オブジェクト指向論(Q)

第12回講義資料
(プログラミング第6回 OOP6)

2023/6/26

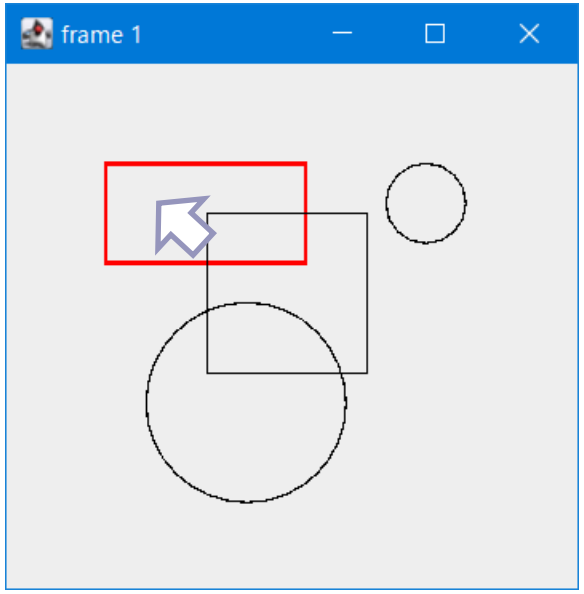
來村 徳信

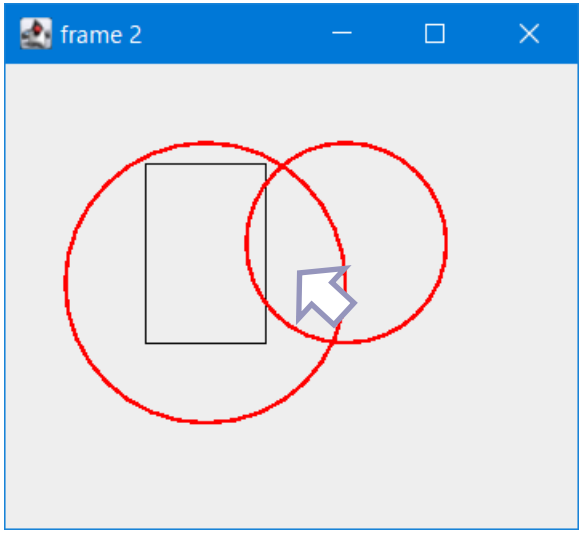
今回／次回講義のテーマと流れ

- イベント処理
  - イベントモデル
    - イベントリスナーの登録による「委譲」
    - 今回 → 1）Window/Panelに対するマウスイベント
      - 今回の目標（レポート課題前半）：図形の内側をクリックして選択する．選択された図形は赤太線で，描画する．
      - ステップ１(B1）：mouseEventクラス
        - マウスクリックに反応して，座標値をコンソールに出力する．
      - ステップ２(B2)：クリックによる図形選択
        - 内側をクリックされた図形を判定して，コンソールに出力する．
      - ステップ３(B3)：選択状態に合わせた描画
        - 選択されている図形を赤太線で描画する．
    - 次回 → 2）UIコンポーネントに対するイベント
      - ActionEventクラス．Button へのクリックに対する反応．
      - ボタンに応じた動作

今回の目標：図形のクリックによる選択

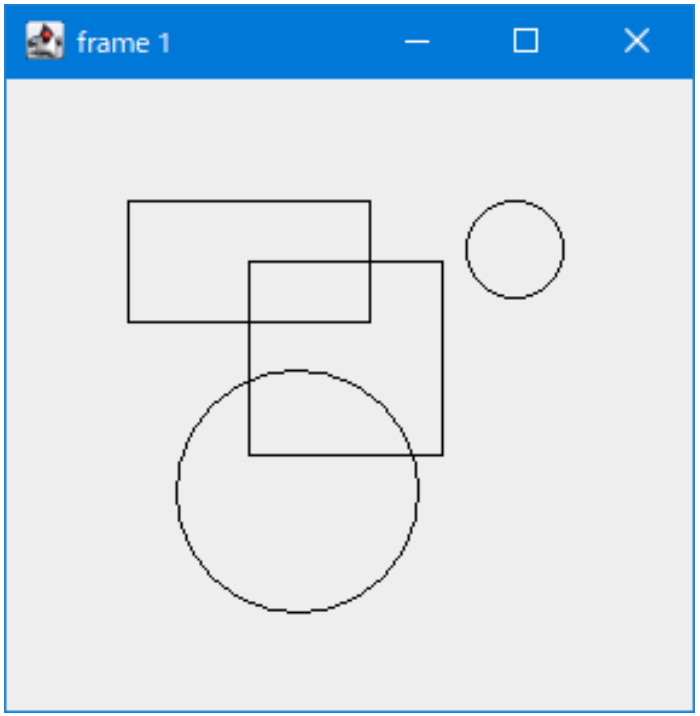
- 四角形または円の内側をクリックして選択する
  - 選択状態にある図形は赤太線で描画する.
    - 非選択状態な図形は黒細線，選択状態は赤太線で描画する.
  - 同時に２つ以上の図形が選択されることもありえる
    - 重なった領域をクリックしたとき

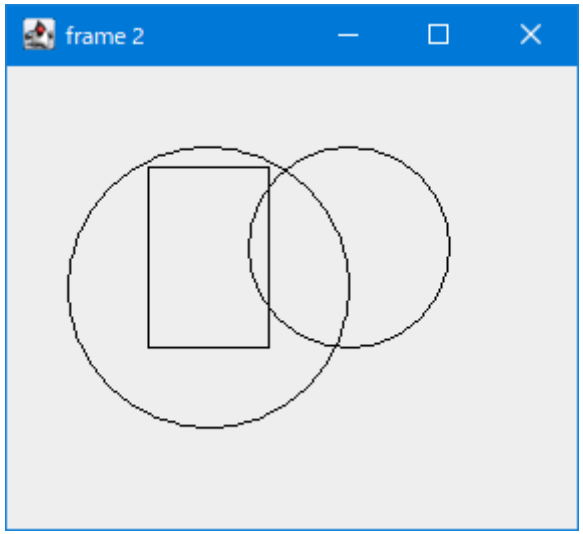




出発点：OOP6-A

- Windowに任意個の図形を描画できる.
  - ArrayListを用いて図形インスタンスを保持.
  - 四角形(Rect)と円(Circle)を黒色で描画する.
  - OOP4のミニ演習の解に，今回の拡張用テンプレートを追加.





イベント駆動型プログラム

- イベント
  - ユーザがGUIを操作するとイベントが発生する.
  - イベントが発生したということが通知される（メッセージパッシング）≒ 特定のメソッドが呼ばれる.
  - ユーザの操作に限らず，例えば画面の「再描画要求」なども システムからプログラムへのメッセージとして送られる.
- オブジェクトがイベントに反応する
  - イベントへの応答を定義することが，オブジェクト指向な GUIプログラミングの中心
  - イベント駆動（ドリブン）型 ⇔ フロー駆動型
  - イベントリスナー（イベントハンドラ）
    - 特定のイベントの種類にどのように反応するかを定義した クラス．そのインスタンスの特定メソッドが呼び出される.
    - あとの回で詳しく説明する
  - 今回・次回で説明・演習する.

イベント(Event)

- イベントの種類の例
  - イベントの種類（クラス）と呼び出されるメソッド名などが，Java API の awt や Swingで定義されている.
  - 例：マウスクリック関連のイベント（後述）
    - 例１：JPanel 内などでマウスボタンがクリックされた
      → MouseEvent. mouseClicked() が呼ばれる（今回）
    - 例２：ボタンの上でマウスボタンがクリックされた
      → ActionEvent. actionPerformed() が呼ばれる（次回）
  - 例：Window の「再描画要求」
    - 例３：JPanel で再描画の必要性が生じると，システムから JPanel の paintComponent() が呼ばれる（OOP4～5）

Java のイベントモデル（1.1以降）

- 委譲(delegation)イベントモデル
  - 「イベント　ソース　」
    - イベントを「発生させる」オブジェクト（例：ウィンドウ）
  - 「イベント　オブジェクト」
    - １回のイベント（例：クリック1回）に対して１つのイベントオブジェクト（インスタンス）が生成される.
    - さまざまな種類がある（クリックやマウスオーバーなど）
    - イベントの情報（クリックなら座標値など）を，インスタンスのフィールド値として持つ.
  - 「イベント　リスナー　」
    - イベントが起こったとき「通知される」オブジェクト
    - イベントの種類によって特定のメソッドが，イベントオブジェクトのインスタンスを引数として，呼び出される.

イベントリスナー

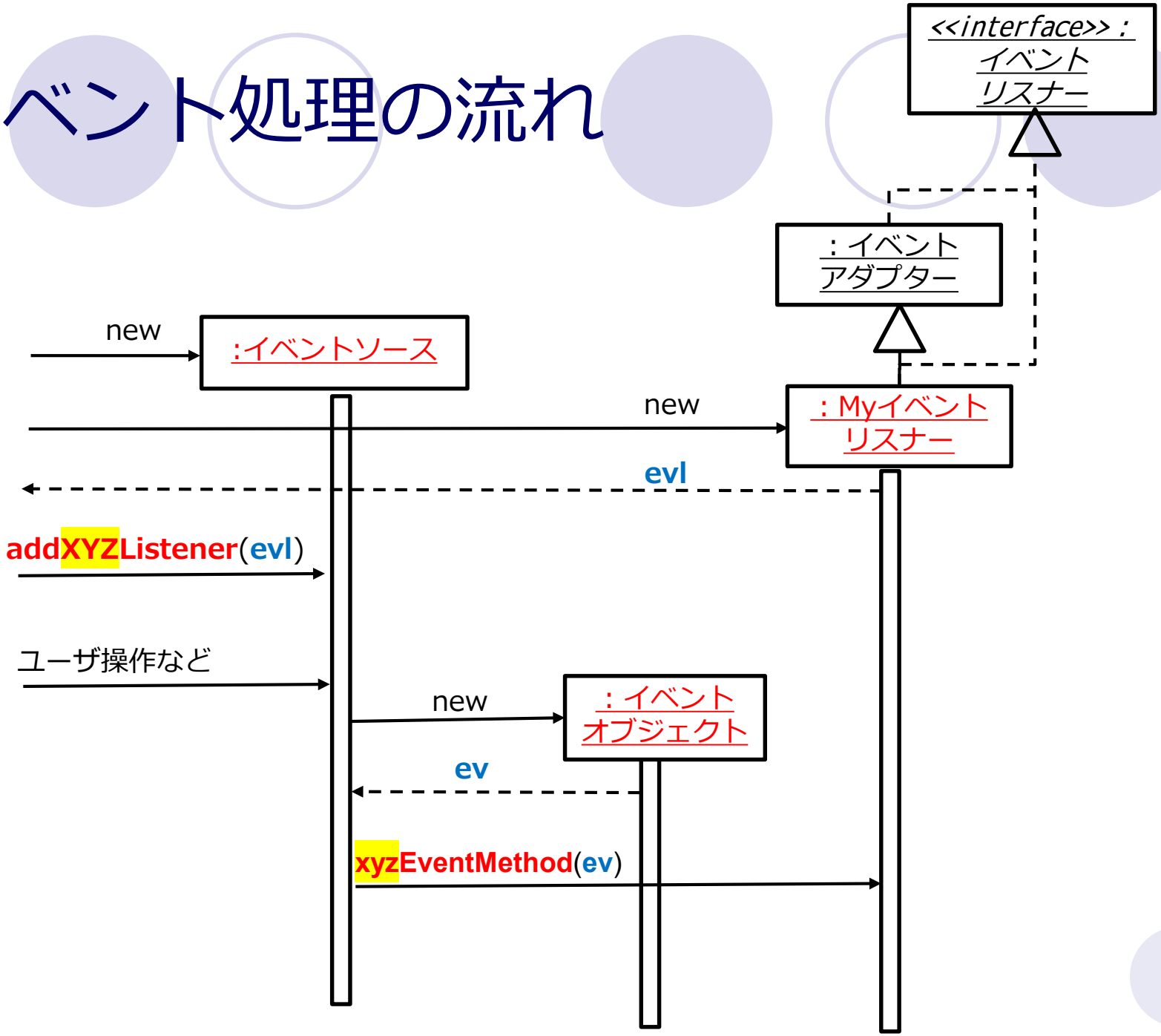
- イベントが起こったときに通知されるオブジェクト
  - そのオブジェクトの特定の名前のメソッドが呼ばれる
    - 例：クリックされたら mouseClicked() が呼ばれる.
    - そのときにどのような反応をするかを定義しておく
  - 事前に，イベントリスナーのインスタンスを作成し，イベント　ソース　に「　登録　」しておく
    - 「私があなたで起こるイベントに反応する」．複数可.
  - イベントの種類ごとにイベントリスナーのインタフェースが定義されているので，それを implements し，メソッドの処理内容を実装することで，自分のプログラムとしての反応を定義する.
  - 引数はイベントオブジェクトのインスタンス.
  - イベントアダプタ(EventAdapter)：イベントリスナーを implements し，空のメソッド（なにもしない）が定義されているもの（後述）

重要

イベント処理のポイント

- 委譲(delegation)イベントモデル
  - イベントの発生源であるオブジェクト（イベントソース）は，それへの対処方法を知らない．発生したということを（システム経由で）通知するのみ．
  - 対処方法は，イベントリスナーという別のオブジェクトが担当する（「権限の 委譲 」）．
  - イベントソースに，イベントリスナーインスタンスを「登録」することで、イベントの通知が実現されている．
    - <イベントソース>.addXYZListener(イベントリスナー)
    - これが図形描画プログラムの，図形オブジェクトのPanelオブジェクトへの登録と描画に似ていることに留意．
  - イベントに対する反応を柔軟に切り替えることができる．
  - １つのイベントに対して，複数のイベントリスナーがそれぞれの反応ができる（マルチキャスト）

イベント処理の流れ


<<interface>> :
イベント
リスナー
：イベント
アダプター
new
:イベントソース
new
：Myイベント
リスナー
evl
addXYZListener(evl)
ユーザ操作など
new
：イベント
オブジェクト
ev
xyzEventMethod(ev)

今回／次回講義のテーマと流れ(再掲1)

- イベント処理
  - イベントモデル
    - イベントリスナーの登録による「委譲」
    - 今回 → 1）Window/Panelに対するマウスイベント
      - 今回の目標（レポート課題前半）：図形の内側をクリックして選択する．選択された図形は赤太線で，描画する．
      - → ステップ１(B1）：mouseEventクラス
        - マウスクリックに反応して，座標値をコンソールに出力する．
      - ステップ２(B2)：クリックによる図形選択
        - 内側をクリックされた図形を判定して，コンソールに出力する．
      - ステップ３(B3)：選択状態に合わせた描画
        - 選択されている図形を赤太線で描画する．
    - 2）UIコンポーネントに対するイベント
      - ActionEventクラス．Button へのクリックに対する反応．
      - ボタンに応じた動作

マウスに関するEventのレベル

- マウスの「イベントオブジェクト」のレベル
  - 対応するイベントリスナーのインタフェース

次回

- ActionEvent：高レベル・イベント
  - UIコンポーネントごとに定義された「意味のある」イベントを表す.
  - ボタンなら「押されて離された」時のみが重要なので，そのイベントのみを処理できる.
    - ActionListener インタフェースのメソッドは actionPerformed(ActionEvent e) だけ.

今回

- MouseEvent ：低レベル・イベント
  - マウスのボタン操作や移動を扱える.
  - インタフェース：MouseListener
    - 中レベル．マウスボタンを押す／離す，クリックなど.
  - インタフェース：MouseMotionListener
    - 低レベル．マウスカーソルの移動やドラッグも扱える.

MouseEvent

- awt.event.MouseEvent クラス
  - 「イベントオブジェクト」の一種
  - Window 内でのマウスイベントを表す
    - マウスボタンを押す・離す・クリックする
    - マウスカーソルが移動する．マウスをドラッグする．
    - マウスカーソルがあるコンポーネント領域に入る／出る．
  - 1つのマウスイベントに対して，1つのインスタンスが生成される．そのイベントに関する情報（座標等）を保持する．
  - 「イベントソース」：JFrame/JPanelクラスなど
    - これらの内部で起こるマウスイベントを処理できる．
- MouseEventクラスのメソッドの例
  - int getX()：クリックされたX座標値（整数）を得る
  - int getY() :クリックされたY座標値（整数）を得る

MouseEventのリスナークラス

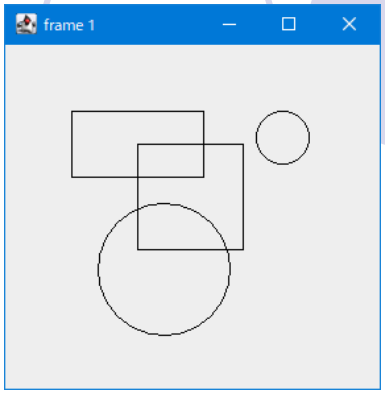
- MouseListener インタフェース
  - 全メソッドが宣言されている
    - mousePressed() : マウスボタンが押された
    - mouseReleased() : マウスボタンが離された
    - mouseClicked() : マウスボタンがクリックされた
    - mouseEntered() : マウスカーソルが領域に入った.
    - mouseExited() ：マウスカーソルが領域から出た
    - 引数は MouseEventクラスのインスタンス
- アダプタークラス: MouseAdapter クラス
  - 上記の全メソッドに対して，空メソッドが定義されている.
  - MouseAdapter を継承して、必要なメソッドだけを オーバーライドする方が簡単
- マウスカーソルの移動やドラッグは扱わない.
  - 扱う場合は，MouseMotionListener インタフェースを使う.

MouseEventのリスナークラスの定義

- MouseAdapter クラスを継承(extends)して，必要なメソッドのみをオーバーライドする.
  - https://docs.oracle.com/javase/jp/8/docs/api/java/awt/event/MouseAdapter.html
  - ここではメソッド: mouseClicked() をオーバーライドする
    - 引数はMouseEvent インスタンス.
    - Panel内でマウスクリックされると，そのPanelインスタンスに登録された MouseAdapterインスタンスの mouseClicked メソッドが呼び出される.
    - mouseClicked() の処理内容を「実装する」
- リスナーのインスタンスを，Panel などのイベントソースに，addMouseListener() で登録する.
  - 例：JPanel のインスタンス *p* に，MouseAdapter のインスタンス *ma* を追加.

```
p.addMouseListener(ma)
```

ステップ１：クリックに反応させる



- パネル内でマウスクリックされたら，コンソールに，クリックされた座標を表示する．
- 方法
  - MyMouseListenerというクラスを定義する．
    - OOP6-A.zip にMyMouseListener.java が含まれている．
    - MyPanel からコンストラクタを呼び出す．MyPanel インスタンスを，コンストラクタの引数としてとり，コンストラクタ内で MyPanel への参照を覚えておく．後で使うから．
  - MouseAdapter クラスを継承(extends)する．
    - mouseClicked() をオーバーライドして，実装する．
    - 引数の MouseEvent のインスタンスに，getX(), getY() メソッドを用いることで，クリックされた座標が分かる．
  - MyPanelのインスタンスに「登録」する
    - JPanel クラスの addMouseListener() を使う（前ページ）

MyMouseListener の定義(B1)

- MouseAdapter クラスを継承(extends)する.
- mouseClicked メソッドを実装する.

```
public class MyMouseListener
                extends MouseAdapter {
    private MyPanel panel;

    MyMouseListener(MyPanel panel) {
        super();
        this.panel = panel;
        }
    @Override
    public void mouseClicked(MouseEvent e) {
        int x = (1)          ;  // Step1. to be changed
        int y = (2)          ;  // Step1. to be changed
        System.out.println("Clicked at ("+x+","+y+")");
        // this.panel.method() と書くことで，イベントソースの
        // MyPanel のインスタンスに対して，method を呼べる.ここで
        }}
```

イベントソースのパネルを覚えておく

パネル内でクリックされると
このメソッドが呼ばれる

使うため

PanelへのListenerの登録(B1)

- MyMouseListener のインスタンスを作る.
  - 自分自身を引数として渡す．覚えてもらう.
- MyPanel のインスタンス（自分．イベントソース）に登録する．自分で生じるイベントの処理を任せる.

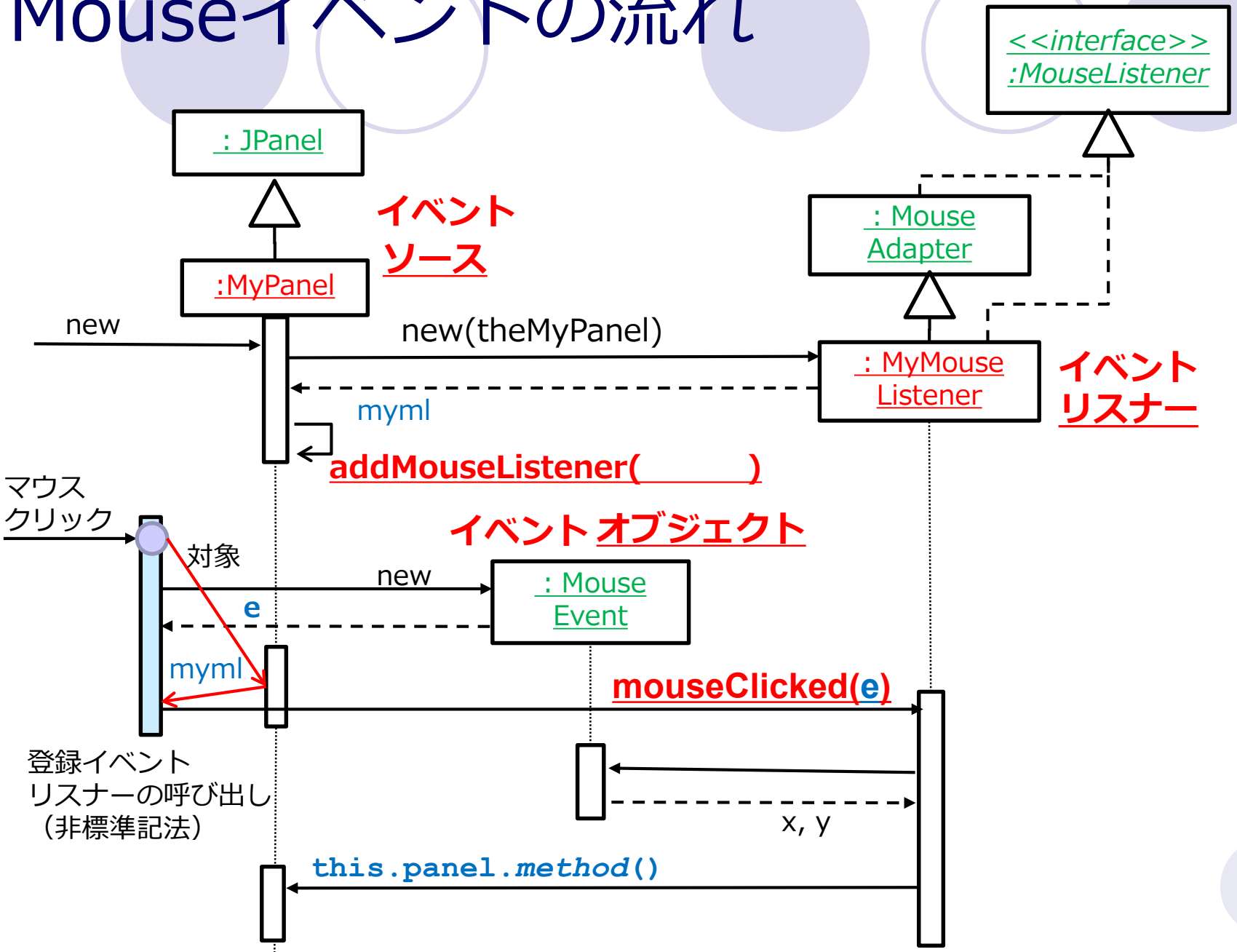
```
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.event.*;
public class MyPanel extends JPanel {

  public MyPanel() {
    super();
    ...
    MouseAdapter myml
      = new MyMouseListener((3)_____);
    (4)_____.addMouseListener((5)_____);
    ...
```

自分で定義したマウスリスナーのインスタンスを作る

生成したマウスリスナーインスタンスを自分自身に登録する.

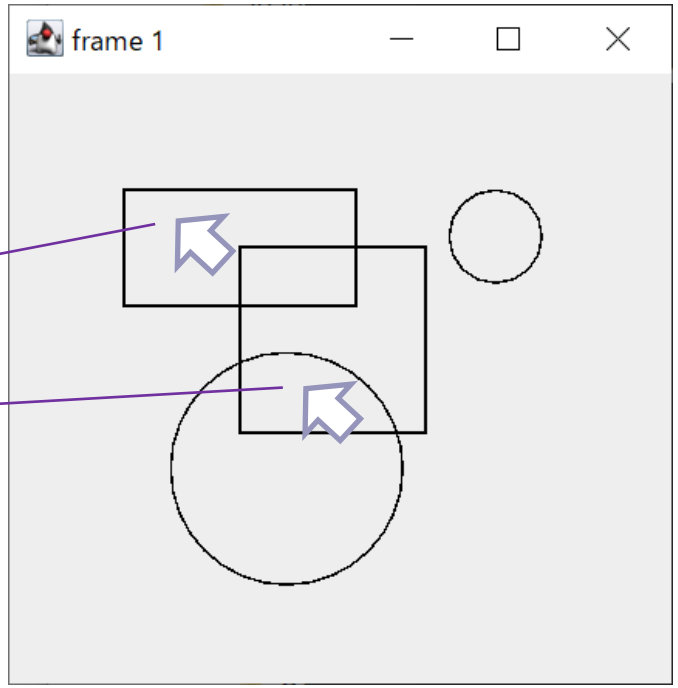
Mouseイベントの流れ



ステップ1(B1)の確認

- クリックしてみて，正しそうな座標が表示されるかを確認する.

```
MyPanel painting ...
Drawing... R (50, 50)-(150, 100)
Drawing... R (100, 75)-(180, 155)
Drawing... C (120, 170) r=50
Drawing... C (210, 70) r=20
Clicked at (72,69)
Clicked at (132,139)
```



今回／次回講義のテーマと流れ(再掲2)

- イベント処理
  - イベントモデル
    - イベントリスナーの登録による「委譲」
  - 1）Window/Panelに対するマウスイベント
    - 今回の目標（レポート課題前半）：図形の内側をクリックして選択する．選択された図形は赤太線で，描画する．
    - ステップ1(B1）：mouseEventクラス
      - マウスクリックに反応して，座標値をコンソールに出力する．
    - ステップ2(B2)：クリックによる図形選択
      - 内側をクリックされた図形を判定して，コンソールに出力する．
    - ステップ3(B3)：選択状態に合わせた描画
      - 選択されている図形を赤太線で描画する．
  - 2）UIコンポーネントに対するイベント
    - ActionEventクラス．Button へのクリックに対する反応．
    - ボタンに応じた動作
      - 選択されている図形に対する処理など

課題への大まかな方針

- manaba+R の OOP6-A をもとにする
  - まず，ステップ1の拡張をする．動作確認をする．
- ステップ2：クリックによる図形の選択
  - クリックされたら，そのMyPanelインスタンスに登録されている全図形インスタンスに，クリックされた座標値を伝える．
  - 各図形インスタンスは，伝えられた座標が自分の内側にあるかどうかを判定し，内側であれば，自分のフィールド変数 selected の値を true にセットする．
    - boolean 型フィールド selected は選択状態を表す．選択されている状態なら値がtrue．でなければ false．
    - 「内側」には枠線の上も含むものとする．
  - ステップ2では選択された図形の座標をコンソールに出力する．
- ステップ3：選択状態に応じた図形の描画
  - 各図形インスタンスを描画する前に，その図形の選択状態に応じて，描画する線の色と太さをセットする．

Shape 抽象クラスの再拡張

- Shape: 描画可能な図形（拡張前）
  - 数学的図形（抽象クラス Shape）
    - 幅(getWidth()) や面積(getArea()) を持つもの.
  - ＋描画可能という機能（Drawable インタフェース）
    - draw という機能(メソッド)を「必ず」持つ
- Shape: クリックで選択可能な描画図形（拡張後）
  - 「クリックで選択可能」という機能を持つように拡張する.
  - Selectable インタフェースとして定義しよう.
    - ある点をクリックされることで「選択」状態になる機能.
    - どのようなメソッドが必要だろうか？
  - Shape 抽象クラスで，Selectable インタフェースを implements する．Shape抽象クラスまたは下位クラスで実際の処理を実装する.

Selectable インタフェース

- 必要なフィールド（OOP6-A の Shape クラスで定義済）
  - private boolean selected; 選択状態であれば true.
- 必要となるメソッドは？
  - (1) クリックによって自分を「選択状態」にするメソッド
    - selectByClick()という名前にする.
    - Selectableで宣言する．実装は?
    - 引数：クリックされた点の座標を表す整数(int 型) x, y
    - 仕様：そのクリック座標によって自分が選択状態になる場合，自分の selected を true にセットする.
    - 今回の内部動作：座標(x,y)が自分の内側かどうかを判定するメソッド contains(x,y)に呼んで，selectedの値をセットする.
  - (2) ある座標が自分の「内側」かどうかを判定するメソッド
    - contains() という名前にする．どこで宣言・実装する？
    - 引数は，x座標とy座標（整数(int)型）
    - 戻り値：boolean 型．その座標が，自分の内側であれば true を返す．そうでなければ，false を返す.

Selectable インタフェースと Shape抽象クラスの定義

- Selectable, implements, selected は OOP6-A で記述済

```
public interface Selectable {
// public abstract void selectByClick(int x, int y);
}
```

※OOP6-Aではコメントアウトされているので，先頭の // を削除してください．
ただし，selectByClick() が実装されていないと，コンパイルエラーになります．

```
public abstract class Shape
        implements Drawable, Selectable {
    private boolean selected = false;

        ※protected ではありません．
    ...   private のままで実装してください
    }
```

MyMouseListener からの呼び出し

- mouseClicked メソッドから，クリックされた座標値を引数として MyPanel の panelClicked() メソッドを呼ぶ.

```
import java.awt.event.*;

public class MyMouseListener extends MouseAdapter {
    private MyPanel panel;

    MyMouseListener(MyPanel panel) {
        super();
        this.panel = panel;
        }
    @Override
    public void mouseClicked(MouseEvent e) {
        int x = __________;
        int y = __________;
        System.out.println("Clicked at ("+x+","+y+")");
        // Step 2. insert here
        }
    }
```

イベントソースのパネルを覚えておく

パネル内でクリックされると
このメソッドが呼ばれる

this.panel へ向けて
panelClicked(x, y) を呼ぶ.

ここで
使うため

MyPanel : panelClicked メソッド

- void panelClicked(int x, int y) メソッド
  - Panel内がクリックされたときに，MyMouseListener クラスの mouseClicked メソッドから，クリックされた座標値を引数として，呼び出される.
  - 自分に登録されている全ての図形インスタンスに向けて selectByClick(x,y) メソッドを呼び出し，座標(x,y)がクリックされたことを伝えて，自分が選択状態になるかどうかを決めさせる
    - ヒント : draw メソッドの呼び出し方をまねる
  - 処理の終了後に repaint() を呼ぶ（記述済）
    - クリックされた図形を赤太線で描画し直すために，再描画が必要.

```
public class MyPanel extends JPanel{
   public void panelClicked(int x, int y) {
        // Step 2. insert here
        this.repaint();
        }
```

クリックによる選択

- void selectByClick(int x, int y) メソッド
  - どこで宣言する？　Selectable インタフェース
  - 仕様：引数であるクリックされた座標 x, y によって，自分は選択状態になるべきかを判断して，自分の selected フィールドの値をセットする.
    - 内側クリックで選択状態にする(trueをセットする)のはもちろん，外側クリックで非選択状態になるのも重要.
  - 今回の動作：「座標(x,y) が自分の内側かどうか」の判定結果に応じて，selected フィールドの値をセットする.
  - 「自分の内側に座標(x,y) が含まれるか」を判定するメソッドは，別のメソッド contains(x,y) として定義して，呼び出そう.
    - Q. 別のメソッドとした方がよいのはなぜか？
  - selectByClick()メソッドはどこで実装する？
    - 「Rect/Circle クラスで実装する」と思うかもしれないが...

内側の判定

- boolean contains(int x, int y) メソッド
  - 「自分の図形が引数の座標(x,y) を内側に含むからどうか」を判定して，boolean 値を返す．線上も内側と見なす．
    - 判定するだけ．selected フィールドは変更しない．
  - どこで宣言する？　図形の内側という概念は？？
    - 抽象(abstract)メソッドとしての宣言が必要
  - どこで実装する？　Rectクラス，Circleクラスで実装する．
    - 簡単な実装方法：引数の座標値(x,y)と自分の座標値を比べる．
    - 四角形(Rect)の場合：
      - フィールド(x1,y1)は左上座標，(x2,y2)は右下座標を表す．x2>x1かつ y2>y1 と仮定してよい．それらと座標値(x,y)を比べれば，内側かどうかが分かる．なお，問題文より，x1などに等しい場合（線の上）は「内側」と見なす．
      - 論理演算子　&& (AND) を使うと短く，書ける．
    - 円(Circle)は？　ヒント：円の内側＝中心からの距離が...
      - int ***x*** の絶対値をとる Math.abs(***x***) を使ってもよい．平方根は使わない方が簡単．円がMyPanel の左上にはみ出している（座標値がマイナスの値の）可能性は考慮しなくてよい．

ステップ3：選択状態による図形の描画

- draw() メソッドの拡張
  - RectとCircle のdraw() で，図形を描画する前に，選択状態に応じて，線の色と太さをセットする必要がある．
    - 「選択状態」なら「赤太線」に，そうでなければ「黒細線」にセットする．
  - Graphics g の描画色をセットするコード

```
g.setColor(Color.BLACK);
```

```
g.setColor(Color.RED);
```

  - Graphics g の線の太さを2（太線）にセットするコード
    - 2で太い線，1で細い線になる．

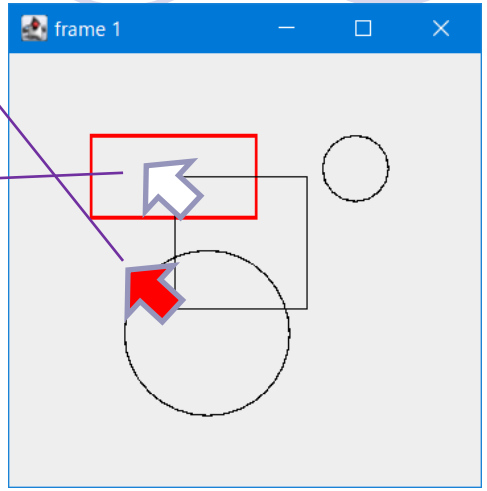
```
Graphics2D g2 = (Graphics2D)g;
float strokeWidth = 2;
g2.setStroke(new BasicStroke(strokeWidth));
```

  - このコードは Rect/Circle クラスに共通である．従って，どのクラスのメソッドに記述するのがよい？
    - OOP6-A ですでに黒細線のコードが書いてある．

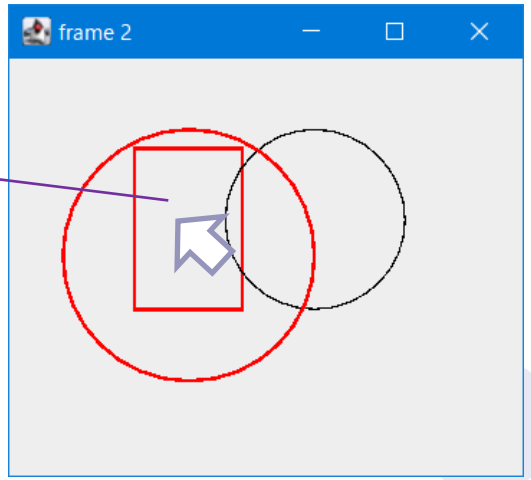
ステップ3の確認

- クリックして正しく描画されることを確認する.
  - 円の外側などをクリックしたときに正しく「選択解除される」ことも重要.

```
Clicked at (80,69)
Selected = R (50, 50)-(150, 100)
MyPanel painting ...
Drawing... R (50, 50)-(150, 100)
Drawing... R (100, 75)-(180, 155)
Drawing... C (120, 170) r=50
Drawing... C (210, 70) r=20
```




```
Clicked at (90,82)
Selected = R (70, 50)-(130, 140)
Selected = C (100, 110) r=70
MyPanel painting ...
Drawing... R (70, 50)-(130, 140)
Drawing... C (170, 90) r=50
Drawing... C (100, 110) r=70
```



今回の講義のまとめ

- イベント処理
  - イベントモデル
    - イベントリスナーの登録による「委譲」

今回 ➡ 1）Window/Panelに対するマウスイベント

- 今回の目標（レポート課題前半）：図形の内側をクリックして選択する．選択された図形は赤太線で，描画する．
- ステップ1(B1）：mouseEventクラス
  - マウスクリックに反応して，座標値をコンソールに出力する．
- ステップ2(B2)：クリックによる図形選択
  - 内側をクリックされた図形を判定して，コンソールに出力する．
- ステップ3(B3)：選択状態に合わせた描画
  - 選択されている図形を赤太線で描画する．

次回 ➡ 2）UIコンポーネントに対するイベント

- ActionEventクラス．Button へのクリックに対する反応．
- ボタンに応じた動作
  - 選択されている図形に対する処理など