがようきゅう）

}

｝

repaint()で再描画されることがポイントです。

これを、JPanelのインスタンスの中で、インスタンス化します。（３項の具体例）

**class** ActionPanel1 **extends** JPanel { // 再掲（下記の2行を追加する）

**public** EventPanel3() { // コンストラクタ

setBackground(Color.**ORANGE**);

//画像（がぞう）のよみこみ等の処理

//イベントを有効（ゆうこう）にする　　ココです

MyMouseAdapter **adapter** = **new** MyMouseAdapter();

addMouseListener(adapter);

}

他　マウスドラッグなどを検出するには、MouseMotionAdapterをextendsします

また、右クリックか、ダブルクリックかなどの判定もできます。元の資料を確認してください。

（３）キーイベントの場合

class MyKeyAdapter extends KeAdapterでクラスを定義します

KeyPressed；キーが押された

KeyTypeed : 文字が入力された

などのイベントを捉えることができます　Swing\_5の資料を参照してください。

（４）タイマーイベントの場合

パネルクラスは、Timerを使うため、ActionListenerをimplementsしておきます。

Timer timer = New Timer(100,this);//100msごとにイベントを起動する。

timer.start();

のように、Panelのコンストラクタ内で記述し、タイマーをインスタンス化します。

タイマーイベント処理のメソッドが呼び出されるので、

ActonPerformed(ActionEvent){ repaint(); }

タイマーごとの繰り返されます。

　注意：　Javaには、java.util.Timer」と「javax.management.timer.Timer」がありますが、

　今回使うのは、javax.swing.Timer;です。これをインポートしてください。

**７．GUIインターフェース　ボタン、テキスト、テキストエリア**

（１）パネルにボタン、ラジオボタン、一行テキスト入力や、複数行のテキストエリアを配置します。

JButton 変数 = new JButton("ボタンの文字");

JRadioButton 変数 = new JRadioButton("ラジオボタンの文字");

JTextField 変数 = new JTextField("テキストの文字", 桁数);

JTextArea 変数 = new JTextArea("テキストエリアの文字", 行数, 列数);

JList<要素の型> JListの変数 = new JList<要素の型>(配列の変数);

JComboBox<要素の型> コンボボックスの変数 = new JComboBox<要素の型>(配列の変数);

コンボボックスはプルダウンメニューです。

JCheckBox 変数 = new JCheckBox("チェックボックスの文字");

上記で作成したインスタンスを張り付けるのは、下記で実施します。

パネルクラス.add(変数);　//コンテナでも可

1. ボタンが押されたときにイベントを有効にするには

パネルのクラス定義で、ActionListenerをimplementsしておき

コンストラクタの中で、buttonの変数名.addActionListener(this)；とします。

さらにメソッドactionPerformed(actionEvent e);をオーバライドして

イベントをキャッチし、このメソッドの中で、処理を呼び出すとともに

repaint()します。どのボタンが押されたかは、JButton pushButtonに

名前を代入するgetSource()のメソッドを使います。

少し複雑なので、テンプレートを示します。

@Override

**public** **void** actionPerformed(ActionEvent e) {

JButton pushButton = (JButton) e.getSource();

**if** (pushButton == buttonL) {

flag = **false**;

} **else** {

flag = **true**;

}

repaint();

}

1. その他の部品の注意事項　（Swing\_6\_2\_GUIの資料の後半を参照）

ラジオボタンの場合、グループ化して、ひとつしか選べないようにします。

コンボボックスの選択中の文字を得る

String 変数 = (String) コンボボックスの変数.getSelectedItem();

テキストボックスにスクロールボタンを付ける

JScrollPane pane = new JScrollPane(テキストエリアの変数);

テキストボックスの自動改行

キストエリアの変数.setLineWrap(true);