第4章 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。

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、Project Explorer`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。

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。

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。