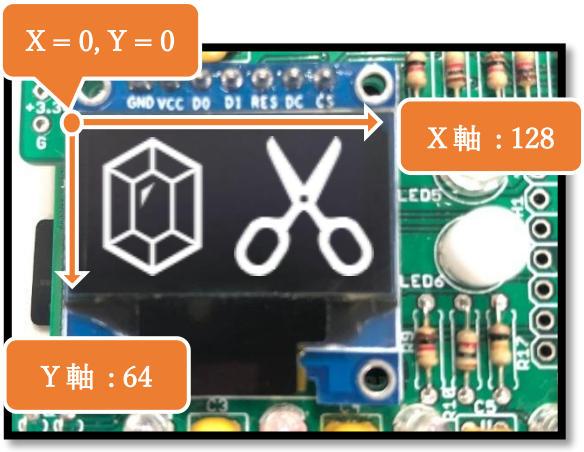
有機 EL（Organic Light Emitting Diode）とは

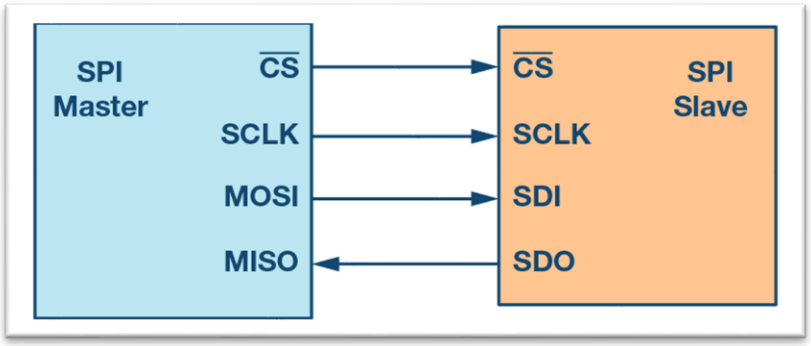


有機 EL（OLED）とは、電流を流すと自ら発光する発光素子のこと。テレビや照明など様々な電子製品に利用されており、OLED も言わば製品のひとつである。拡張ボードには 1 個搭載されており、SPI 通信を使用して 128×64 ドットのディスプレイに単色の文字や画像を表示することができる。

SPI（Serial Peripheral Interface）通信とは

SPI 通信とは、クロックに同期させてデータの通信を行う同期式シリアル通信のひとつ。以下の 4 本の信号線で通信する為、数 Mbps の通信が可能である。

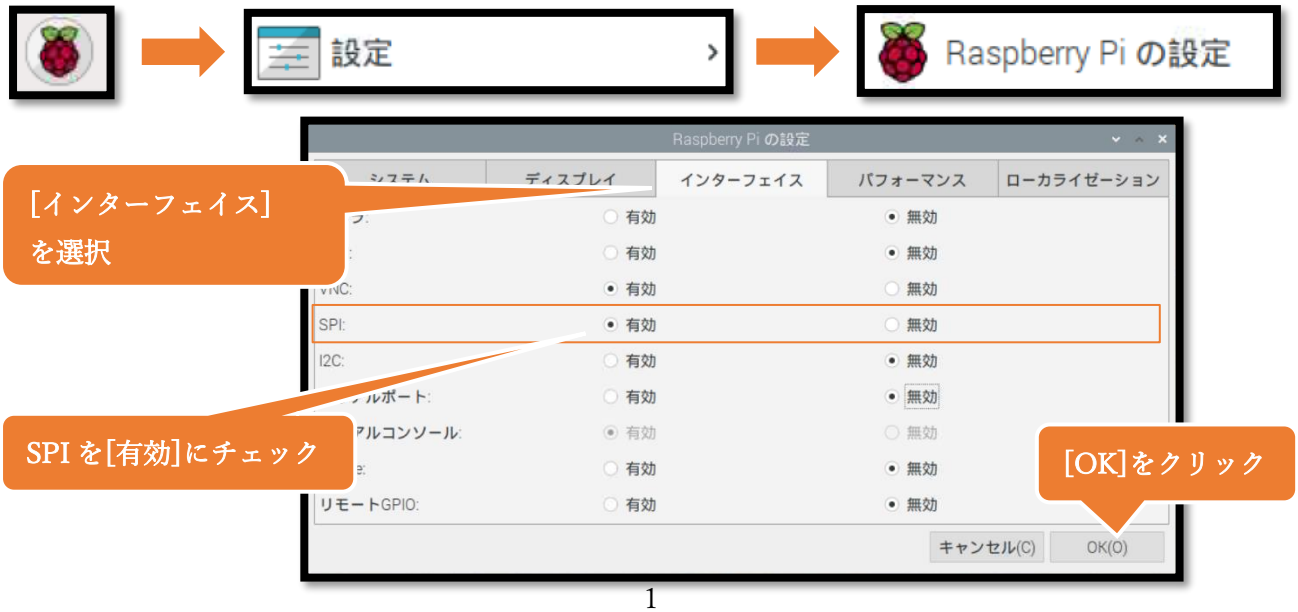
・チップセレクト（CS）
・クロック（SCLK）
・データ入力（SDI）
・データ出力（SDO）



SPI 通信上ではクロック信号を生成する側（Raspberry Pi）をマスタと言い、マスタに対してデータを送受信する側（OLED）をスレーブと言う。1 つのマスタに対して複数のスレーブとやり取りが行える。

SPI 通信の有効化

Raspberry Pi で SPI 通信を使用する場合、有効化の設定を行う必要がある。



練習（TryDisplay.java）

以下のサンプルプログラムを記述して実行し、実行時にディスプレイの右側にメニューが表示され、
SW1 を押したときにグー（石）の画像がディスプレイの左側に表示され、
SW4 を押したときにディスプレイの左側（グーの画像）がクリアされるか確認しなさい。

```
/*
 * TryDisplay.java
 * Date    : 2022/01/01
 * Author : IE1A 99 K.Murakami
 */

// GPIO ピンを利用するために必要なクラスを読み込む
import com.pi4j.wiringpi.Gpio;
import com.pi4j.wiringpi.GpioUtil;
// SPI 通信やディスプレイ描画に必要なクラスを読み込む
import com.pi4j.io.spi.SpiChannel;
import com.pi4j.io.spi.SpiFactory;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.RaspiPin;
import eu.ondryaso.ssd1306.Display;
import javax.imageio.ImageIO;
import java.awt.*;
import java.io.IOException;

public class TryDisplay {
    // Thread.sleep メソッドで発生する割り込み例外を throws する
    public static void main (String[] args) throws IOException, InterruptedException {

        System.out.println("プログラム開始");

        // デジタル出力信号を定数化
        final int HIGH = 1;
        final int LOW = 0;
        // デジタル入力信号を定数化
        final int ON = 0;
        final int OFF = 1;
```

インポートが多数必要なので忘れずに

```
// LED のピン番号を宣言
final int LED1 = 15;
final int LED4 = 27;
// SW のピン番号を宣言
final int SW1 = 7;
final int SW4 = 22;

// GPIO を初期化
Gpio.wiringPiSetup();
System.out.println("GPIO 初期化完了");

// 各 LED を出力に設定
Gpio.pinMode(LED1, Gpio.OUTPUT);
Gpio.pinMode(LED4, Gpio.OUTPUT);
// 各 SW を入力に設定
Gpio.pinMode(SW1, Gpio.INPUT);
Gpio.pinMode(SW4, Gpio.INPUT);
System.out.println("GPIO 入出力設定完了");

// ディスプレイを初期化
Display disp = new Display(128, 64, GpioFactory.getInstance(),
                           SpiFactory.getInstance(SpiChannel.CS0, 8000000),
                           RaspiPin.GPIO_05, RaspiPin.GPIO_04);
disp.begin();
System.out.println("ディスプレイ初期化完了");

// ディスプレイの書式設定
disp.getGraphics().setFont(new Font("Serif", Font.PLAIN, 12));
disp.getGraphics().setColor(Color.WHITE);
// 文字を設定（文字列, X 座標, Y 座標）
disp.getGraphics().drawString("１：グー", 65, 10);
disp.getGraphics().drawString("２：", 65, 25);
disp.getGraphics().drawString("３：", 65, 40);
disp.getGraphics().drawString("４：クリア", 65, 55);
disp.displayImage();     //ディスプレイ描画

// 画像オブジェクトの作成
Image rImg = ImageIO.read(TryDisplay.class.getResourceAsStream("./image/rock.png"));
```

使用する画像のパスを指定する

```
// 各 SW の状態を保持する変数
int sw1Data, sw4Data;
// 前回の各 SW の状態を保持する変数（初期値は OFF）
int sw1DataOld = OFF;
int sw4DataOld = OFF;

// プログラムを終了させない為に無限ループする
while(true){
    // 現在の各 SW の状態を読み取る
    sw1Data = Gpio.digitalRead(SW1);
    sw4Data = Gpio.digitalRead(SW4);

    // 現在の SW の状態が、前回の SW の状態と違う（変化があった）場合
    if(sw1Data != sw1DataOld){
        if(sw1Data == ON){
            Gpio.digitalWrite(LED1, HIGH);  //LED 点灯

            // 画像を設定（画像データ, X 座標, Y 座標, null）
            disp.getGraphics().drawImage(rImg, 0, 0, null);
            disp.displayImage();    //ディスプレイ描画
        }else{
            Gpio.digitalWrite(LED1, LOW);   //LED 消灯
        }
    }

    // 現在の SW の状態が、前回の SW の状態と違う（変化があった）場合
    if(sw4Data != sw4DataOld){
        if(sw4Data == ON){
            Gpio.digitalWrite(LED4, HIGH);  //LED 点灯

            // 指定範囲をクリア（X 座標, Y 座標, X の幅, Y の高さ）
            disp.getGraphics().clearRect(0, 0, 64, 64);
            disp.displayImage();    //ディスプレイ描画
        }else{
            Gpio.digitalWrite(LED4, LOW);   //LED 消灯
        }
    }
```

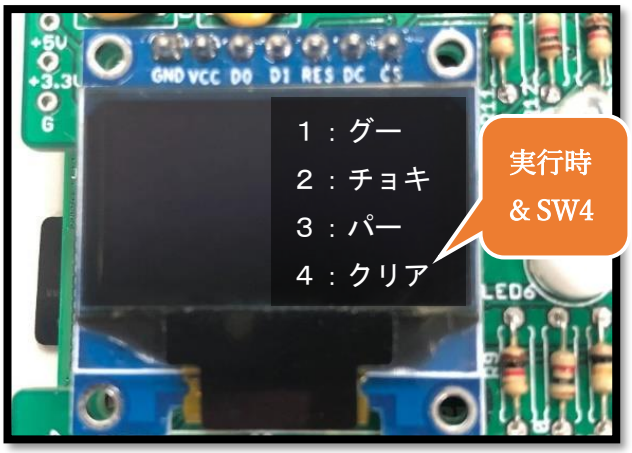
ディスプレイのクリアは「クリアしたデータを描画する」という考え方

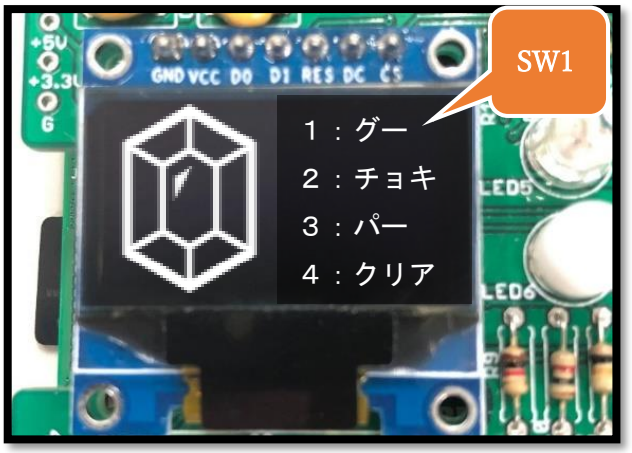
```
                // 次回の為に現在の各 SW の状態を保存する
                sw1DataOld = sw1Data;
                sw4DataOld = sw4Data;

                // 100 ミリ秒待機する
                Thread.sleep(100);
            }
        }
}
```

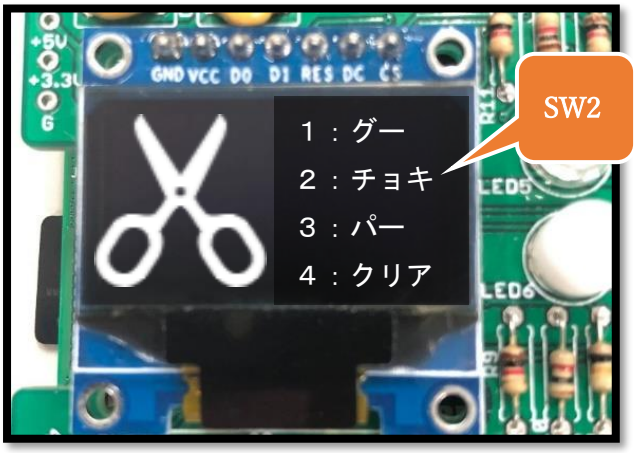
課題 1 （DisplayJanken.java）

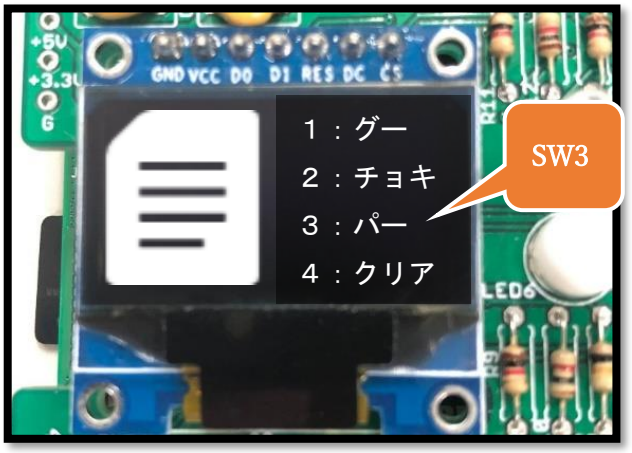
前述の練習を別名保存し、以下の条件でじゃんけんの手を表示するプログラムを追記しなさい。

・SW1 → 押している間は座標（0, 0）にグーの画像を表示し、LED1 を点灯する。離すと消灯する。
・SW2 → 押している間は座標（0, 0）にチョキの画像を表示し、LED2 を点灯する。離すと消灯する。
・SW3 → 押している間は座標（0, 0）にパーの画像を表示し、LED3 を点灯する。離すと消灯する。
・SW4 → 練習のまま。押している間は画像（左側）をクリアし、LED4 を点灯する。離すと消灯する。
※画像を表示した状態で次の画像を表示した場合は重複されてしまう為、間にクリア処理を挟むこと。



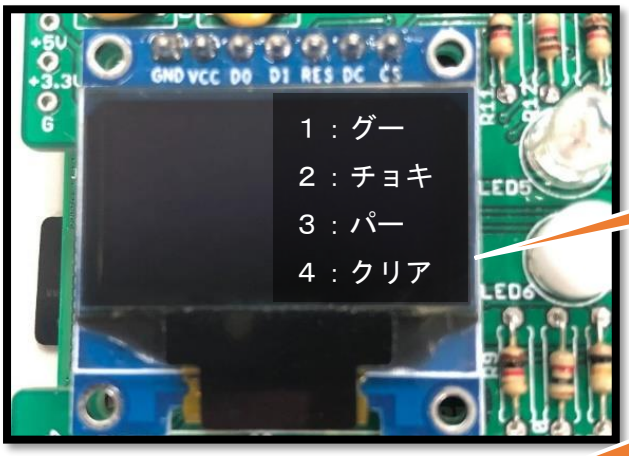




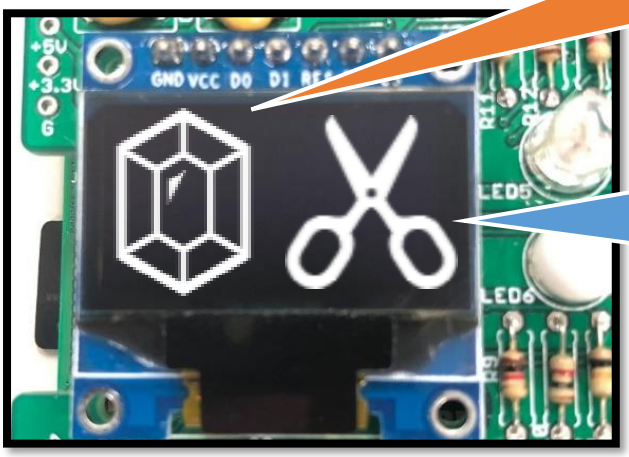


課題 2 （DisplayBattle.java）

前述の課題 1 を別名保存し、以下の条件でコンピュータとじゃんけんを行う処理を追加・修正しなさい。
【SW1・2・3 を押した場合】
画面の左側には前述と同様に自分が押した SW（1：グー・2：チョキ・3：パー）の手の画像を表示し、右側にはコンピュータの手の画像をランダムで表示する。
【SW4 を押した場合】
実行時の状態にクリアする（画面の左側は何もなし、右側は SW のメニュー文字列を表示する）



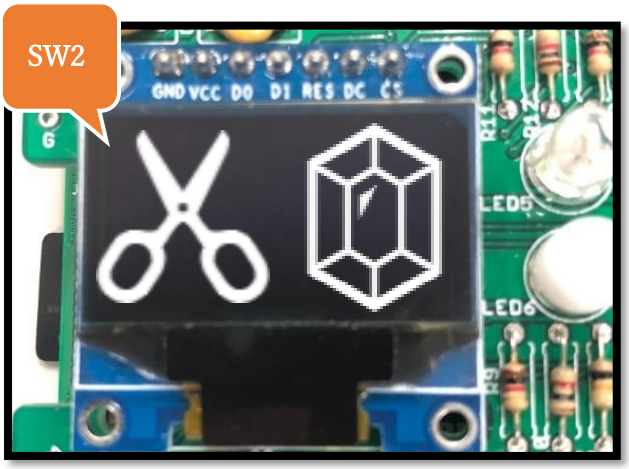
実行時、または SW4 を押した場合は
この画面の状態にクリアする



左側（自分）は前述の課題と同様で、
押した SW に対応するグー・チョキ・パーの
画像を表示する

右側（コンピュータ）は SW1・2・3 が
押された場合にグー・チョキ・パーの
いずれかの画像をランダムに表示する

SW2



SW3