5 回目　ウィンドウに画像を表示してみよう

今日の講義で学ぶ内容

・画像の表示
・画像のエフェクト
・画像のビューポート指定

画像の表示

§1　画像を表示してみましょう

画像の表示はクラス ImageView により管理されます。

ソースファイル名：Sample5_1.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像の表示
public class Sample5_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像を生成／設定します
      ImageView iv = new ImageView("Sunset.jpg");

      // 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(iv);

      // 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("画像");

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

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

■画像の表示を管理するクラス ImageView

画像の表示はクラス ImageView により管理され、表示を処理するメソッドが準備されています。

・画像のオープン　　→ new ImageView("Sunset.jpg");

ファイル Sunset.jpg がオープンされ、オブジェクトに読み込まれます。

■画像ファイルの相対パス指定と絶対パス指定

・ファイルの相対パス指定

　→ new ImageView("Book/Sunset.jpg");

・ファイルの絶対パス指定（ドライブの指定があります）

　→ new ImageView("file:///C:/Book/Sunset.jpg");

・ファイルの絶対パス指定（ドライブ C:のルートとみなされます）

　→ new ImageView("/Book/Sunset.jpg");

・ネットワーク上のファイルの指定

　→ new ImageView("http://www.test.co.jp/Book/Sunset.jpg");

■レイアウト HBox に画像を配置するには

ボタンなどの GUI 部品と同じように、画像をレイアウトに配置できます。

〔コード例〕

```
1. HBox hb = new HBox();                              // レイアウトの生成
2. ObservableList<Node> lst = hb.getChildren();       // GUI 部品リストの取得
3. lst.add(iv);                                       // 画像の追加
```

※ImageView クラスのオブジェクト iv を GUI 部品リストに追加します

■利用したクラスの一覧

ImageView クラス	
ImageView(String s){…}	画像ファイル s をオープンします。（コンストラクタ）
ObservableList<Node>クラス	
boolean add(Node e){…}	新しい部品 e を GUI 部品リストに追加します。
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 ImageView クラスのスーパークラスです。

§2　画像のサイズを調べてみましょう

画像はクラス Image により表現されます。Image クラスのメソッドにより画像情報を取得できます。

ソースファイル名：Sample5_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像のサイズの表示
public class Sample5_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像を生成／設定します
      ImageView iv = new ImageView("Sunset.jpg");

      // 画像の属性を取得します
      Image img = iv.getImage();
      double width = img.getWidth();
      double height = img.getHeight();
      System.out.println("画像のサイズ　幅="+width+"　高さ="+height);

      // 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(iv);

      // 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("画像");

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

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

実行結果

```
画像のサイズ　幅=320.0　高さ=240.0
```

■画像を管理するクラス Image

画像はクラス Image により管理され、画像情報を取得するメソッドが準備されています。

・画像の横の長さ（ピクセル）の取得　→ getWidth();

・画像の縦の長さ（ピクセル）の取得　→ getHeight();

※この他、画像の形式情報や画素情報を取得するメソッドが準備されています。

■利用したクラスの一覧

ImageView クラス	
Image getImage(){…}	Image クラス型の画像オブジェクトを取得します。
Image クラス	
double getWidth(){…}	画像の横の長さ（ピクセル）を取得します。
double getHeight(){…}	画像の縦の長さ（ピクセル）を取得します。

§3　画像のレイアウト枠を変更してみましょう（1）

画像が描画されるレイアウト枠のサイズを変更することができます。

ソースファイル名：Sample5_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像を指定サイズへフィット
public class Sample5_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像を生成／設定します
      ImageView iv = new ImageView("Sunset.jpg");
      iv.setFitWidth(200);
      iv.setFitHeight(300);

      // 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(iv);

      // 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("画像");

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

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

■レイアウト枠のサイズを変更するには

ImageView クラスにレイアウト枠を指定するメソッドが準備されています。

・レイアウト枠の横サイズ（ピクセル）を指定　→ setFitWidth(200);

・レイアウト枠の縦サイズ（ピクセル）を指定　→ setFitHeight(300);

横が 200 ピクセル、縦が 300 ピクセルのレイアウト枠になります。画像はこの枠に収まるように引き伸ばされ、または縮められて描画されます。

§4　画像のレイアウト枠を変更してみましょう（2）

画像の縦横比を維持したままレイアウト枠のサイズを変更することができます。

ソースファイル名：Sample5_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像を縦横比を固定したまま指定サイズへフィット
public class Sample5_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像を生成／設定します
      ImageView iv = new ImageView("Sunset.jpg");
      iv.setPreserveRatio(true);
      iv.setFitWidth(200);

      // 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(iv);

      // 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("画像");

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

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

■レイアウト枠の縦横比を固定するには

ImageView クラスにレイアウト枠の縦横比を固定するメソッドが準備されています。

・レイアウト枠の縦横比の固定　　→ setPreserveRatio(true);

レイアウト枠の縦横比が固定された場合は、その横の長さまたは縦の長さの一方を設定します。

■利用したクラスの一覧

ImageView クラス

`void setFitWidth(double v){…}`	レイアウト枠の横の長さ（ピクセル）を v にします。
`void setFitHeight(double v){…}`	レイアウト枠の縦の長さ（ピクセル）を v にします。
`void setPreserveRatio(boolean b){…}`	引数が true の場合、レイアウト枠の縦横比を固定します。

§5　複数の画像を表示してみましょう

複数の ImageView オブジェクトを準備して複数の画像を表示できます。

ソースファイル名：Sample5_5.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 複数の画像の表示
public class Sample5_5 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ４枚の画像を生成／設定します
      ImageView[] ivs = new ImageView[4];
      ivs[0] = new ImageView("Sunset.jpg");
      ivs[1] = new ImageView("Beach.jpg");
      ivs[2] = new ImageView("Cafe.jpg");
      ivs[3] = new ImageView("Food.jpg");

      // 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(ivs);

      // 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("画像");

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

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

§6　画像にセピアやぼかしのエフェクトをつけてみましょう

エフェクトを表現するクラスを用いて、画像に様々なエフェクトをつけることができます。

ソースファイル名：Sample5_6.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像のエフェクト（セピアとぼかし）
public class Sample5_6 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ４枚の画像を生成／設定します
      ImageView[] ivs = new ImageView[4];
      ivs[0] = new ImageView("Sunset.jpg");
      ivs[1] = new ImageView("Sunset.jpg");
      ivs[2] = new ImageView("Cafe.jpg");
      ivs[3] = new ImageView("Cafe.jpg");

      // エフェクトを生成し、画像に適用します
      SepiaTone spa = new SepiaTone();
      spa.setLevel(0.9);
      ivs[1].setEffect(spa);
      GaussianBlur gb = new GaussianBlur();
      gb.setRadius(10.0);
      ivs[3].setEffect(gb);

      // 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(ivs);

      // 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("画像");

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

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

■セピア（エフェクト）を表現するクラス SepiaTone

セピアはクラス SepiaTone で表現されます。エフェクトの強さなどの設定を行うことができます。

・セピアエフェクトの生成　　→ `new SepiaTone();`

・エフェクトの強さ設定　　→ `setLevel(0.9);`

セピアエフェクトを生成し、強さを 0.9 に設定します。強さは最大 1.0 から最小 0.0 で設定します。

強さ 0.2

強さ 0.5

強さ 0.8

■ぼかし（エフェクト）を表現するクラス GaussianBlur

ぼかしはクラス GaussianBlur で表現されます。エフェクトの強さなどの設定を行うことができます。

・ぼかしエフェクトの生成　　→ `new GaussianBlur();`

・ぼかし半径の設定　　→ `setRadius(10.0);`

ぼかしエフェクトを生成し、ぼかし半径を 10.0 にします。半径は最大 63.0 から最小 0.0 で設定します。

ぼかし半径 2.0

ぼかし半径 8.0

ぼかし半径 32.0

■画像にエフェクトをつけるには

クラス ImageView には各種エフェクトをつけるメソッドが準備されています。

・画像にエフェクトを設定　　→ `setEffect(…);`

引数にエフェクトを表現するクラスのオブジェクトを渡すと、そのエフェクトが画像に施されます。

■利用したクラスの一覧

ImageView クラス	
void setEffect(Effect e){…}	エフェクト e を画像に適用します。 ※クラス Effect はクラス SepiaTone と クラス GaussianBlur のスーパークラスです。
SepiaTone クラス	
SepiaTone(){…}	セピアエフェクトを生成します。（コンストラクタ）
void setLevel(double v){…}	セピアの強さ（0.0 ~ 1.0）を設定します。
GaussianBlur クラス	
GaussianBlur(){…}	ぼかしエフェクトを生成します。（コンストラクタ）
void setRadius(double v){…}	ぼかし半径（0.0 ~ 63.0）を設定します。

§7　画像にインナシャドーや輝きのエフェクトをつけてみましょう

エフェクトを表現するクラスを用いて、画像に様々なエフェクトをつけることができます。

ソースファイル名：Sample5_7.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像のエフェクト（インナシャドーと輝き）
public class Sample5_7 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ４枚の画像を生成／設定します
      ImageView[] ivs = new ImageView[4];
      ivs[0] = new ImageView("Sunset.jpg");
      ivs[1] = new ImageView("Sunset.jpg");
      ivs[2] = new ImageView("Cafe.jpg");
      ivs[3] = new ImageView("Cafe.jpg");

      // エフェクトを生成し、画像に適用します
      InnerShadow ins = new InnerShadow();
      ins.setOffsetX(4);
      ins.setOffsetY(4);
      ivs[1].setEffect(ins);
      Glow gw = new Glow();
      gw.setLevel(0.6);
      ivs[3].setEffect(gw);

      // 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(ivs);

      // 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("画像");

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

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

■インナシャドー（エフェクト）を表現するクラス InnerShadow

インナシャドーはクラス InnerShadow で表現されます。影の大きさなどの設定を行うことができます。

- インナシャドーエフェクトの生成　→ new InnerShadow();
- 影の長さ（X 軸方向）　→ setOffsetX(4);
- 影の長さ（Y 軸方向）　→ setOffsetY(4);

X 軸方向 4 ピクセル、Y 軸方向 4 ピクセルの長さの影を生成します。

X=2, Y=2

X=8, Y=8

X=2, Y=8

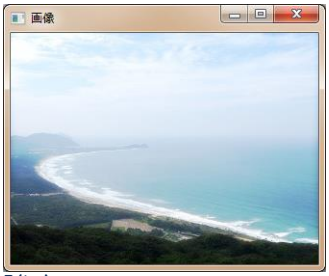
■輝き（エフェクト）を表現するクラス Glow

輝きはクラス Glow で表現されます。エフェクトの強さなどの設定を行うことができます。

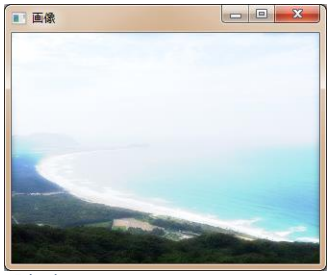
- 輝きエフェクトの生成　→ new Glow();
- 輝きの強さの設定　→ setLevel(0.6);

輝きエフェクトを生成し、強さを 0.6 に設定します。強さは最大 1.0 から最小 0.0 で設定します。

強さ 0.2



強さ 0.5



強さ 0.8

■利用したクラスの一覧

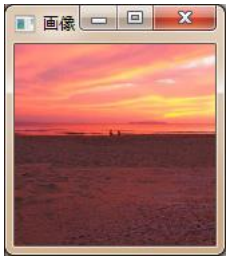
InnerShadow クラス	
InnerShadow(){…}	インナシャドーエフェクトを生成します。（コンストラクタ）
void setOffsetX(double v){…}	影の X 軸方向の長さ（ピクセル）を v に設定します。
void setOffsetY(double v){…}	影の Y 軸方向の長さ（ピクセル）を v に設定します。
Glow クラス	
Glow(){…}	輝きエフェクトを生成します。（コンストラクタ）
void setLevel(double v){…}	輝きの強さ（0.0 ~ 1.0）を v に設定します。

§8　画像の部分を切り出してみましょう

ビューポートを用いて、画像の一部分を切り出すことができます。

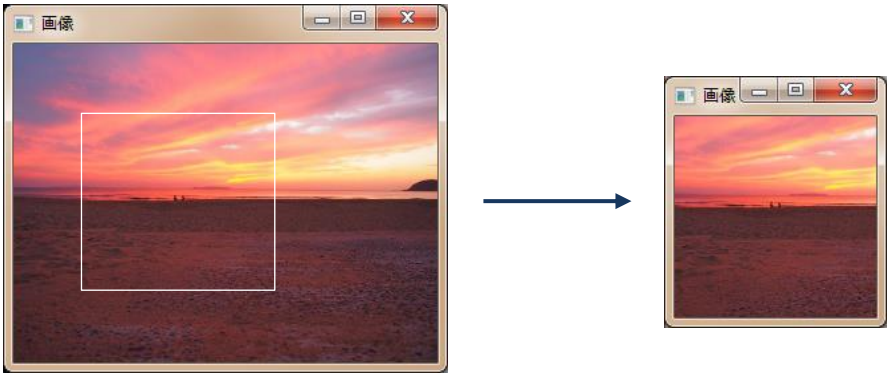
ソースファイル名：Sample5_8.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像のビューポート指定
public class Sample5_8 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像を生成／設定します
      ImageView iv = new ImageView("Sunset.jpg");

      // ビューポートを生成し、画像に適用します
      Rectangle2D rt = new Rectangle2D(50, 50, 150, 150);
      iv.setViewport(rt);

      // 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(iv);

      // 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("画像");

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

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



■ビューポートとは

ビューポートは画像を切り出す矩形です。矩形の中に入る画像が切り出され表示されます。



ビューポートはクラス Rectangle2D により表現されます。

・ビューポートの指定　　→ `new Rectangle2D(50, 50, 150, 150);`

始点座標（50,50）から幅 150 ピクセル、高さ 150 ピクセルの矩形（ビューポート）が生成されます。

■画像にビューポートを指定するには

クラス ImageView にビューポートを指定するメソッドが準備されています。

・ビューポートの指定　　→ `setViewport(…);`

引数にビューポートを表す Rectangle2D クラスのオブジェクトを渡します。指定した大きさの矩形で画像が切り出されます。

■利用したクラスの一覧

ImageView クラス

`void setViewport(Rectangle2D r){…}`　ビューポート r を画像に設定します。

Rectangle2D クラス

`Rectangle2D(double minX, double minY, double width, double height){…}`

指定された大きさの矩形を生成します。（コンストラクタ）