8 回目　ボタン、チェックボックス、ラジオボタン

今日の講義で学ぶ内容

・ボタンとアクションイベント
・ボタンのカスタマイズ
・チェックボックスとラジオボタン

ボタンとアクションイベント

§1　ボタンを配置してみましょう

ボタンは、ラベルと同じようにフォントやその色、画像の貼り付けなどを設定できます。

ソースファイル名：Sample8_1.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ボタンの配置
public class Sample8_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button bt1 = new Button("メリークリスマス¥n ボタン１");
      Button bt2 = new Button("メリークリスマス¥n ボタン２");
      bt1.setGraphic(new ImageView("xmas.jpg"));
      bt1.setTextFill(Color.RED);
      bt1.setFont(new Font(24));
      bt2.setGraphic(new ImageView("gift.jpg"));
      bt2.setTextFill(Color.RED);
      bt2.setFont(new Font(24));

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

      // 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("イベント処理");
```

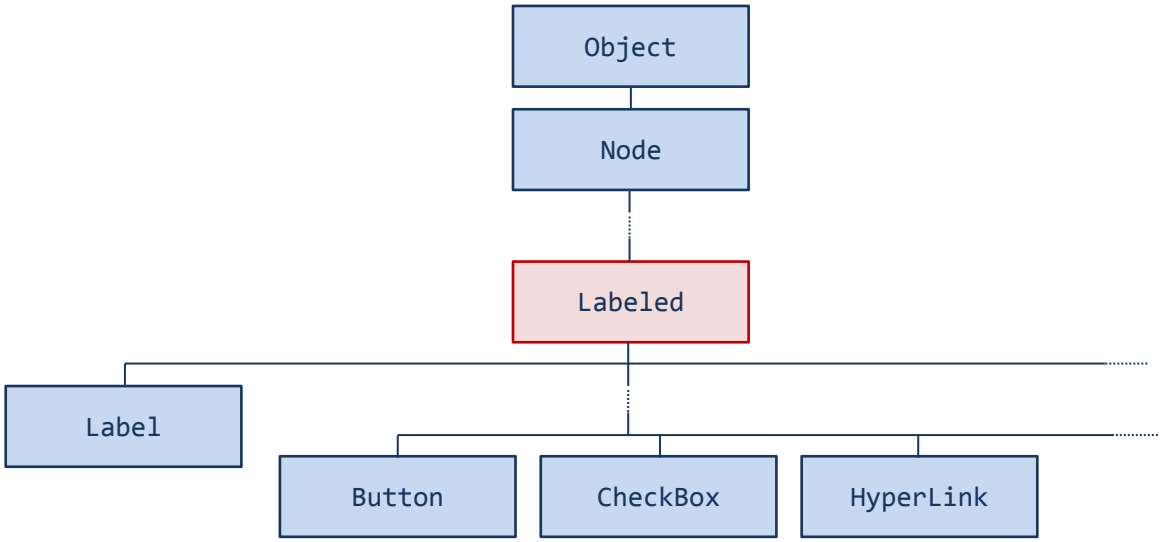
```
        // ステージを表示します
        stage.show();
    }

    public static void main(String[] args)

    {
        launch(args);
    }
}
```

■ボタンを管理するクラス Button

ボタンはクラス Button により表現されます。前回学習したラベルを表現するクラス Label と同じスーパークラス Labeled を持ちます。クラス Labeled のメソッドを用いて、文字のフォントや色、画像の貼り付け、画像の相対位置などラベルと同様の設定ができます。

- ボタンの生成 → `new Button("メリークリスマス");`
- 画像の貼り付け → `setGraphic(new ImageView("xmas.jpg"));`
- 文字の色 → `setTextFill(Color.RED);`
- 文字のフォント指定 → `setFont(new Font(24));`

24 ポイントで赤色の「メリークリスマス」という名前で画像 xmas.jpg が貼られたボタンを生成します。



■利用したクラスの一覧

Button クラス

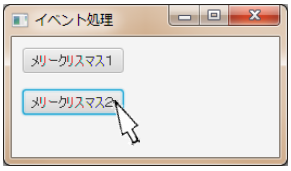
`Button(String s){…}`	文字列 s のボタンを生成します。
`void setGraphic(Node n){…}`	GUI 部品 n をボタンに貼ります。 ※Node クラスは ImageView クラスのスーパークラスです。
`void setTextFill(Paint p){…}`	文字の色を p に設定します。 ※Paint クラスは Color クラスのスーパークラスです。
`void setFont(Font f){…}`	フォントを f に設定します。

§2　ボタンから発生するアクションイベントを受け取ってみましょう

アクションイベントは、ボタンをクリックすると発生します。

ソースファイル名：Sample8_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ボタンとアクションイベント
public class Sample8_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button bt1 = new Button("メリークリスマス１");
      Button bt2 = new Button("メリークリスマス２");
      bt1.setId("Button1");
      bt2.setId("Button2");

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      bt1.addEventHandler(ActionEvent.ANY, actionhandler);
      bt2.addEventHandler(ActionEvent.ANY, actionhandler);

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

      // 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("イベント処理");

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

   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
      {
         Button bt = (Button)e.getTarget();
         System.out.println(bt.getId());
      }
   }
```



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

実行結果

```
Button1 ← メリークリスマス１を押す
Button2 ← メリークリスマス２を押す
Button1 ← メリークリスマス１を押す
   :
```

■アクションイベントとは

アクションイベントは、ボタンやチェックボックスなど主に GUI 部品に変化があったときに発生するイベントです。これらのイベントが発生したタイミングで、各処理を実行させることができます。

■アクションイベントを表現するクラス ActionEvent

クラス ActionEvent により表現され、以下の種類があります。

・ActionEvent.ACTION　　→ 唯一のイベントです。

この他、すべてのイベントを表現するイベントがあります。実際に発生するイベントではなく、すべてのイベントを受け取りたいときに利用します。

・ActionEvent.ANY　　→ 上記すべてのイベントを表現します

※今後アクションイベントの種類が増えることを想定しています。

■アクションイベントを処理するイベントハンドラインタフェース EventHandler<ActionEvent>

アクションイベントはイベントハンドラクラスで受け取り、対応する処理を行います。

1. EventHandler<ActionEvent>インタフェースを実装してイベントハンドラクラスを宣言

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

※発生したイベントがメソッドの引数 e に渡されて呼び出されます

〔コード例〕

```
1. class MyEventHandler implements EventHandler<ActionEvent>{
2.    public void handle(ActionEvent e)
3.    {
4.       // ここにイベントに対応する処理を記述します
5.    }
6. }
```

■ボタンへイベントハンドラを登録

GUI 部品やシーン、ステージは様々なイベントを発生します。これらを受け止めるためにイベントハンドラをそれぞれに登録する必要があります。アクションイベントはボタンなど GUI 部品を表現するクラスで受け取ることができます。ボタンにイベントハンドラを登録します。

〔コード例〕

```
1. MyEventHandler eh = new MyEventHandler();
2. bt.addEventHandler(ActionEvent.ANY, eh);
```

※オブジェクト eh をイベントハンドラとして Button クラスのオブジェクト bt に登録します

■ボタンと識別子

複数のボタンを 1 つのイベントハンドラで処理するとき、イベントが発生したらどのボタンから発生したのかを知りたい場合があります。ボタンには識別子を設定することができます。識別子を用いてどのボタンから発生したイベントなのかを判断します。

Button クラスに識別子の設定と取得を行うメソッドが準備されています。

- 識別子の設定（"Button1"を識別子として）　→ `setId("Button1");`
- 識別子の取得（String 型）　→ `getId();`

■利用したクラスの一覧

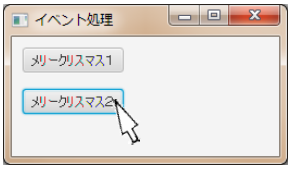
ActionEvent クラス	
`ActionEvent.ACTION`	主に GUI 部品に変化があったときに発生するイベントです。
`EventTarget getTarget(){…}`	イベントが発生した GUI 部品を取得します。
EventHandler<ActionEvent>インタフェース	
`void handle(ActionEvent e);`	イベントが発生したときに実行されます。
Button クラス	
`void setId(String s){…}`	文字列 s をボタンの識別子に設定します。
`String getId(){…}`	ボタンの識別子を取得します。
`void addEventHandler(EventType<ActionEvent> e, EventHandler<ActionEvent> h){…}`	イベント e を受け取るハンドラ h を登録します。

§３　ボタン毎に処理を分岐してみましょう

ボタンの識別子を利用して、ボタン毎に別々の処理を記述できます。

ソースファイル名：Sample8_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ボタン毎に処理を分岐
public class Sample8_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button bt1 = new Button("メリークリスマス１");
      Button bt2 = new Button("メリークリスマス２");
      bt1.setId("Button1");
      bt2.setId("Button2");

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      bt1.addEventHandler(ActionEvent.ANY, actionhandler);
      bt2.addEventHandler(ActionEvent.ANY, actionhandler);

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

      // 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("イベント処理");

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

   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
      {
         Button bt = (Button)e.getTarget();
         String id = bt.getId();
         if(id.equals("Button1")){
            System.out.println("ボタン１が押されました");
```



```
        }else if(id.equals("Button2")){
          System.out.println("ボタン２が押されました");
        }
      }
   }

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

実行結果

```
ボタン１が押されました ← メリークリスマス１を押す
ボタン２が押されました ← メリークリスマス２を押す
ボタン１が押されました ← メリークリスマス１を押す
   :
```

■ボタンの識別と分岐

ボタンがもつ識別子は文字列（String 型）で表現されます。String クラスに文字列を比較するメソッドが準備されています。

・文字列の比較（boolean 型）　→ equals("Button1");
　・文字列"Button1"と等しい場合、true を戻します。
　・文字列"Button1"と等しくない場合、false を戻します。

■利用したクラスの一覧

String クラス

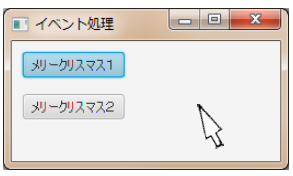
boolean equals(Object s){…}　与えられた文字列 s と等しいかどうか（true/false）を判断します。

§4　デフォルトボタン／キャンセルボタンを指定してみましょう

リターンキーやエスケープキーで動作するボタンを指定できます。

ソースファイル名：Sample8_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// デフォルトボタンとキャンセルボタン
public class Sample8_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button bt1 = new Button("メリークリスマス１");
      Button bt2 = new Button("メリークリスマス２");
      bt1.setId("Button1");
      bt1.setDefaultButton(true);
      bt2.setId("Button2");
      bt2.setCancelButton(true);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      bt1.addEventHandler(ActionEvent.ANY, actionhandler);
      bt2.addEventHandler(ActionEvent.ANY, actionhandler);

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

      // 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("イベント処理");

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

   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
```



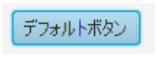
```
    {
      Button bt = (Button)e.getTarget();
      System.out.println(bt.getId());
    }
  }

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

実行結果

```
Button1 ← リターン（Enter）キーを押す
Button1 ← リターン（Enter）キーを押す
Button2 ← エスケープ（Esc）キーを押す
  :
```

■デフォルトボタンとは

リターン（Enter）キーを押すと、このボタンから ActionEvent が発生します。デフォルトボタンは他のボタンとは異なる色で表示されます。



Button クラスにデフォルトボタンを設定するメソッドが準備されています。

・デフォルトボタンの設定　　→ setDefaultButton(true);

■キャンセルボタンとは

エスケープ（Esc）キーを押すと、このボタンから ActionEvent が発生します。

Button クラスにキャンセルボタンを設定するメソッドが準備されています。

・キャンセルボタンの設定　　→ setCancelButton(true);

■利用したクラスの一覧

Button クラス

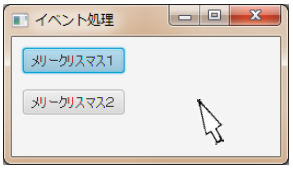
void setDefaultButton(boolean b){…}	デフォルトボタンの設定を行います。
void setCancelButton(boolean b){…}	キャンセルボタンの設定を行います。

§5　ボタンにニーモニックを指定してみましょう

[Alt]+[A] や [Alt]+[X] でボタンが動作するニーモニックを指定できます。

ソースファイル名：Sample8_5.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ボタンとニーモニック
public class Sample8_5 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ボタンを生成／設定します
      Button bt1 = new Button("メリークリスマス１ _A");
      Button bt2 = new Button("メリークリスマス２ _B");
      bt1.setId("Button1");
      bt1.setDefaultButton(true);
      bt2.setId("Button2");
      bt2.setCancelButton(true);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      bt1.addEventHandler(ActionEvent.ANY, actionhandler);
      bt2.addEventHandler(ActionEvent.ANY, actionhandler);

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

      // 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("イベント処理");

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

   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
      {
         Button bt = (Button)e.getTarget();
         System.out.println(bt.getId());
```



```
    }
  }

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

実行結果

```
Button1 ← [Alt]+[A]キーを押す
Button1 ← [Alt]+[A]キーを押す
Button2 ← [Alt]+[B]キーを押す
  :
```

■ニーモニックとは

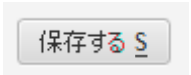
マウスでボタンをクリックしなくても、キーコンビネーションでボタンを押すことができます。これをニーモニックといいます。たとえば、[Alt]+[S] や [Alt]+[C] があります。

ニーモニックは、ボタンの名前（文字列）の**最後に、半角スペースを入れ、アンダーバーとこれに続く英数字**で指定されます。[Alt]キーと、指定した英数字を同時に押すと、該当するボタンからアクションイベントが発生します。たとえば、

```
new Button("開く _O");
```

[Alt]+[O] で開くボタンを押すことができます。

```
new Button("保存する" _S);
```



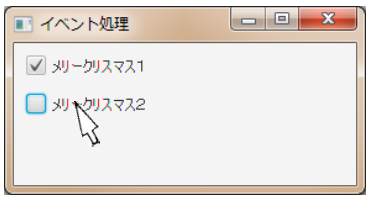
[Alt]+[S] で保存するボタンを押すことができます。

§6　チェックボックスから発生するアクションイベントを受け取ってみましょう

チェックボックスのチェックを入れたり外したりするとアクションイベントが発生します。

ソースファイル名：Sample8_6.java

```
// ※HP よりインポート文をここへ貼り付けてください

// チェックボックスとアクションイベント
public class Sample8_6 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // チェックボックスを生成／設定します
      CheckBox cb1 = new CheckBox("メリークリスマス１");
      CheckBox cb2 = new CheckBox("メリークリスマス２");
      cb1.setId("CheckBox1");
      cb1.setSelected(true);
      cb2.setId("CheckBox2");

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      cb1.addEventHandler(ActionEvent.ANY, actionhandler);
      cb2.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト VBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.add(cb1);
      lst.add(cb2);
      vb.setPadding(new Insets(10));
      vb.setSpacing(15);

      // 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("イベント処理");

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

   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
      {
         CheckBox cb = (CheckBox)e.getTarget();
```



```
        boolean on = cb.isSelected();
        System.out.println(cb.getId()+"/"+on);
      }
   }

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

実行結果

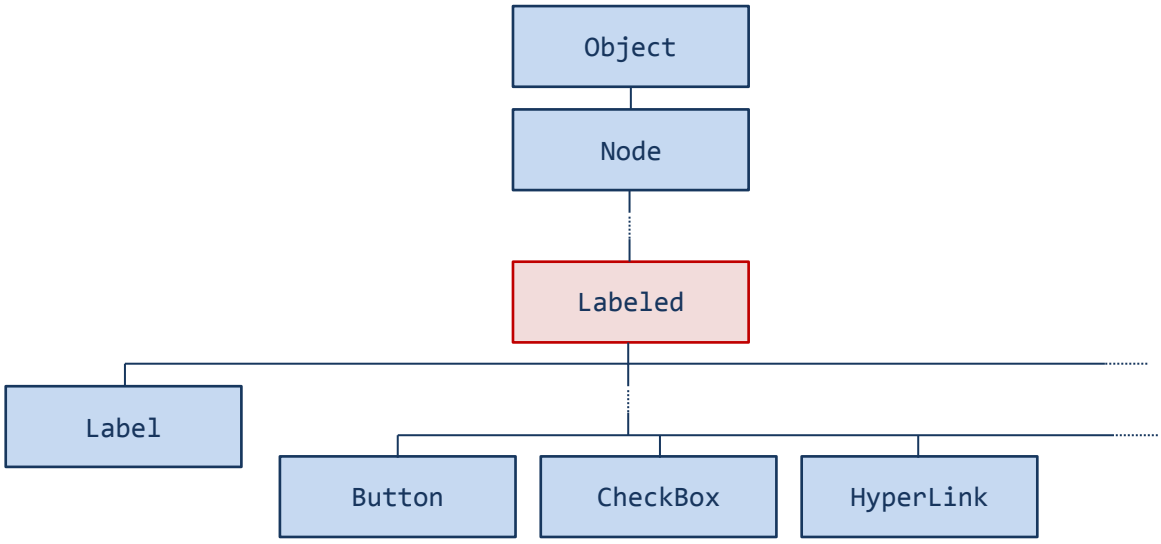
```
CheckBox1/false   ← メリークリスマス１のチェックを外す
CheckBox1/true    ← メリークリスマス１のチェックを入れる
CheckBox2/true    ← メリークリスマス２のチェックを入れる
CheckBox2/false   ← メリークリスマス２のチェックを外す
          :
```

■チェックボックスとは

チェックボックスは、選択状態と解除状態を切り替えることができる GUI 部品です。チェックボックスのチェックを入れたり外したりすると、アクションイベントが発生します。

■チェックボックスを管理するクラス CheckBox

チェックボックスはクラス CheckBox により表現されます。前回学習したラベルを表現するクラス Label と同じスーパークラス Labeled を持ちます。クラス Labeled のメソッドを用いて、文字のフォントや色、画像の貼り付け、画像の相対位置などラベルと同様の設定ができます。



■アクションイベントを処理するイベントハンドラインタフェース EventHandler<ActionEvent>

イベントハンドラクラスの宣言は、ボタンの場合と同じように行います。

1. EventHandler<ActionEvent>インタフェースを実装してイベントハンドラクラスを宣言

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

■チェックボックスへイベントハンドラを登録

チェックボックスから発生するイベントを受け取ります。それぞれのチェックボックスにイベントハンドラを登録します。

〔コード例〕

```
1. MyEventHandler eh = new MyEventHandler();
2. cb.addEventHandler(ActionEvent.ANY, eh);
```

※オブジェクト eh をイベントハンドラとして CheckBox クラスのオブジェクト cb に登録します

■チェックボックスと識別子

複数のチェックボックスを 1 つのイベントハンドラで処理するときは、ボタンの場合と同じように、識別子を用いてどのチェックボックスから発生したイベントなのかを判断します。

CheckBox クラスに識別子の設定と取得を行うメソッドが準備されています。

- 識別子の設定（"CheckBox1"を識別子として） → `setId("CheckBox1");`
- 識別子の取得（String 型） → `getId();`

■チェックボックスの状態を設定したり確認したりするには

チェック状態を設定／取得するメソッドがクラス CheckBox に準備されています。

- チェック状態の設定 → `setSelected(true);`
- チェック状態の取得（boolean 型） → `isSelected();`
  - 戻り値が true であればチェックされています。
  - 戻り値が false でればチェックされていません。

■利用したクラスの一覧

CheckBox クラス

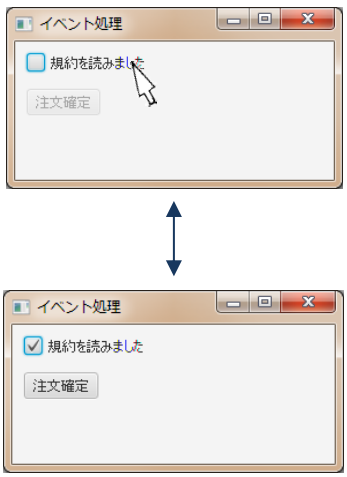
`CheckBox(String s){…}`	文字列 s をもつチェックボックスを生成します。
`void setId(String s){…}`	文字列 s をチェックボックスの識別子に設定します。
`String getId(){…}`	チェックボックスの識別子を取得します。
`void addEventHandler(EventType<ActionEvent> e, EventHandler<ActionEvent> h){…}`	イベント e を受け取るハンドラ h を登録します。
`void setSelected(boolean b){…}`	チェック状態（true/false）を設定します。
`boolean isSelected(){…}`	チェック状態（true/false）を返します。

§7　チェックボックスとボタンを連携してみましょう

チェックボックスのチェック状態をボタンの有効／無効に反映してみましょう。

ソースファイル名：Sample8_7.java

```
// ※HP よりインポート文をここへ貼り付けてください

// チェックボックスとボタンの連携
public class Sample8_7 extends Application
{
   private Button bt;

   public void start(Stage stage) throws Exception
   {
      // チェックボックスとボタンを生成／設定します
      CheckBox cb = new CheckBox("規約を読みました");
      bt = new Button("注文確定");
      cb.setSelected(false);
      bt.setDisable(true);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      cb.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト VBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.add(cb);
      lst.add(bt);
      vb.setPadding(new Insets(10));
      vb.setSpacing(15);

      // 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("イベント処理");

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

   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
      {
         CheckBox cb = (CheckBox)e.getTarget();
         if(cb.isSelected()) bt.setDisable(false);
         else bt.setDisable(true);
```



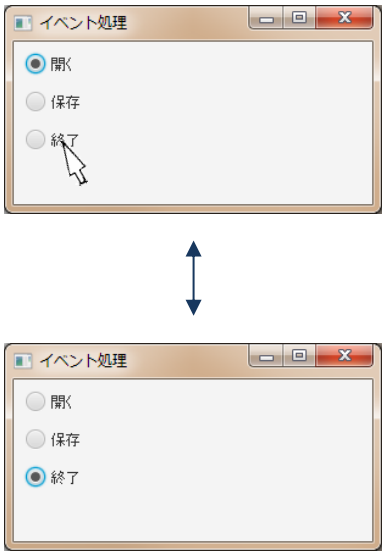
```
    }
  }

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

§8　ラジオボタンから発生するアクションイベントを受け取ってみましょう

ラジオボタンを選択するとアクションイベントが発生します。

ソースファイル名：Sample8_8.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ラジオボタンとアクションイベント
public class Sample8_8 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ラジオボタンを生成／設定します
      RadioButton rb1 = new RadioButton("開く");
      RadioButton rb2 = new RadioButton("保存");
      RadioButton rb3 = new RadioButton("終了");
      rb1.setId("Open");
      rb2.setId("Save");
      rb3.setId("Quit");

      // ラジオボタンをグループ化します
      ToggleGroup gp = new ToggleGroup();
      rb1.setToggleGroup(gp);
      rb2.setToggleGroup(gp);
      rb3.setToggleGroup(gp);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      rb1.addEventHandler(ActionEvent.ANY, actionhandler);
      rb2.addEventHandler(ActionEvent.ANY, actionhandler);
      rb3.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト VBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.add(rb1);
      lst.add(rb2);
      lst.add(rb3);
      vb.setPadding(new Insets(10));
      vb.setSpacing(15);

      // 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("イベント処理");

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



```
   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
      {
         RadioButton rb = (RadioButton)e.getTarget();
         System.out.println(rb.getId());
      }
   }

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

実行結果

```
Open   ← 開くを選択する
Quit   ← 終了を選択する
Save   ← 保存を選択する
  :
```

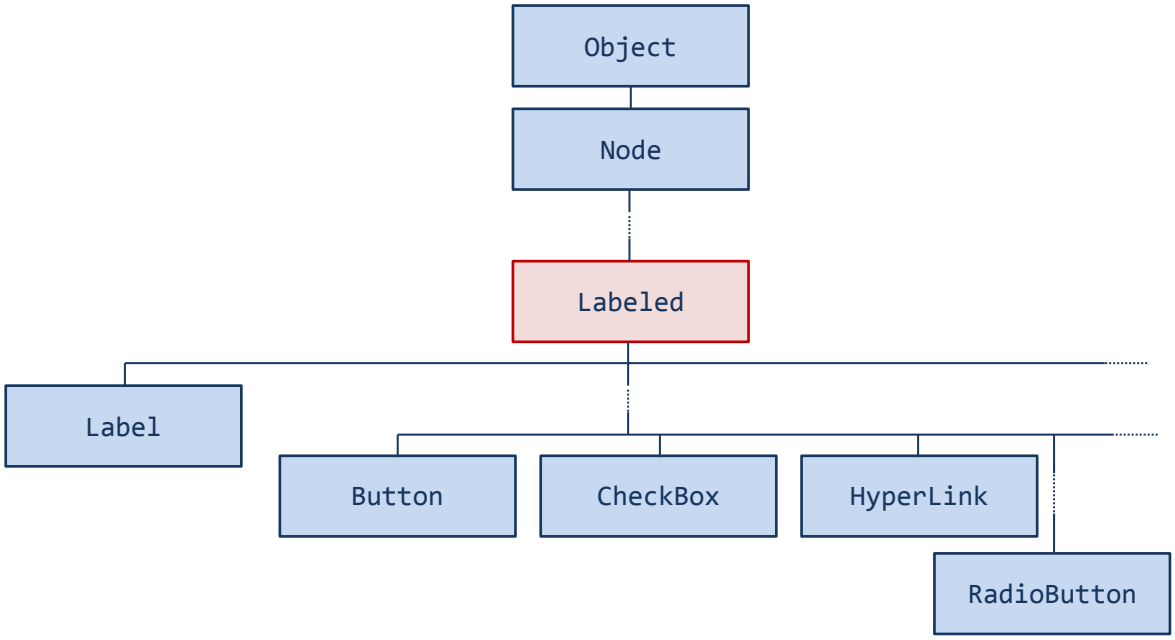
■ラジオボタンとは

ラジオボタンは、選択状態と解除状態が排他的に切り替わるボタンです。

ボタンが 3 個ある場合、ボタン 1 を押したら、ボタン 2 とボタン 3 は戻ります。ボタン 2 を押したら、ボタン 1 とボタン 3 は戻ります。同時に複数のボタンを押すことはできません。

■ラジオボタンを管理するクラス RadioButton

ラジオボタンはクラス RadioButton により表現されます。前回学習したラベルを表現するクラス Label と同じスーパークラス Labeled を持ちます。クラス Labeled のメソッドを用いて、文字のフォントや色、画像の貼り付け、画像の相対位置などラベルと同様の設定ができます。



■ラジオボタンをグループ化するクラス ToggleGroup

ラジオボタンはグループ化することにより、排他的に切り替わるボタンの範囲を指定します。クラス ToggleGroup によりグループは管理されます。

・グループの生成　　→ new ToggleGroup();

クラス RadioButton に、グループを指定するメソッドが準備されています。

・ラジオボタンをグループへ登録　　→ setToggleGroup(gp);

複数のラジオボタンで、クラス ToggleGroup の同じオブジェクト gp を指定することで、グループが指定されます。

■アクションイベントを処理するイベントハンドラインタフェース EventHandler<ActionEvent>

イベントハンドラクラスの宣言は、ボタンの場合と同じように行います。

1. EventHandler<ActionEvent>インタフェースを実装してイベントハンドラクラスを宣言

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

■ラジオボタンへイベントハンドラを登録

ラジオボタンから発生するイベントを受け取ります。それぞれのラジオボタンにイベントハンドラを登録します。

〔コード例〕

```
1. MyEventHandler eh = new MyEventHandler();
2. rb.addEventHandler(ActionEvent.ANY, eh);
```

※オブジェクト eh をイベントハンドラとして RadioButton クラスのオブジェクト rb に登録します

■ラジオボタンと識別子

複数のラジオボタンを 1 つのイベントハンドラで処理するときは、ボタンの場合と同じように、識別子を用いてどのラジオボタンから発生したイベントなのかを判断します。

RadioButton クラスに識別子の設定と取得を行うメソッドが準備されています。

- 識別子の設定（"Open"を識別子として）　→ `setId("Open");`
- 識別子の取得（String 型）　→ `getId();`

■ラジオボタンの状態を設定したり確認したりするには

チェック状態を設定／取得するメソッドがクラス RadioButton に準備されています。

- チェック状態の設定　→ `setSelected(true);`
- チェック状態の取得（boolean 型）　→ `isSelected();`
  - 戻り値が true であればチェックされています。
  - 戻り値が false でればチェックされていません。

■利用したクラスの一覧

RadioButton クラス	
`RadioButton(String s){…}`	文字列 s をもつラジオボタンを生成します。
`void setId(String s){…}`	文字列 s をラジオボタンの識別子に設定します。
`String getId(){…}`	ラジオボタンの識別子を取得します。
`void addEventHandler(EventType<ActionEvent> e, EventHandler<ActionEvent> h){…}`	イベント e を受け取るハンドラ h を登録します。
`void setSelected(boolean b){…}`	チェック状態（true/false）を設定します。
`boolean isSelected(){…}`	チェック状態（true/false）を返します。
`void setToggleGroup(ToggleGroup g){…}`	ラジオボタンのグループを g に設定します、
ToggleGroup クラス	
`ToggleGroup(){…}`	ラジオボタンのグループを作ります。