4 回目　いろいろなレイアウトを試してみよう

今日の講義で学ぶ内容

・レイアウトの利用
・様々なレイアウト
・レイアウトの重ね合わせ

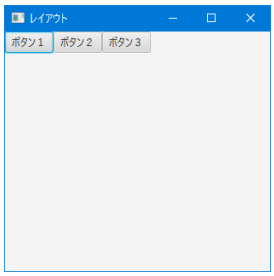
レイアウトの利用

§1　ボタンをレイアウトに配置してみましょう。

レイアウトを用いると、ボタンなどの GUI 部品をきれいに配置できます。

ソースファイル名：Sample4_1.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ボタンをレイアウトに配置
public class Sample4_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[3];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");

      // レイアウトを生成／設定します
      FlowPane flowpane = new FlowPane();
      ObservableList<Node> lst = flowpane.getChildren();
      lst.addAll(btn);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト");

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

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



■ボタンを表現するクラス

ボタンはクラス Button により表現され、ボタンを設定するメソッドが準備されています。

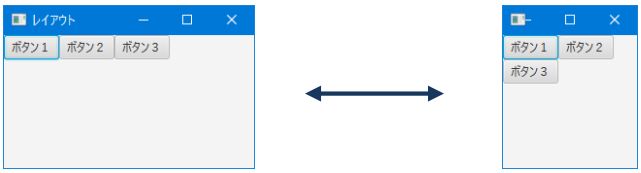
・ボタンの生成と名前　　→ new Button("ボタン１");

※この他、名前のフォントやイメージボタンなどボタンを設定するメソッドが多数準備されています。

「ボタン１」という名前のボタンが生成されます。

■レイアウト FlowPane

このレイアウトはボタンなどの GUI 部品を横方向優先で配置します。横方向に収まりきらない場合は自動的に改行され、次の行へ配置されます。



■レイアウト FlowPane にボタンを配置するには

レイアウトに配置されている GUI 部品のリストを取得し、そのリストにボタンを追加します。

〔コード例〕

```
1. FlowPane flowpane = new FlowPane();                    // レイアウトの生成
2. ObservableList<Node> lst = flowpane.getChildren();     // GUI 部品リストの取得
3. lst.addAll(btn);                                       // ボタンの追加
```

※Button クラスの配列 btn を GUI 部品リストに追加します

※ボタンを個別に追加するときは lst.add(…); を用います

■ジェネリクスとは

引数を用いてクラスの宣言を行います。クラスを使用するときは引数に値（クラス名）を入れます。

```
class Data<E>{
   E value;
}
```

```
Data<String> title=new Data<String>();
title.value = "HCI プログラミング";
Data<Integer> price=new Data<Integer>();
price.value = 108;
```

■利用したクラスの一覧

FlowPane クラス ← Pane クラス ← Region クラス ← Parent クラス ← Node クラス ← Object クラス

ObservableList<Node> getChildren(){…}　GUI 部品リストを取得します。

ObservableList<Node>インタフェース

boolean addAll(Node[] e){…}	新しい部品群 e を GUI 部品リストに追加します。 ※GUI 部品リストが更新されたら true が戻ります。 ※Node クラスは Button クラスのスーパークラスです。
Boolean add(Node e){…}	新しい部品 e を GUI 部品リストに追加します。

§2　ボタンのサイズを変更してみましょう。

ボタンは Button クラスにより表現されます。Button クラスのメソッドを用いてボタンの設定を行うことができます。

ソースファイル名：Sample4_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ボタンの推奨サイズ
public class Sample4_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[3];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");
      btn[0].setPrefSize(200, 100);
      btn[1].setPrefSize(100, 200);
      btn[2].setPrefSize(200, 200);

      // レイアウトを生成／設定します
      FlowPane flowpane = new FlowPane();
      ObservableList<Node> lst = flowpane.getChildren();
      lst.addAll(btn);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト");

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

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

■ボタンのサイズを変更するには

ボタンを表現する Button クラスにボタンのサイズを変更するメソッドが準備されています。

・ボタンのサイズ　　→ setPrefSize(200, 100);

横幅が 200 ピクセル、縦幅が 100 ピクセルのボタンになります。

§3　レイアウト周りに空白エリアを入れてみましょう

レイアウトの各種設定はレイアウトクラスのメソッドを用いて行うことができます。

ソースファイル名：Sample4_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// レイアウト周りの空白エリア
public class Sample4_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[3];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");

      // レイアウトを生成／設定します
      FlowPane flowpane = new FlowPane();
      ObservableList<Node> lst = flowpane.getChildren();
      lst.addAll(btn);
      flowpane.setPadding(new Insets(20));

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト");

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

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

■レイアウト周りに空白エリアを入れるには

FlowPane クラスに空白エリアを指定するメソッドが準備されています。

・周辺の空白エリア　　→ setPadding(new Insets(20));

レイアウトの上下左右に 20 ピクセルの空白エリアが入ります。Insets クラスは上下左右の微調整を表現するクラスです。new Insets(10,20,25,15);で上 10 右 20 下 25 左 15 の微調整を指定できます。

§4　ボタン間に空白エリアを入れてみましょう

レイアウトクラスのメソッドを用いてボタン間にも空白エリアを入れることができます。

ソースファイル名：Sample4_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 各ボタン周りの空白エリア
public class Sample4_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[3];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");

      // レイアウトを生成／設定します
      FlowPane flowpane = new FlowPane();
      ObservableList<Node> lst = flowpane.getChildren();
      lst.addAll(btn);
      flowpane.setPadding(new Insets(20));
      flowpane.setHgap(10);
      flowpane.setVgap(10);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト");

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

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

■ボタン同士の間に空白エリアを入れるには

FlowPane クラスにボタン間に空白エリアを指定するメソッドが準備されています。

- 横隣りの空白エリア　→ setHgap(10);
- 縦隣りの空白エリア　→ setVgap(10);

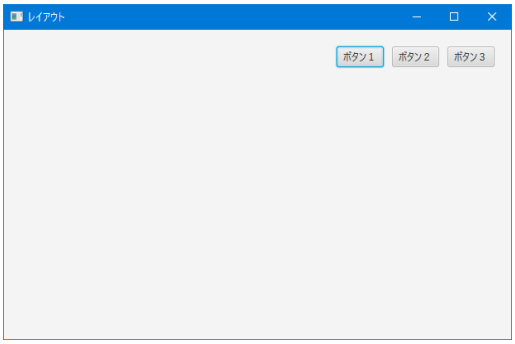
ボタンの横隣りに 10 ピクセル、縦隣りに 10 ピクセルの空白エリアが入ります。

§5　レイアウトの位置を指定してみましょう

GUI 部品の全体的なレイアウトの位置を右上や中央、左下などに変更することができます。

ソースファイル名：Sample4_5.java

```
// ※HP よりインポート文をここへ貼り付けてください

// レイアウト内のボタンの位置合わせ
public class Sample4_5 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[3];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");

      // レイアウトを生成／設定します
      FlowPane flowpane = new FlowPane();
      ObservableList<Node> lst = flowpane.getChildren();
      lst.addAll(btn);
      flowpane.setPadding(new Insets(20));
      flowpane.setHgap(10);
      flowpane.setVgap(10);
      flowpane.setAlignment(Pos.TOP_RIGHT);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト");

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

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



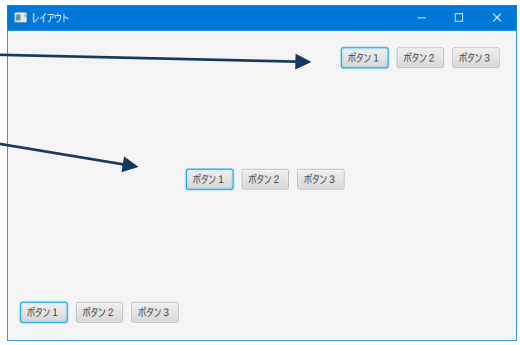
■レイアウトの位置を指定するには

FlowPane クラスに GUI 部品の全体的なレイアウトの位置を指定するメソッドが準備されています。

・レイアウトの位置（右上の場合）　　→ setAlignment(Pos.TOP_RIGHT);

レイアウトが右上に置かれます。この他、以下の場所に置くこともできます。

Pos.TOP_CENTER	上
Pos.TOP_LEFT	左上
Pos.TOP_RIGHT	右上
Pos.CENTER	中央
Pos.CENTER_LEFT	左
Pos.CENTER_RIGHT	右
Pos.BOTTOM_CENTER	下
Pos.BOTTOM_LEFT	左下
Pos.BOTTOM_RIGHT	右下



■列挙型とは

特別なクラスです。この列挙型の変数には列挙されている値のみ代入が可能です。

```
enum Gender{
   Male,        // 代入可能な値をカンマで区切って列挙します（暗黙的に、クラス変数です）
   Female;
}
```

```
Gender me = Gender.Male;
Gender you = Gender.Female;
```

■利用したクラスの一覧

Button クラス ←… Control クラス ← Region クラス ←Parent クラス Node クラス ← Object クラス

void setPrefSize(double w, double h){…} ボタンのサイズを横 w×縦 h に設定します。

FlowPane クラス ← Pane クラス ← Region クラス ← Parent クラス ← Node クラス ← Object クラス

void setPadding(Insets v){…}	レイアウト周りの空白エリアを v に指定します。
void setHgap(double v){…}	GUI 部品の横隣りの空白エリアを v に指定します。
void setVgap(double v){…}	GUI 部品の縦隣りの空白エリアを v に指定します。
void setAlignment(Pos v){…}	GUI 部品の全体的なレイアウトを v に指定します。

Insets クラス ← Object クラス

Insets(double d){…}	上下左右の微調整幅を d にします。（コンストラクタ）
Insets(double top, double right, double bottom, double left){…}	（コンストラクタ） 上下左右の微調整幅をそれぞれ指定します。

Pos 列挙型 ← Enum<Pos>クラス ← Object クラス

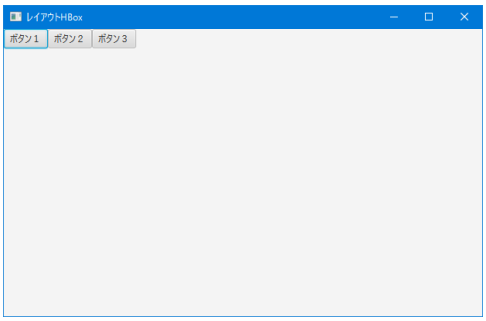
Pos.TOP_RIGHT	右上を指定します。

§6　レイアウト HBox を使ってみましょう

レイアウトには FlowPane の他に HBox や VBox、BorderPane、GridPane があります。レイアウト HBox はボタンなどの GUI 部品を横方向一列に配置します。

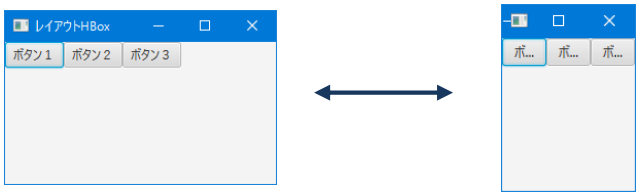
ソースファイル名：Sample4_6.java

```
// ※HP よりインポート文をここへ貼り付けてください

// レイアウト HBox
public class Sample4_6 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[3];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");

      // レイアウト HBox を生成／設定します
      HBox hb = new HBox();
      ObservableList<Node> lst = hb.getChildren();
      lst.addAll(btn);

      // 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("レイアウト HBox");

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

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



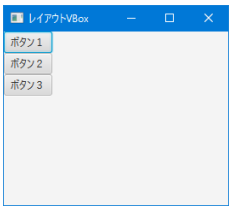
■レイアウト HBox

このレイアウトはボタンなどの GUI 部品を横方向一列に配置します。



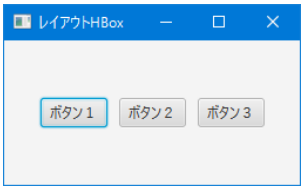
■レイアウト VBox

このレイアウトはボタンなどの GUI 部品を縦方向一列に配置します。



■レイアウト HBox/VBox の空白エリアの設定

- レイアウト周りの空白エリア → `setPadding(new Insets(20));`
- ボタンなど GUI 部品間の空白エリア → `setSpacing(10);`
  ※setHgap()/setVgap()と同じように使用
- ボタンなど GUI 部品の全体的なレイアウトの位置 → `setAlignment(Pos.CENTER);`



■利用したクラスの一覧

HBox クラス ← Pane クラス ← Region クラス ←Parent クラス Node クラス ← Object クラス	
HBox(){…}	レイアウト HBox を生成します。（コンストラクタ）
ObservableList<Node> getChildren(){…}	GUI 部品リストを取得します。
void setSpacing(double d){…}	GUI 部品の横隣りの空白エリアを指定します。
VBox クラス ← Pane クラス ← Region クラス ←Parent クラス Node クラス ← Object クラス	
VBox(){…}	レイアウト VBox を生成します。（コンストラクタ）
ObservableList<Node> getChildren(){…}	GUI 部品リストを取得します。
void setSpacing(double d){…}	GUI 部品の縦隣りの空白エリアを指定します。

§7　レイアウト BorderPane を使ってみましょう（1）

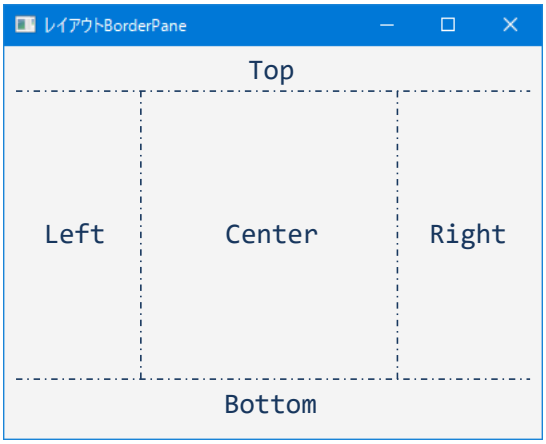
レイアウト BorderPane はボタンなどの GUI 部品を上下左右および中央に配置します。

ソースファイル名：Sample4_7.java

```
// ※HP よりインポート文をここへ貼り付けてください

// レイアウト BorderPane
public class Sample4_7 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[5];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");
      btn[3] = new Button("ボタン４");
      btn[4] = new Button("ボタン５");

      // レイアウト BorderPane を生成／設定します
      BorderPane bp = new BorderPane();
      bp.setTop(btn[0]);
      bp.setLeft(btn[1]);
      bp.setCenter(btn[2]);
      bp.setRight(btn[3]);
      bp.setBottom(btn[4]);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト BorderPane");

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

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

■レイアウト BorderPane

このレイアウトはボタンなどの GUI 部品を上領域（Top）下領域（Bottom）左領域（Left）右領域（Right）および中央領域（Center）に配置します。



■レイアウト BorderPane にボタンを配置するには

各領域にボタンなどの GUI 部品を配置するメソッドが BorderPane クラスに準備されています。

- 上領域（Top）へ配置　→ setTop(btn[0]);
- 左領域（Left）へ配置　→ setLeft(btn[1]);
- 中央領域（Center）へ配置　→ setCenter(btn[2]);
- 右領域（Right）へ配置　→ setRight(btn[3]);
- 下領域（Bottom）へ配置　→ setBottom(btn[4]);

各ボタンが指定された領域に配置されます。

■利用したクラスの一覧

BorderPane クラス ← Pane クラス ← Region クラス ←Parent クラス Node クラス ← Object クラス

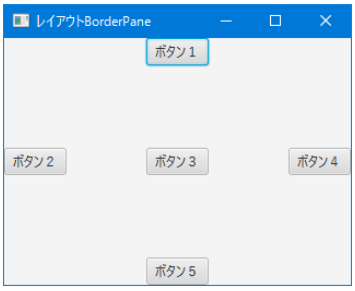
BorderPane(){…}	レイアウト BorderPane を生成します。（コンストラクタ）
void setTop(Node n){…}	GUI 部品 n を上領域に配置します。
void setBottom(Node n){…}	GUI 部品 n を下領域に配置します。
void setLeft(Node n){…}	GUI 部品 n を左領域に配置します。
void setRight(Node n){…}	GUI 部品 n を右領域に配置します。
void setCenter(Node n){…}	GUI 部品 n を中央領域に配置します。
	※Node クラスは Button クラスのスーパークラスです。

§8　レイアウト BorderPane を使ってみましょう（2）

レイアウト BorderPane の領域ごとに GUI 部品の配置を調整できます。

ソースファイル名：Sample4_8.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 各ボタンの位置合わせ
public class Sample4_8 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[5];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");
      btn[3] = new Button("ボタン４");
      btn[4] = new Button("ボタン５");

      // レイアウト BorderPane を生成／設定します
      BorderPane bp = new BorderPane();
      bp.setTop(btn[0]);
      bp.setLeft(btn[1]);
      bp.setCenter(btn[2]);
      bp.setRight(btn[3]);
      bp.setBottom(btn[4]);
      bp.setAlignment(btn[0], Pos.CENTER);
      bp.setAlignment(btn[1], Pos.CENTER);
      bp.setAlignment(btn[2], Pos.CENTER);
      bp.setAlignment(btn[3], Pos.CENTER);
      bp.setAlignment(btn[4], Pos.CENTER);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト BorderPane");

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

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



■レイアウト BorderPane の各領域に配置されたボタンの位置調整をするには

クラス BorderPane にはボタン毎に位置調整を行うメソッドが準備されています。

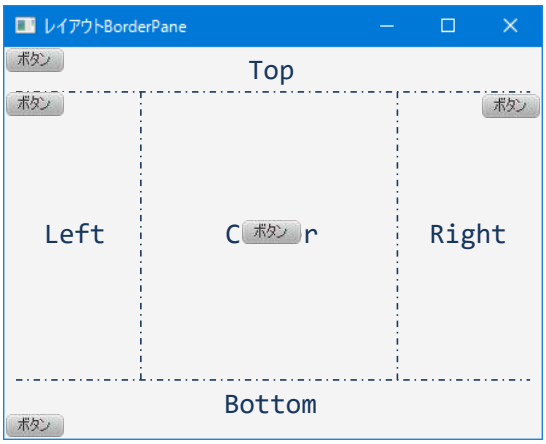
・ボタンの位置調整　　→ setAlignment(btn[0], Pos.CENTER);

指定されたボタンの位置調整を中央配置（Pos.CENTER）にします。

■レイアウト BorderPane の各領域に配置されるボタンの初期位置は

ボタンの位置調整を行わない場合、自動的に次のようになります。

- 上領域（Top）のボタン → Pos.TOP_LEFT
- 下領域（Bottom）のボタン → Pos.BOTTOM_LEFT
- 左領域（Left）のボタン → Pos.TOP_LEFT
- 右領域（Right）のボタン → Pos.TOP_RIGHT
- 中央領域（Center）のボタン → Pos.CENTER



■レイアウト BorderPane の空白エリアの設定

- レイアウト周りの空白エリア → setPadding(new Insets(20));
- ボタンなど GUI 部品間の空白エリア → なし
- ボタンなど GUI 部品の全体的なレイアウトの位置 → なし

■利用したクラスの一覧

BorderPane クラス ← Pane クラス ← Region クラス ←Parent クラス Node クラス ← Object クラス

`void setAlignment(Node n, Pos v){…}` GUI 部品 n の位置調整を v に設定します。

※クラス Node はクラス Button のスーパークラスです。

§9　レイアウト GridPane を使ってみましょう

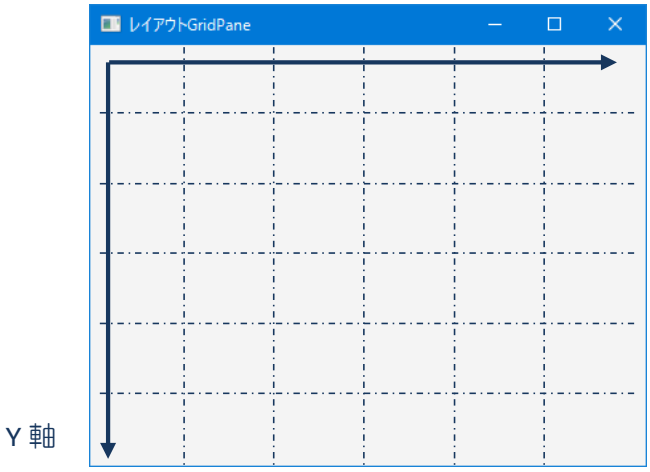
レイアウト GridPane はボタンなどの GUI 部品を縦横の格子状に配置します。

ソースファイル名：Sample4_9.java

```
// ※HP よりインポート文をここへ貼り付けてください

// レイアウト GridPane
public class Sample4_9 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button[] btn = new Button[5];
      btn[0] = new Button("ボタン１");
      btn[1] = new Button("ボタン２");
      btn[2] = new Button("ボタン３");
      btn[3] = new Button("ボタン４");
      btn[4] = new Button("ボタン５");

      // レイアウト GridPane を生成／設定します
      GridPane gp = new GridPane();
      gp.add(btn[0], 0, 0);
      gp.add(btn[1], 0, 1);
      gp.add(btn[2], 0, 2);
      gp.add(btn[3], 1, 0);
      gp.add(btn[4], 1, 1);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウト GridPane");

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

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

■レイアウト GridPane

このレイアウトはボタンなどの GUI 部品を格子状に配置します。



■レイアウト GridPane にボタンを配置するには

各格子にボタンなどの GUI 部品を配置するメソッドが準備されています。

- X 軸 0 Y 軸 0 へ配置　→ `add(btn[0], 0, 0);`
- X 軸 0 Y 軸 1 へ配置　→ `add(btn[1], 0, 1);`

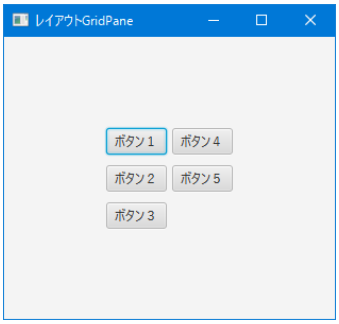
　　:　　:

- X 軸 i Y 軸 j へ配置　→ `add(btn[n], i, j);`

各ボタンが指定された格子に配置されます。

■レイアウト GridPane の空白エリアの設定

- レイアウト周りの空白エリア　→ `setPadding(new Insets(20));`
- ボタンなど GUI 部品間の空白エリア　→ `setHgap(5); setVgap(10);`
- ボタンなど GUI 部品の全体的なレイアウトの位置　→ `setAlignment(Pos.CENTER);`



■利用したクラスの一覧

GridPane クラス ← Pane クラス ← Region クラス ←Parent クラス Node クラス ← Object クラス

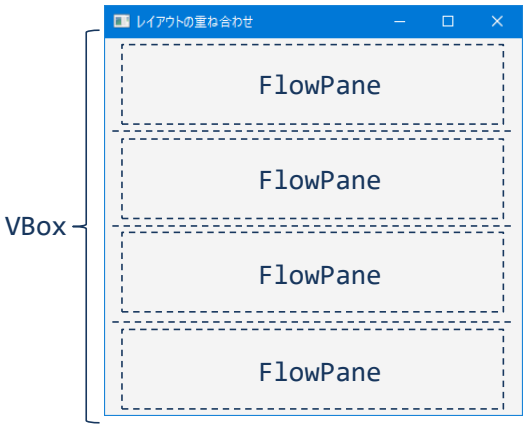
`GridPane(){…}`	レイアウト GridPane を生成します。（コンストラクタ）
`void add(Node n, int c, int r){…}`	GUI 部品 n を x 軸 c と y 軸 r の位置に配置します。
	※クラス Node はクラス Button のスーパークラスです。

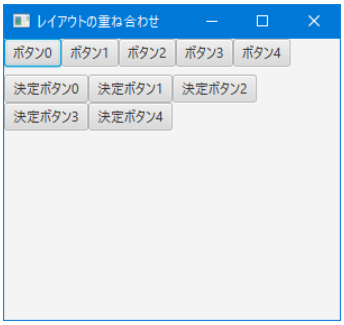
§10　レイアウトを別のレイアウトの上に配置してみましょう

レイアウトを重ね合わせながら、より柔軟にレイアウトを構成できます。

ソースファイル名：Sample4_10.java

```
// ※HP よりインポート文をここへ貼り付けてください

// レイアウトの重ね合わせ
public class Sample4_10 extends Application
{
   public void start(Stage stage) throws Exception{
      ObservableList<Node> lst;

      // ボタンを生成／設定します
      Button[] btn1 = new Button[5];
      Button[] btn2 = new Button[5];
      for(int i=0;i<btn1.length;i++)
         btn1[i] = new Button("ボタン"+i);
      for(int i=0;i<btn2.length;i++)
         btn2[i] = new Button("決定ボタン"+i);

      // ボタンをレイアウト FlowPane に載せます
      FlowPane[] fps = new FlowPane[2];
      fps[0] =  new FlowPane();
      fps[1] =  new FlowPane();
      lst = fps[0].getChildren();
      lst.addAll(btn1);
      lst = fps[1].getChildren();
      lst.addAll(btn2);

      // レイアウト FlowPane をレイアウト VBox に載せます
      VBox vb = new VBox();
      lst = vb.getChildren();
      lst.addAll(fps);
      vb.setSpacing(7);

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

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("レイアウトの重ね合わせ");

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

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





■レイアウトを別のレイアウトの上に配置するには

レイアウトにボタンを配置するのと同じように、新しい別のレイアウトを配置できます。この場合、上に配置されたレイアウトにボタンなどの GUI 部品を配置していきます。

たとえば、画像ビュアーは VBox レイアウトの上に HBox レイアウトを載せて構成できます。