3 回目　ウィンドウを表示してみよう

今日の講義で学ぶ内容

・ウィンドウの表示
・ウィンドウの最大／最小サイズと半透明化
・複数のウィンドウと親子関係

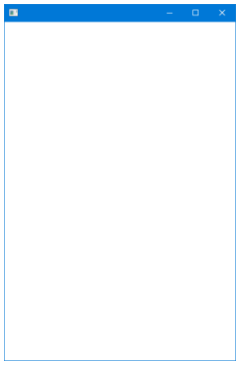
ウィンドウの表示

§1　ウィンドウを表示してみましょう。

ウィンドウアプリケーションは、Application クラスを拡張して作成します。

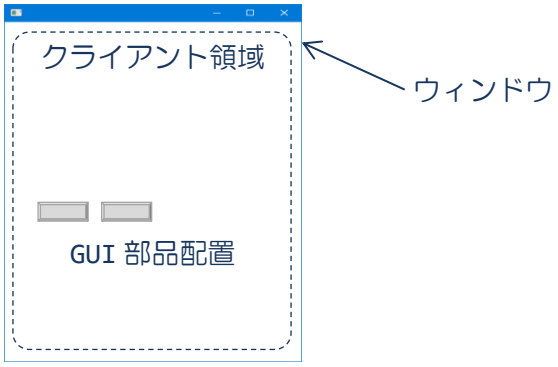
ソースファイル名：Sample3_1.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ウィンドウの表示
public class Sample3_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // レイアウトを生成します
      FlowPane flowpane = new FlowPane();

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

      // ステージにシーンを設定します
      stage.setScene(scene);

      // ステージ（ウィンドウ）を表示します
      stage.show();
   }

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



■ウィンドウの構成

ウィンドウはタイトルバーとウィンドウ枠、クライアント領域からなります。タイトルバーには最小化／最大化ボタンや閉じるボタンがあります。クライアント領域にはボタンなどGUI部品が配置されます。



これらは次のような対応するクラスを用いて表現されます。

• ウィンドウ	ステージ	Stage クラス
• クライアント領域	シーン	Scene クラス
• ボタンなどの GUI 部品の配置方法	レイアウト	FlowPane クラスや HBox クラスなど

■ウィンドウアプリケーションを作成するには

Application クラスを拡張したサブクラスを宣言し、

1. 継承された launch()メソッドを main()メソッド内で実行します。
   ※launch()メソッドの引数には、main()メソッドの引数をそのまま渡します。

2. 継承された start()メソッドをサブクラスでオーバーライドして実行したい処理を記述します。
   ※start()メソッドの引数には、ウィンドウを表現する Stage オブジェクトが渡されます。
   ※通常この Stage オブジェクトにシーンやレイアウトを設定します。

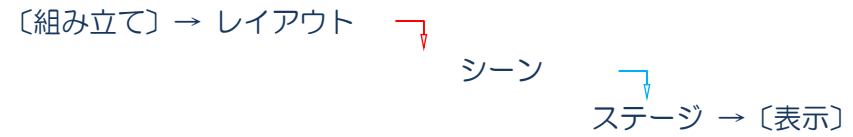
■ウィンドウアプリケーションの動作の流れ

main()メソッドから処理が始まります。ここで Application クラスから継承された launch()メソッドが実行されます。launch()メソッドはアプリケーション開始の準備を行いながら内部で start()メソッドを実行し、アプリケーションを開始します。

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

■ウィンドウにシーンとレイアウトを設定するには

ステージの上にシーンを、シーンの上にレイアウトを置いてウィンドウを組み立てます。それぞれ対応するクラスのオブジェクトを関連付け組み立てます。



〔コード例〕

```
1. FlowPane fp = new FlowPane();     // レイアウトを生成します
2. Scene sc = new Scene(fp);         // レイアウトをシーンにのせ、シーンを生成します
3. st.setScene(sc);                  // シーンをステージにのせます
```

※start()メソッドに渡されたウィンドウを表現する Stage オブジェクト st にシーンを設定します。設定は setScene()メソッドを使用します。

■ウィンドウを表示するには

シーンとレイアウトが設定されたウィンドウを可視状態にします。ウィンドウを表現する Stage オブジェクトの show()メソッドを使用します。

■利用したクラスの一覧

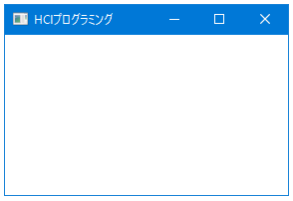
FlowPane クラス（レイアウトクラス）	
FlowPane(){…}	レイアウトを生成します。（コンストラクタ）
Scene クラス	
Scene(Parent a){…}	レイアウト a を設定したシーンを生成します。（コンストラクタ） ※Parent クラスは FlowPane クラスのスーパークラスです。
Stage クラス	
Stage(){…}	ステージを生成します。（コンストラクタ）
void setScene(Scene a){…}	シーン a をステージに貼り付けます。
void show(){…}	ステージを表示します。
Application クラス	
void launch(String[] args){…}	アプリケーションを開始します。
void start(Stage a) {…}	アプリケーションの本体です。オーバーライドして利用します。

§2　ウィンドウの表示位置とサイズを変更してみましょう。

ウィンドウは Stage クラスにより表現されます。Stage クラスのメソッドを用いてウィンドウの設定を行うことができます。

ソースファイル名：Sample3_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ウィンドウの属性
public class Sample3_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // レイアウトを生成します
      FlowPane flowpane = new FlowPane();

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

      // ステージにシーンを設定します
      stage.setScene(scene);

      // ステージ（ウィンドウ）の属性を変更します
      stage.setTitle("HCI プログラミング");
      stage.setX(100);
      stage.setY(100);
      stage.setWidth(300);
      stage.setHeight(200);

      // ステージ（ウィンドウ）を表示します
      stage.show();
   }

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



■ウィンドウの設定を行うには

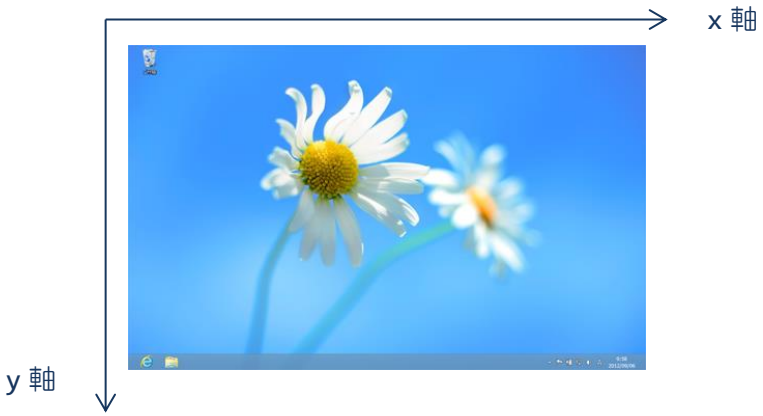
ウィンドウを表現する Stage クラスにウィンドウの設定を行うメソッドが準備されています。

・ウィンドウタイトル　→ `setTitle("HCI プログラミング");`
・ウィンドウの位置（x 座標）　→ `setX(100);`
・ウィンドウの位置（y 座標）　→ `setY(100);`
・ウィンドウのサイズ（横幅ピクセル）　→ `setWidth(300);`
・ウィンドウのサイズ（縦幅ピクセル）　→ `setHeight(200);`

タイトルが「HCI プログラミング」、横幅が 300 ピクセル、縦幅が 200 ピクセルのウィンドウが、スクリーン座標系上の座標(100,100)の位置に表示されます。

■スクリーン座標系とは

ディスプレイ画面上の座標系です。原点は左上です。x 軸正は右方向、y 軸正は下方向です。



■利用したクラスの一覧

Stage クラス

`void setTitle(String s){…}`	ウィンドウのタイトルを文字列 s に設定します。
`void setX(double d){…}`	ウィンドウの位置（x 座標）を d に設定します。
`void setY(double d){…}`	ウィンドウの位置（y 座標）を d に設定します。
`void setWidth(double d){…}`	ウィンドウのサイズ（横幅ピクセル）を d に設定します。
`void setHeight(double d){…}`	ウィンドウのサイズ（縦幅ピクセル）を d に設定します。

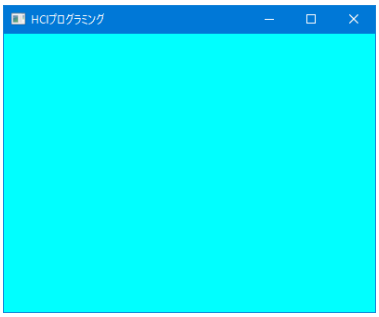
※スーパークラスから継承されるメソッドも含めて紹介しています。以後紹介するクラスの一覧も同様です。

§3　シーンのサイズと色を変更してみましょう

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

ソースファイル名：Sample3_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// シーンの属性
public class Sample3_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // レイアウトを生成します
      FlowPane flowpane = new FlowPane();

      // シーンを生成します
      Scene scene = new Scene(flowpane, 400, 300);
      scene.setFill(Color.AQUA);

      // ステージにシーンを設定します
      stage.setScene(scene);
      stage.setTitle("HCI プログラミング");
      stage.setX(100);
      stage.setY(100);

      // ステージ（ウィンドウ）を表示します
      stage.show();
   }

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



■シーンの設定を行うには

シーンの設定を行うコンストラクタやメソッドが Scene クラスに準備されています。

・レイアウトとシーンのサイズ　　→ `new Scene(flowpane, 400, 300);`

・シーンの色　　→ `setFill(Color.AQUA);`

※この他、キーボードやマウスの入力イベントを設定するメソッドが多数準備されています。

横幅が 400 ピクセル、縦幅が 300 ピクセルで、背景の色がアクア色のシーンになります。

※シーンとステージのサイズが両方とも指定された場合はステージが優先されます。

■使用できる色は

色は Color クラスで宣言される定数で表現されています。Color.ORANGE や Color.SKYBLUE など数多くあります。詳しくは、Color クラスを検索してみましょう。「javaFX Color クラス」ググってみて!!

■利用したクラスの一覧

Scene クラス

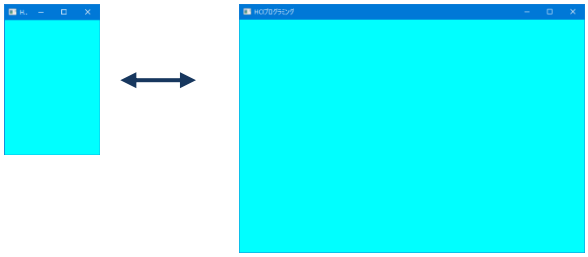
`Scene(Parent a, double w, double h){…}`	レイアウト a とシーンのサイズ（横幅×縦幅）w×h ピクセルを設定したシーンを生成します。（コンストラクタ） ※Parent クラスは FlowPane クラスのスーパークラスです。 ※サイズにタイトルバーとウィンドウ枠の幅は含みません。
`void setFill(Paint v){…}`	シーンの色を v に設定します。 ※Paint クラスは Color クラスのスーパークラスです。

§4　ウィンドウの最大サイズと最小サイズを指定してみましょう。

ウィンドウ表現する Stage クラスのメソッドを用いてウィンドウのサイズを制限できます。

ソースファイル名：Sample3_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ウィンドウの最大サイズと最小サイズ
public class Sample3_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // レイアウトを生成します
      FlowPane flowpane = new FlowPane();

      // シーンを生成します
      Scene scene = new Scene(flowpane, 400, 300);
      scene.setFill(Color.AQUA);

      // ステージにシーンを設定します
      stage.setScene(scene);
      stage.setTitle("HCI プログラミング");
      stage.setX(100);
      stage.setY(100);
      stage.setMaxWidth(700);
      stage.setMaxHeight(500);
      stage.setMinWidth(200);
      stage.setMinHeight(300);

      // ステージ（ウィンドウ）を表示します
      stage.show();
   }

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



■ウィンドウのサイズを制限するには

ウィンドウの最大サイズと最小サイズを指定するメソッドが Stage クラスに準備されています。

・ウィンドウの最大サイズ（横幅ピクセル）　→ `setMaxWidth(700);`
・ウィンドウの最大サイズ（縦幅ピクセル）　→ `setMaxHeight(500);`
・ウィンドウの最小サイズ（横幅ピクセル）　→ `setMinWidth(200);`
・ウィンドウの最小サイズ（縦幅ピクセル）　→ `setMinHeight(300);`

ウィンドウの最大サイズが横幅 700 ピクセル、縦幅 500 ピクセルに、最小サイズが横幅 200 ピクセル、縦幅 300 ピクセルに設定されます。マウスによるウィンドウのサイズ変更は、指定したサイズ以内でのみ可能です。

■利用したクラスの一覧

Stage クラス

`void setMaxWidth(double d){…}`	ウィンドウの最大サイズ（横幅ピクセル）を d に設定します。
`void setMaxHeight(double d){…}`	ウィンドウの最大サイズ（縦幅ピクセル）を d に設定します。
`void setMinWidth(double d){…}`	ウィンドウの最小サイズ（横幅ピクセル）を d に設定します。
`void setMinHeight(double d){…}`	ウィンドウの最小サイズ（縦幅ピクセル）を d に設定します。

§5　ウィンドウを半透明化してみましょう。

ウィンドウを表現する Stage クラスのメソッドを用いてウィンドウを半透明化できます。

ソースファイル名：Sample3_5.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ウィンドウの半透明化
public class Sample3_5 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // レイアウトを生成します
      FlowPane flowpane = new FlowPane();

      // シーンを生成します
      Scene scene = new Scene(flowpane, 400, 300);
      scene.setFill(Color.AQUA);

      // ステージにシーンを設定します
      stage.setScene(scene);
      stage.setTitle("HCI プログラミング");
      stage.setX(100);
      stage.setY(100);
      stage.setMaxWidth(700);
      stage.setMaxHeight(500);
      stage.setMinWidth(200);
      stage.setMinHeight(300);
      stage.setOpacity(0.5);

      // ステージ（ウィンドウ）を表示します
      stage.show();
   }

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

■ウィンドウを半透明化するには

ウィンドウの透明度を指定するメソッドが Stage クラスに準備されています。

・ウィンドウの透明度　　→ setOpacity(0.5);

ウィンドウの透明度が 0.5 に設定されます。完全な不透明は 1.0 とし、完全な透明は 0.0 です。

■利用したクラスの一覧

Stage クラス

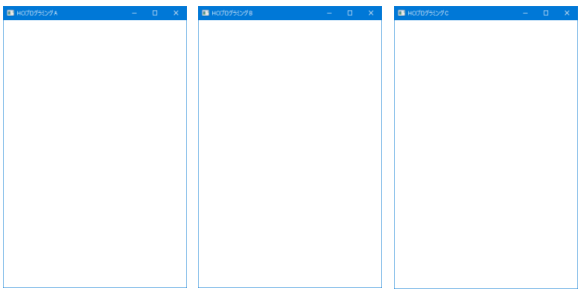
void setOpacity(double d){…}	透明度を d（透明 0.0~不透明 1.0）に設定します。

§6　複数のウィンドウを表示してみましょう。

複数のウィンドウを生成し、表示することができます。

ソースファイル名：Sample3_6.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 複数のウィンドウ
public class Sample3_6 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ３個のレイアウトを生成します
      FlowPane flowpane1 = new FlowPane();
      FlowPane flowpane2 = new FlowPane();
      FlowPane flowpane3 = new FlowPane();

      // ３個のシーンを生成します
      Scene scene1 = new Scene(flowpane1);
      Scene scene2 = new Scene(flowpane2);
      Scene scene3 = new Scene(flowpane3);

      // ３個のステージを生成します
      Stage stage1 = new Stage();
      Stage stage2 = new Stage();
      Stage stage3 = new Stage();
      stage1.setScene(scene1);
      stage2.setScene(scene2);
      stage3.setScene(scene3);
      stage1.setTitle("HCI プログラミングＡ");
      stage2.setTitle("HCI プログラミングＢ");
      stage3.setTitle("HCI プログラミングＣ");
      stage1.setX(0);    stage1.setY(0);
      stage2.setX(100);  stage2.setY(100);
      stage3.setX(200);  stage3.setY(200);

      // ３個のステージ（ウィンドウ）を表示します
      stage1.show();
      stage2.show();
      stage3.show();
   }

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



■複数のウィンドウを表示するには

レイアウトとシーン、ステージをそれぞれ準備します。シーンやウィンドウの設定はこれまで同様です。

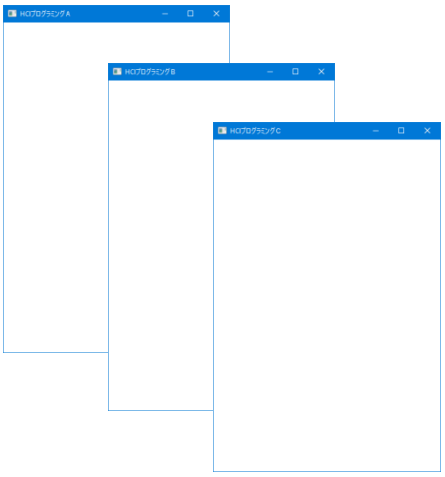
また、start()メソッドから渡された Stage オブジェクトを必ずしも使う必要はありません。

§7　ウィンドウには親子関係があります。

ウィンドウに親子関係を指定することができます。

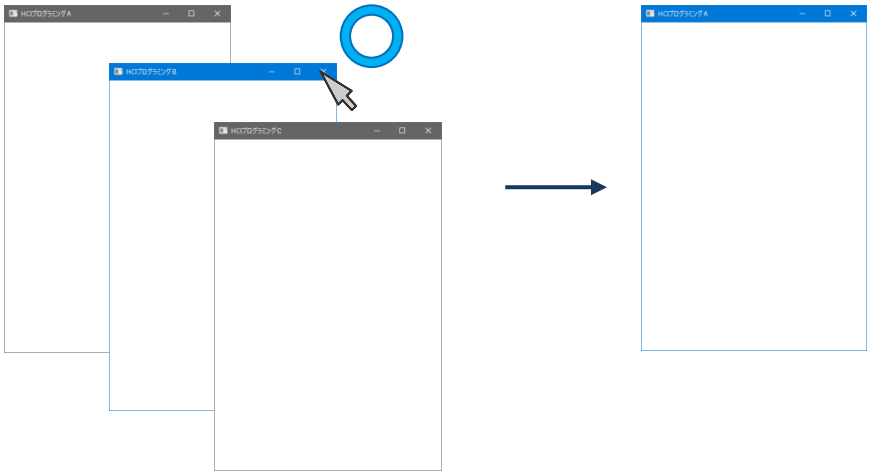
ソースファイル名：Sample3_7.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ウィンドウの親子関係
public class Sample3_7 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ３個のレイアウトを生成します
      FlowPane flowpane1 = new FlowPane();
      FlowPane flowpane2 = new FlowPane();
      FlowPane flowpane3 = new FlowPane();

      // ３個のシーンを生成します
      Scene scene1 = new Scene(flowpane1);
      Scene scene2 = new Scene(flowpane2);
      Scene scene3 = new Scene(flowpane3);

      // ３個のステージを生成します
      Stage stage1 = new Stage();
      Stage stage2 = new Stage();
      Stage stage3 = new Stage();
      stage1.setScene(scene1);
      stage2.setScene(scene2);
      stage3.setScene(scene3);
      stage1.setTitle("HCI プログラミングＡ");
      stage2.setTitle("HCI プログラミングＢ");
      stage3.setTitle("HCI プログラミングＣ");
      stage1.setX(0);    stage1.setY(0);
      stage2.setX(100);  stage2.setY(100);
      stage3.setX(200);  stage3.setY(200);

      // 親子関係を設定します
      stage2.initOwner(stage1);
      stage3.initOwner(stage2);

      // ３個のステージ（ウィンドウ）を表示します
      stage1.show();
      stage2.show();
      stage3.show();
   }

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



■ウィンドウの親子関係とは

親ウィンドウが閉じられると、そのすべての子ウィンドウは閉じられます。



■ウィンドウの親子関係を指定するには

ウィンドウを表現する Stage クラスに、親子関係を設定するメソッドが準備されています。

・ウィンドウの親子関係　　→ `initOwner(stage1);`

ウィンドウ stage1 を親ウィンドウに指定します。親ウィンドウ stage1 が閉じられると、その子ウィンドウも閉じられます。

■利用したクラスの一覧

Stage クラス

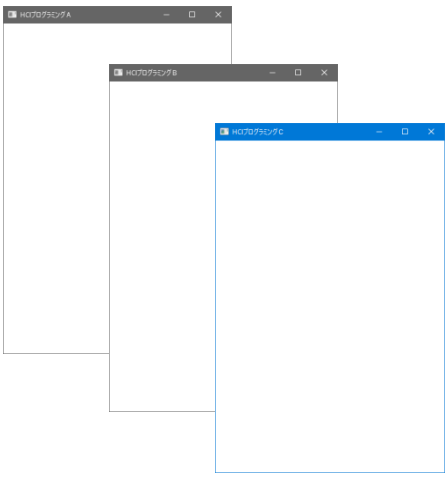
`void initOwner(Window w){…}`	ウィンドウ w を親ウィンドウに設定します。 ※Window クラスは Stage クラスのスーパークラスです。

§8　ウィンドウにフォーカス移動制限をつけてみましょう。

親子関係にあるウィンドウにフォーカス移動制限（モダリティ）を指定することができます。

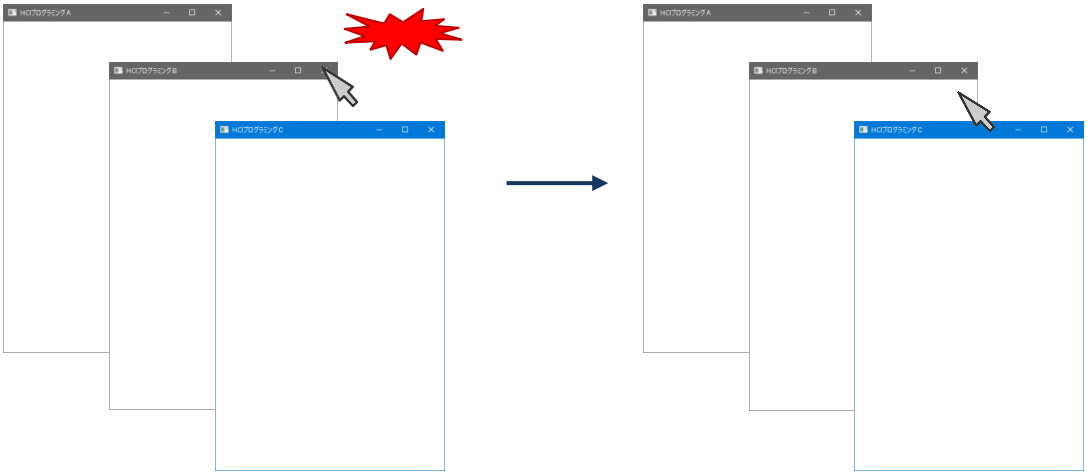
ソースファイル名：Sample3_8.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ウィンドウの親子関係とモーダルウィンドウ
public class Sample3_8 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ３個のレイアウトを生成します
      FlowPane flowpane1 = new FlowPane();
      FlowPane flowpane2 = new FlowPane();
      FlowPane flowpane3 = new FlowPane();

      // ３個のシーンを生成します
      Scene scene1 = new Scene(flowpane1);
      Scene scene2 = new Scene(flowpane2);
      Scene scene3 = new Scene(flowpane3);

      // ３個のステージを生成します
      Stage stage1 = new Stage();
      Stage stage2 = new Stage();
      Stage stage3 = new Stage();
      stage1.setScene(scene1);
      stage2.setScene(scene2);
      stage3.setScene(scene3);
      stage1.setTitle("HCI プログラミングＡ");
      stage2.setTitle("HCI プログラミングＢ");
      stage3.setTitle("HCI プログラミングＣ");
      stage1.setX(0);    stage1.setY(0);
      stage2.setX(100);  stage2.setY(100);
      stage3.setX(200);  stage3.setY(200);

      // 親子関係を設定します
      stage2.initOwner(stage1);
      stage3.initOwner(stage2);

      // モダリティを指定します
      stage2.initModality(Modality.WINDOW_MODAL);
      stage3.initModality(Modality.WINDOW_MODAL);

      // ３個のステージ（ウィンドウ）を表示します
      stage1.show();
      stage2.show();
      stage3.show();
   }

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



■ウィンドウのフォーカス移動制限とは

親ウィンドウは、そのすべての子ウィンドウが閉じられるまで、閉じることができません。



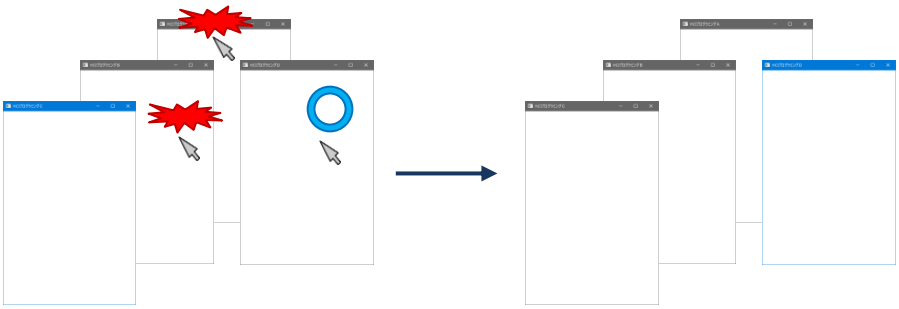
■ウィンドウのフォーカス移動制限を指定するには

ウィンドウを表現する Stage クラスに、フォーカス移動制限を設定するメソッドが準備されています。

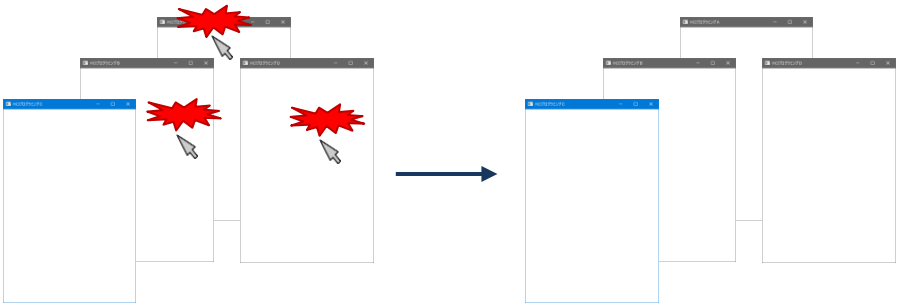
・フォーカス移動制限　　→ `initModality(Modality.WINDOW_MODAL);`

ウィンドウ毎に、次の引数のいずれかを用いて指定します。

・`Modality.WINDOW_MODAL`　　親ウィンドウへのフォーカス移動を禁止します。



・`Modality.APPLICATION_MODAL`　　他のウィンドウへのフォーカス移動も禁止します。



■利用したクラスの一覧

Stage クラス

`void initModality(Modality m){…}`　　フォーカス移動制限を m に設定します。