第 3 章 Swing

この章では、Swingというパッケージを使って、簡単なものを作っていきます。

1. Swingとは

Javaで作成するプログラムをアプリケーションといいます。この中で、GUIアプリケーションを作るのに使われるパッケージ（ライブラリ）が、Swingです。　　ちなみに、GUIとは、Graphical　　User Interfaceの略で、画面が出てくるアプリケーションのことです。

2. Swing の基本スタイル

今回は、このテンプレートを使って、ゲームを作っていきたいと思います。まず、以下のコードを実行してみましょう。

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

public class MyJPanel extends JPanel{
    public void game(int width, int height){
        Image img  = createImage(width, height);
        Graphics g = img.getGraphics();
        Graphics wg = getGraphics();

        while(true){
            wg.drawImage(img, 0, 0, null);
        }
    }
}
```

実行結果：

[CSSC2016]SwingSample1の実行結果.png[result1] | *img/*

何もないウィンドウが表示されたと思います。Javaコースではこのウィンドウを使って、いろいろなアプリケーションを作っていくことになります。　　mainメソッドの中に、`height`と`width`という変数があると思いますが、これの変数は画面の縦横の大きさが格納されています。
画面を大きさを変更したいときは、heightとwidthの値を変えてやれば、画面が大きくなったり小さくなったりします。

何もないと味気ないので、文字を表示するアプリケーションを作ってみましょう。「Hello World!」と表示するために、さっきのコードを少し変えましょう。　　変えるのは、MyJPanelクラスの中の、while文の中にひとこと追加するだけです。

```
while(true){
    g.drawString("Hello, world!" ,100,100); //追加！
    wg.drawImage(img, 0, 0, null);
}
```

最初に作ったウィンドウに、Hello, Worldという文字が表示されると思います。

2.1. drawString(string, x, y)

ここで、drawStringの定義を見てみましょう。

第１引数　：　表示する文字列

第２引数　：　表示するｘ座標

第３引数　：　表示するｙ座標

注意点として、ここでいうxとｙは、左から数えてxピクセル、上から数えてｙピクセルという意味です。
Swingは基準点が左上なので、気をつけてください。

2.2. 余力のある人は、フォントも設定してみよう

`drawString`メソッドは、フォントを設定することもできます。
`setFont`メソッドを使って以下のようにすると、フォントを設定できます。

```
            wg.drawImage(img, 0, 0, null);
        }
    }
}
```

3. 図形、画像の表示

今度は、Graphicsクラスを使って図形を描いてみましょう。Graphicsクラスを使うと、線を引いたり四角形を描いたりできます。 他にも背景の色を変更したり、見た目をいろいろと変更することができます。

まず、以下のコードを実行してみましょう。while文の中を変えるだけですね。

```
public class MyJPanel2 extends JPanel{
    public void game(int width, int height){
        Image img  = createImage(width, height);
        Graphics g = img.getGraphics();
        Graphics wg = getGraphics();

        while(true){
            g.setColor(Color.BLUE);
            g.fillOval(0, 0, 100, 100);

            g.setColor(Color.BLACK);
            g.drawRect(0, 0, 300, 100);

            g.setColor(Color.GREEN);
            g.drawLine(150, 100, 150, 300);

            g.setColor(Color.YELLOW);
            g.fillArc(200, 200, 100, 100, 30, 300);

            wg.drawImage(img, 0, 0, null);
        }
    }
}
```

実行結果：

[CSSC2016]SwingSample3の実行結果.png[result2] | *img/*

それぞれのメソッドについて解説します。

3.1. 四角形を描く

四角形を描くには、`drawRect`もしくは`fillRect`メソッドを使います。

```
drawRect(x, y, width, height)
fillRect(x, y, width, height)
```

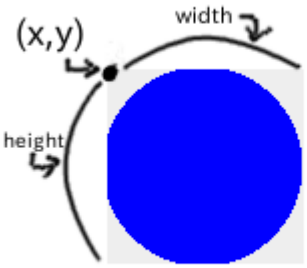
`drawRect(x, y, width, height)`と指定すると、座標(x,y)を四角形の左上の頂点として、高さがheight、幅がwidthの長方形が描かれます。 `drawRect`は四角形の外枠だけが描かれて、`fillRect`は中が塗りつぶされた四角形を描くことができます。

3.2. 円を描く

円を描くには、`drawOval`もしくは`fillOval`メソッドを使います。

```
drawOval(x, y, width, height)
fillOval(x, y, width, height)
```

drawOval`メソッドは、座標（x,y）を左上とした、高さ`height、幅`width`の四角形の中にすっぽり入る円を描くことができます。 `fillOval`で、中が塗りつぶされた円を描くこともできます。



3.3. 角度を指定して、円（孤）を描く

角度を指定して円を描くには、`drawArc`もしくは`fillArc`メソッドを使います。

```
drawArc(x1, y1, x2, y2, arc1, arc2)
fillArc(x1, y1, x2, y2, arc1, arc2)
```

`drawArc`メソッドは、座標(x1,y1）と（x2,y2）を対角線（右下がり）とする四角形の中にすっぽり入るような円が描けます。 `arc1`と`arc2`で角度を指定すれば、arc1からarc2までの角度の扇型を描くことができます。例えば、0,360とすれば、一回転なので、ふつうの円が描けます。 30,300とすれば、パックマンが出来上がります（笑）。

3.4. 色を指定する

描く図形の色を指定したい場合は、`setColor`を使います。

```
setColor(Color.BLUE)
```

図形を描く前に、`setColor`を使えば、図形に色をつけられます。 例えば、

```
g.setColor(Color.BLUE);
g.fillOval(0, 0, 100, 100);
```

とすれば、青色の円が描けると思います。

他にも様々な色が指定できます。

コード	色
Color.BLACK	黒
Color.BLUE	青
Color.GRAY	灰色
Color.GREEN	緑
Color.ORANGE	オレンジ
Color.PINK	ピンク
Color.RED	赤
Color.WHITE	白
Color.YELLOW	黄色

また、数字で厳密にカラーコードを指定することも可能です。

```
g.setColor(new Color(127,255,212));
```

3.5. 画像を表示する

次に、画像を表示してみましょう。
今回は、こちらでは特に画像を用意していないので、自分で好きな画像をネットからダウンロードしましょう。

3.5.1. 画像を保存する手順

1. firefoxを起動

2. 画像を適当に検索して、画像を保存。

3. eclipse で、**img** というフォルダを作る。

srcを右クリックして、`New → Folder`を選択します。
画像を入れるフォルダなので、フォルダ名は`img`としておきましょう。

4. 保存した画像を、**eclipse**の**img**フォルダへドラッグ＆ドロップ

3.5.2. 画像を表示してみよう

画像の保存が無事終わったところで、今度は画像を表示するコードをかいてみましょう。

```
    public void game(int width, int height) {
        Image img = createImage(width, height);
        Graphics g = img.getGraphics();
        Graphics wg = getGraphics();

        //追加！
        Image image = null;
        try {
            image = ImageIO.read(new File("img/aizu.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }

        while (true) {
            g.drawImage(image, 0, 0, this); //追加！

            wg.drawImage(img, 0, 0, null);
        }
    }
```

画像を読み込むには、`ImageIO.read`メソッドを使い、画像を表示するには、`drawImage`メソッドを使います。サンプルでは、aizu.pngを読み込んでいます。`new File`は、ファイルを開く命令です。それらを囲んでいる`try`と`catch`は、例外処理といって、エラーを処理するときに使いますが、今はあまり気にしなくて大丈夫です。

画像を読み込んだら、あとは`drawImage`メソッドで描くだけです。

```
drawImage(image, x, y, this);
```

第１引数`image`には、先ほどファイルを読み込んで格納した変数をいれます。ここでは、変数`image`がそれにあたります。

第２、３引数のx,yは、画像の位置ですね。左上が基準点になっているので注意です。

第４引数は、this（もしくは`null`）を入れれば大丈夫です。