6 回目　いろいろなラベルを作ってみよう

今日の講義で学ぶ内容

・ラベルの表示
・ツールチップの表示
・マウスカーソルの変更

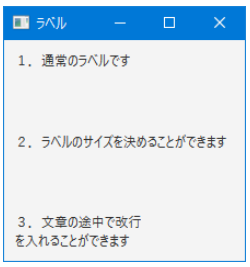
ラベルの表示

§1　ラベルを表示してみましょう

ラベルはクラス Label により管理されます。

ソースファイル名：Sample6_1.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ラベルの表示
public class Sample6_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ラベルを生成／設定します
      Label[] lb = new Label[3];
      lb[0] = new Label("１．通常のラベルです");
      lb[1] = new Label("２．ラベルのサイズを決めることができます");
      lb[2] = new Label("３．文章の途中で改行¥n を入れることができます");
      lb[1].setPrefSize(200,100);

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



■ラベルを管理するクラス Label

ラベルはクラス Label により管理され、各種設定を行うメソッドが準備されています。

・ラベルの生成　　→ new Label("１．通常のラベルです");
・ラベルのサイズ　　→ setPrefSize(200,100);

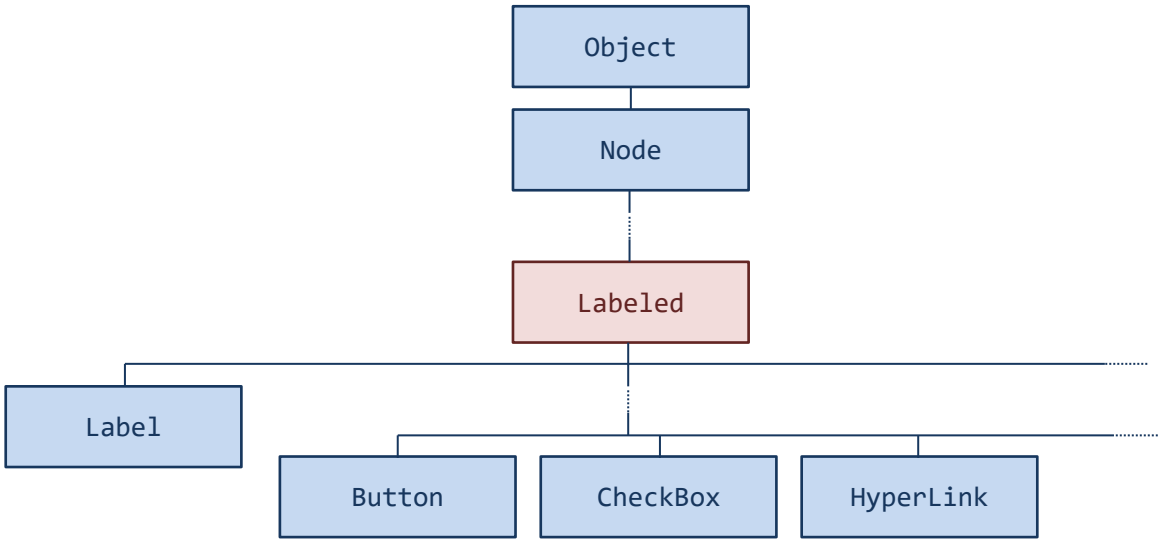
ラベル文字が「１．通常のラベルです」で横 200 ピクセル×縦 100 ピクセルのラベルが作成されます。

■GUI 部品がもつ文字表示を管理するクラス Labeled

ボタンやチェックボックス、メニュー、リストなどの文字列と関係する GUI 部品は Labeled クラスをスーパークラスに持ちます。代表的なサブクラスとして次のようなクラスがあります。

- **ラベル　Label**
- **ボタン　Button, CheckBox, HyperLink, MenuButton, ToggleButon**
- **リスト　ListCell, CheckBoxListCell, ChoiceBoxListCell, …**
- **テーブル　TableCell, CheckBoxTableCell, ChoiceBoxTableCell, …**
- **ツリー　TreeCell, CheckBoxTreeCell, ChoiceBoxTreeCell, …**

講義ではラベルを管理する Label クラスを例にして利用の仕方を学習します。本講義で紹介するメソッドは全て Labeled クラスから継承されたものですので、上記の他の GUI 部品でも同じように使うことができます。



■利用したクラスの一覧

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

Label(String s){…}　　文字列 s をもつラベルを生成します。

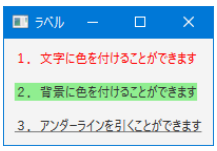
void setPrefSize(double w, double h){…}
　　サイズを横 w ピクセル×縦 h ピクセルに設定します。

§2　ラベル文字を修飾してみましょう

Label クラスのメソッドを用いて、ラベル文字を修飾することができます。

ソースファイル名：Sample6_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ラベル文字の修飾
public class Sample6_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ラベルを生成／設定します
      Label[] lb = new Label[3];
      lb[0] = new Label("１．文字に色を付けることができます");
      lb[1] = new Label("２．背景に色を付けることができます");
      lb[2] = new Label("３．アンダーラインを引くことができます");
      lb[0].setTextFill(Color.RED);
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGREEN,null,null)));
      lb[2].setUnderline(true);

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



■ラベル文字を修飾するには

ラベルはクラス Label により管理され、ラベル文字の修飾設定を行うメソッドが準備されています。

・ラベル文字の色　→ setTextFill(Color.RED);
・ラベル文字の背景色　→ setBackground(…(Color.LIGHTGREEN,null,null));
・ラベル文字のアンダーライン　→ setUnderline(true);

文字の色が赤で背景色が薄緑のアンダーラインのついたラベル文字になります。

■利用したクラスの一覧

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

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 setBackground(Background b){…}	ラベルの背景を指定した背景素材 b に設定します。
void setUnderline(boolean b){…}	引数が true のときアンダーラインを設定します。

Background クラス ← Object クラス

Background(BackgroundFill b){…}	塗りつぶし設定 b をもつ背景素材を生成します。

BackgroundFill クラス ← Object クラス

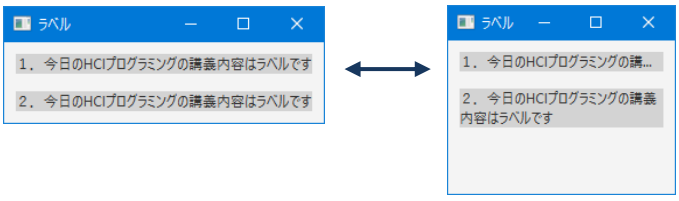
BackgroundFill(Paint p, CornerRadii r, Insets i){…}	塗りつぶし色 p と角の丸み（半径）r、外枠の幅 i の塗りつぶし設定を生成します。 ・p が null の場合、色は透明に設定 ・r が null の場合、角の丸みなしに設定 ・i が null の場合、外枠の幅なしに設定

§3　ラベル文字を自動改行してみましょう

ウィンドウ枠に応じてラベル文字を自動改行することができます。

ソースファイル名：Sample6_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ラベル文字の自動改行
public class Sample6_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ラベルを生成／設定します
      Label[] lb = new Label[2];
      lb[0] = new Label("１．今日の HCI プログラミングの講義内容はラベルです");
      lb[1] = new Label("２．今日の HCI プログラミングの講義内容はラベルです");
      lb[1].setWrapText(true);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



■ラベル文字を自動改行するには

Label クラスに自動改行の有無を指定するメソッドが準備されています。

・ラベル文字の自動改行　　→ setWrapText(true);

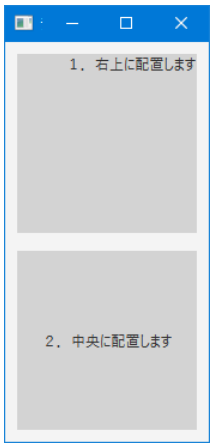
ラベル文字がウィンドウ枠に収まらない場合、下の行へ自動的に改行されます。

§4　ラベル文字の表示位置を変えてみましょう

ラベルのサイズが縦横に広い場合、ラベル文字の表示位置を指定することができます。

ソースファイル名：Sample6_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ラベル文字の表示位置合わせ
public class Sample6_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ラベルを生成／設定します
      Label[] lb = new Label[2];
      lb[0] = new Label("１．右上に配置します");
      lb[1] = new Label("２．中央に配置します");
      lb[0].setPrefSize(150,150);
      lb[1].setPrefSize(150,150);
      lb[0].setAlignment(Pos.TOP_RIGHT);
      lb[1].setAlignment(Pos.CENTER);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



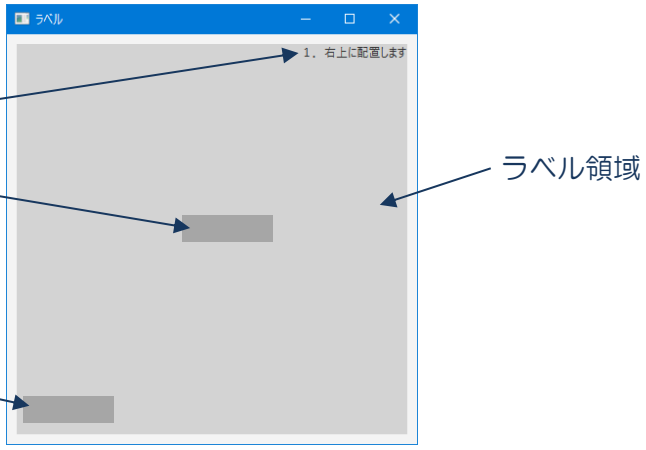
■ラベル上のラベル文字の表示位置を変更するには

Label クラスにラベル文字の表示位置を指定するメソッドが準備されています。

・ラベル文字の表示位置（右上の場合）　→ `setAlignment(Pos.TOP_RIGHT);`

ラベルのサイズが十分に広いとき、ラベル文字は右上に表示されます。この他、次の指定ができます。

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



■利用したクラスの一覧

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

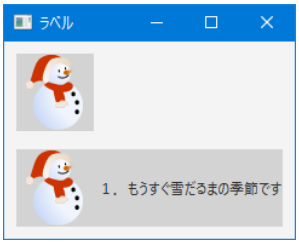
`void setWrapText(boolean b){…}`	引数が true のとき、ラベル文字の自動改行を行います。
`void setAlignment(Pos p){…}`	ラベル文字の表示位置を p にします。

§5　イメージラベルを表示してみましょう

ラベルに画像を貼ることもできます。

ソースファイル名：Sample6_5.java

```
// ※HP よりインポート文をここへ貼り付けてください

// イメージラベルの表示
public class Sample6_5 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像ファイルを準備します
      ImageView icon1 = new ImageView("snowman.png");
      ImageView icon2 = new ImageView("snowman.png");

      // ラベルを生成／設定します
      Label[] lb = new Label[2];
      lb[0] = new Label();
      lb[1] = new Label("１．もうすぐ雪だるまの季節です");
      lb[0].setGraphic(icon1);
      lb[1].setGraphic(icon2);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



■ラベルに画像を貼るには

Label クラスに画像の表示を管理する ImageView クラスのオブジェクトをイメージラベルとして指定するメソッドが準備されています。

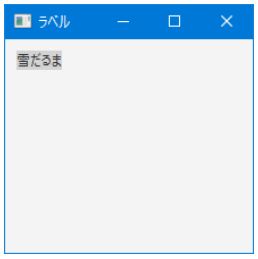
・画像をラベルに貼る　　→ setGraphic(icon1);

ImageView クラスのオブジェクト icon1 をラベルに貼ります。

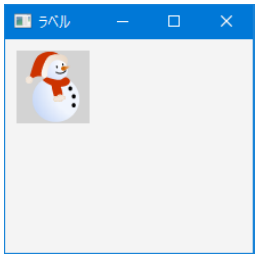
■ラベル文字と画像を同時に表示すると

ラベルには、ラベル文字のみ、画像のみ、ラベル文字と画像の３パターンで表示が可能です。

・ラベル文字のみ

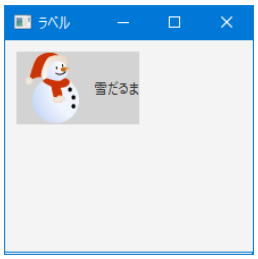


・画像のみ



・ラベル文字と画像

※ラベル文字は標準で右側に表示されます



■利用したクラスの一覧

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

void setGraphic(Node n){…}　　GUI 部品 n をラベルに貼ります。

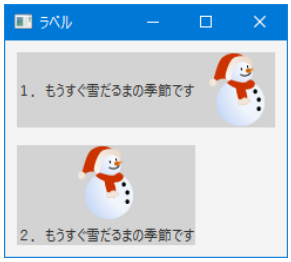
※Node クラスは ImageView クラスのスーパークラスです。

§6　イメージラベルとラベル文字の相対位置を調整してみましょう

ラベル上のラベル文字と画像の相対位置を変更することができます。

ソースファイル名：Sample6_6.java

```
// ※HP よりインポート文をここへ貼り付けてください

// イメージとラベル文字の位置調整
public class Sample6_6 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像ファイルを準備します
      ImageView icon1 = new ImageView("snowman.png");
      ImageView icon2 = new ImageView("snowman.png");

      // ラベルを生成／設定します
      Label[] lb = new Label[2];
      lb[0] = new Label("１．もうすぐ雪だるまの季節です");
      lb[1] = new Label("２．もうすぐ雪だるまの季節です");
      lb[0].setGraphic(icon1);
      lb[1].setGraphic(icon2);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // ラベルと画像の相対位置の調整
      lb[0].setContentDisplay(ContentDisplay.RIGHT);
      lb[1].setContentDisplay(ContentDisplay.TOP);

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



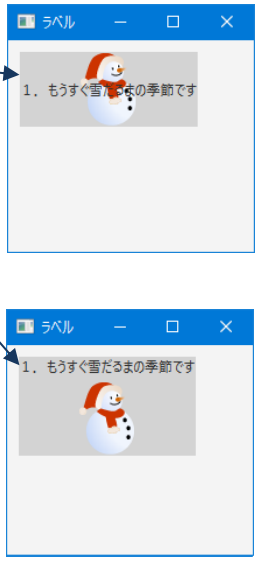
■ラベル文字と画像の相対位置を変更するには

Label クラスにラベル文字と画像の位置関係を５通りから指定できるメソッドが準備されています。

・ラベル文字と画像の位置関係（画像を上に）　→ setContentDisplay(ContentDisplay.TOP);

画像をラベル文字の上（ContentDisplay.TOP）に置きます。この他、次の指定ができます。

ContentDisplay.TOP	画像を上に
ContentDisplay.CENTER	画像を中央に
ContentDisplay.BOTTOM	画像を下に
ContentDisplay.LEFT	画像を左に
ContentDisplay.RIGHT	画像を右に



■利用したクラスの一覧

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

void setContentDisplay(ContentDisplay c){…} 画像の相対位置を c に設定します。

ContentDisplay 列挙型 ← Enum<ContentDisplay>クラス ← Object クラス

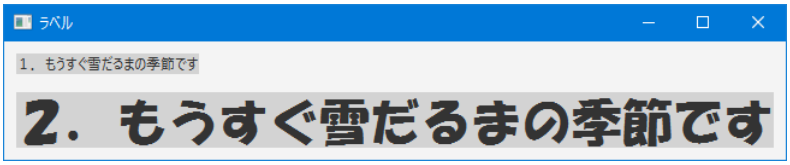
ContentDisplay.TOP　　ラベル上のコンテンツの位置 TOP を表します。

§7　ラベル文字のフォントを指定してみましょう

ラベル文字のフォントとしてパソコンにインストールしてあるフォントを指定できます。

ソースファイル名：Sample6_7.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ラベル文字のフォント指定
public class Sample6_7 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // フォントを生成します
      Font ft1 = new Font(12);
      Font ft2 = new Font("HG 創英角ﾎﾟｯﾌﾟ体",42);

      // ラベルを生成／設定します
      Label[] lb = new Label[2];
      lb[0] = new Label("１．もうすぐ雪だるまの季節です");
      lb[1] = new Label("２．もうすぐ雪だるまの季節です");
      lb[0].setFont(ft1);
      lb[1].setFont(ft2);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



■フォントを管理するクラス Font

フォントはクラス Font により管理され、各種設定を行うコンストラクタが準備されています。

・指定サイズのデフォルトフォントの生成　　→ new Font(12);

・指定サイズの指定フォントの生成　　→ new Font("HG 創英角ﾎﾟｯﾌﾟ体",42);

それぞれ 12 ポイントのデフォルトフォント、42 ポイントの HG 創英角ポップ体フォントを生成します。指定するフォント名は英単語で表現されるもので HP に一覧を載せていますのでご覧ください。以下に一部を載せております。

〔フォント一覧（一部)〕

MS Gothic	HGP 教科書体	HGS 明朝 E
MS Mincho	HGP 明朝 B	HGS 行書体
MS Outlook	HGP 明朝 E	HGS ｺﾞｼｯｸ E
MS PGothic	HGP 行書体	HGS ｺﾞｼｯｸ M
MS PMincho	HGP ｺﾞｼｯｸ E	HG 丸ｺﾞｼｯｸ M-PRO
MS Reference Sans Serif	HGP ｺﾞｼｯｸ M	HG 創英角ｺﾞｼｯｸ UB
MS Reference Specialty	HGS 創英角ｺﾞｼｯｸ UB	HG 創英角ﾎﾟｯﾌﾟ体
MS UI Gothic	HGS 創英角ﾎﾟｯﾌﾟ体	HG 創英ﾌﾟﾚｾﾞﾝｽ EB
HGP 創英角ｺﾞｼｯｸ UB	HGS 創英ﾌﾟﾚｾﾞﾝｽ EB	HG 教科書体
HGP 創英角ﾎﾟｯﾌﾟ体	HGS 教科書体	HG 明朝 B
HGP 創英ﾌﾟﾚｾﾞﾝｽ EB	HGS 明朝 B	HG 明朝 E

■ラベル文字のフォントを指定するには

Label クラスにラベル文字のフォントを指定できるメソッドが準備されています。

・ラベル文字のフォント指定　　→ setFont(ft1);

Font クラスのオブジェクト ft1 がラベル文字のフォントになります。

■利用したクラスの一覧

Font クラス ← Object クラス

Font(double d){…}	サイズが d ポイントのデフォルトフォントを生成します。
Font(String s, double d){…}	サイズが d ポイントのフォント s を生成します。

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

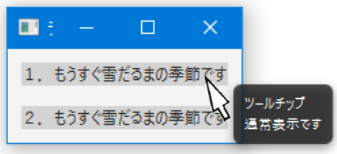
void setFont(Font f){…}	フォントを f に設定します。

§8　ラベルにツールチップをつけてみましょう（1）

ツールチップを表現するクラスを用いて、ラベルに様々な情報を付加することができます。

ソースファイル名：Sample6_8.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ツールチップの指定１
public class Sample6_8 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ツールチップを生成します
      Tooltip tp1 = new Tooltip("ツールチップ¥n 通常表示です");
      Tooltip tp2 = new Tooltip("ツールチップ¥n サイズを決めることもできます");
      tp2.setPrefSize(200,200);

      // ラベルを生成／設定します
      Label[] lb = new Label[2];
      lb[0] = new Label("１．もうすぐ雪だるまの季節です");
      lb[1] = new Label("２．もうすぐ雪だるまの季節です");
      lb[0].setTooltip(tp1);
      lb[1].setTooltip(tp2);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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

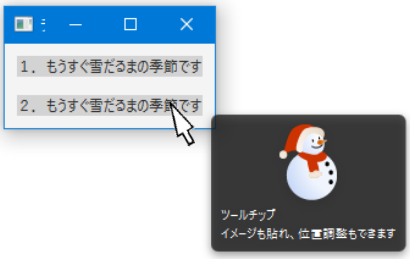


§9　ラベルにツールチップをつけてみましょう（2）

ツールチップの文字のフォントを変更したり、画像を貼ったりすることができます。

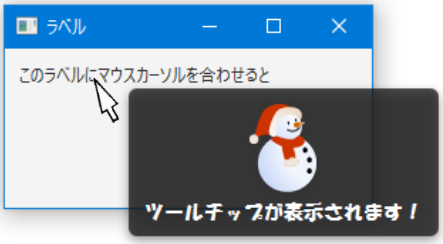
ソースファイル名：Sample6_9.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ツールチップの指定２
public class Sample6_9 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // 画像ファイルを準備します
      ImageView icon=new ImageView("snowman.png");

      // フォントを生成します
      Font ft = new Font("HG 創英角ﾎﾟｯﾌﾟ体",42);

      // ツールチップを生成します
      Tooltip tp1 = new Tooltip("ツールチップ¥n フォントも指定できます");
      Tooltip tp2 = new Tooltip("ツールチップ¥n イメージも貼れ、位置調整もできます");
      tp1.setFont(ft);
      tp2.setGraphic(icon);
      tp2.setContentDisplay(ContentDisplay.TOP);

      // ラベルを生成／設定します
      Label[] lb = new Label[2];
      lb[0] = new Label("１．もうすぐ雪だるまの季節です");
      lb[1] = new Label("２．もうすぐ雪だるまの季節です");
      lb[0].setTooltip(tp1);
      lb[1].setTooltip(tp2);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



■ツールチップとは

ボタンやテキストボックスなどの GUI 部品に注釈を付加する方法の 1 つです。マウスを GUI 部品に合せると、周辺に新しい領域が出現し、説明や詳細などの付加的な情報が表示されます。



■ツールチップを表現するクラス Tooltip

ツールチップはクラス Tooltip で表現されます。メッセージやフォント、画像を貼ったりするなど各種設定を行うことができます。

・ツールチップの生成とメッセージ設定　→ new Tooltip("ツールチップ¥n 通常表示です");
・ツールチップのサイズ　→ setPrefSize(200,200);
・メッセージのフォントの指定　→ setFont(ft);
・画像の指定　→ setGraphic(icon);
・メッセージと画像の相対位置　→ setContentDisplay(ContentDisplay.TOP);

Font クラスのオブジェクト ft と ImageView クラスのオブジェクト icon を、それぞれのツールチップのフォントと画像に指定し、文字列「ツールチップ…」の上に画像が配置される横 200 ピクセル×縦 200 ピクセルのツールチップを生成します。

■ラベルにツールチップをつけるには

クラス Label にツールチップをつけるメソッドが準備されています。

・ラベルにツールチップを設定　→ setTooltip(tp1);

Tooltip クラスオブジェクト tp1 をラベルのツールチップに設定します。

■利用したクラスの一覧

Tooltip クラス ← PopupControl クラス ← PopupWindow クラス ← Window クラス ← Object クラス

Tooltip(String s){…}	文字列 s のツールチップを生成します。
void setPrefSize(double w, double h){…}	サイズを横 w ピクセル、縦 h ピクセルに設定します。
void setFont(Font f){…}	フォント f をツールチップに適用します。
void setGraphic(Node n){…}	GUI 部品 n をツールチップに貼り付けます。
	※Node クラスは ImageView クラスのスーパークラスです。

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

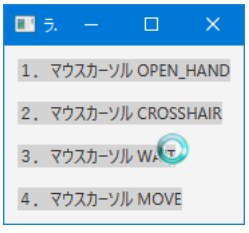
void setTooltip(Tooltip t){…}	ツールチップ t をラベルに設定します。

§10　マウスカーソルを自由に変更してみましょう

GUI 部品に応じてマウスカーソルを手アイコンや移動アイコンなどに変更することができます。

ソースファイル名：Sample6_10.java

```
// ※HP よりインポート文をここへ貼り付けてください

// マウスカーソルの変更
public class Sample6_10 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ラベルを生成／設定します
      Label[] lb = new Label[4];
      lb[0] = new Label("１．マウスカーソル OPEN_HAND");
      lb[1] = new Label("２．マウスカーソル CROSSHAIR");
      lb[2] = new Label("３．マウスカーソル WAIT");
      lb[3] = new Label("４．マウスカーソル MOVE");
      lb[0].setCursor(Cursor.OPEN_HAND);
      lb[1].setCursor(Cursor.CROSSHAIR);
      lb[2].setCursor(Cursor.WAIT);
      lb[3].setCursor(Cursor.MOVE);
      lb[0].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[1].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[2].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));
      lb[3].setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.addAll(lb);
      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("ラベル");

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

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



■マウスカーソルの種類

マウスカーソルはクラス Cursor で管理されています。以下のようなカーソルの種類があります。

Cursor.CLOSED_HAND		Cursor.CROSSHAIR	
Cursor.V_RESIZE		Cursor.H_RESIZE	
Cursor.HAND		Cursor.MOVE	
Cursor.OPEN_HAND		Cursor.TEXT	
Cursor.WAIT		などなど	

※この他、利用可能なすべてのカーソルはクラス Cursor に宣言されています。

■カーソルをラベルに指定するには

クラス Label にカーソルを変更するメソッドが準備されています。

・カーソルの指定　　→ setCursor(Cursor.OPEN_HAND);

このラベルにマウスカーソルを合わせると、OPEN_HAND の形状になります。

■利用したクラスの一覧

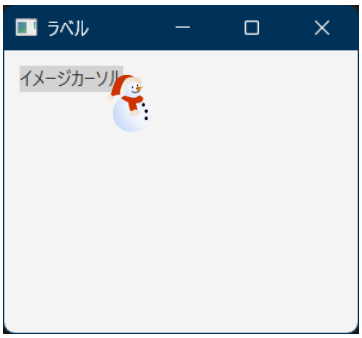
Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス	
void setCursor(Cursor c){…}	カーソル c をラベルに設定します。
Cursor クラス ← Object クラス	
Cursor.OPEN_HAND	手形状のマウスカーソルです。
Cursor.CROSSHAIR	照準形状のマウスカーソルです。
Cursor.WAIT	処理待ちを表すマウスカーソルです。
Cursor.MOVE	ドラッグ移動を表すマウスカーソルです。

§11　イメージカーソルを設定してみましょう

マウスカーソルを好みのアイコン画像に変更することができます。

ソースファイル名：Sample6_11.java

```
// ※HP よりインポート文をここへ貼り付けてください

// マウスカーソルの変更
public class Sample6_10 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // イメージカーソルを生成します
      ImageCursor icr = new ImageCursor(new Image("snowman.png"));

      // ラベルを生成／設定します
      Label lb = new Label("イメージカーソル");
      lb.setCursor(icr);
      lb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // レイアウト HBox を生成／設定します
      VBox vb = new VBox();
      ObservableList<Node> lst = vb.getChildren();
      lst.add(lb);
      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("ラベル");

      // 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

Label クラス←Labeled クラス←Control クラス←Region クラス←Parent クラス←Node クラス←Object クラス

void setCursor(Cursor c){…}	カーソル c をラベルに設定します。

ImageCursor クラス ← Cursor クラス ← Object クラス

ImageCursor(Image img){…}	アイコン画像 img をもつマウスカーソルを生成します。