11 回目　グラフィックス

今日の講義で学ぶ内容

・キャンバスと図形描画
・マウスを用いたインタラクション
・ラジオボタンなど GUI 部品を用いたインタラクション

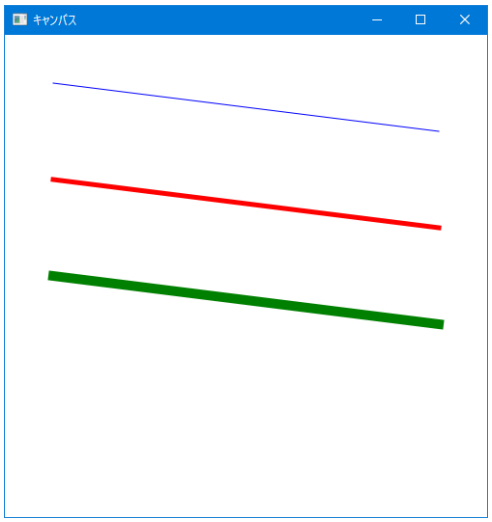
キャンバスと図形描画

§1　キャンバスに線を引いてみましょう

画用紙を表すキャンバスに図形を描くことができます。

ソースファイル名：Sample11_1.java

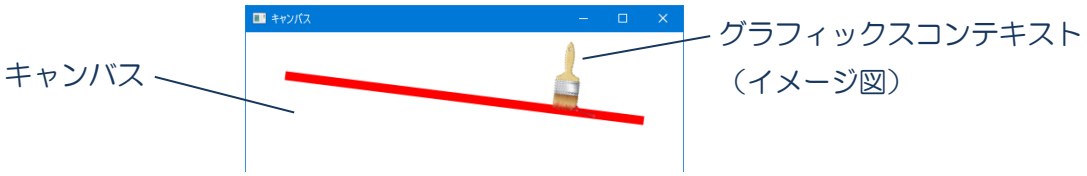
```
// ※HP よりインポート文をここへ貼り付けてください

// 線の描画
public class Sample11_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // キャンバスを生成／設定します
      Canvas cv = new Canvas(500,500);
      GraphicsContext gc = cv.getGraphicsContext2D();

      // 線を引きます
      gc.setLineWidth(1);
      gc.setStroke(Color.BLUE);
      gc.strokeLine(50.0, 50.0, 450.0, 100.0);

      gc.setLineWidth(5);
      gc.setStroke(Color.RED);
      gc.strokeLine(50.0, 150.0, 450.0, 200.0);

      gc.setLineWidth(10);
      gc.setStroke(Color.GREEN);
      gc.strokeLine(50.0, 250.0, 450.0, 300.0);

      // 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(cv);

      // 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("キャンバス");

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

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

■キャンバスとは

キャンバスは、線や矩形などの図形や画像などを描くことができる画用紙を表します。またグラフィックスコンテキストは、図形などを描くときの筆やペンを表します。グラフィックスコンテキストを用いてキャンバスに図形や画像を描画します。



■キャンバスクラス Canvas

キャンバスはクラス Canvas 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。

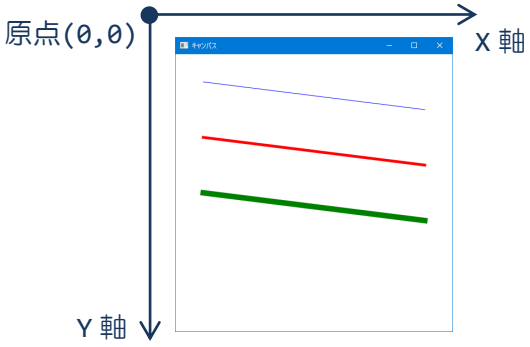
・キャンバスの生成（500x500px）　→ `new Canvas(500,500);`

・グラフィックスコンテキストの取得 → `getGraphicsContext2D();`

クラス Canvas のオブジェクトを生成するときにキャンバスのサイズを指定します。また、生成したキャンバスに図形を描くためのグラフィクスコンテキストを取得することができます。

■キャンバスの座標系は?

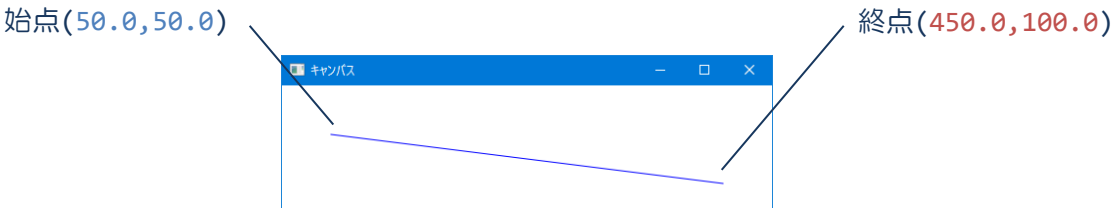
キャンバスは左上隅が原点(0,0)です。X 軸正は右方向であり、Y 軸正は下方向です。図形を描く際の点の指定はこの座標系で行います。



■グラフィックスコンテキストクラス GraphicsContext

グラフィックスコンテキストはクラス GraphicsContext により表現され、図形を描画する各種設定を行うメソッドが準備されています。

- 線の太さの設定（1px）　→ `setLineWidth(1);`
- 線の色を設定（青）　→ `setStroke(Color.BLUE);`
- 線を描画　→ `strokeLine(50.0, 50.0, 450.0, 100.0);`



線の太さを 1px に、色を青色に設定し、線を始点(50.0,50.0)から終点(450.0,100.0)まで引きます。
線を描くときは、そのときに設定されている線の太さと色で描画されます。

■利用したクラスの一覧

Canvas クラス←Node←Object

`Canvas(double w, double h){…}`　幅 w ピクセル、高さ h ピクセルのキャンバスを生成します。

`GraphicsContext getGraphicsContext2D(){…}`
キャンバス用のグラフィックスコンテキストを取得します。

GraphicsContext クラス←Object

`void setLineWidth(double w){…}`　線の太さを w ピクセルに設定します。

`void setStroke(Paint p){…}`　線の色を p に設定します。
※クラス Paint はクラス Color のスーパークラスです。

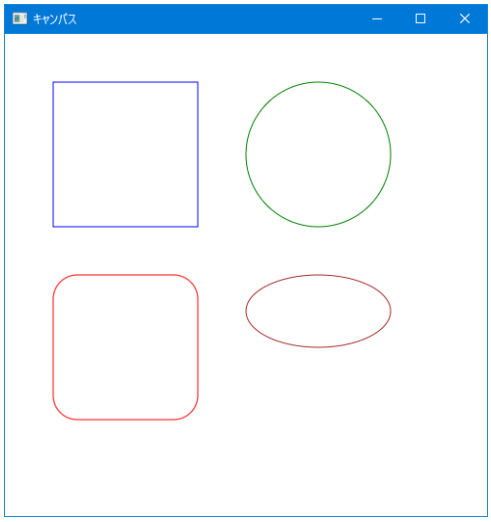
`void strokeLine(double x1, double y1, double x2, double y2){…}`
始点(x1,y1)から終点(x2,y2)までの線を引きます。

§2　キャンバスに図形を描いてみましょう（1）

四角や円などの基本的な図形をキャンバスに描くことができます。

ソースファイル名：Sample11_2.java

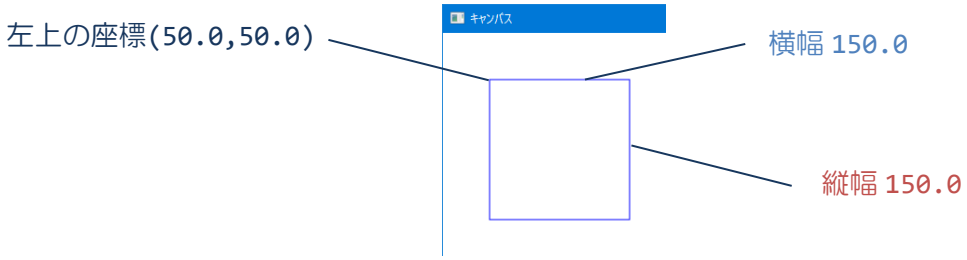
```
// ※HP よりインポート文をここへ貼り付けてください

// 四角と円の描画
public class Sample11_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // キャンバスを生成／設定します
      Canvas cv = new Canvas(500,500);
      GraphicsContext gc = cv.getGraphicsContext2D();

      // 四角と角丸四角を描きます
      gc.setLineWidth(1);
      gc.setStroke(Color.BLUE);
      gc.strokeRect(50.0, 50.0, 150.0, 150.0);

      gc.setStroke(Color.RED);
      gc.strokeRoundRect(50.0, 250.0, 150.0, 150.0, 50, 50);

      // 楕円と円を描きます
      gc.setStroke(Color.GREEN);
      gc.strokeOval(250.0, 50.0, 150.0, 150.0);

      gc.setStroke(Color.BROWN);
      gc.strokeOval(250.0, 250.0, 150.0, 75.0);

      // 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(cv);

      // 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("キャンバス");

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

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



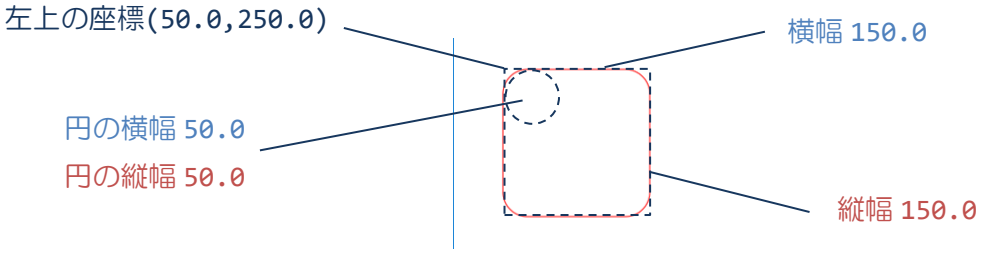
■四角形や円の描画

グラフィックスコンテキストクラス GraphicsContext には基本的な図形を描画するメソッドが準備されています。

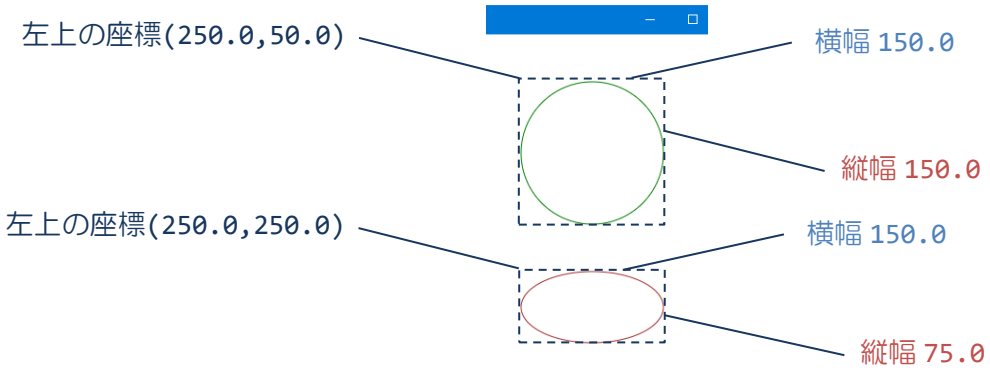
・四角形を描画　→ strokeRect(50.0, 50.0, 150.0, 150.0);



・角丸四角形を描画　→ strokeRoundRect(50.0, 250.0, 150.0, 150.0, 50, 50);



・円や楕円を描画　→ strokeOval(250.0, 50.0, 150.0, 150.0);
→ strokeOval(250.0, 250.0, 150.0, 75.0);



■利用したクラスの一覧

GraphicsContext クラス←Object

void strokeRect(double x, double y, double w, double h){…}
始点(x,y)幅 w 縦 h の四角形を描きます。

void strokeRoundRect(double x, double y, double w, double h, double aw, double ah){…}
さらに、角丸の横幅 aw 縦幅 ah の角丸四角形を描きます。

void strokeOval(double x, double y, double w, double h){…}
始点(x,y)幅 w 縦 h の四角形に収まる円／楕円を描きます。

§3　キャンバスに図形を描いてみましょう（2）

多角形など複雑な図形や文字列も簡単にキャンバスに描くことができます。

ソースファイル名：Sample11_3.java

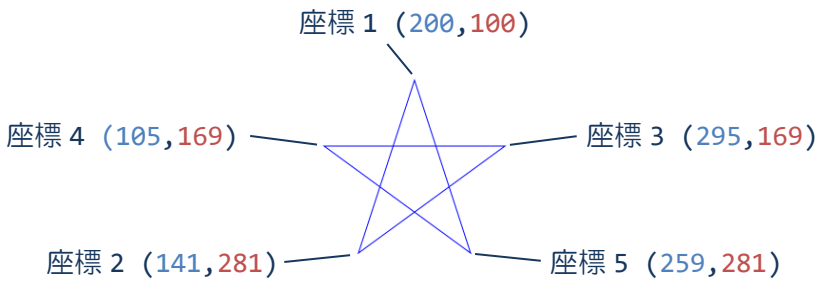
```
// ※HP よりインポート文をここへ貼り付けてください

// 多角形と文字列の描画
public class Sample11_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // キャンバスを生成／設定します
      Canvas cv = new Canvas(500,500);
      GraphicsContext gc = cv.getGraphicsContext2D();

      // 多角形を描きます
      double[] px={200, 141, 295, 105, 259};
      double[] py={100, 281, 169, 169, 281};
      gc.setLineWidth(1);
      gc.setStroke(Color.BLUE);
      gc.strokePolygon(px, py, 5);

      // 文字列を描きます
      gc.setStroke(Color.GREEN);
      gc.setFont(new Font(52));
      gc.strokeText("HCI プログラミング", 40, 70);

      // 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(cv);

      // 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("キャンバス");

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

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

■多角形の描画

グラフィックスコンテキストクラス **GraphicsContext** には多角形や文字列を描画するメソッドが準備されています。

・多角形を描画

```
double[] px={200, 141, 295, 105, 259};
double[] py={100, 281, 169, 169, 281};
→ strokePolygon(px, py, 5);
```



座標配列(200,100)→(141,281)→(295,169)→(105,169)→(259,281)の順番で線を引きます。最後に終点から始点に線が引かれます。線の色と太さも設定可能です。

・文字列を描画

```
→ setFont(new Font(52));
→ strokeText("HCI プログラミング", 40, 70);
```

フォント 52pt のデフォルトフォントで左下座標が(40,70)の位置から文字列を描画します。線の色と太さも設定可能です。

■利用したクラスの一覧

GraphicsContext クラス←Object

`void strokePolygon(double[] px, double[] py, int c){…}`
座標配列 px,py から c 個の座標を用いて多角形を描きます。

`void setFont(Font f){…}`
テキスト描画用のフォントを f に設定します。

`void strokeText(String s, double x, double y){…}`
左下座標(x,y)から文字列 s を描画します。

§4　図形を塗りつぶしてみましょう

図形や文字列はキャンバスに塗りつぶして描くこともできます。

ソースファイル名：Sample11_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 塗りつぶして描画
public class Sample11_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // キャンバスを生成／設定します
      Canvas cv = new Canvas(500,500);
      GraphicsContext gc = cv.getGraphicsContext2D();

      // 四角形と円を塗りつぶして描きます
      gc.setFill(Color.LIGHTBLUE);
      gc.fillRect(50.0, 50.0, 150.0, 150.0);
      gc.setFill(Color.PINK);
      gc.fillRoundRect(50.0, 250.0, 150.0, 150.0, 50, 50);

      gc.setFill(Color.LIGHTGREEN);
      gc.fillOval(250.0, 50.0, 150.0, 150.0);
      gc.setFill(Color.SANDYBROWN);
      gc.fillOval(250.0, 250.0, 150.0, 75.0);

      // 多角形と文字列を塗りつぶして描きます
      double[] px={200, 141, 295, 105, 259};
      double[] py={100, 281, 169, 169, 281};
      gc.setFill(Color.ROYALBLUE);
      gc.fillPolygon(px, py, 5);

      gc.setFill(Color.GREEN);
      gc.setFont(new Font(52));
      gc.fillText("HCI プログラミング", 40, 70);

      // 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(cv);

      // 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("キャンバス");

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

```
  }

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

■図形の塗りつぶし

グラフィックスコンテキストクラス GraphicsContext には図形を塗りつぶして描画するメソッドが準備されています。線で描画するメソッドと次のように対応します。

図形	線描画	塗りつぶし描画
・四角形	strokeRect(…);	fillRect(…);
・角丸四角形	strokeRoundRect(…);	fillRoundRect(…);
・円／楕円	strokeOval(…);	fillOval(…);
・多角形	strokePolygon(…);	fillPolygon(…);
・文字列	strokeText(…);	fillText(…);

※各引数リストは同じです。

また、線の色や太さの設定と同様に、塗りつぶし色の設定ができます。

属性	線	塗りつぶし
・太さ	setLineWidth(…);	なし
・色	setStroke(…);	setFill(…);

※引数リストは同じです。

■利用したクラスの一覧

GraphicsContext クラス←Object

```
void fillRect(double x, double y, double w, double h){…}
```
始点(x,y)幅 w 縦 h の四角形を塗りつぶします。

```
void fillRoundRect(double x, double y, double w, double h, double aw, double ah){…}
```
さらに、角丸の横幅 aw 縦幅 ah の角丸四角形を塗りつぶします。

```
void fillOval(double x, double y, double w, double h){…}
```
始点(x,y)幅 w 縦 h の四角形に収まる円／楕円を塗りつぶします。

```
void fillPolygon(double[] px, double[] py, int c){…}
```
座標配列 px,py から c 個の座標を用いて多角形を塗りつぶします。

```
void fillText(String s, double x, double y){…}
```
左下座標(x,y)から文字列 s を塗りつぶします。

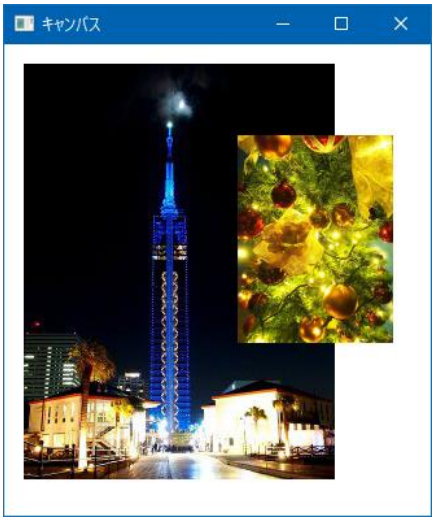
```
void setFill(Paint p){…}
```
塗りつぶし色を p に設定します。
※クラス Paint はクラス Color のスーパークラスです。

§5　画像を描画してみましょう

画像をキャンバスの好きな位置に描くこともできます。

ソースファイル名：Sample11_5.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 画像の描画
public class Sample11_5 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // キャンバスを生成／設定します
      Canvas cv = new Canvas(500,500);
      GraphicsContext gc = cv.getGraphicsContext2D();

      // 画像を準備します
      Image img1 = new Image("FukuokaTower.jpg");
      Image img2 = new Image("XmasTree.jpg");

      // 画像を描きます
      gc.drawImage(img1,  15, 15);
      gc.drawImage(img2, 180, 70, img2.getWidth()*0.5, img2.getHeight()*0.5);

      // 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(cv);

      // 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("キャンバス");

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

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



■画像の描画

グラフィックスコンテキストクラス **GraphicsContext** には画像を描くメソッドが準備されています。

・画像を描画　→ `drawImage(img1, 15, 15);`
　　※画像 `img1` のオリジナルサイズで表示されます.

・画像を描画　→ `drawImage(img2, 180, 70, img2.getWidth()*0.5, img2.getHeight()*0.5);`
　　※画像 `img2` の半分のサイズで表示されます.

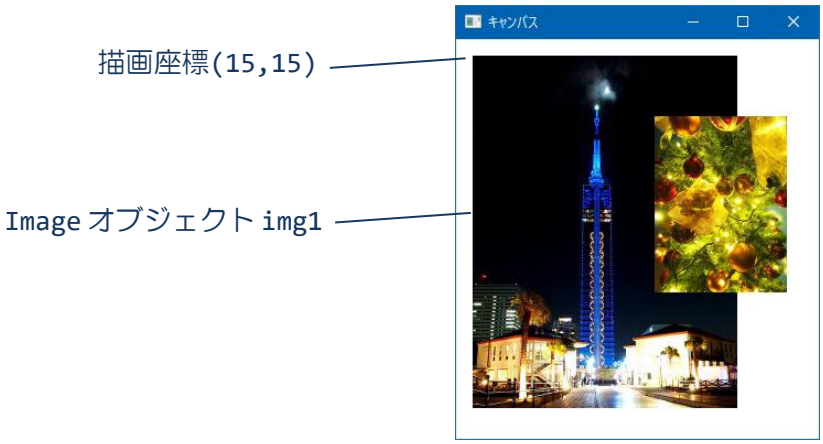


Image クラスのオブジェクト img1（画像）を座標(15,15)に描画します．画像の左上隅が指定した座標に配置されます．また，オブジェクト img2（画像）を座標(180,70)に半分のサイズで描画します．

■利用したクラスの一覧

GraphicsContext クラス←Object

`void drawImage(Image img, double x, double y){…}`
　　座標(x,y)を画像 img の左上隅として描きます.

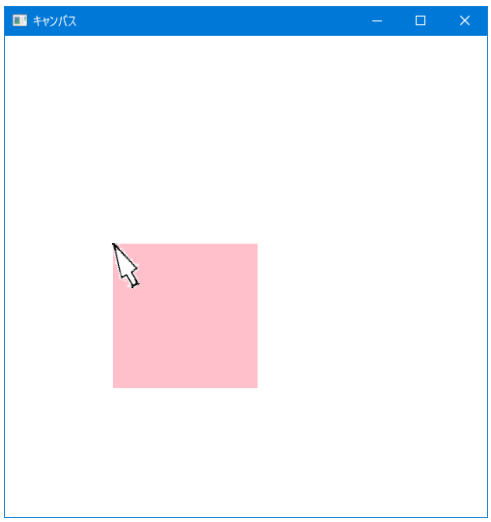
`Void drawImage(Image img, double x, double y, double w, double h){…}`
　　座標(x,y)を画像 img の左上隅として指定サイズ w×h で描きます.

§6　マウス位置を取得して図形を描いてみましょう

マウスイベントを取得して図形をインタラクティブに描画することができます。

ソースファイル名：Sample11_6.java

```
// ※HP よりインポート文をここへ貼り付けてください

// マウス位置に四角形を表示
public class Sample11_6 extends Application
{
   private Canvas cv;
   private double mx, my;

   public void start(Stage stage) throws Exception
   {
      // キャンバスを生成／設定します
      cv = new Canvas(500,500);

      // キャンバスにマウスのイベントハンドラを設定します
      MyEventHandler mousehandler = new MyEventHandler();
      cv.addEventHandler(MouseEvent.ANY, mousehandler);

      // 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(cv);

      // 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("キャンバス");

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

   // キャンバスを描画するメソッド
   private void drawCanvas()
   {
      GraphicsContext gc = cv.getGraphicsContext2D();

      // キャンバスをクリアする
      gc.setFill(Color.WHITE);
      gc.fillRect(0, 0, cv.getWidth(), cv.getHeight());

      // 四角形を塗りつぶして描きます
      gc.setFill(Color.PINK);
```



```
        gc.fillRect(mx, my, 150.0, 150.0);
    }

    // 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<MouseEvent>
    {
        public void handle(MouseEvent e)
        {
            EventType<? extends MouseEvent> type = e.getEventType();
            if(type ==  MouseEvent.MOUSE_CLICKED){
                mx = e.getX();
                my = e.getY();
                drawCanvas();
            }
        }
    }

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

■キャンバスを描画する処理を 1 つのメソッドにまとめましょう

これまでは起動時に一度だけ実行される start()メソッドの中で描画処理を記述していました。アニメーションなどマウスの動作に合わせて描画を行う場合は、描画処理を繰返し実行します。

描画処理をまとめて一つのメソッド（例では drawCanvas()メソッド）として宣言し、start()メソッド内ではキャンバスの生成とレイアウトのみを行うように役割分担をしておくと便利です。

メソッド drawCanvas()は描画をするとき、start()メソッドで生成したキャンバスのグラフィックスコンテキストを必要とします。Canvas クラスの変数をクラスのメンバー変数として宣言し、他のメソッドからもキャンバスを参照できるようにしておきます。

■キャンバスの再描画とキャンバスのクリア

キャンバスを再描画するとき、以前描いた内容を白紙にもどします。これをキャンバスのクリアといいます。特別なメソッドはなく、白色で矩形を塗りつぶし白紙にします。ここでキャンバスのサイズを取得する Canvas クラスのメソッドを用いると便利です。

・キャンバスの横幅（ピクセル）の取得　　→ getWidth();
・キャンバスの縦幅（ピクセル）の取得　　→ getHeight();

■マウスのクリック座標などのマウス情報は共有しましょう

マウスを用いたインタラクティブな描画処理を記述する場合、マウスの座標などのマウス情報は、マウスハンドラで取得され、描画するメソッドでこれらの情報を利用することになります。

マウス情報をクラスのメンバー変数として宣言し、複数のメソッドから参照できるようにしておきます。

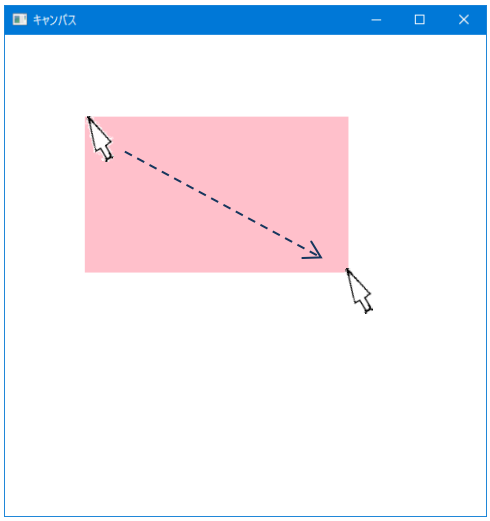
§7　マウスドラッグで四角形を描いてみましょう

複数のマウスイベントを処理してサイズなどを調整しながら図形をインタラクティブに描画できます。

ソースファイル名：Sample11_7.java

```
// ※HP よりインポート文をここへ貼り付けてください

// マウスドラッグで四角形を描画
public class Sample11_7 extends Application
{
   private Canvas cv;
   private double mx, my, nx, ny;

   public void start(Stage stage) throws Exception
   {
      // キャンバスを生成／設定します
      cv = new Canvas(500,500);

      // キャンバスにマウスのイベントハンドラを設定します
      MyEventHandler mousehandler = new MyEventHandler();
      cv.addEventHandler(MouseEvent.ANY, mousehandler);

      // 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(cv);

      // 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("キャンバス");

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

   // キャンバスを描画するメソッド
   private void drawCanvas(){
      GraphicsContext gc = cv.getGraphicsContext2D();

      // キャンバスをクリアする
      gc.setFill(Color.WHITE);
      gc.fillRect(0, 0, cv.getWidth(), cv.getHeight());

      // 四角形を塗りつぶして描きます
      gc.setFill(Color.PINK);
      gc.fillRect(mx, my, nx-mx, ny-my);
   }
```

```
   // 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<MouseEvent>
   {
      public void handle(MouseEvent e)
      {
         EventType<? extends MouseEvent> type = e.getEventType();
         if(type ==  MouseEvent.MOUSE_PRESSED){
            mx = e.getX();
            my = e.getY();
            nx = mx;
            ny = my;
            drawCanvas();
         }
         if(type ==  MouseEvent.MOUSE_DRAGGED){
            nx = e.getX();
            ny = e.getY();
            drawCanvas();
         }
         if(type ==  MouseEvent.MOUSE_RELEASED){
            nx = e.getX();
            ny = e.getY();
            drawCanvas();
         }
      }
   }

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



§8　ラジオボタンを用いて図形の色を変更してみましょう

GUI 部品のアクションイベントを取得して色を変えながら図形をインタラクティブに描画できます。

ソースファイル名：Sample11_8.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ラジオボタンによる色変更
public class Sample11_8 extends Application
{
   private Canvas cv;
   private Color clr;

   public void start(Stage stage) throws Exception
   {
      // ラジオボタンを生成／設定します
      RadioButton[] rbs = new RadioButton[3];
      rbs[0] = new RadioButton("赤色");
      rbs[1] = new RadioButton("青色");
      rbs[2] = new RadioButton("黄色");
      rbs[0].setId("red");
      rbs[1].setId("blue");
      rbs[2].setId("yellow");

      // ラジオボタンをグループ化します
      ToggleGroup tg = new ToggleGroup();
      rbs[0].setToggleGroup(tg);
      rbs[1].setToggleGroup(tg);
      rbs[2].setToggleGroup(tg);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      rbs[0].addEventHandler(ActionEvent.ANY, actionhandler);
      rbs[1].addEventHandler(ActionEvent.ANY, actionhandler);
      rbs[2].addEventHandler(ActionEvent.ANY, actionhandler);

      // キャンバスを生成／設定します
      cv = new Canvas(500,500);
      clr = Color.BLACK;
      drawCanvas();

      // 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(rbs);
      hb.setPadding(new Insets(10));
      hb.setSpacing(10);

      // レイアウト VBox を生成／設定します
      VBox vb = new VBox();
```

```
        lst = vb.getChildren();
        lst.add(hb);
        lst.add(cv);

        // 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("キャンバス");

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

    // キャンバスを描画するメソッド
    private void drawCanvas(){
        GraphicsContext gc = cv.getGraphicsContext2D();

        // キャンバスをクリアする
        gc.setFill(Color.WHITE);
        gc.fillRect(0, 0, cv.getWidth(), cv.getHeight());

        // 四角形を塗りつぶして描きます
        gc.setFill(clr);
        gc.fillRect(100, 100, 150.0, 150.0);
    }

    // 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 target = (RadioButton)e.getTarget();
            String id = target.getId();

            if(id.equals("red")) clr = Color.RED;
            if(id.equals("blue")) clr = Color.BLUE;
            if(id.equals("yellow")) clr = Color.YELLOW;
            drawCanvas();
        }
    }

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