10 回目　テキストフィールドとキーイベント

今日の講義で学ぶ内容

・テキストフィールドの利用
・キーイベントの処理

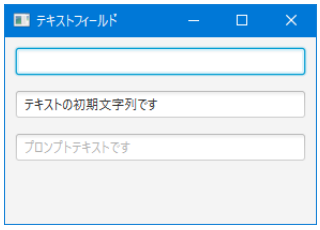
テキストフィールドの利用

§1　テキストフィールドを配置してみましょう

テキストフィールドを用いることにより、数値や文字列などのデータ入力が可能になります。

ソースファイル名：Sample10_1.java

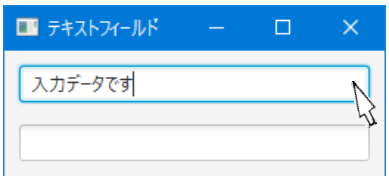
```
// ※HP よりインポート文をここへ貼り付けてください

// テキストフィールドの表示
public class Sample10_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // テキストフィールドを生成／設定します
      TextField[] tf = new TextField[3];
      tf[0] = new TextField();
      tf[1] = new TextField("テキストの初期文字列です");
      tf[2] = new TextField();
      tf[2].setPromptText("プロンプトテキストです");

      // レイアウト VBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(tf);
      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("テキストフィールド");

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



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

■テキストフィールドとは

キーボード入力によるテキストを受け付ける入力フィールドです。
入力フィールドをマウスでクリックしてフォーカスを移動した後、
キーボードから文字列を入力することができます。



■テキストフィールドクラス TextField

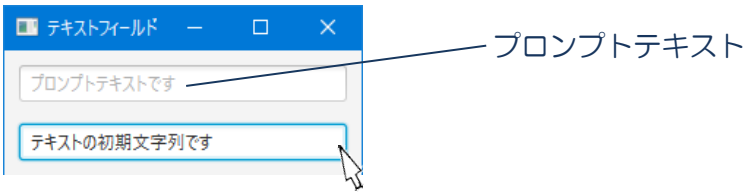
テキストフィールドはクラス TextField により表現され、各種設定を行うメソッドが準備されています。

・テキストフィールドの生成　　→ `new TextField();`
・テキストフィールドの初期文字列　→ `new TextField("テキストの初期文字列です");`
・プロンプトテキストの指定　　→ `setPromptText("プロンプトテキストです");`

クラス TextField のオブジェクトを生成するときに入力フィールドに最初に表示される文字列を指定します。指定しないときは、空の入力フィールドが生成されます。

■プロンプトテキストとは

入力フィールドに最初に表示される文字列です。初期文字列と異なり、薄い灰色で表示されます。キーボードから入力フィールドに文字列を入力すると、プロンプトテキストは消えます。プロンプトテキストには入力例や注意点を指定しておくと良いでしょう。



■利用したクラスの一覧

TextField クラス

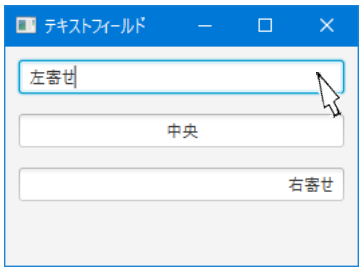
`TextField(){…}`	テキストフィールドを生成します。
`TextField(String txt){…}`	初期文字列 txt をもつテキストフィールドを生成します。
`void setPromptText(String txt){…}`	文字列 txt をプロンプトテキストに指定します。

§2　入力フィールド内の文字位置を調整してみましょう

入力フィールド内の文字列を左寄せや中央寄せ、右寄せで配置することができます。

ソースファイル名：Sample10_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// テキストフィールド内の文字位置
public class Sample10_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // テキストフィールドを生成／設定します
      TextField[] tf = new TextField[3];
      tf[0] = new TextField("左寄せ");
      tf[1] = new TextField("中央");
      tf[2] = new TextField("右寄せ");
      tf[0].setAlignment(Pos.TOP_LEFT);
      tf[1].setAlignment(Pos.TOP_CENTER);
      tf[2].setAlignment(Pos.TOP_RIGHT);

      // レイアウト VBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(tf);
      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("テキストフィールド");

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

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



■利用したクラスの一覧

TextField クラス

`void setAlignment(Pos p){…}`　　テキストフィールドの文字列の配置を p に指定します。

§3　キーイベントを取得してみましょう

入力フィールドでキー入力を行うと、キーを押すたびにキーイベントが発生します。

ソースファイル名：Sample10_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// キーイベントの取得
public class Sample10_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // テキストフィールドを生成／設定します
      TextField tf = new TextField();

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

      // 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(tf);
      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("テキストフィールド");

      // 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<KeyEvent>
   {
      public void handle(KeyEvent e)
      {
         EventType<KeyEvent> type = e.getEventType();
         if(type == KeyEvent.KEY_PRESSED){
            System.out.println("キーが押されました");
         }
         if(type == KeyEvent.KEY_RELEASED){
            System.out.println("キーが離されました");
         }
```

```
        if(type == KeyEvent.KEY_TYPED){
           String str=e.getCharacter();
           System.out.println("キーがタイプされました("+str+")");
        }
     }
  }

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

実行結果

```
キーが押されました          ← キー「A」を押す
キーがタイプされました(a)   ← キー「A」を離す
キーが離されました
  :
```

■キーイベントとは

キーイベントは、キーの入力によって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、各処理を実行することができます。

■キーイベントを表現するクラス KeyEvent

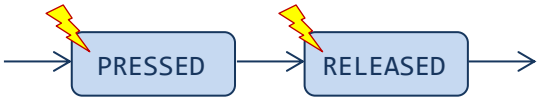
クラス KeyEvent により表現され、以下の種類があります。

- KeyEvent.KEY_PRESSED → キーが押されたときに発生します
- KeyEvent.KEY_RELEASED → キーが離されたときに発生します
- KeyEvent.KEY_TYPED → キーがタイプされた（押されて離された）ときに発生します

これらのイベントは次の順番で発生します。

□通常の文字キーの場合：

□特殊キー（ファンクションキー、Alt キー、Ctrl キーなど）の場合：



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。

- KeyEvent.ANY → 上記すべてのイベントを表現します

また、**イベント KEY_TYPED が発生したとき**、入力された文字を受け取ることができます。
・入力文字の取得（String 型）　→ getCharacter();

※他のイベント PRESSED と RELEASED のときは CHAR_UNDEFINED（String 型）を返します。

■キー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<KeyEvent>
キー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<KeyEvent>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(KeyEvent 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<KeyEvent>{
2.   public void handle(KeyEvent 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. }
```

■テキストフィールド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。キーイベントはテキストフィールドで受け取ることができます。テキストフィールドにイベントハンドラを登録します。

〔コード例〕
```
1. MyEventHandler kh = new MyEventHandler();
2. tf.addEventHandler(KeyEvent.ANY, kh);
```
※オブジェクト kh をイベントハンドラとして TextField クラスのオブジェクト tf に登録します

■利用したクラスの一覧

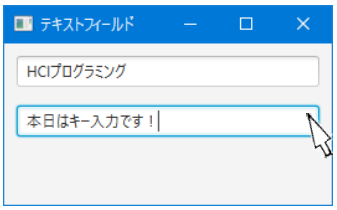
KeyEvent クラス	
KeyEvent.KEY_PRESSED	キーを押したときに発生するイベントです。
EventType<KeyEvent> getEventType(){…}	イベントの種類を取得します。
String getCharacter(){…}	入力文字を取得します。
EventHandler<KeyEvent>インタフェース	
void handle(KeyEvent e);	イベントが発生したときに実行されます。
TextField クラス	
void addEventHandler(EventType<KeyEvent> e, EventHandler<KeyEvent> h){…}	イベント e を受け取るハンドラ h を登録します。

§4　アクションイベントを取得してみましょう

入力フィールドに文字列を入力した後、リターンキーを押すとアクションイベントが発生します。

ソースファイル名：Sample10_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// アクションイベントの取得
public class Sample10_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // テキストフィールドを生成／設定します
      TextField tf1 = new TextField();
      TextField tf2 = new TextField();
      tf1.setId("textfield1");
      tf2.setId("textfield2");

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      tf1.addEventHandler(ActionEvent.ANY, actionhandler);
      tf2.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(tf1);
      lst.add(tf2);
      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("テキストフィールド");

      // 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)
      {
         TextField tf = (TextField)e.getTarget();
         System.out.println(tf.getId()+"/"+tf.getText());
      }
   }
```



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

実行結果

```
textfield1/HCI プログラミング      ← 上のテキストフィールドに入力後、リターンを押す
textfield2/本日はキー入力です！    ← 下のテキストフィールドに入力後、リターンを押す
  :
```

■テキストフィールドとアクションイベント

入力フィールドに文字列を入力したあとリターンキーを押すとアクションイベントが発生します。これらのイベントが発生したタイミングで、各処理を実行することができます。

■キーイベントとアクションイベントの違いは?

キーイベントは入力フィールドでキー入力があるたびに発生するイベントです。一方、アクションイベントは入力フィールドでリターンキーを押すと発生するイベントです。

キーイベントが発生したときに受け取ることができる文字データは**1文字**です。一方、アクションイベントが発生したときに受け取ることができる文字データは入力フィールドに**入力された文字列**です。

■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

■テキストフィールド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。アクションイベントはテキストフィールドで受け取ることができます。テキストフィールドにイベントハンドラを登録します。

〔コード例〕

```
1. MyEventHandler ah = new MyEventHandler();
2. tf.addEventHandler(ActionEvent.ANY, ah);
```

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

■テキストフィールドに入力された文字列を受け取るには?

クラス TextField のメソッドを用いて現在入力されている文字列を取り出すことができます。また、ボタンやラベルなどの GUI 部品と同じように識別子を設定／取得するメソッドもあります。

- 入力フィールド内の文字列の取得（String 型）　→ `getText();`
- 識別子の設定（"textfield1"を識別子として）　→ `setId("textfield1");`
- 識別子の取得（String 型）　→ `getId();`

■利用したクラスの一覧

TextField クラス

`void setOnAction(EventHandler<ActionEvent> h){…}`	イベントを受け取るハンドラ h を登録します。
`String getText(){…}`	入力フィールド内の文字列を取得します。