14 回目　センサーを用いたインタラクティブプログラミング 1

今日の講義で学ぶ内容

・センサーでボタンを選択してクリック

センサーでボタンを選択してクリック

§1　まずはボタンを配置してボタンのイベント処理をしましょう

これまでどおり、ボタンなどの GUI 部品をイベントハンドラの宣言と登録を行いましょう。

ソースファイル名：Sample14_1.java

```
// ※HP よりインポート文をここへ貼り付けてください

// センサーでボタンを選択してクリック１
public class Sample14_1 extends Application
{
   private Button bt1, bt2;   // ボタン

   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      bt1 = new Button("霜の花 クリック！");
      bt2 = new Button("紅茶 クリック！");
      bt1.setGraphic(new ImageView("frost.jpg"));
      bt1.setTextFill(Color.RED);
      bt1.setFont(new Font(24));
      bt2.setGraphic(new ImageView("tea.jpg"));
      bt2.setTextFill(Color.RED);
      bt2.setFont(new Font(24));
      bt1.requestFocus();

      // ボタンにイベントハンドラを設定します
      ButtonEventHandler bh = new ButtonEventHandler();
      bt1.addEventHandler(ActionEvent.ANY, bh);
      bt2.addEventHandler(ActionEvent.ANY, bh);

      // レイアウト HBox を生成／設定します
      HBox hb = new HBox();
      ObservableList<Node> lst = hb.getChildren();
      lst.add(bt1);
      lst.add(bt2);
      hb.setPadding(new Insets(10));
      hb.setSpacing(15);

      // シーンを生成／設定します
```

```
        Scene scene = new Scene(hb);

        // ステージを設定します
        stage.setScene(scene);
        stage.setTitle("センサーを傾けてクリック！");

        // ステージを表示します
        stage.show();
    }

    // ボタン用イベントリスナー
    private class ButtonEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent ac){

            Button src=(Button)ac.getTarget();
            String name=src.getText();
            System.out.println(name);
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```

実行結果

```
霜の花 クリック！                    ← 霜の花ボタンをクリックします
霜の花 クリック！                    ← 霜の花ボタンをクリックします
紅茶 クリック！                      ← 紅茶ボタンをクリックします
        :
```

§2　次にセンサーの準備と重力データの取得を行いましょう

次に、センサー周りの準備を加えていきます。

ソースファイル名：Sample14_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// センサーでボタンを選択してクリック２
public class Sample14_2 extends Application
{
   private SpatialPhidget sp; // センサー
   private Button bt1, bt2;   // ボタン

   public void start(Stage stage) throws Exception
   {
      // センサーを準備します
      sp = new SpatialPhidget();

      // センサーにイベントハンドラを登録します
      sp.addSpatialDataListener(new mySpatialDataListener());

      // センサーをオープンします
      sp.openAny();

      // ボタンを生成／設定します
      bt1 = new Button("霜の花 クリック！");
      bt2 = new Button("紅茶 クリック！");
      bt1.setGraphic(new ImageView("frost.jpg"));
      bt1.setTextFill(Color.RED);
      bt1.setFont(new Font(24));
      bt2.setGraphic(new ImageView("tea.jpg"));
      bt2.setTextFill(Color.RED);
      bt2.setFont(new Font(24));
      bt1.requestFocus();

      // ボタンにイベントハンドラを設定します
      ButtonEventHandler bh = new ButtonEventHandler();
      bt1.addEventHandler(ActionEvent.ANY, bh);
      bt2.addEventHandler(ActionEvent.ANY, bh);

      // レイアウト HBox を生成／設定します
      HBox hb = new HBox();
      ObservableList<Node> lst = hb.getChildren();
      lst.add(bt1);
      lst.add(bt2);
      hb.setPadding(new Insets(10));
      hb.setSpacing(15);

      // シーンを生成／設定します
      Scene scene = new Scene(hb);
```

```
        // ステージを設定します
        stage.setScene(scene);
        stage.setTitle("センサーを傾けてクリック！");

        // ステージを表示します
        stage.show();
    }

    public void stop() throws Exception
    {
        // センサーをクローズします
        sp.close();
    }

    // ボタン用イベントリスナー
    private class ButtonEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent ac){

            Button src=(Button)ac.getTarget();
            String name=src.getText();
            System.out.println(name);
        }
    }

    // イベントハンドラ（センサーからデータを受け取ったとき）
    private class mySpatialDataListener implements SpatialDataListener
    {
        public void data(SpatialDataEvent e)
        {
            // 重力データを取り出します
            SpatialEventData[] set = e.getData();
            double[] data = set[0].getAcceleration();

            System.out.println("x="+data[0]+" Y="+data[1]+" z="+data[2]);
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```

実行結果

```
            :
x=0.12549 Y=0.01074 z=1.00098
x=0.12695 Y=0.00635 z=0.99902
霜の花 クリック！                    ← 霜の花ボタンをクリックします
x=0.12793 Y=0.00439 z=0.99658
            :
```

§3　センサーを傾けてボタンのフォーカスを変えてみましょう

センサーが左へ傾いたら左のボタンにフォーカスが移動するようにしてみましょう。

ソースファイル名：Sample14_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// センサーでボタンを選択してクリック３
public class Sample14_3 extends Application
{
   private SpatialPhidget sp; // センサー
   private Button bt1, bt2;   // ボタン

   public void start(Stage stage) throws Exception
   {
      // センサーを準備します
      sp = new SpatialPhidget();

      // センサーにイベントハンドラを登録します
      sp.addSpatialDataListener(new mySpatialDataListener());

      // センサーをオープンします
      sp.openAny();

      // ボタンを生成／設定します
      bt1 = new Button("霜の花 クリック！");
      bt2 = new Button("紅茶 クリック！");
      bt1.setGraphic(new ImageView("frost.jpg"));
      bt1.setTextFill(Color.RED);
      bt1.setFont(new Font(24));
      bt2.setGraphic(new ImageView("tea.jpg"));
      bt2.setTextFill(Color.RED);
      bt2.setFont(new Font(24));
      bt1.requestFocus();

      // ボタンにイベントハンドラを設定します
      ButtonEventHandler bh = new ButtonEventHandler();
      bt1.addEventHandler(ActionEvent.ANY, bh);
      bt2.addEventHandler(ActionEvent.ANY, bh);

      // レイアウト HBox を生成／設定します
      HBox hb = new HBox();
      ObservableList<Node> lst = hb.getChildren();
      lst.add(bt1);
      lst.add(bt2);
      hb.setPadding(new Insets(10));
      hb.setSpacing(15);

      // シーンを生成／設定します
      Scene scene = new Scene(hb);
```

```
    // ステージを設定します
    stage.setScene(scene);
    stage.setTitle("センサーを傾けてクリック！");

    // ステージを表示します
    stage.show();
}

public void stop() throws Exception
{
    // センサーをクローズします
    sp.close();
}

// ボタン用イベントリスナー
private class ButtonEventHandler implements EventHandler<ActionEvent>
{
    public void handle(ActionEvent ac){

        Button src=(Button)ac.getTarget();
        String name=src.getText();
        System.out.println(name);
    }
}

// イベントハンドラ（センサーからデータを受け取ったとき）
private class mySpatialDataListener implements SpatialDataListener
{
    public void data(SpatialDataEvent e)
    {
        // 重力データを取り出します
        SpatialEventData[] set = e.getData();
        double[] data = set[0].getAcceleration();

        if(bt1!=null && bt2!=null)
        {
            // x 軸傾きに応じてボタンのフォーカスを変更する
            if(data[0] > 0.4)
            {
                if(bt1.isFocused()==false)
                    Platform.runLater(new Button1Focus());
            }
            else if(data[0] < -0.4)
            {
                if(bt2.isFocused()==false)
                    Platform.runLater(new Button2Focus());
            }
        }
    }
}

// JavaFX アプリケーションスレッドで実行したい命令１
private class Button1Focus implements Runnable
{
    public void run(){
        bt1.requestFocus();
    }
```

```
    }

    // JavaFX アプリケーションスレッドで実行したい命令２
    private class Button2Focus implements Runnable
    {
       public void run(){
          bt2.requestFocus();
       }
    }

    public static void main(String[] args)
    {
       launch(args);
    }
}
```

実行結果

■JavaFX アプリケーションスレッド

JavaFX アプリケーションは複数のスレッドで動作します。代表的なスレッドとそのスレッドが実行するメソッドの例を下に示します。

1. Main スレッド　　main()メソッドと launch()メソッド
2. JavaFX アプリケーションスレッド　　start()メソッドと stop()メソッド

〔開始〕→ main() → launch() → start() →〔実行中〕→ stop() →〔終了〕

※〔実行中〕の GUI 部品のイベントハンドラは JavaFX アプリケーションスレッドにより実行されます。

規則：GUI 部品へのアクセス（ボタンを生成／設定したり、ボタン名などの状態を変更したりするなど）は JavaFX アプリケーションスレッドからのみ許されます。

■センサーのイベントハンドラを実行するスレッドは?

センサーの接続や切断、データーの取得を行うイベントハンドラは JavaFX アプリケーションスレッドとは別のスレッドで実行されます。これはセンサーのライブラリが JavaFX の外部のものであることが理由として考えられます。

■センサーのイベントハンドラ内で GUI 部品にアクセスするには?

実行したいコードを Platform クラスのクラスメソッド runLater()メソッドに引数として渡し、JavaFX アプリケーションスレッドで実行するように登録します。

まずは、JavaFX アプリケーションスレッドで実行したい処理を宣言します。

1. Runnable インタフェースを実装してサブクラスを宣言

2. 継承される void run();メソッドをオーバーライドして処理を記述

〔コード例〕

```
1.   class Sub implements Runnable
2.   {
3.      public void run(){
4.         // ここに JavaFX アプリケーションスレッドで実行したい処理を記述します
5.      }
6.   }
```

次に、このクラスのオブジェクトを生成し、クラス Platform を用いてこのオブジェクトを登録します。

```
Platform.runLater(new Sub());
```

※クラス Sub の中の run()メソッドが JavaFX アプリケーションスレッドで実行されます。

■利用したクラス／インタフェースの一覧

Button クラス	
boolean isFocused(){…}	ボタンにフォーカスがあるかどうか調べます。
void requestFocus(){…}	このボタンにフォーカスを合せます。
Platform クラス	
static void runLater(Runnable r){…}	JavaFX アプリケーションスレッドで実行したいコードを登録します。