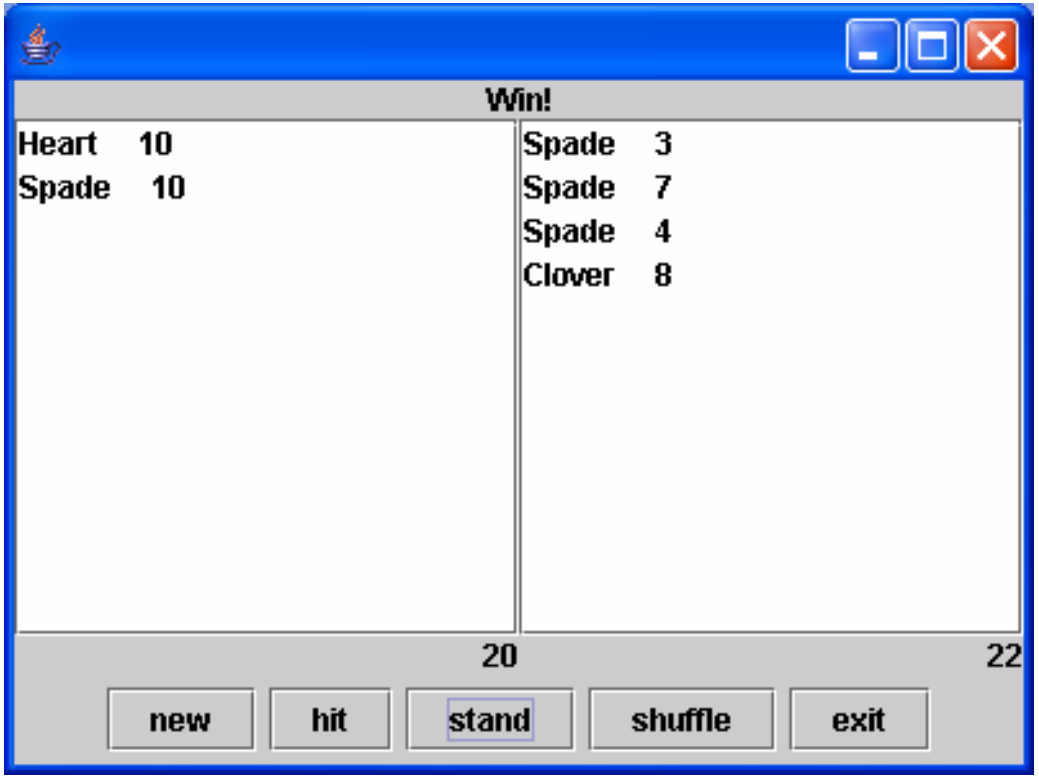
（１０）JavaのGUI

■ここでは、blackjackのプログラムで用いたJavaのGUI(Graphic)について説明していきます。

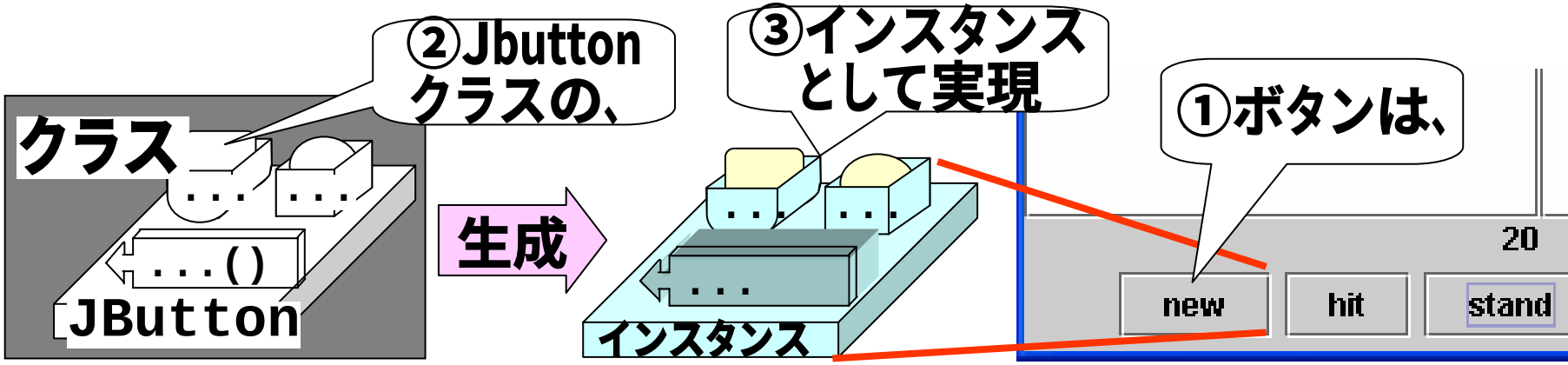
■系統的に説明を行う訳ではなく、あくまでblackjackのプログラムを直感的に理解するための説明です。



(10．1) 画面中の要素

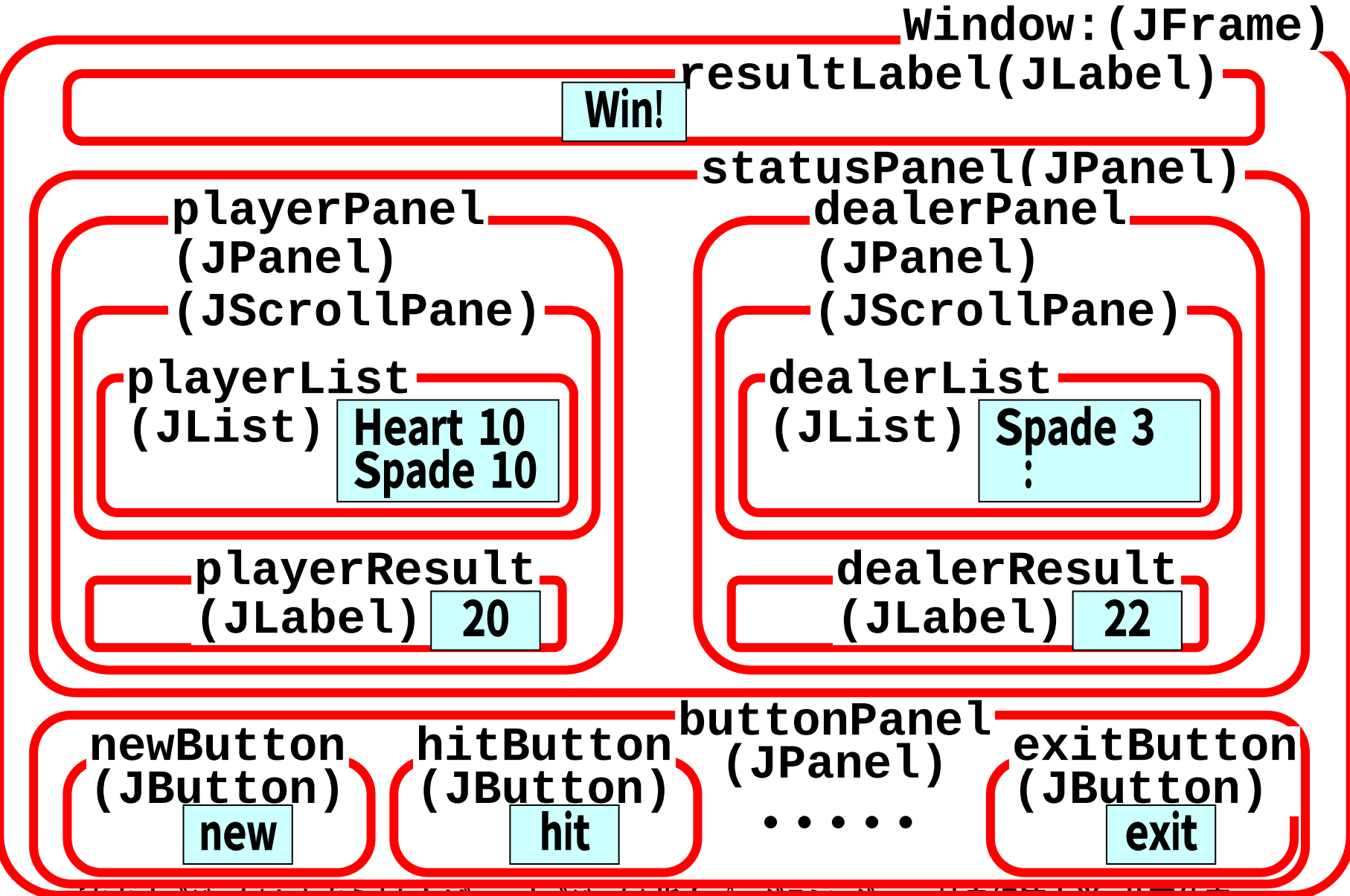
■画面に見える“要素”は、それぞれ特定のクラスのインスタンスとして実現されています。

■たとえば、画面中のボタンは、Jbuttonクラスによりその外見･機能が実現されています。



■本スライドでは、これら各々のクラスについて、Blackjack内での使い方のみを説明します。各々の意味づけや仕様については、オンラインマニュアル等を参照してください。

(10.2.1) 画面上の要素の構成



(10．2．2) 前スライドの説明

■前スライド中、() 内はクラス名を示しています。クラス名は、パッケージ名の“javax.swing”を略しています。

■下記の記述は、Windowクラスが、JFrameクラスを拡張していることを示します。

```
Window:(JFrame)
```

■下記のような記述では、JListクラスのインスタンスが、クラス型変数“playerList”に保持されていることを示し、これにより、画面中の

Heart 10
Spade 10

の表示が実現されていることを示します。

```
playerList(JList)
```

Heart 10
Spade 10

(10.3) 要素中に追加する

■スライド(10.2.1)に記した画面中の要素は、“Window”クラス内で定義されています。

- 例：“newButton” ← JButtonクラスのインスタンス

```
newButton = new JButton("new");
```

- 例：“buttonPanel” ← JPanelクラスのインスタンス

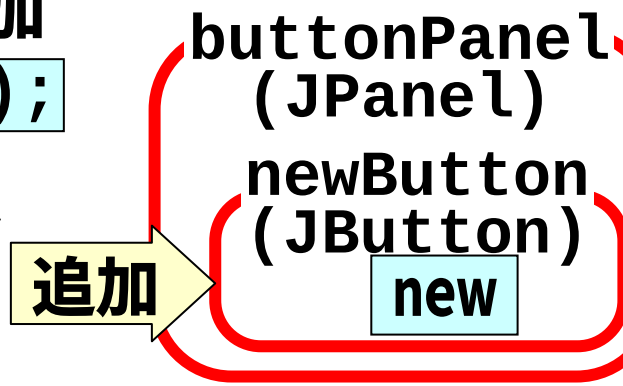
```
JPanel buttonPanel = new JPanel();
```

■スライドのように構成するために、コンストラクタ内で要素中に他の要素を追加する処理を行っています。

- 例：buttomPanel内に、newButtonを追加

```
buttonPanel.add(newButton);
```

■他の要素についても、おおよそ同じ方法で追加を行っています。



(10．4) レイアウト

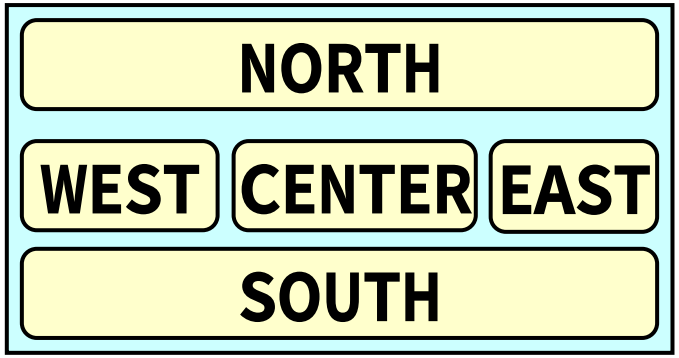
■レイアウトとは、ボタンなどの要素 (コンポーネントと呼びます) を、どのような位置に配置するかを決めることを指します。

■レイアウトには種類がありますが、Blackjackで用いられているのは、次のイアウトです。

フローレイアウト (FlowLayout)

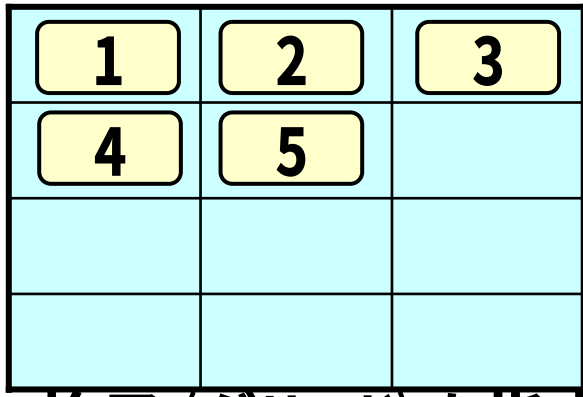
・左から右へ、上から下へ、順に配置する (デフォルトは中央揃え)

ボーダーレイアウト (BorderLayout)



・上記のような、東西南北中央を指定して配置する

グリッドレイアウト (GridLayout)



・格子 (グリッド) を指定 (上記では4行3列) し、その升目に従い、左から右へ上から下へ順に配置する

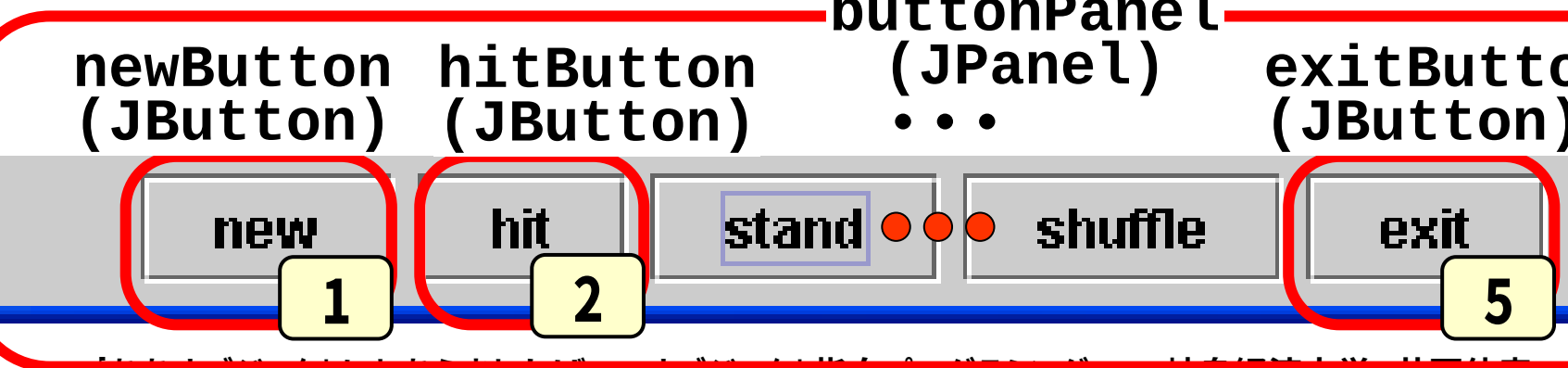
(10．4．1) フローレイアウト

■Jpanelクラスのインスタンスは、レイアウトについて何も指定しないと、フローレイアウトが適用されます。

■Blackjackでは、buttonPanel内にボタンが追加される際、フローレイアウトとなっています。

```
JPanel buttonPanel = new JPanel();
buttonPanel.add(newButton);
buttonPanel.add(hitButton);
buttonPanel.add(standButton);
buttonPanel.add(shuffleButton);
buttonPanel.add(exitButton);
```

指定がないので、フローレイアウト



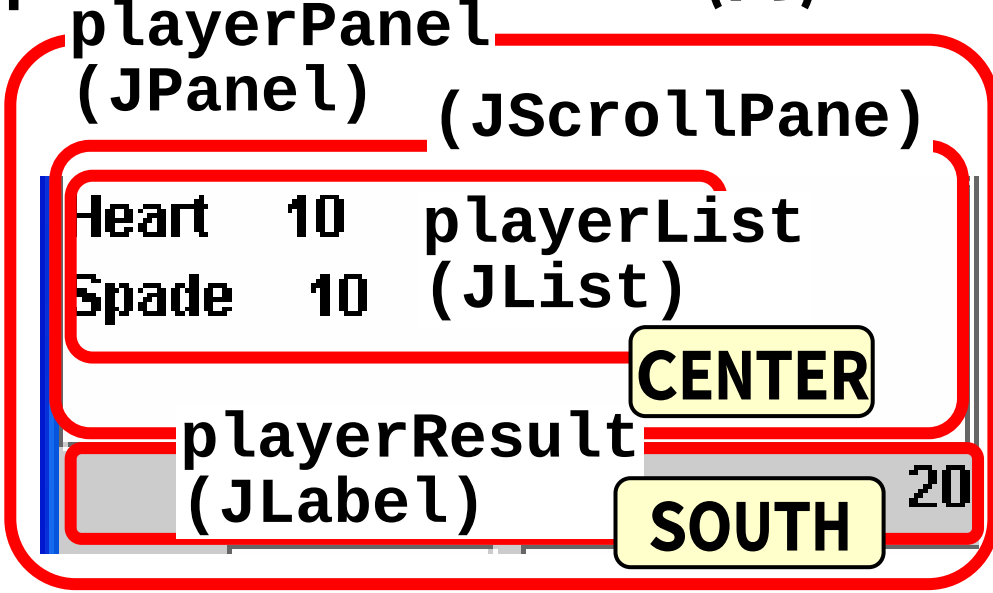
(10．4．2) ボーダーレイアウト

■plyerPanelではボーダーレイアウトが使われています。

```
playerPanel = new JPanel(new BorderLayout());
```

■playerListを内包したJScrollPanelクラスのインスタンスがＣＥＮＴＥＲ(中央)に、playerResultがSOUTH(南)に配置されています。

(この2つ以外は
配置されず、
結果として
2段重ねの
配置になっている。)




```
layerPanel.add(
 new JScrollPane(playerList), BorderLayout.CENTER)
layerPanel.add(playerResult, BorderLayout.SOUTH);
```

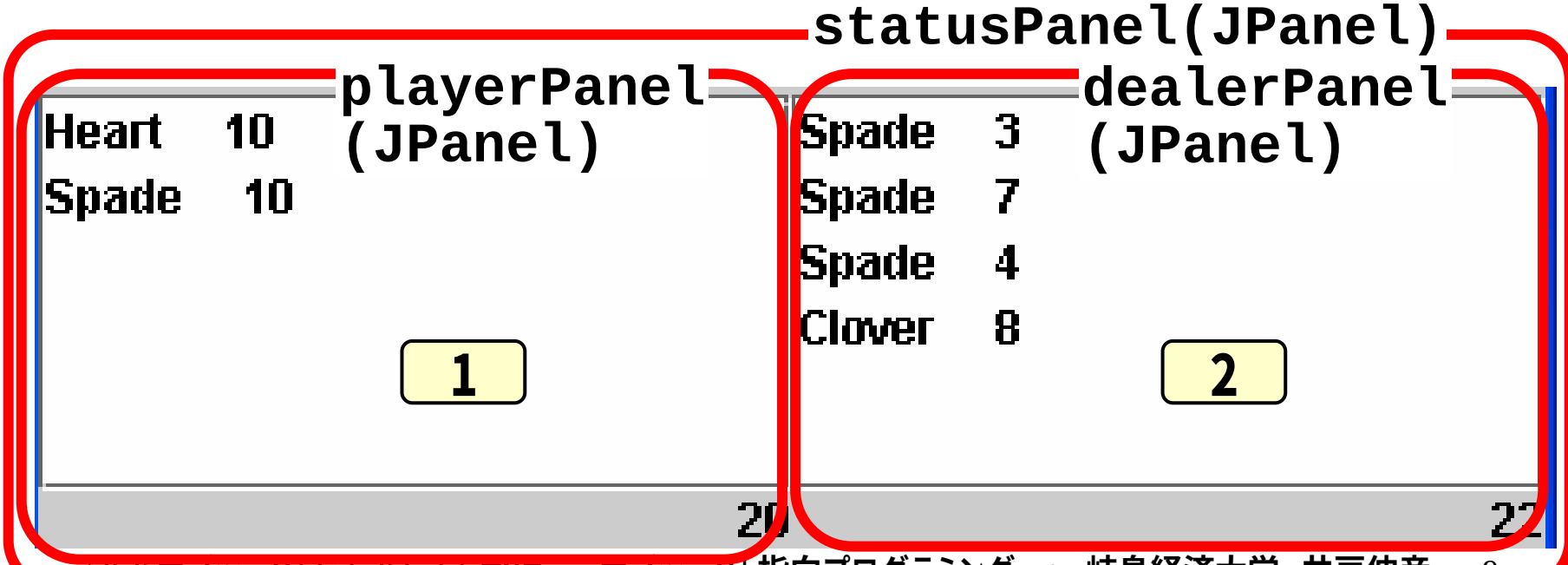
(10．4．3)グリッドレイアウト

■statusPanelでは、1行2列のグリッドレイアウトが使われています。

```
Panel statusPanel=new JPanel(new GridLayout(1, 2))
```

■1つめにplayerPanelが、2つめにdealerPanelが配置されています。

```
statusPanel.add(playerPanel);
statusPanel.add(dealerPanel);
```



(10.4.4)もうひとつのBorderLayout

■Windowクラスは、JFrameクラスを継承しています。

```
public class Window extends JFrame …… {
```

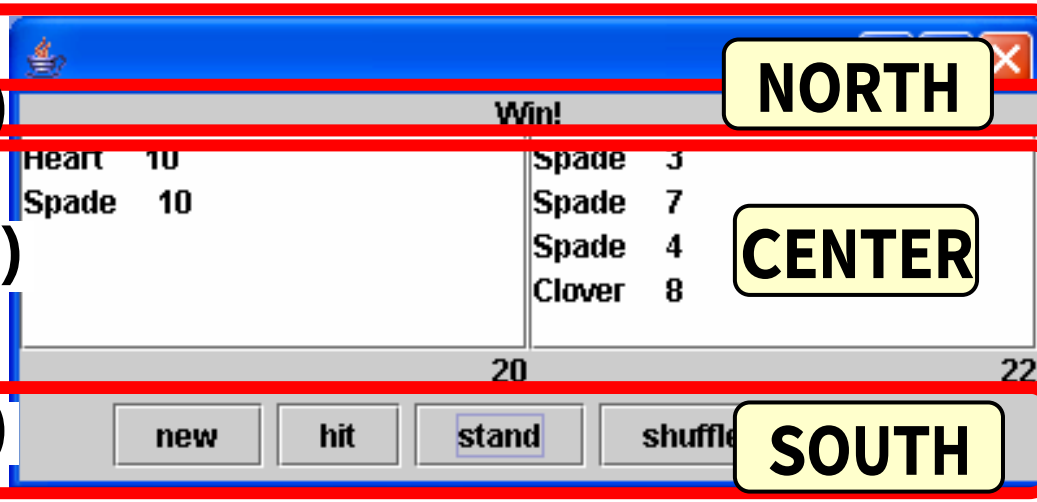
■Jframeクラスは、デフォルトでボーダーレイアウトとなっています。この上に、resultLabelがNORTH(北)に、statusPanelがCENTER(中央)に、buttonPanelがSOUTH(南)に配置されています。

Window:(JFrame)

resultLabel(JLabel)

statusPanel(JPanel)

buttonPanel(JPanel)



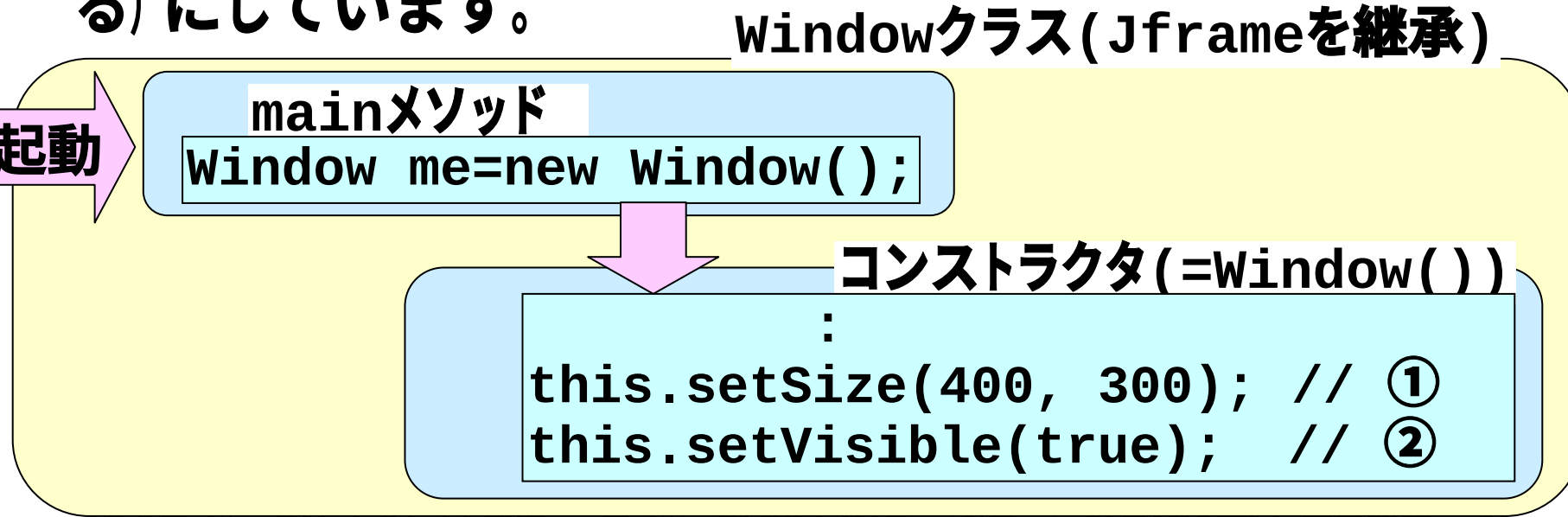

```
his.getContentPane().add(resultLabel, BorderLayout.NORTH)
his.getContentPane().add(statusPanel, BorderLayout.CENTER
his.getContentPane().add(buttonPanel, BorderLayout.SOUTH)
```

(10.5.1) JFrameの表示

■Windowクラスが起動されると、mainメソッドが呼び出されます。

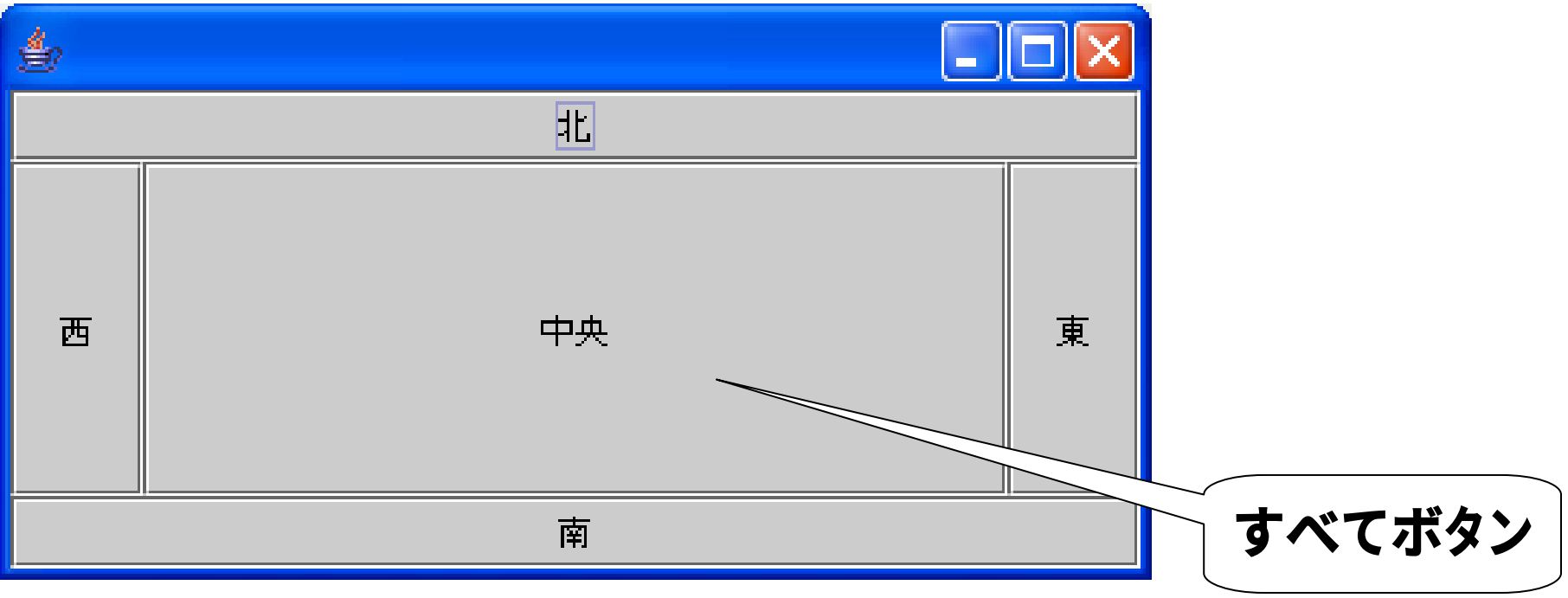
■mainメソッドでは、Windowクラスのインスタンスを生成しており、この際に起動されるコンストラクタ内でスライド(10.2.1)のような画面が構成されます。

■コンストラクタの最後では、①JFrameにより表示されるウィドウのサイズを決め、②可視状態（見えるようになる）にしています。



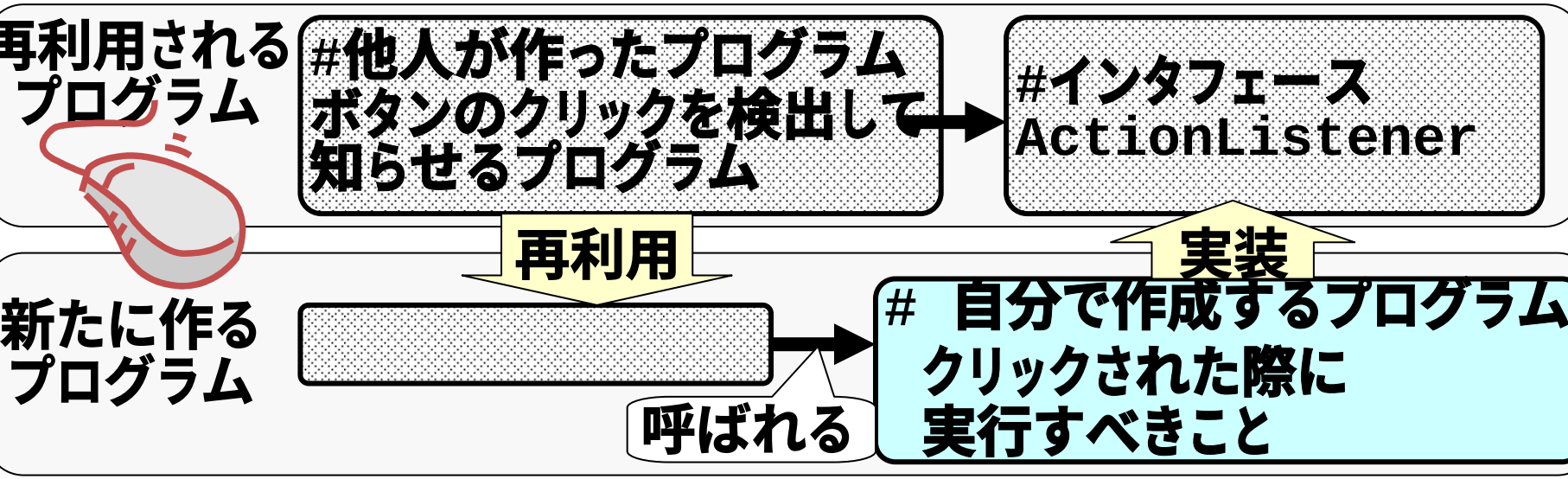
(10．5．2) 発展演習 (課題10－1)

■blackjackのプログラム、および、スライド (10) を参考にして、次のようなウインドウを表示するプログラムを作ってください。



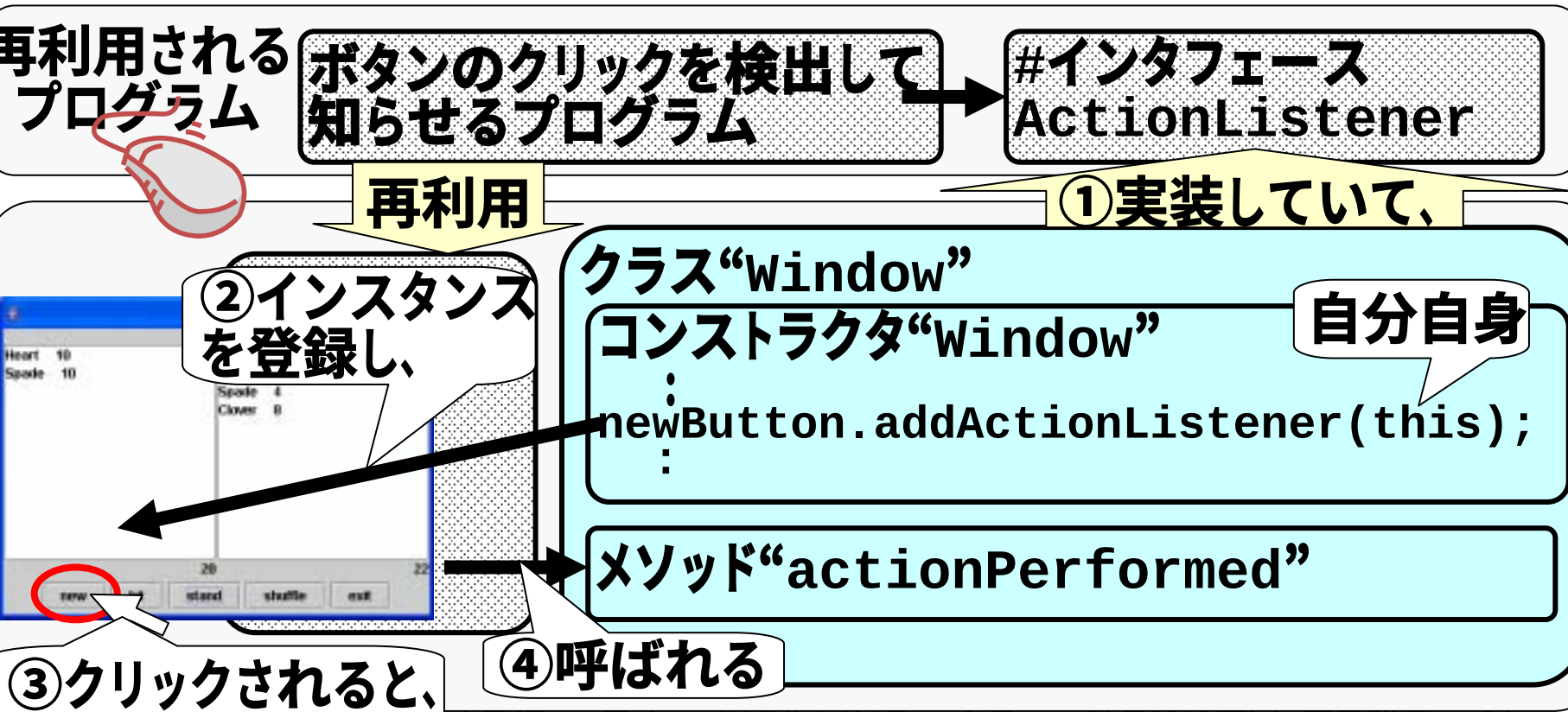
(10.6) ボタンのクリックを処理する

- 「ボタンがクリックを検出して知らせるプログラム」は、自分で作成するのではなく、Javaのデフォルトの環境で利用できるプログラムを利用します。
- 自分で作成する必要があるのは、「ボタンをクリックしたらすべきことが記されたプログラム」です。
- このようなプログラムを作成する際に、インタフェース“ActionListener”を実装します。



(10.6.1) ActionListenerの実装による動作

■Windowクラスが①ActionLinerを実装 (implements) しており、②ボタンに対して自分自身のインスタンスを登録しているので、③ボタンがクリックされた際、④このクラスのactionPerformedメソッドが呼び出されます。



（10．6．2）Windows.java

■Window.javaのファイル中、前スライドに関係する部分は次のとおりです。

①ActionListene
を実装していて、

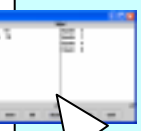
```
public class Window extends JFrame
                    implements ActionListener {
      :
  private javax.swing.JButton newButton;
      :
  public Window() {
      :
    newButton = new JButto   new");
      :
    newButton.addActionListener(this);
  }
  public void actionPerformed(ActionEvent ae) {
    // ボタンがクリックされた際に動くプログラム
      :
  }
```

②インスタンスを登録し、

“new”ボタン

自分自身

④呼ばれる

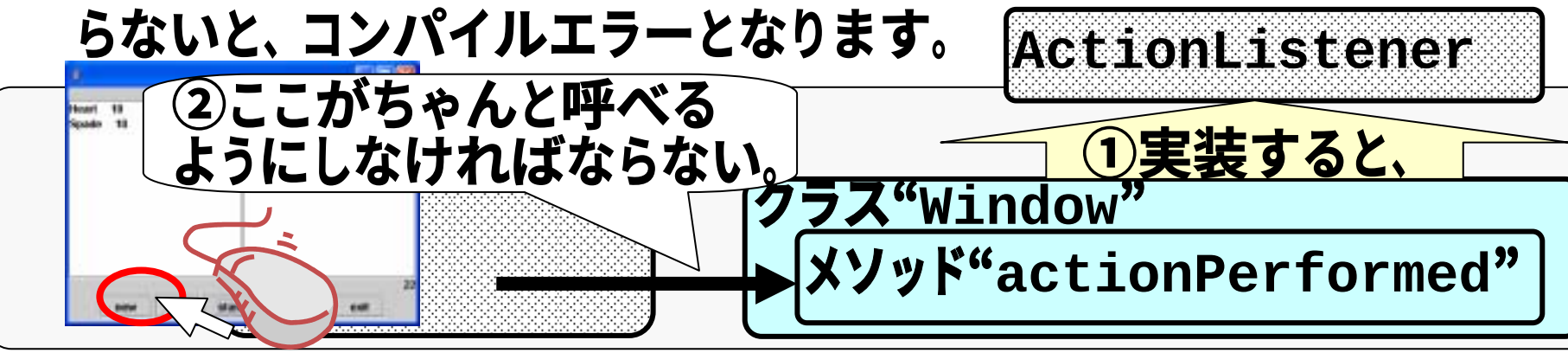


③クリックされると、

(10．6．3) インタフェース

■ActionListenerは、インタフェースと呼ばれるJavaの仕組みです。

■マウスでボタンをクリックした際に、どのようなメソッドが呼ばれるかについては、約束ごとが必要です。“actionPerformed”というメソッドが呼ばれるのが約束なのですが、ActionListenerインタフェースを実装 (implements) することにより、この約束事を守らせるようにしています。具体的には、“actionPerformed”を作らないと、コンパイルエラーとなります。



■インタフェースについては、「Javaオブジェクト指向スライド」の(11)章を参照してください。該資料では、マウスのウインドウ内 (＝ボタンでは無い) でのクリックを検出する際の例が示されています。

(10.6.4) actionPerformedメソッド

■actionPerformedでは、クリックされたボタンを入力パラメタのActionEventから判定して、必要な処理を行っています。

```
public void actionPerformed(ActionEvent ae) {

  Object obj = ae.getSource();


  if(obj.equals(newButton)) {
    // newボタンがクリックされた際の処理
          :
  }else if(obj.equals(hitButton)) {
    // hitボタンがクリックされた際の処理
          :
  }else ……
          :
}
```

①入力パラメタから、イベントの元(source)となったオブジェクトを取り出す

②元となったオブジェクトがnewButtonならば、(newボタンがクリックされたのならば)

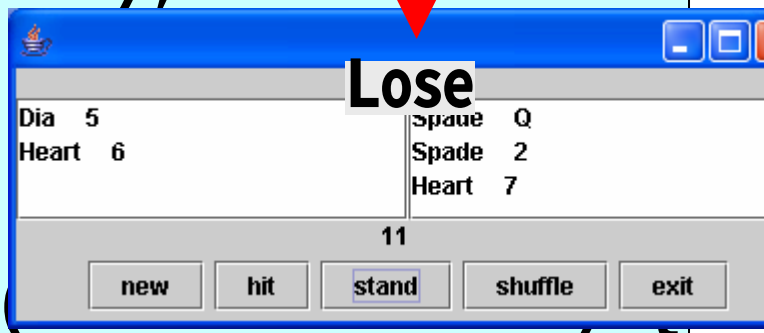
③(hitボタンがクリックされたのならば)

(10.6.5) JLabelへのテキスト設定

■ボタンをクリックされた際、Window.javaでは様々な処理を行っています。例えば、負けた際の表示は、次のようにJLabelへのテキスト設定にて行っています。

```
public void showResult(boolean isWin) {
        :
    resultLabel.setText("Lose");
        :
}

public void actionPerformed(
        :
  }else if(obj.equals(hitButton)){
        :
      showResult(false);
        :
}
```



（１０．６．６）発展問題（課題１０－２）

■blackjackのプログラム、および、スライド（１０）を参考にして、次のように、クリックしたボタンを表示するウインドウのプログラムを作ってください。