Javaゲームプログラミングコース

Table of Contents

1. 導入

1.1. まず、Javaコースでは何をするの？

このコースでは、Javaで簡単なゲームを作っていきます。

１日目は、Javaの基礎知識やSwingの使い方、簡単なゲームの作り方を学びます。

２、３日目は、初日で学んだ知識をもとにゲームを作っていきます。何人かのチームに分かれてゲームを作っていきましょう。

最終日は、作ったゲームをみんなでプレイしましょう！

1.2. Javaってなあに？

Java（じゃば）は、プログラミング言語の１つで、オブジェクト指向という考え方をもとに作られました。

1.3. Swingってなあに？

Swing(スウィング)とは、画面を出すためのライブラリです。（後に詳しく説明します）

1.4. Javaって何に使われているの？

一番メジャーなのは、Androidでしょう。そう、みなさんが持っているであろう、Androidのスマホは実はJavaで動いているのです。

2. Eclipeを使おう

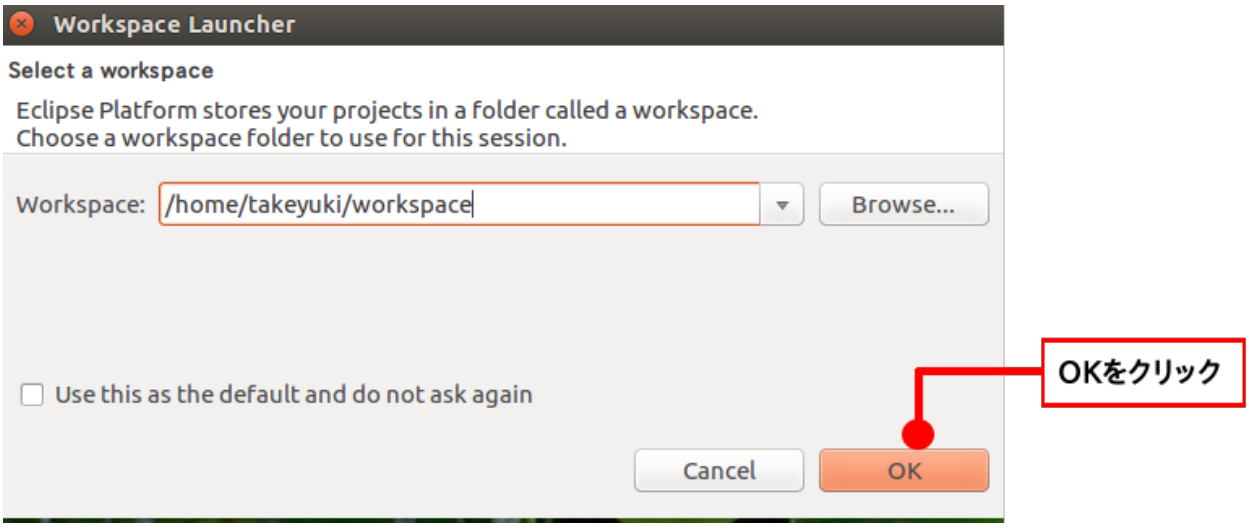
2.1. Eclipseとは

EclipseとはJavaの統合開発環境と呼ばれるもので、これひとつで、Javaプログラミング、コンパイル、実行のすべてができるツールです。さらにデバッグも自動で行ってくれるため、プログラミングの効率も向上するでしょう。

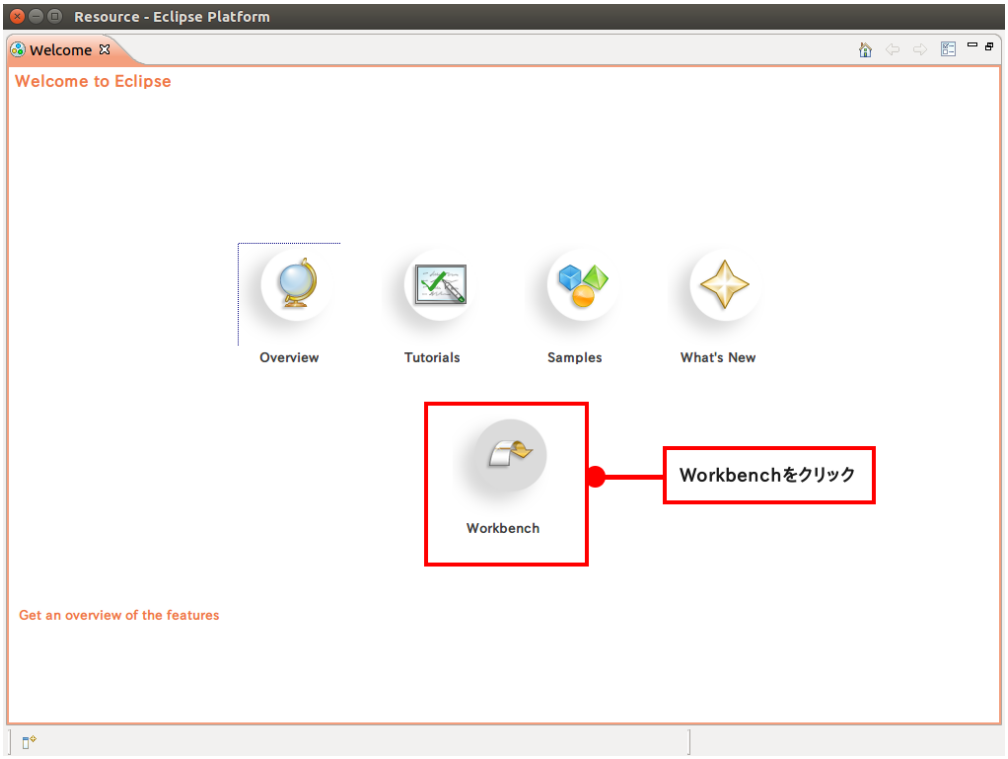
2.2. Eclipseの使い方

では、さっそく Eclipseを使ってみましょう。

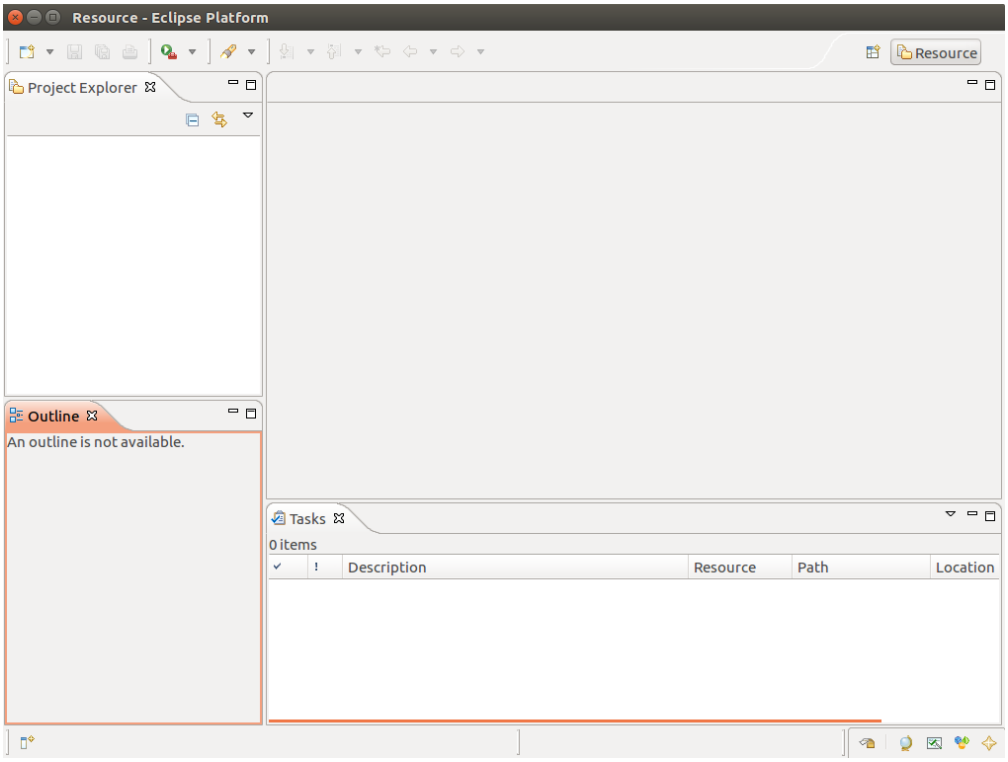
まず、Eclipseを起動すると、このようなものが出てきます。 ここは、そのままOKを押してください。



すると↓の画面がでてくるので、workbenchをクリック

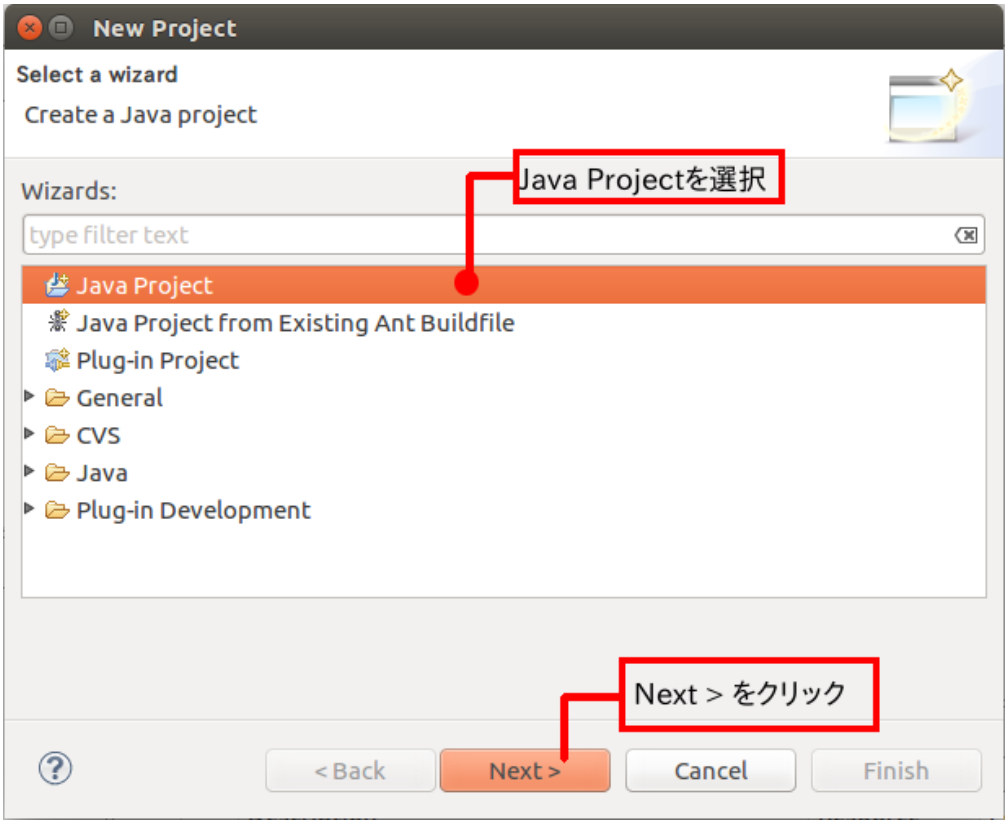


この画面が出てきます。

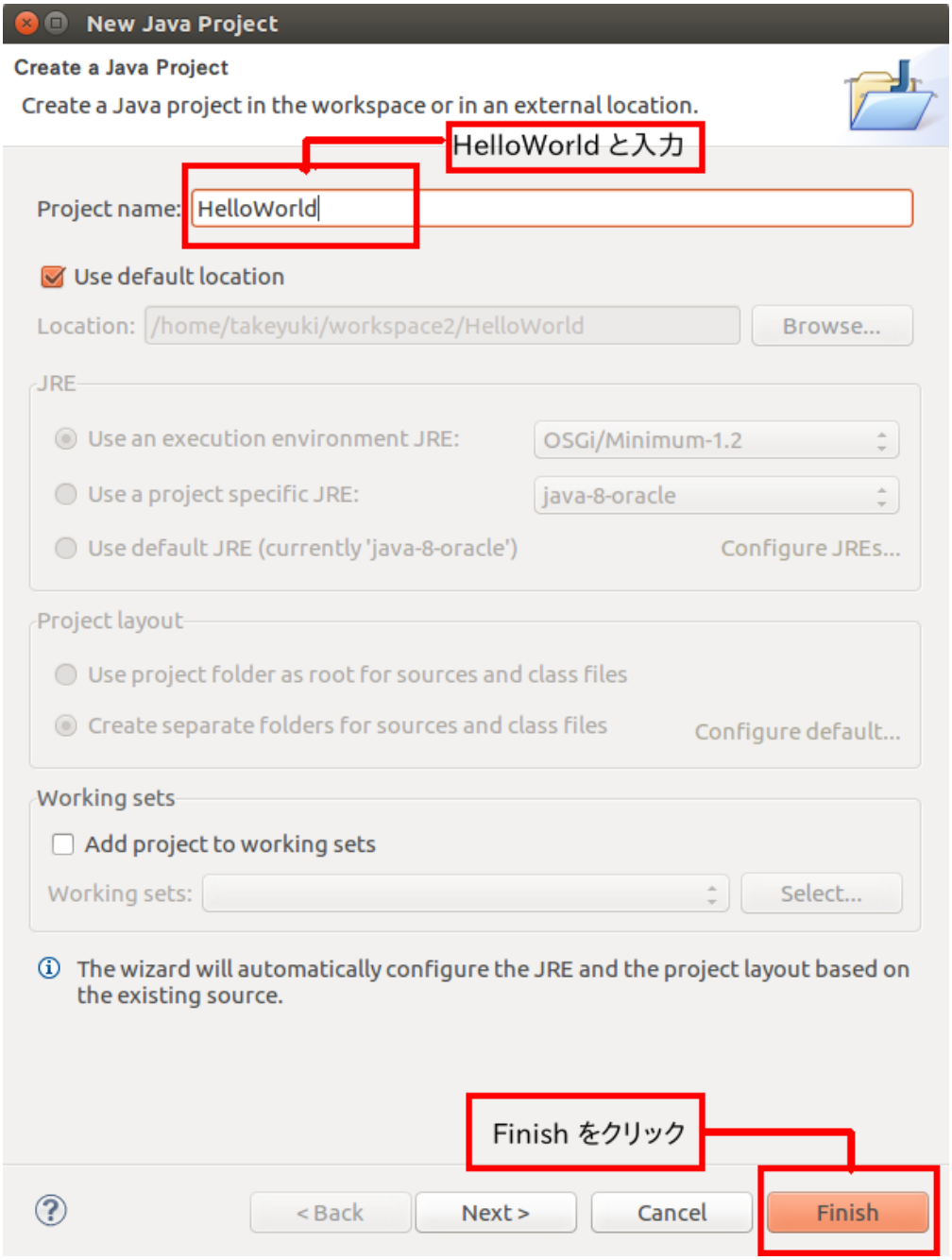


次に、Project Explorerとかいてあるところの下の、白い枠を右クリックし、New → Projectを選択します。

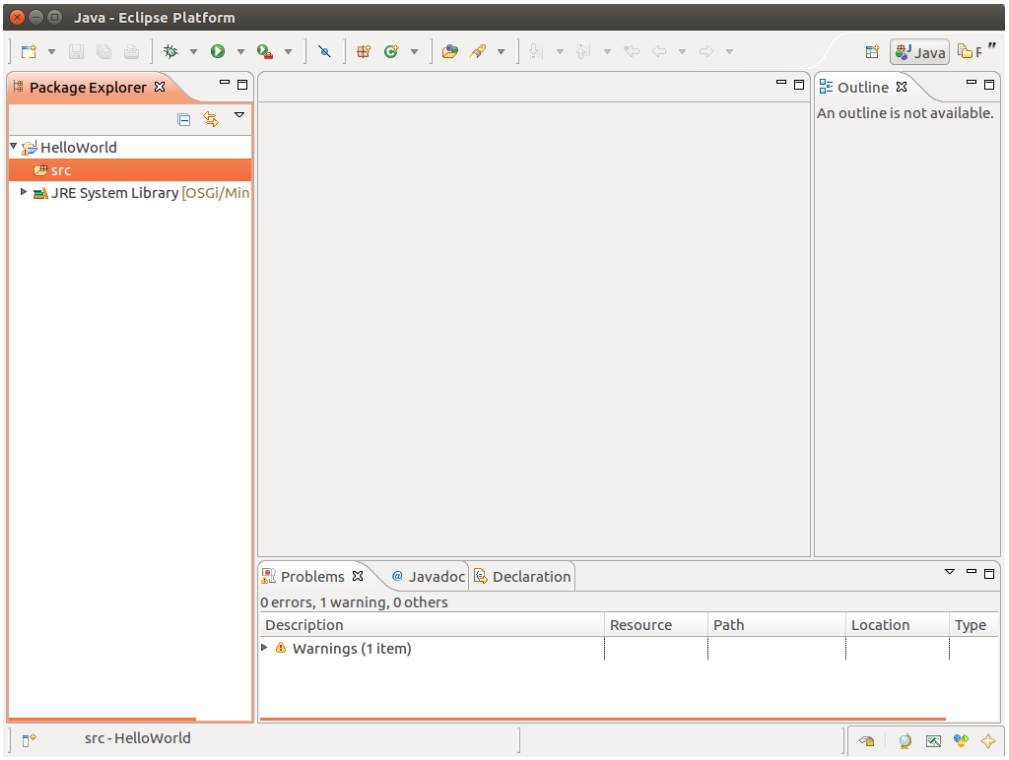
Java Project を選択して、Next> をクリック



Project name: のところに、今回は、HelloWorldと入力。



このような画面がでてきます。

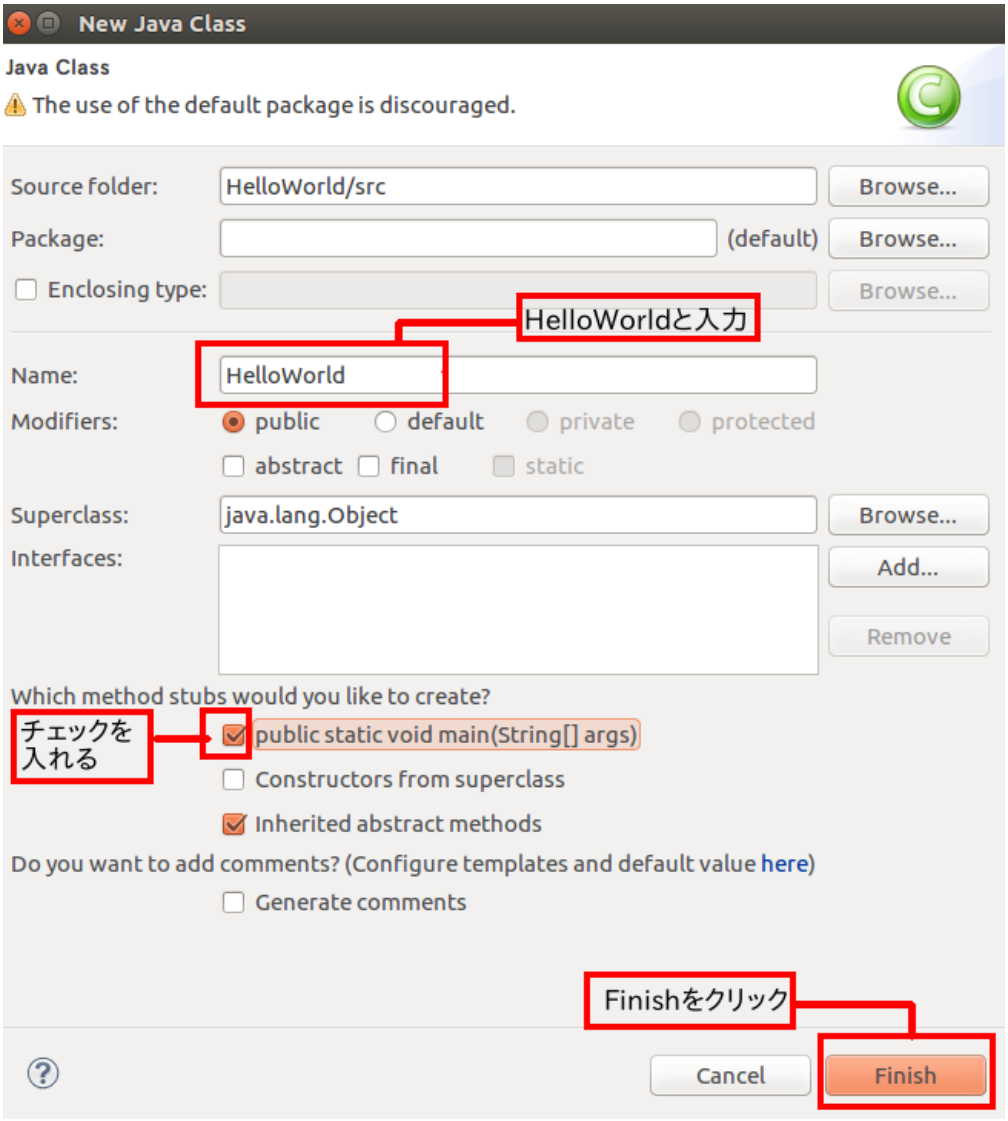


HelloWorldのところをダブルクリックすると、srcというフォルダが出てきます。

srcを右クリック → New → Class を選択

次に

1. Name: のところに、HelloWorldと入力
2. public static void main(String[] args)　のところにチェックを入れる
3. Finishをクリック



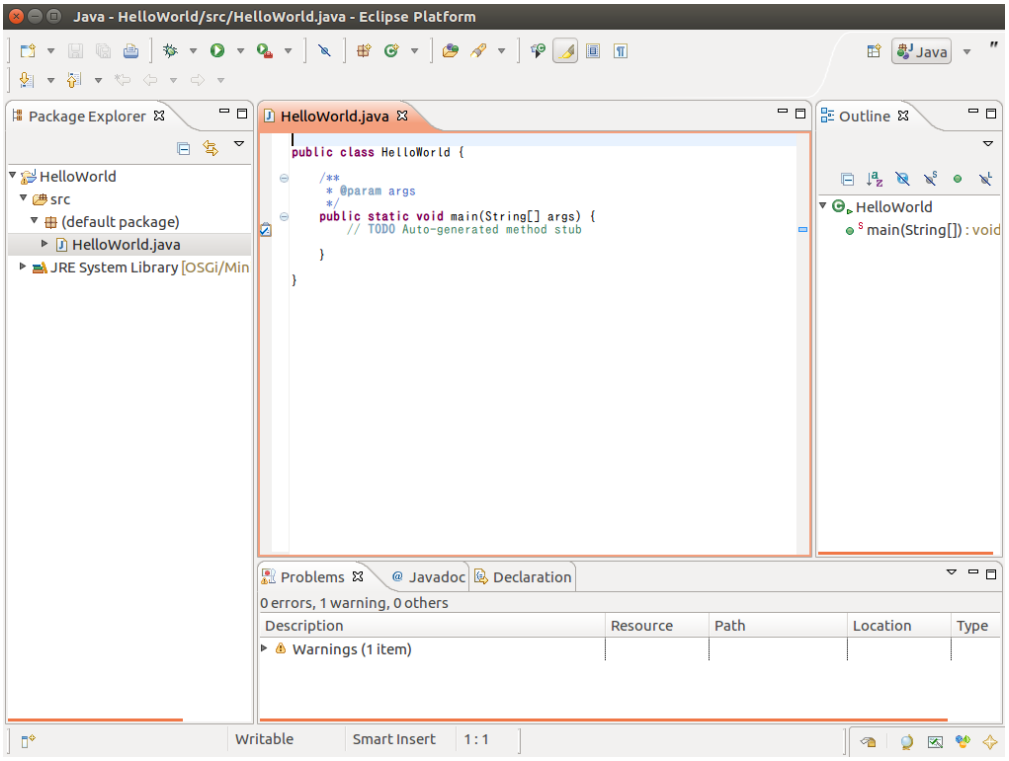
NOTE

ここで２つ注意しておくことがあります。

- １つのファイルには、１つのクラスのみかく。
- クラスの名前とファイルの名前は同じにする。

これらのことに注意してプログラムをするようにしてください。

Finishをクリックすると、この画面がでてきます。

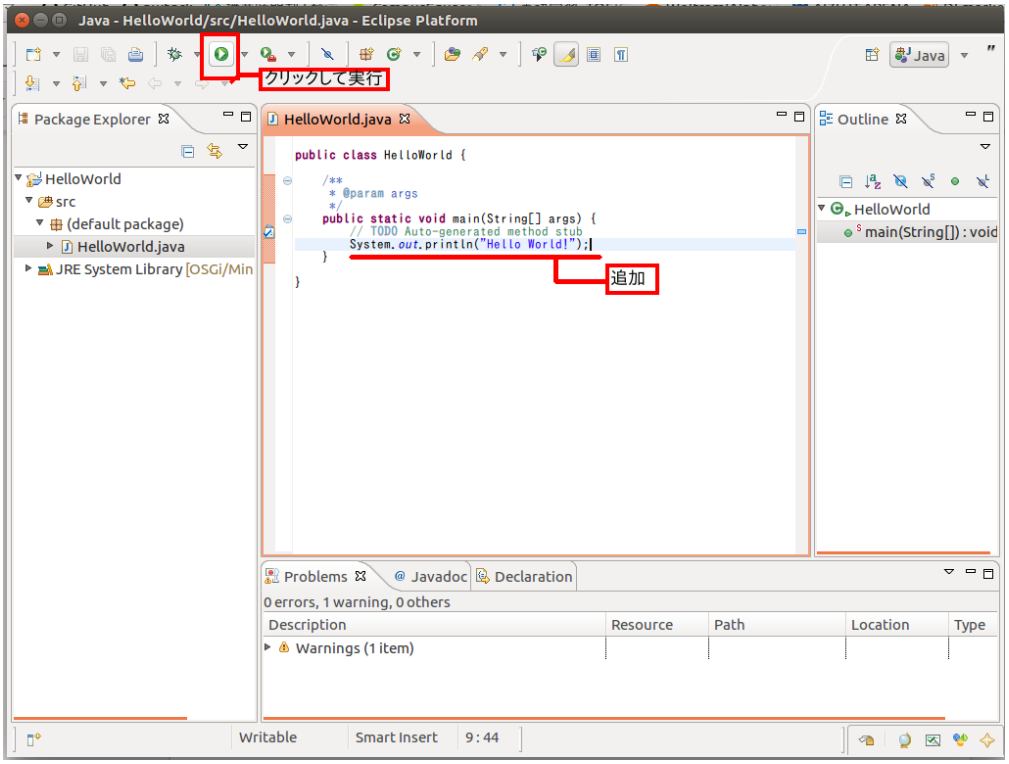


ここで、ちょとしたプログラムをかいてみます。 `public static void main(String[] args){` という行の下に、

```
System.out.println("HelloWorld");
```

と追加します。

プログラムを実行するときは、左上のほうの、緑色の三角ボタンを押してください。



3. 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。

3.1. 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。

3.2. 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：

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。

3.2.1. 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。

3.2.2. 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。

```
        while (true) {
            g.setFont(new Font("Arial", Font.BOLD, 24));
            g.drawString("hello world", 100, 100);

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

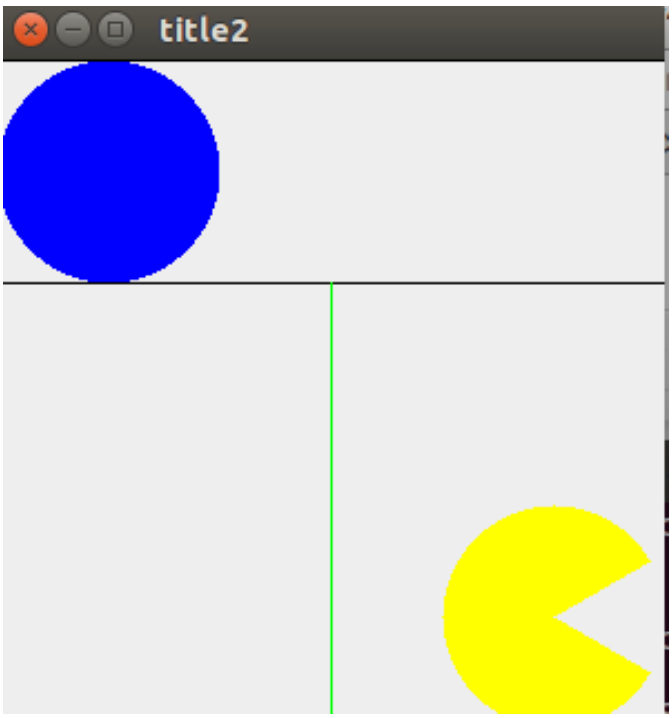
3.3. 図形、画像の表示

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：



それぞれのメソッドについて解説します。

3.3.1. 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。

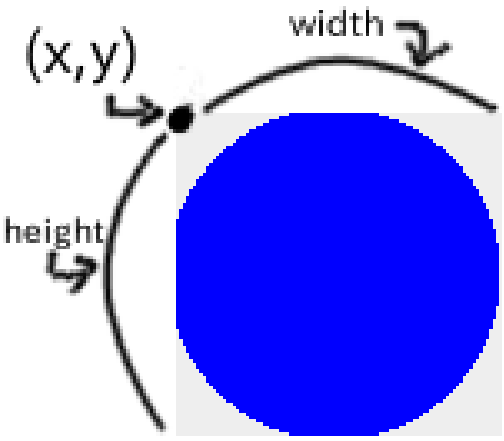
drawRectは四角形の外枠だけが描かれて、fillRectは中が塗りつぶされた四角形を描くことができます。

3.3.2. 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。



3.3.3. 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）。

3.3.4. 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));
```

3.3.5. 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。

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

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。

4. キー入力

この章では、ゲーム作りでも重要な、キー入力をどうすればよいかを学びます。

4.1. 新しいプロジェクトを作ろう

今まで、１つのプロジェクトの中でプログラムをかいてきましたが、ここで、新しく別のプロジェクトを作ってみましょう。

第２章でやったように、ProjectExplorerのところを右クリックして、New→Projectで新しくプロジェクトが作れます。 名前は適当につけてください。

プロジェクトを作ったら、以前のプロジェクトから、2つのファイルをコピー＆ペーストしてきましょう。

次に、以下のコードを、Key.javaとしてクラスを作りましょう。

```
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class Key implements KeyListener{
    static boolean up = false;
    static boolean down = false;
    static boolean right = false;
    static boolean left = false;
    static boolean enter = false;

    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_UP) {
            up = true;
        }
        if (e.getKeyCode() == KeyEvent.VK_DOWN) {
            down = true;
        }
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            left = true;
        }
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            right = true;
        }

        if(e.getKeyCode() == KeyEvent.VK_ENTER) {
            enter = true;
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_UP) {
```

```
            up = false;
        }
        if (e.getKeyCode() == KeyEvent.VK_DOWN) {
            down = false;
        }
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            left = false;
        }
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            right = false;
        }
        if (e.getKeyCode() == KeyEvent.VK_ENTER){
            enter = false;
        }
    }

    @Override
    public void keyTyped(KeyEvent e) {
    }
}
```

このクラスでやっていることは、キーが押されている間、変数がtrueになるということだけです。
例えば、Upキーが押されていると、その間だけ、変数upの値がtrueになります。
そのキーを離すと、変数の値がfalseになります。

Keyクラスが作れたところで、今度はこのクラスの使い方を学びましょう。

今度は、MyJPanel.javaを編集していきます。変更箇所は、while文の中身です。

```
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) {
            g.setColor(Color.WHITE);
            g.fillRect(0,0,width, height);

            if(Key.up){
                g.setColor(Color.BLACK);
                g.drawString("UP!",100,100);
            }
            if(Key.down){
                g.setColor(Color.BLACK);
                g.drawString("DOWN!",100,100);
            }
            if(Key.right){
                g.setColor(Color.BLACK);
                g.drawString("RIGHT",100,100);
            }
            if(Key.left){
                g.setColor(Color.BLACK);
                g.drawString("LEFT",100,100);
            }

            wg.drawImage(img, 0, 0, null);
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

また、mainメソッドの中身も少しだけ追加します。終わりのほうに、`frame.addKeyListener(new Key());`を追加します。

```
import javax.swing.*;

public class SwingSample1 {
    public static void main(String[] args){
        final int height = 600;
        final int width = 600;
        final String titleName = "title";

        JFrame frame = new JFrame(titleName);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(width, height);
        frame.setVisible(true);

        MyJPanel myJPanel = new MyJPanel();
        frame.getContentPane().add(myJPanel);

        myJPanel.setBounds(0,0, width, height);

        frame.addKeyListener(new Key()); // 追加！
        myJPanel.game(width, height);
    }
}
```

addKeyListenerをしないと、キー操作を受け付けないので、追加し忘れないようにしましょう。

ここまでくれば、プログラムがちゃんと実行できるようになると思います。

4.1.1. Keyクラスの使い方

```
if(Key.up){
    g.setColor(Color.BLACK);
    g.drawString("UP!",100,100);
}
```

if(Key.up)で、Upキーが押されているかどうかを判定します。もし押されていれば、括弧の中の処理を実行します。 他のキーでも同じです。

4.1.2. Thread.sleep（_ _） ..zzzZZ

また、while文の終わりの方にThread.sleep(10)などの文が追加されていますね。これは、10ミリ秒だけプログラムの動きを止めます。

```
try {
    Thread.sleep(10);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

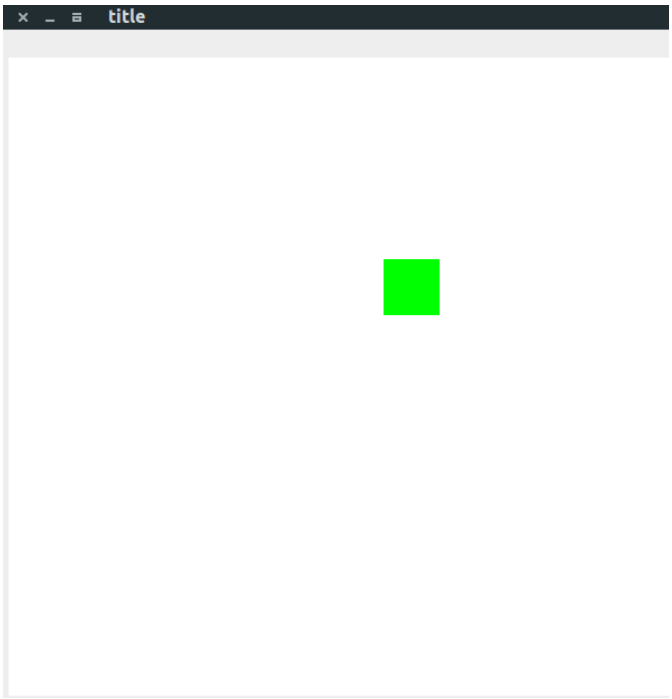
なぜこうするかというと、動きを少しずつ止めていかないと、描画のスピードが追いつかなくなったり、動きが重くなったりするからです。

`Thread.sleep`を囲んでいる`try catch`文は、例外処理というおまじないです。

5. 簡単なゲームを作ってみよう

この章では、今までで学んだ知識を元に、簡単なゲームを作っていきたいと思います。

今回は、以下のような、主人公をキー操作で動かすようなプログラムを作ってみましょう。



5.1. 新しくプロジェクトを作る

この章でも、新しくプロジェクトを作ってゲームを作っていきましょう。
先ほどと同じように、新しいプロジェクトを作ります。次に、先ほど作ったプログラム（３つのJavaファイル）をこのプロジェクトにコピーします。

5.2. 主人公を描いてみる

早速、主人公を描画してみましょう。画像を使って描いてもいいですが、とりあえず、四角形を主人公として描いてみます。

以下のコードを追加してみましょう。追加する場所は、４つif文をかいたところの直後に追加してください。

```
g.setColor(Color.GREEN);
g.fillRect(200,200,50,50);
```

このままでは、主人公が描画されただけで動きませんので、次は主人公を動かすコードをかいてみましょう。

5.3. 主人公を動かす

主人公を動かすにはどうすればいいでしょうか。ずばり、主人公を描画する位置を変えるしかないですね。
位置を変えるには、位置の情報を毎回どこかに保存して、随時それの情報を見ることができるような、”箱

”のようなものが必要です。

その”箱”とは、変数のことです！ 位置を格納する変数xと y を宣言しましょう。

ここでは、while文の直前に宣言します。

```
//while文の上に宣言
int x = 200;
int y = 200;
```

変数を宣言したところで、今度は変数を値を変えられるようにしましょう。例えば、Upキーを押すと、上に動く、みたいな感じです。

これをするために、if文の中をかきかえましょう。

```
            if(Key.up){
                y -= 5;
            }
            if(Key.down){
                y += 5;
            }
            if(Key.right){
                x += 5;
            }
            if(Key.left){
                x -= 5;
            }
```

少し注意することとして、y座標は、上にいくほど値が減っていくので、Upキーが押された時は、yの値を減らしています。

最後に、主人公を描く位置を、xと y にすればOKです。

```
g.setColor(Color.GREEN);
g.fillRect(x,y,50,50);
```

これで無事、動くようになったはずです。

5.3.1. y -= 5 って何？

y -= 5は、y = y - 5と同じ命令です。

では、y = y - 5はどういう命令なのかというと、y - 5した値を、再びyに代入するという処理です。つまり、yの値を5だけ減らしてyを更新します。

5.4. クラスを作ってみよう

このままでも、ゲームは実行できますが、実は、もっとわかりやすくプログラムを記述する方法があります。 クラスというものを使います。

クラスを説明する前に、次のコードを見てみましょう。

```
        while(true){
            g.setColor(Color.WHITE);
            g.fillRect(0,0,width, height);

            //追加！
            player.move();
            player.draw(g);

            wg.drawImage(img, 0, 0, this);
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```

さっきよりも、主人公が何をしているか見やすくなったと思いませんか？

playerがmoveして、さらにplayerをdrawしているのが一目でわかるようになったと思います。

このようなコードを書くためには、`Player`クラスを新たに作る必要があります。

ProjectExplorerのところを右クリックして、New→Classで新しいクラスを作りましょう。クラス名は、Player で。

5.4.1. moveメソッドを定義する。

次に考えるべきことは、Playerが何をするかです。このゲームでプレイヤーは移動をします。つまり、Player は移動するということを定義すればいいわけです。

クラスの中で、移動するなどの動作の定義のことを、メソッドと言います。
では、移動の処理を行う、moveというメソッドを定義してみましょう。

```
public class Player {
    public void move(){

    }
}
```

今度は、moveメソッドの具体的な処理を記述していきましょう。
moveメソッドは、キーが押されたら、その方向にプレイヤーが動けばいいわけです。

```
public class Player {
    public void move(){
        if(Key.up){
            y -= 5;
        }
        if(Key.down){
            y += 5;
        }
        if(Key.right){
            x += 5;
        }
        if(Key.left){
            x -= 5;
        }
    }
}
```

そう言えば、このコードどこかで見たような・・・。

そうです！whileループの中に記述していたコードと全く一緒です！

先ほどかいたwhileループの中のこの処理を、`move`メソッド内にコピペしましょう。

5.4.2. フィールドを追加する

先ほど`move`メソッドを定義した時に、xとyの変数の宣言はコピペせずにやりましたね。では、xとyはどこに宣言すればいいのでしょう？

結論からいうと、以下の場所に宣言します。

```
public class Player {
    int x;
    int y;

    public void move(){
        if(Key.up){
            y -= 5;
        }
        if(Key.down){
            y += 5;
        }
        if(Key.right){
            x += 5;
        }
        if(Key.left){
            x -= 5;
        }
    }
}
```

このxとyは、moveメソッド内に宣言してはいけません。なぜなら、メソッド内に定義すると、その変数はローカル変数となり、メソッドの実行が終了するたびに変数の状態が消えてしまうからです。

これを防ぐため、メソッドの外に定義します。このメソッドの外に定義された変数のことを、フィールドと呼びます。 こうすれば、Playerはきちんと位置の情報を保持することが可能になります。

5.4.3. コンストラクタを作る

ここで、フィールドを見てみると、フィールドが初期化されていませんね。
フィールドの初期化は、コンストラクタというものを使います。

Playerクラス内に以下のように定義します。

```
Player(int x, int y){
    this.x = x;
    this.y = y;
}
```

this.xは、フィールドのxを指し、ただのxは、コンストラクタの引数に渡ってきたxです。

他のクラスから使う時に、xとyの値を指定してPlayerを作ることができます。

5.4.4. drawメソッドを定義する。

moveメソッドを定義したように、今度はdrawメソッドも定義してみましょう。また、whileループの中の一部分をコピペすれば良いですね。

```
public class Player {
    int x;
    int y;

    Player(int x, int y){
        this.x = x;
        this.y = y;
    }

    public void move(){
        if(Key.up){
            y -= 5;
        }
        if(Key.down){
            y += 5;
        }
        if(Key.right){
            x += 5;
        }
        if(Key.left){
            x -= 5;
        }
    }

    // 追加！
    public void draw(Graphics g){
        g.setColor(Color.GREEN);
        g.fillRect(x,y,50,50);
    }
}
```

これで、drawメソッドが出来上がりました！

さあ、あとは、MyJPanelの方を少し直して完成です。

5.4.5. MyJPanelの修正

whileループの中を以下のように修正します。

```
        while(true){
            g.setColor(Color.WHITE);
            g.fillRect(0,0,width, height);

            //追加！
            player.move();
            player.draw(g);

            wg.drawImage(img, 0, 0, this);
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```

でも、このままでは実行できません。実は、クラスを使うには、クラスのインスタンスというものを作る必要があります。

とりあえず、このクラスのフィールドに、

```
Player player = new Player(0,0);
```

のようなものを追加してみましょう。

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

public class MyJPanel2 extends JPanel {
    Player player = new Player(0,0); //追加

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

        while(true){
            g.setColor(Color.WHITE);
            g.fillRect(0,0,width, height);

            //追加！
            player.move();
            player.draw(g);

            wg.drawImage(img, 0, 0, this);
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

`new Player(0,0)`とすれば、初期座標(0,0)の`Player`を作ることができます。

もちろん、`new Player(200,200)`とすれば、初期座標(200,200)の`Player`を作ることができます。

こうすれば完成です！
あとは、自分で色々試してみるのもありですし、画像を読み込んで遊んでみても面白いと思います。