Java/Swingについて (3)

2005年10月19日

海谷 治彦

目次

- メニューとAbstractAction
- ダイアログ
- ファイルダイヤログ
- Inner Class (内部クラス)
- Anonymous Inner Class (無名内部クラス)
- GUIでもちっとはクラス図を使おう.
- 実行可能アーカイブ (jar)の作り方
  - エクリプス無しでも実行したい.

メニューとAbstractAction

- メニューをつけると「それっぽい」プログラムになります.
- 従来のListener, Adapterではなく, AbstractActionを使って, メニュー選択時の動作を設定することもできる.
- 例題参照.

ダイヤログ

- メインのウインドウとは別のウインドウを出して，
  - 警告
  - 確認
  - 選択肢
  - 簡単な文字列入力

  等ができるもの．
- Swingでは，JOptionPaneクラス内にあるstaticメソッド(クラスメソッド)を利用することで，異常に簡単に実現可能．
- 個人的にはウザいと思うこともある．
  - 特に確認系のダイヤログとか．

警告 showMessageDialog()

- 入力エラーやプログラムの状態の不具合等をユーザーに警告するもの.
- 単にメインウインドウ(の隅の方)に警告が出るよりわかりやすい.

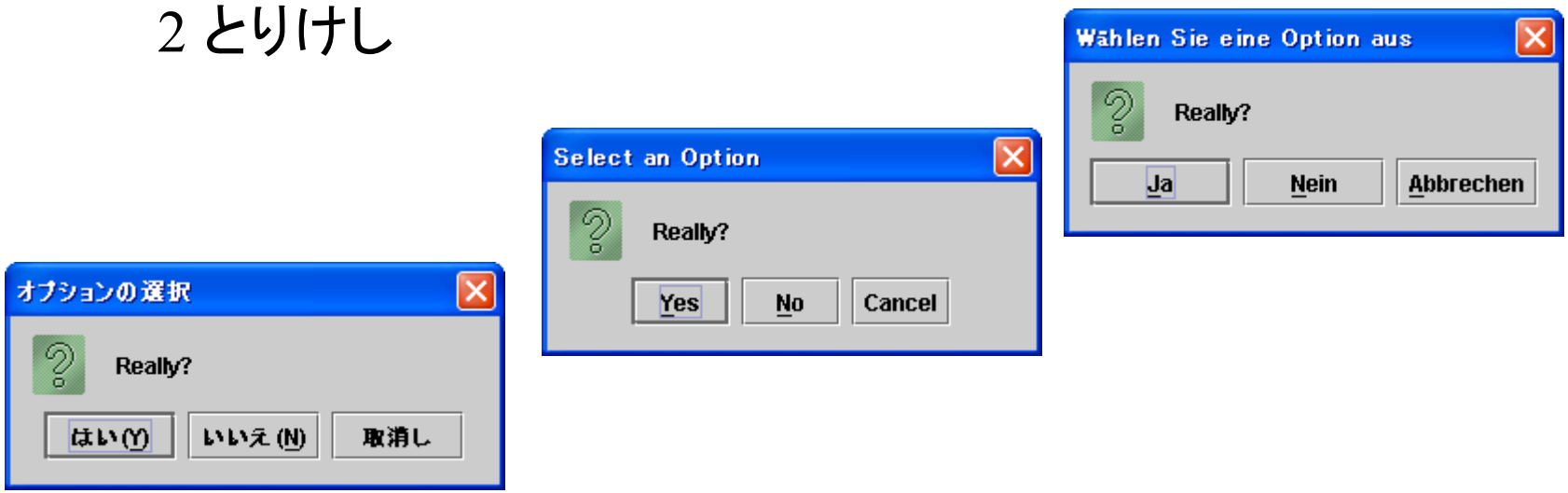
警告の例

```
 *
 * To change the template for this generated type commer
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;C
 */
public class Dialog2 extends MouseAdapter {
    private Component c;
    public static void main(String[] args) {
        // Locale.setDefault(Locale.ENGLISH);
        JFrame frame=new JFrame();
        frame.setSize(200,300);
        JPanel panel=(JPanel)frame.getContentPane().
        JButton b=new JButton("Fire");
        b.addMouseListener(new Dialog2(panel));

        panel.add(b, BorderLayout.NORTH);
        frame.setVisible(true);
    }

    Dialog2(Component c){
        this.c=c;
    }
    public void mousePressed(MouseEvent arg0) {
        JOptionPane.showMessageDialog(c, "Never Push Me!");
    }
}
```

ボタンを押したら「押すな」と警告するプログラム(涙)

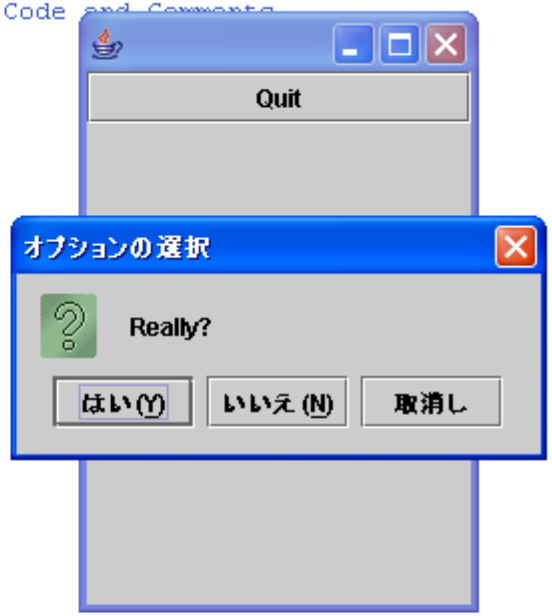
確認

- とりかえしのつかないこと等を実行する場合に，ホントにやっていいかを確認する場合など．
- showConfirmDialog() の返り値として，
  0 はいの場合
  1 いいえ
  2 とりけし



確認の例

```
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class Dialog3 extends MouseAdapter {
    private Component c;
    public static void main(String[] args) {
        //Locale.setDefault(Locale.GERMAN);
        JFrame frame=new JFrame();
        frame.setSize(200,300);
        JPanel panel=(JPanel)frame.getContentPane();
        JButton b=new JButton("Quit");
        b.addMouseListener(new Dialog3(panel));

        panel.add(b, BorderLayout.NORTH);
        frame.setVisible(true);
    }

    Dialog3(Component c){
        this.c=c;
    }
    public void mousePressed(MouseEvent arg0) {
        int ans=JOptionPane.showConfirmDialog(c, "Really?");
        if(ans==0) System.exit(1);
    }
}
```



いきなりプログラムを停止するのではなく,
とりあえず一度確認するとか.

ファイルについて

- 詳細は例題を参照.

Inner Class 内部クラス

- クラス内部で(ローカルな)クラスを定義できる.
- あるクラスの完全下請けなクラス(イベントリスナー等)はコレで実装したほうがよいかもしれない.
- あるクラスの中に定義されているので，親クラス内の属性にアクセスが許可されている.

例 (staticの場合)

```
public class Dialog4  {
  private static JPanel panel;

  public static void main(String[] args) {
    JFrame frame=new JFrame();
    frame.setSize(200,300);
    panel=(JPanel)frame.getContentPane();
    JButton b=new JButton("Quit");
    b.addMouseListener(new D4Adapter());

    panel.add(b, BorderLayout.NORTH);
    frame.setVisible(true);
  }

  static class D4Adapter extends MouseAdapter{
    public void mousePressed(MouseEvent arg0) {
      int ans=JOptionPane.showConfirmDialog(panel, "Really?");
      if(ans==0) System.exit(1);
    }
  }
}
```

例

```
public class Dialog5 extends JFrame {
  private JPanel panel;
  Dialog5(){
    super("5");
    JFrame frame=new JFrame("4");
    frame.setSize(200,300);
    panel=(JPanel)frame.getContentPane();
    frame.setVisible(true);
  }

  public void run(){
    JButton b=new JButton("Quit");
    b.addMouseListener(new D4Adapter());
    panel.add(b, BorderLayout.NORTH);
    panel.validate();
  }
  public static void main(String[] args) {
    new Dialog5().run();
  }

  private class D4Adapter extends MouseAdapter{
    public void mousePressed(MouseEvent arg0) {
      int ans=JOptionPane.showConfirmDialog(panel, "Really?");
      if(ans==0) System.exit(1);
    }
  }
}
```

無名インナークラス

- 以下のような場合，無名インナークラスを用いるのが良い.
  - 1回ポッキリしかインスタンスを作らない.
  - 単にいくつかのメソッドを再定義するだけで済む.
  - インスタンスを参照する場合，実装しているインタフェースもしくはスーパークラスで受ける.
- 無名インナークラスのもとになるのはクラスでもインタフェースでもよい.

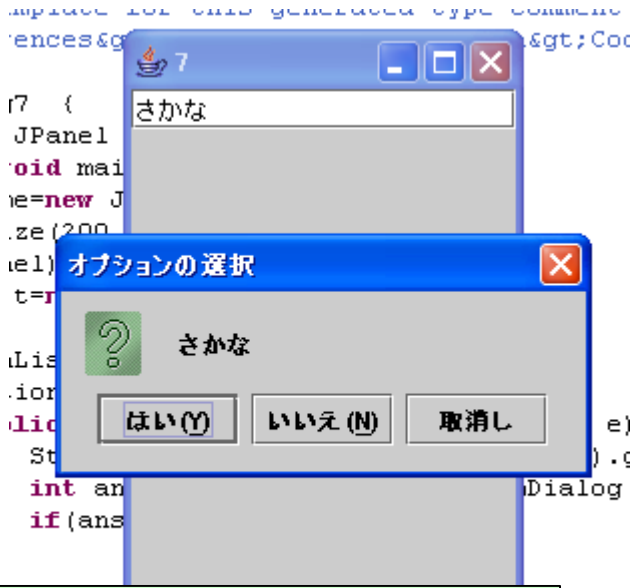
例（クラス拡張）

```
public class Dialog6  {
  private static JPanel panel;
  public static void main(String[] args) {
    JFrame frame=new JFrame("6");
    frame.setSize(200,300);
    panel=(JPanel)frame.getContentPane();
    JButton b=new JButton("Quit");

    b.addMouseListener(
      new MouseAdapter(){
        public void mousePressed(MouseEvent e){
          int ans=JOptionPane.showConfirmDialog(panel, "Really?");
          if(ans==0) System.exit(1);
        }
      }
    );

    panel.add(b,
        BorderLayout.NORTH);
    frame.setVisible(true);
  }
}
```

```
class XX extends  MouseAdapter{
  public void mousePressed(MouseEvent e){
    int ans=JOptionPane.showConfirmDialog(panel, "Really?");
      if(ans==0) System.exit(1);
    }
  }
}

MouseAdapter a=new XX();
```

例 (IF実装)

```
public class Dialog7  {
  private static JPanel panel;
  public static void main(String[] args) {
    JFrame frame=new JFrame("7");
    frame.setSize(200,300);
    panel=(JPanel)frame.getContentPane();
    JTextField t=new JTextField(20);

    t.addActionListener(
      new ActionListener(){
        public void actionPerformed(ActionEvent e){
          String s=((JTextField)e.getSource()).getText();
          int ans=JOptionPane.showConfirmDialog(panel, s);
          if(ans==0) System.exit(1);
        }
      }
    );

    panel.add(t, BorderLayout.NORTH);
    frame.setVisible(true);
  }
}
```




```
class XX implements ActionListener{
  public void actionPerformed(ActionEvent e){
    String s=((JTextField)e.getSource()).getText();
    int ans=JOptionPane.showConfirmDialog(panel, s);
    if(ans==0) System.exit(1);
  }
}

ActionListener a=new XX();
```

内部クラスについての考察

- 推奨するか否かは微妙.
- 前述の例のように場合によっては手間が減る.
- インデントが深いので見難い.
- 変数(属性)の可視スコープをうまく利用したプログラミング(のサボリ)ができる.
  - 逆にスコープを理解してない読みにくくなる.
  - 引数等で明示的にインスタンスへの参照を受け取る方法と相補的.
    - どちらが良いかはケースバイケースだとおもふ.

クラス図からはじめよう

- GUIを作る場合，クラス図を用いて設計はしにくい.
  - 主な部分は「見てくれ」の部分だし.
- しかし，特にイベントの流れに相当する部分はクラス図で把握しておくのもよいかもしれない.
- この実験で紹介したSwingの部品は，基本的にデータの入出力に用いるものである.
  - 計算を行うクラスは従来通りに設計する必要がある.

ソースコード

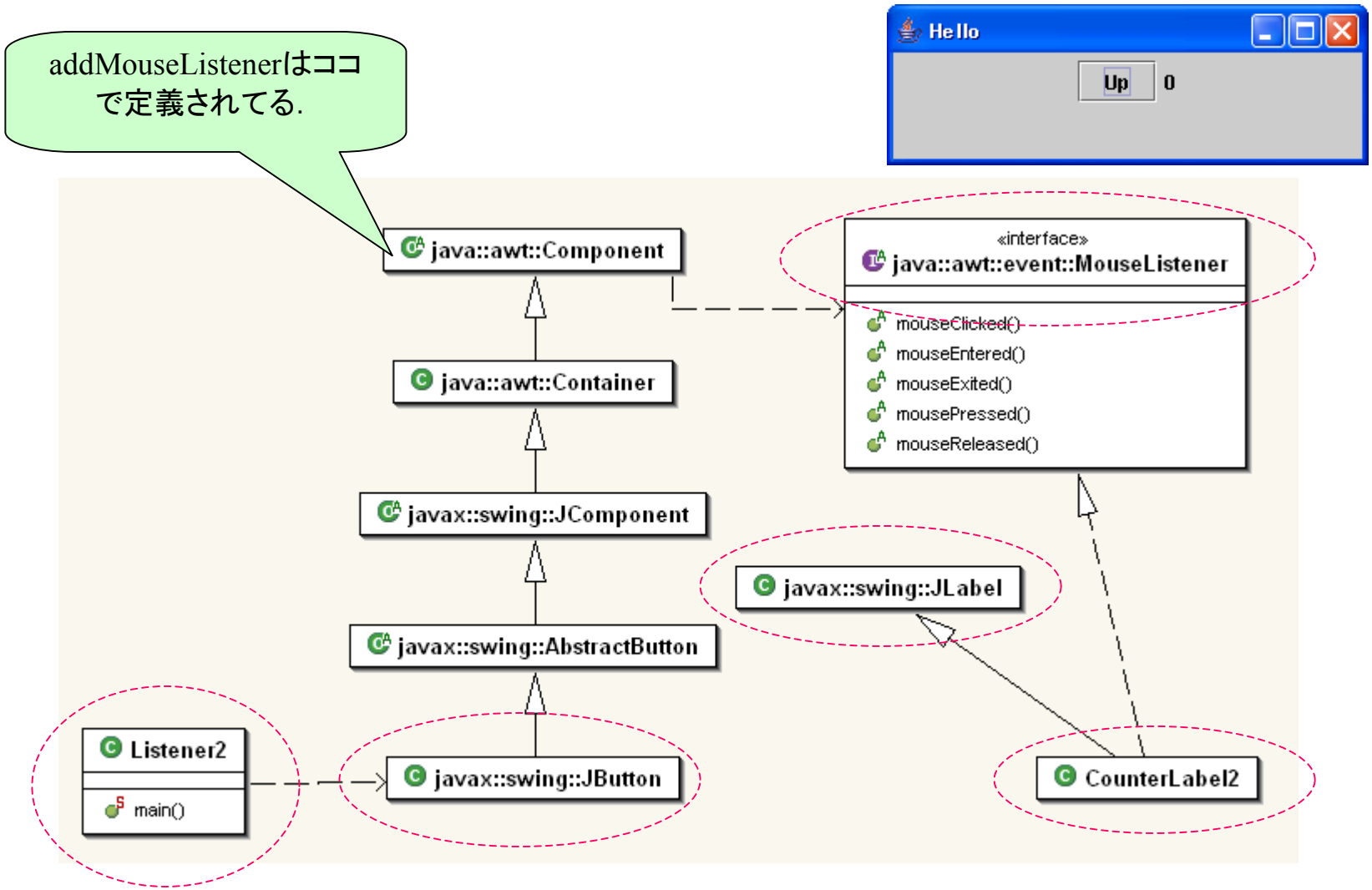
```
public class Listener1 {
  public static void main(String[] args){
    JFrame jf=new JFrame("Hello");
    jf.setSize(300, 100);
    JPanel panel=new JPanel();
    jf.setContentPane(panel);

    JButton button=new JButton("Up");
    panel.add(button);
    CounterLabel counter=new CounterLabel();
    panel.add(counter);

    button.addMouseListener(counter);

    jf.setVisible(true);
  }
}
```

```
public class CounterLabel extends JLabel
                          implements MouseListener {
    private int c=0;

    CounterLabel(){ super(0+""); }
    public void mouseClicked(MouseEvent arg0) {}
    public void mouseEntered(MouseEvent arg0) {}
    public void mouseExited(MouseEvent arg0) {}
    public void mouseReleased(MouseEvent arg0) {}

    public void mousePressed(MouseEvent arg0) {
      c++;
      setText(c+"");
    }
}
```

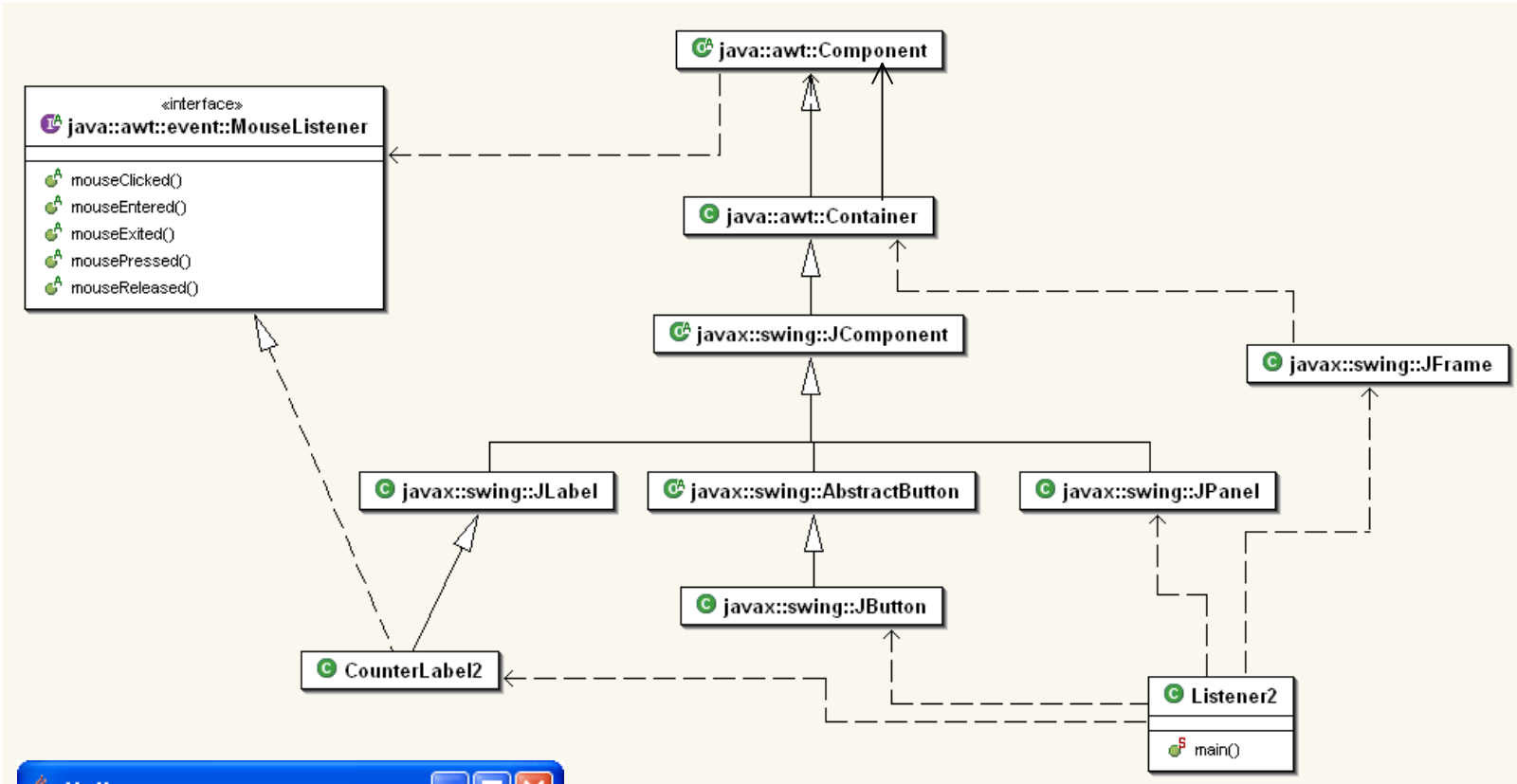
ボタンbuttonのイベントを
ラベルcounterが聞くように指示

ボタン系のイベントに対応して行う処理を，
ラベル(リスナー)内に実装．

参考までにクラス図

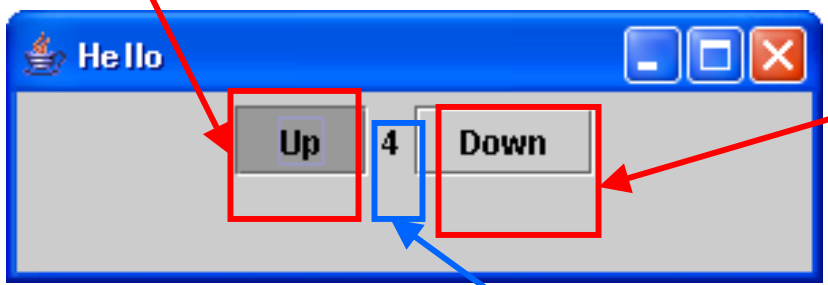

addMouseListenerはココで定義されてる.
Hello
Up
0
java::awt::Component
«interface»
java::awt::event::MouseListener
mouseClicked()
mouseEntered()
mouseExited()
mousePressed()
mouseReleased()
java::awt::Container
javax::swing::JComponent
javax::swing::JLabel
javax::swing::AbstractButton
Listener2
main()
javax::swing::JButton
CounterLabel2

相互の関係を網羅すると以下



ソースが複数の場合は？

イベントソース
押すと増える



イベントソース
押すと減る

リスナー

リスナー側で
「どこからの」イベントかを判別しないといけない.

ソース (抜粋)

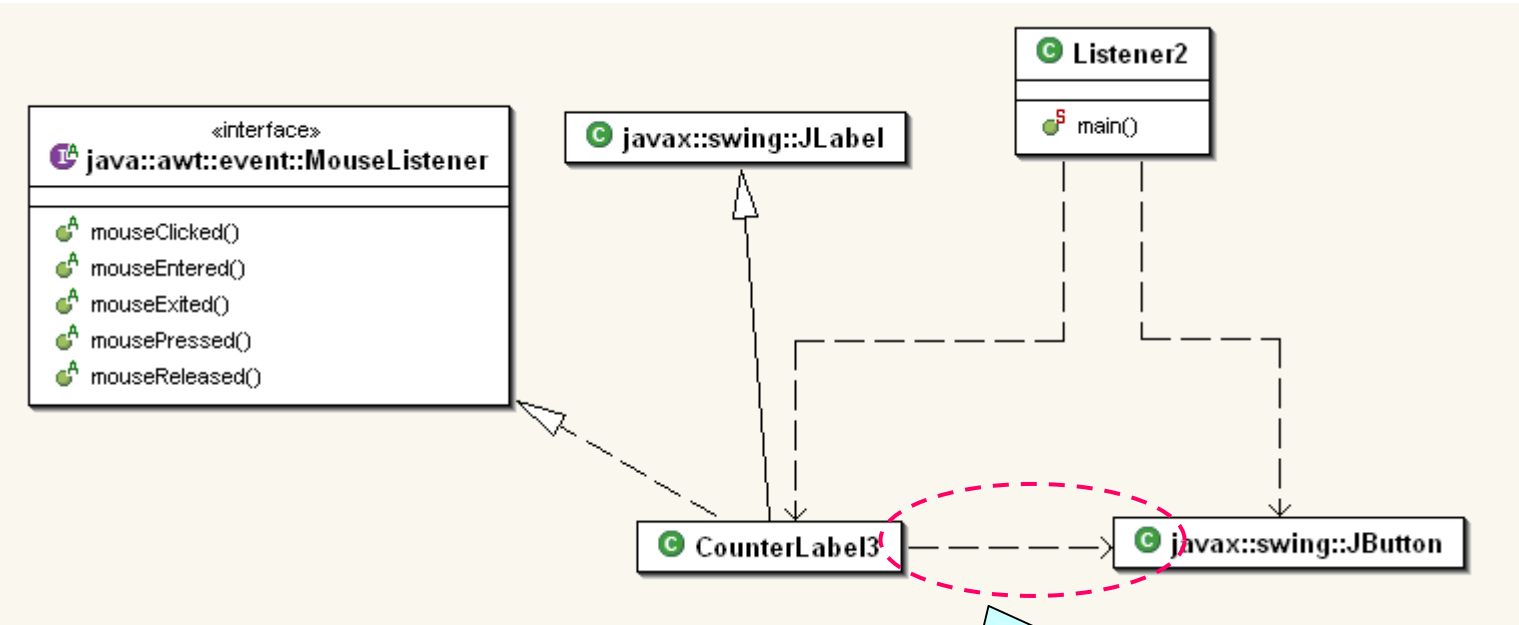
```
public class Listener2 {
  public static void main(String[] args){
    JFrame jf=new JFrame("Hello");
    jf.setSize(300, 100);
    JPanel panel=new JPanel();
    jf.setContentPane(panel);

    JButton up=new JButton("Up");
    JButton down = new JButton("Down");
    CounterLabel3 counter=new CounterLabel3(0, up, down);

    up.addMouseListener(counter);
    down.addMouseListener(counter);

    panel.add(up);
    panel.add(counter);
    panel.add(down);
    jf.setVisible(true);
  }
}
```

ポイントはリスナー側で，複数あるソースの情報を保持していること.

```
public class CounterLabel3 extends JLabel
                              implements MouseListener {
  private int c=0;
  private JButton up;
  private JButton down;

  CounterLabel3(int x, JButton up, JButton down){
    super(x+"");
    c=x;
    this.up=up;
    this.down=down;
  }

  public void mousePressed(MouseEvent arg0) {
    JButton b=(JButton)arg0.getSource();
    if(b==up){
      c++;
    }else if(b==down){
      c--;
    }
    setText(c+"");
  }

  public void mouseReleased(MouseEvent arg0) {}
  public void mouseClicked(MouseEvent arg0) {}
  public void mouseEntered(MouseEvent arg0) {}
  public void mouseExited(MouseEvent arg0) {}
}
```

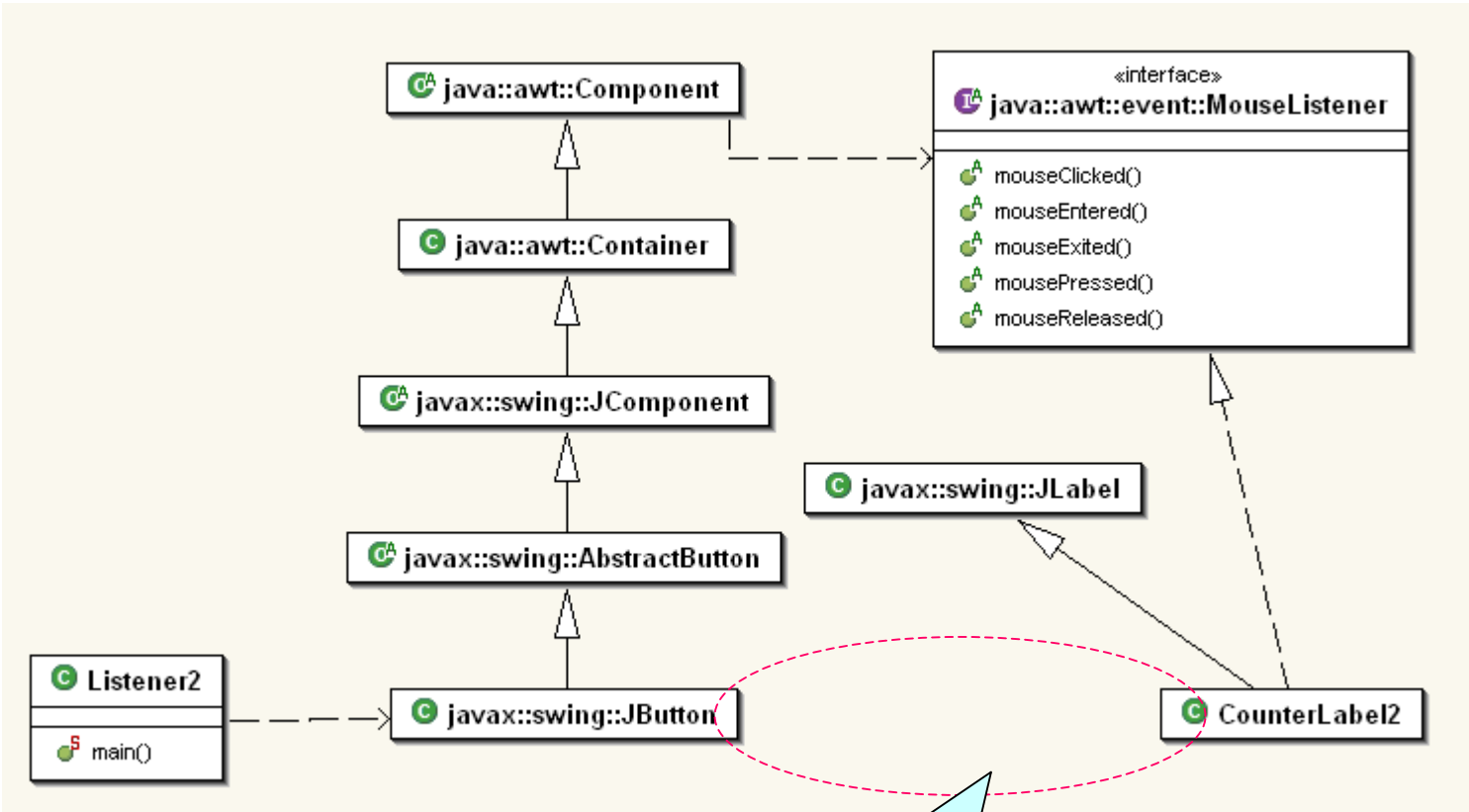
リスナー

クラス図



ココの情報保持があって，複数のボタン(ソース)の区別をしている．

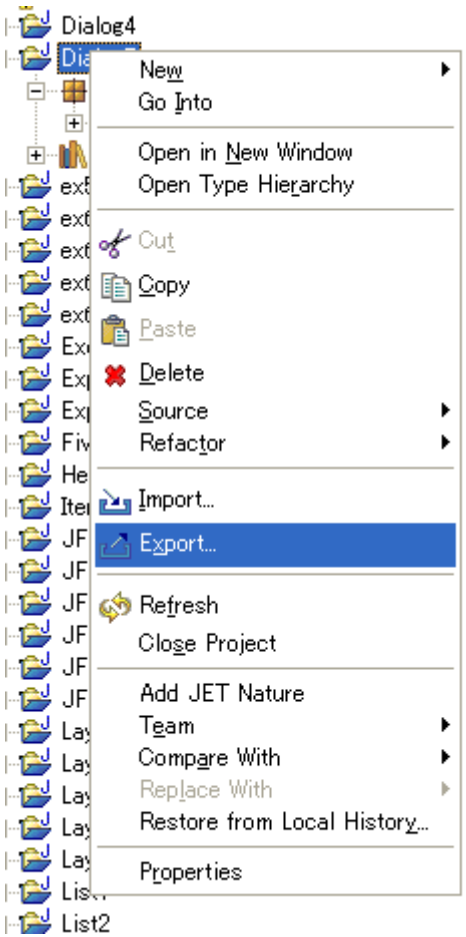
前2つの例のクラス図 (再)

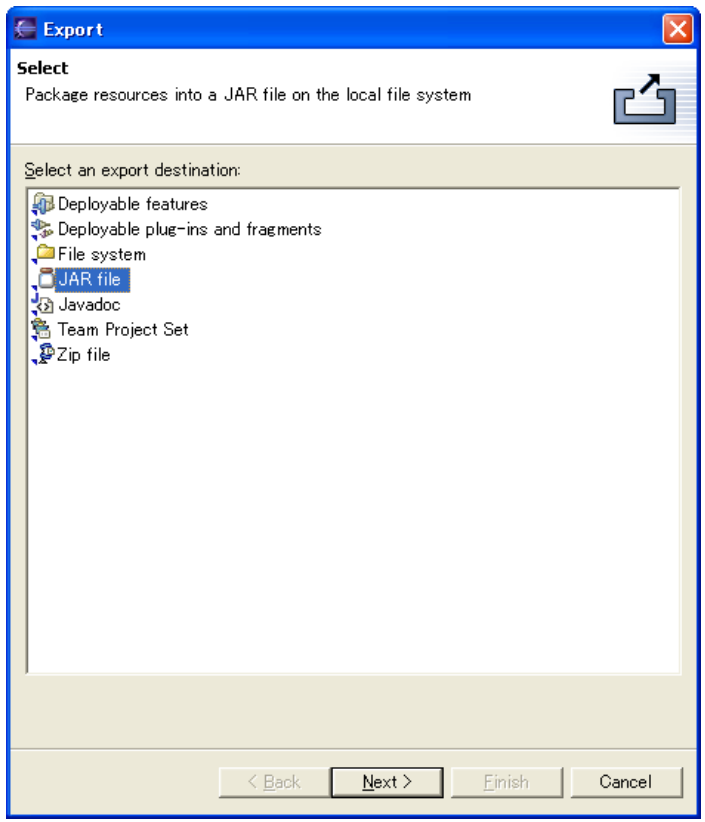


jarアーカイブ

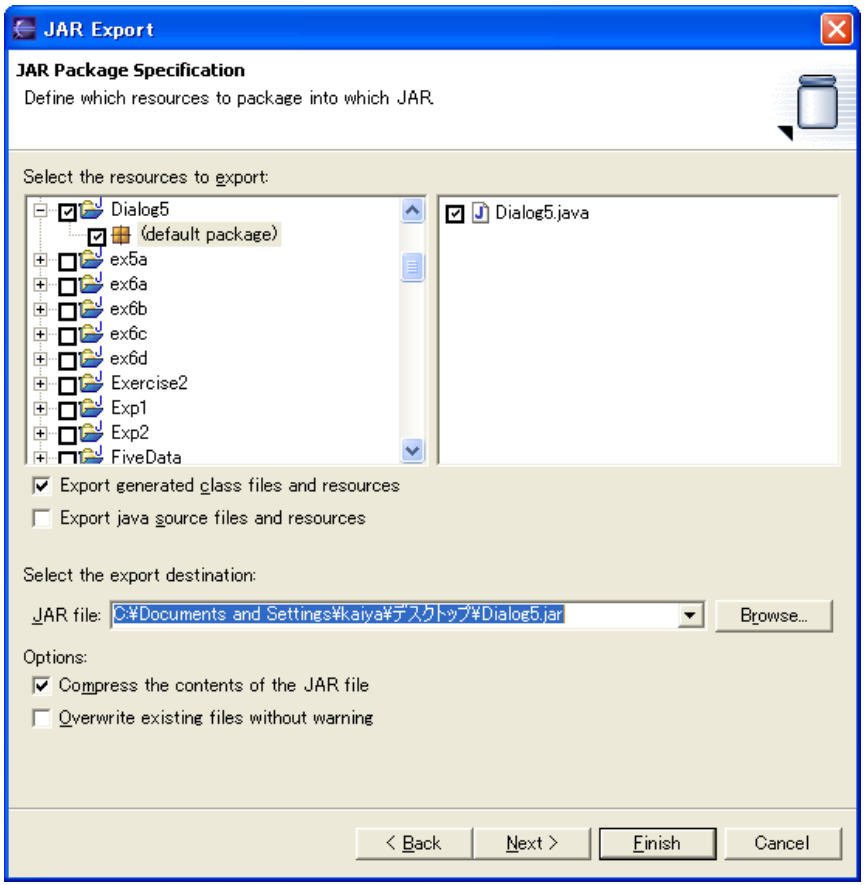
- Eclipseが無くてもJ2SDK, JDKがなくても, JRE (Java Runtime Env.)があれば開発したアプリケーションを実行できる.
- 無論，クラスファイル全部をばらばらに配布しても良いのだが,
- 1つのファイルにまとめておいたほうが扱いやすい.
- そのためのアーカイブ形式が jar

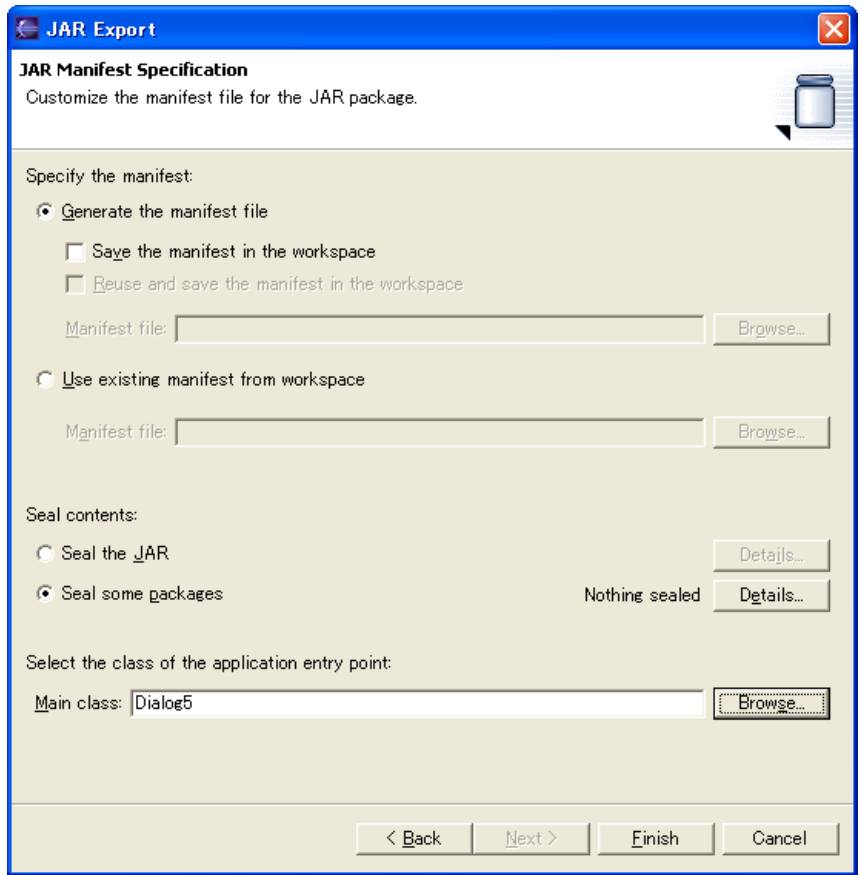
やり方1

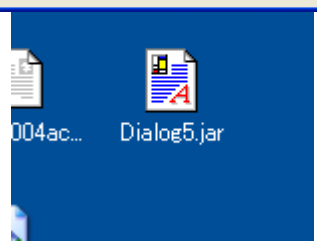

Dialog4
New
Go Into
Open in New Window
Open Type Hierarchy
Cut
Copy
Paste
Delete
Source
Refactor
Import...
Export...
Refresh
Close Project
Add JET Nature
Team
Compare With
Replace With
Restore from Local History...
Properties
List2


Export
Select
Package resources into a JAR file on the local file system
Select an export destination:
Deployable features
Deployable plug-ins and fragments
File system
JAR file
Javadoc
Team Project Set
Zip file
< Back
Next >
Finish
Cancel

やり方 2


JAR Export
JAR Package Specification
Define which resources to package into which JAR.
Select the resources to export:
Dialog5
(default package)
ex5a
ex6a
ex6b
ex6c
ex6d
Exercise2
Exp1
Exp2
FiveData
Dialog5.java
Export generated class files and resources
Export java source files and resources
Select the export destination:
JAR file:
C:¥Documents and Settings¥kaiya¥デスクトップ¥Dialog5.jar
Browse...
Options:
Compress the contents of the JAR file
Overwrite existing files without warning
< Back
Next >
Finish
Cancel


JAR Export
JAR Manifest Specification
Customize the manifest file for the JAR package.
Specify the manifest:
Generate the manifest file
Save the manifest in the workspace
Reuse and save the manifest in the workspace
Manifest file:
Browse...
Use existing manifest from workspace
Manifest file:
Browse...
Seal contents:
Seal the JAR
Details...
Seal some packages
Nothing sealed
Details...
Select the class of the application entry point:
Main class:
Dialog5
Browse...
< Back
Next >
Finish
Cancel


004ac...
Dialog5.jar