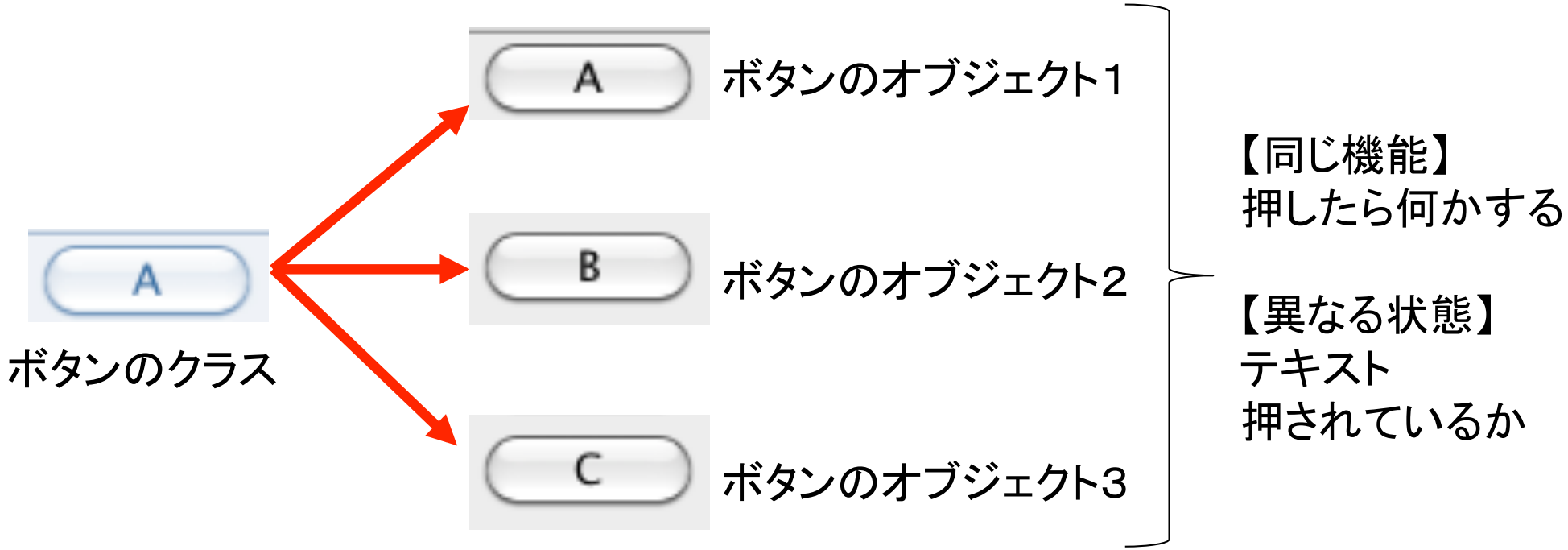
プログラミング基礎

GUI (イベント駆動型プログラミング)

OOP (interface, 継承)

復習：クラスとオブジェクト

- ひとつのクラスから複数のオブジェクトを作る
  - それぞれ自分の状態を覚えている



復習：Java API とクラス

- クラスを作る

```
class Cat{
  int x;
  int y;

  void moveTo(int x, int y){
    this.x = x;
    this.y = y;
  }
```

- クラスを使う

```
c = new Cat();
```

- Java API の場合
  - クラスは用意されている

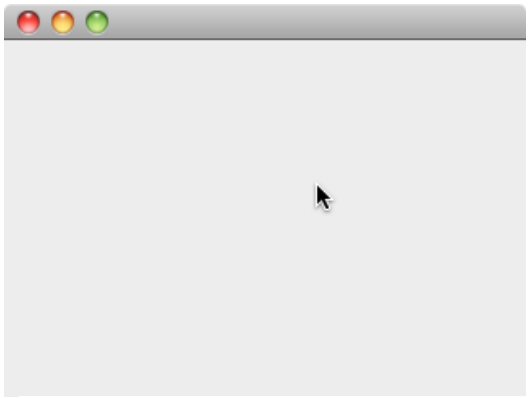
```
public class JButton {
    // イメージ図...
}
```

- クラスを使う

```
JButton button = new JButton("送信");
```

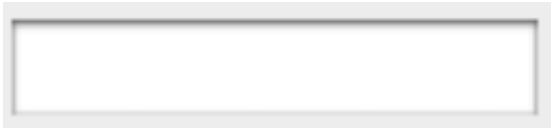
オブジェクト指向は
クラスを作る人と使う人の
連携に便利！

プログラミングの分割（例）



JFrameのオブジェクト

JButtonのオブジェクト

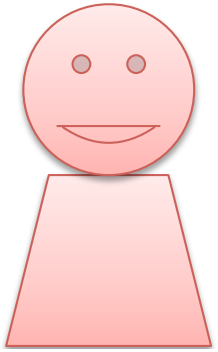


JTextFieldのオブジェクト

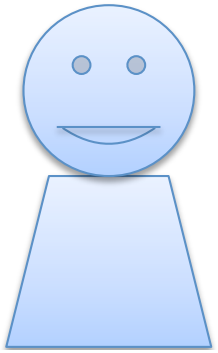
その他の部品

【共通の機能】
画面に表示可能

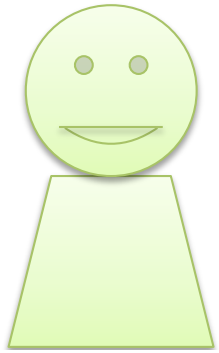
【固有の機能】
押す・テキスト入力
Etc.



土台を作る人

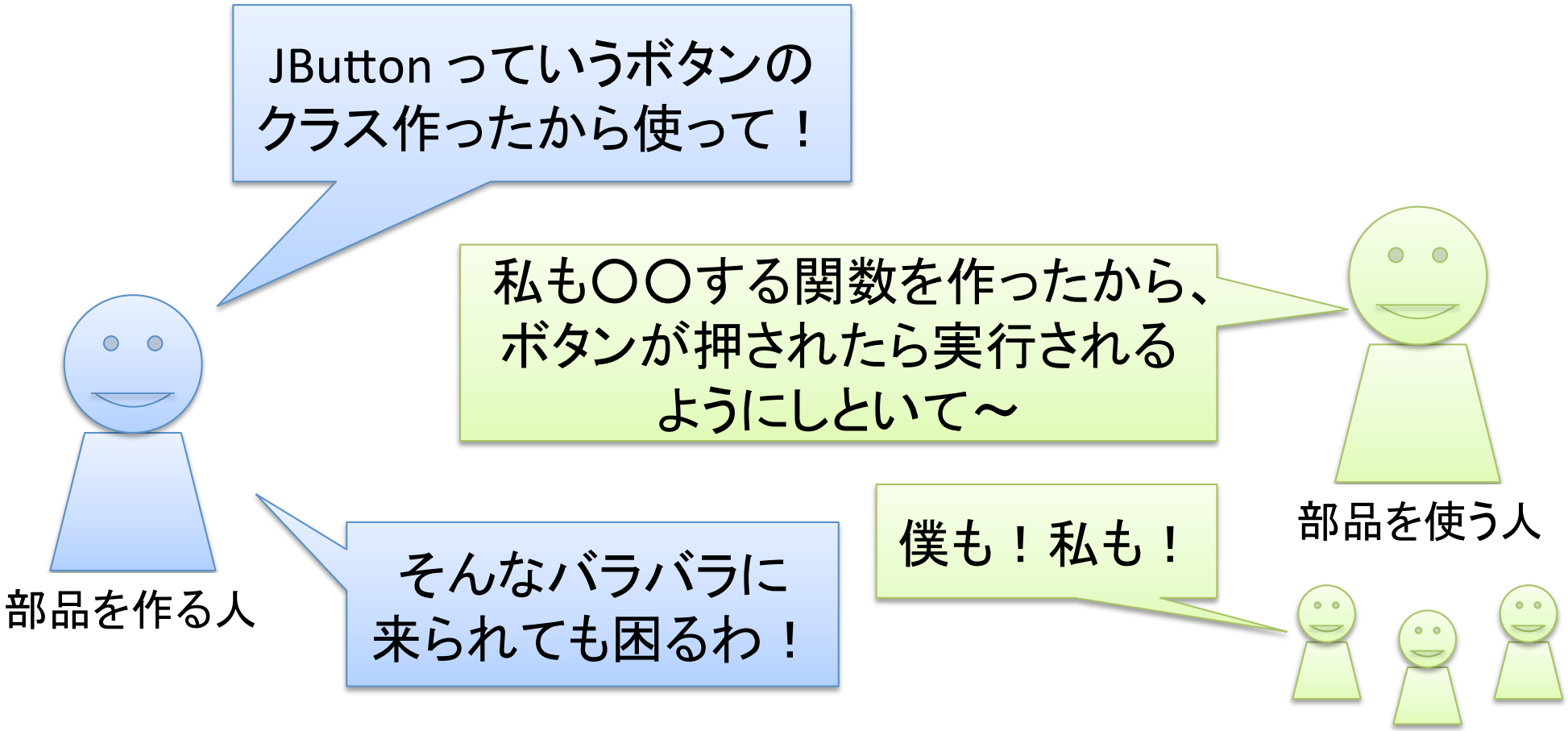


部品を作る人



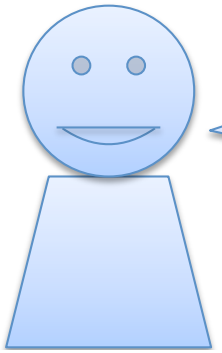
部品を使う人

プログラマのとある一日…



interface

- 必要な関数の一覧を決めておく機能

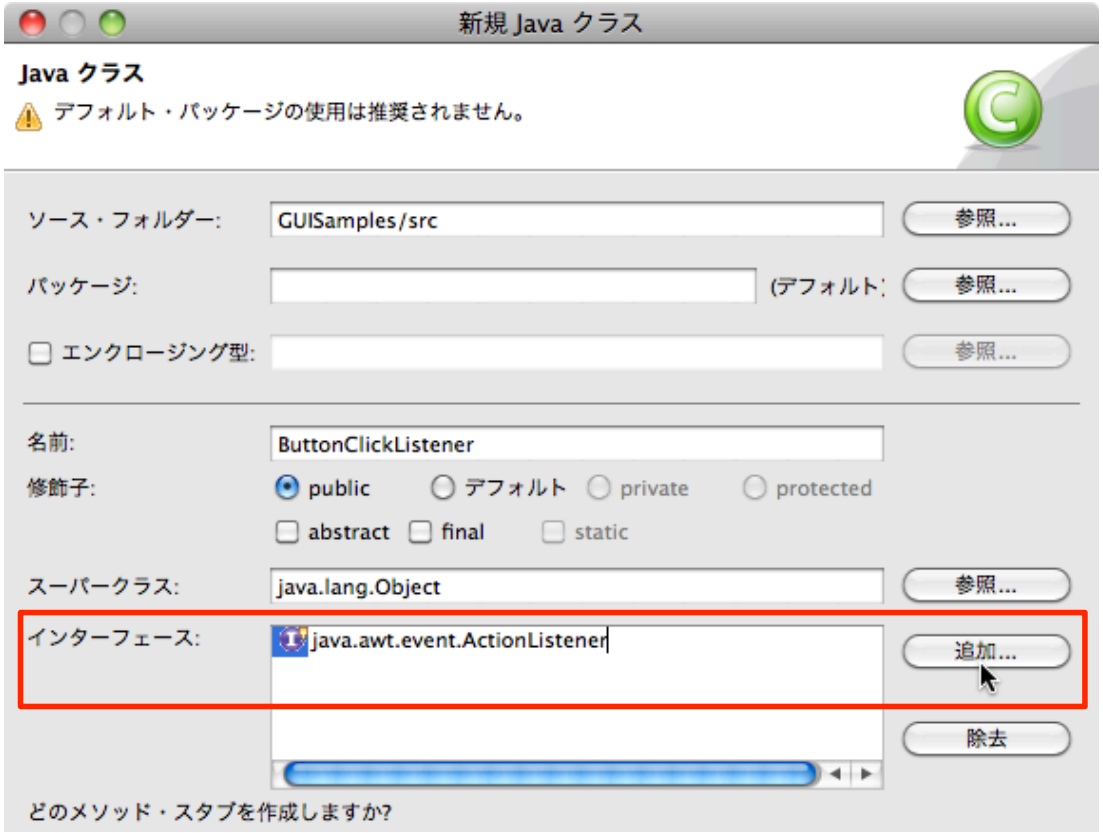


部品を作る人

JButton が押されたときに何かするプログラムが作りたい人は ActionListener という interface に合わせたクラスを作ってね

interface ActionListener を備えたクラスを作る

（先週作ったプログラムのつづきから）



1. 新規＞クラス

2. 追加... から ActionListener を選ぶ

ボタンに登録する

1. actionPerformed 関数の中身を作る

```
public class ButtonClickListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        System.out.println("ボタンが押されたよ");
    }

}
```

2. 今作ったクラスのオブジェクトを登録

```
ButtonClickListener l = new ButtonClickListener();
button.addActionListener(l);
```

ここで1度実行してみよう

部品同士を連携させよう

```
public class ButtonClickListener implements ActionListener {

    private JTextField tf;
    private JTextArea ta;

    public ButtonClickListener(JTextField tf, JTextArea ta){
        this.tf = tf;
        this.ta = ta;
    }

    @Override
    public void actionPerformed(ActionEvent arg0) {
        String s1 = ta.getText();
        String s2 = tf.getText();
        if(s2.length() > 0){
            ta.setText(s1 + "¥n" + s2);
            tf.setText("");
        }
    }

}
```

1. 連携したい部品を覚えておく変数

2. コンストラクタ（new したときに実行される関数）を作る

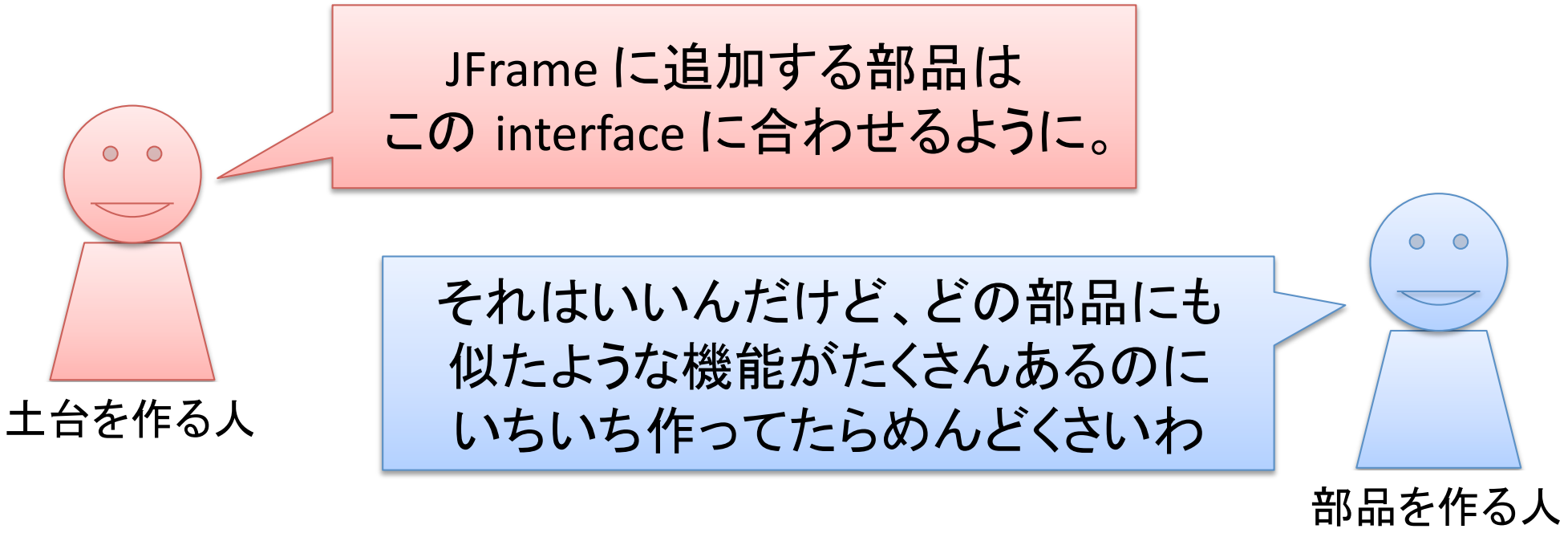
結果の型は不要

3. 覚えておいた変数を使って連携させる

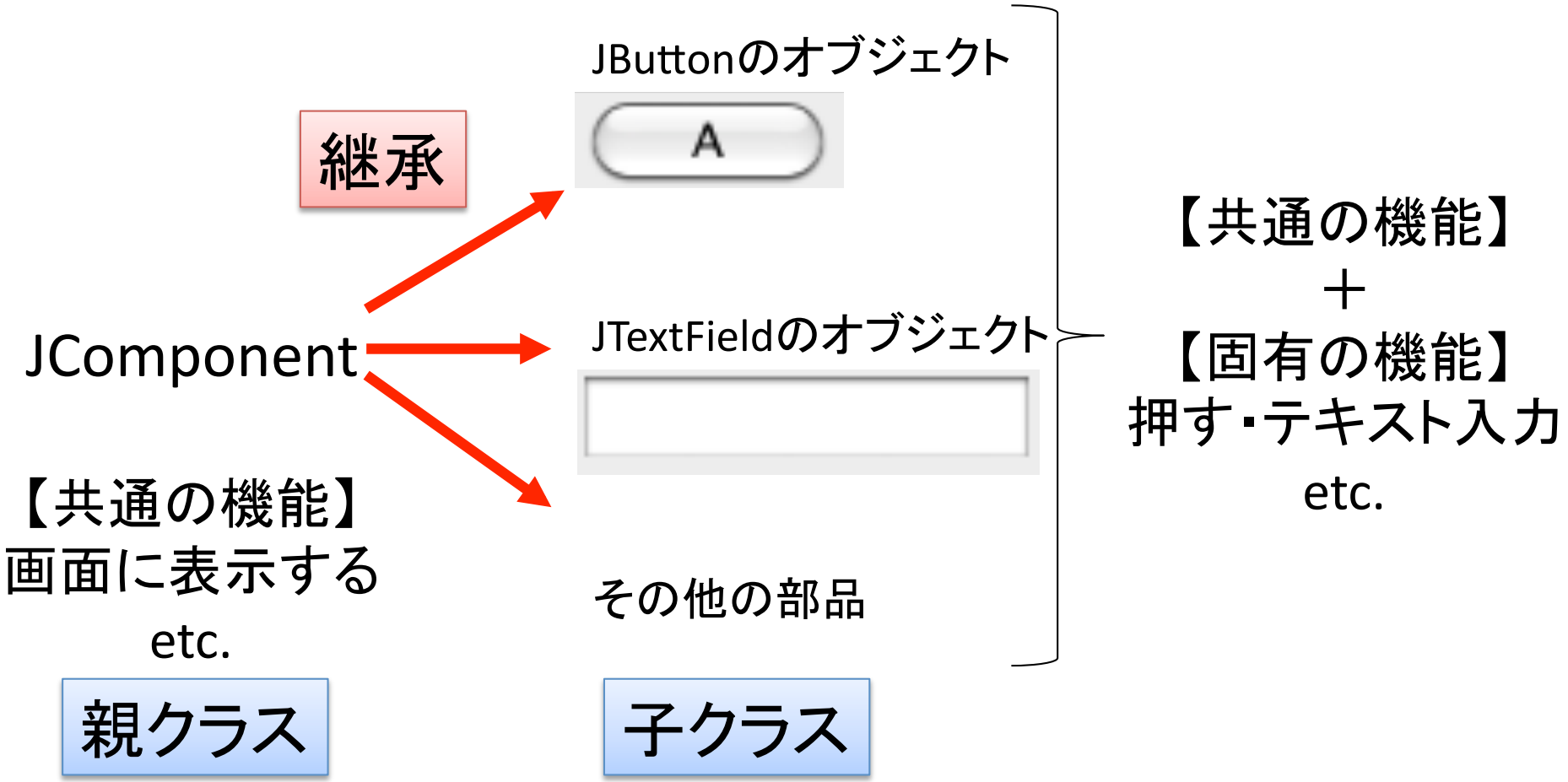
その他 interface を使う局面の例

- ○○が起きた時にすることを登録しておく
  イベント駆動 (event driven)
  - ボタンが押されたとき、クリックされたとき
  - ○○秒経ったとき
- 並び替え
  - 何が先に来るか決める関数を指定する interface
  - たとえば、昇順、降順、アルファベット順など

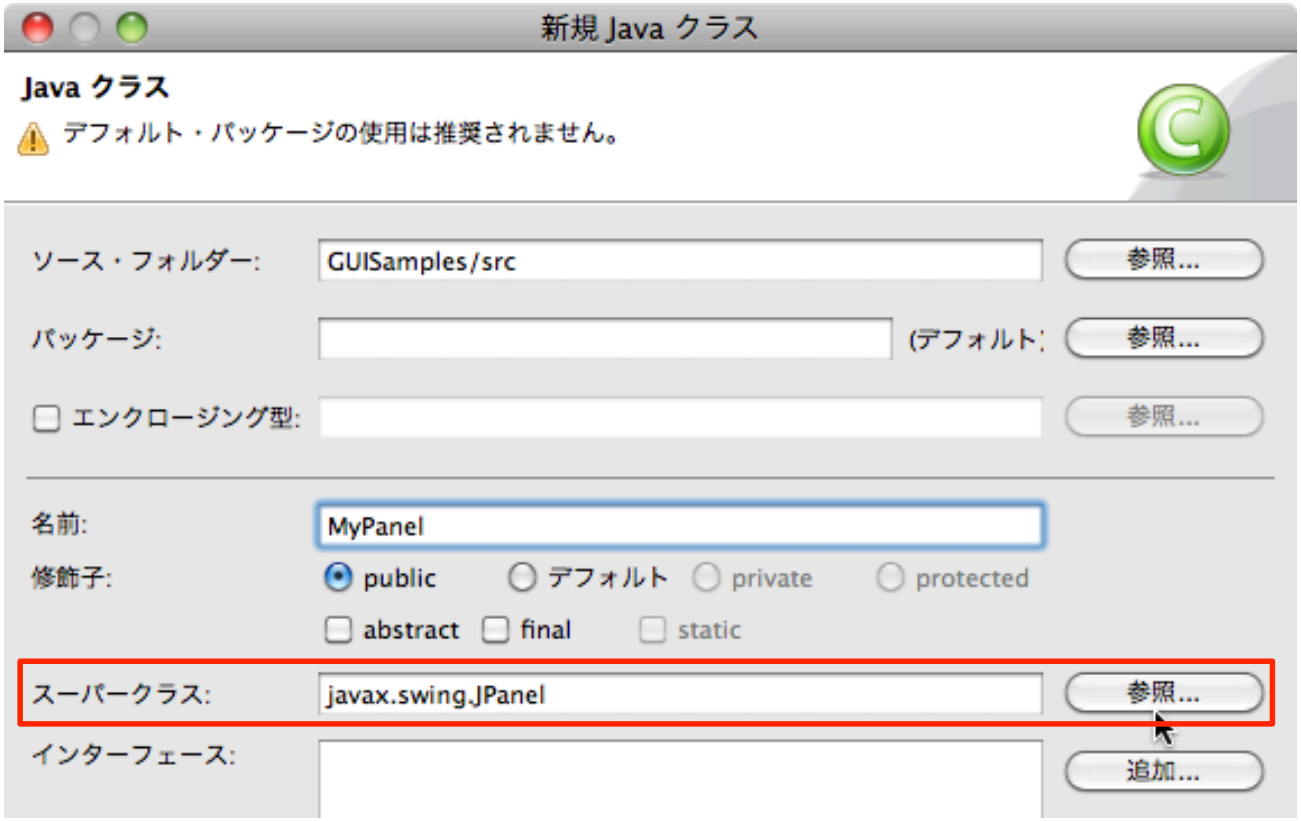
また別の日…


JFrame に追加する部品は
この interface に合わせるように。
土台を作る人
それはいいんだけど、どの部品にも
似たような機能がたくさんあるのに
いちいち作ってたらめんどくさいわ
部品を作る人

クラスの継承


継承
JComponent
【共通の機能】
画面に表示する
etc.
親クラス
JButtonのオブジェクト
A
JTextFieldのオブジェクト
その他の部品
子クラス
【共通の機能】
＋
【固有の機能】
押す・テキスト入力
etc.

既存の部品を継承して
オリジナルの部品を作る

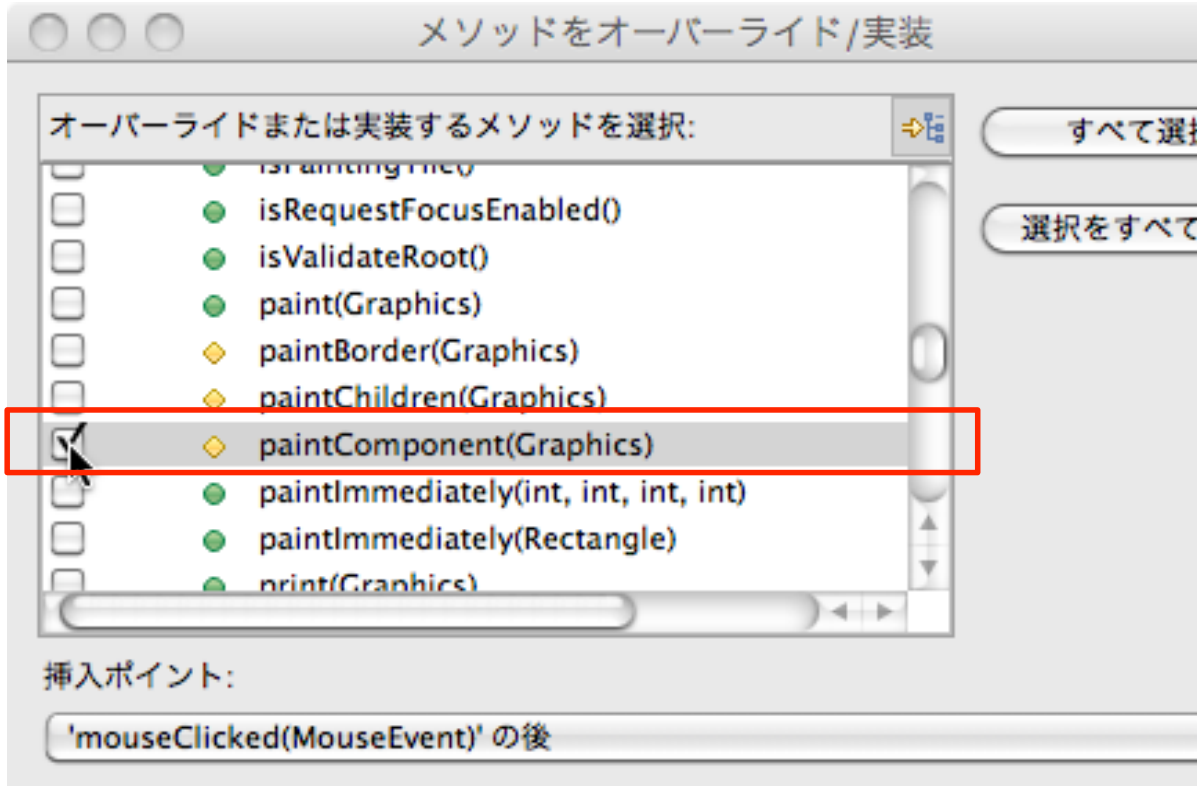


1. 新規＞クラス

2. 参照... から JPanel を選ぶ

メソッドのオーバーライド

- オーバーライド
  - 親クラスの関数を子クラスで書き換えること



1. 右クリック＞ソース
＞メソッドのオーバーライド

2. 出てきたダイアログで
paintComponent に
チェックを入れて OK

メソッドのオーバーライド（つづき）

```
public class MyPanel extends JPanel {

    private boolean fill;

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        if(fill){
            g.fillOval(0, 0, getWidth(), getHeight());
        }
        else{
            g.drawOval(0, 0, getWidth(), getHeight());
        }
    }

}
```

オーバーライドしたという目印

親クラスのpaintComponentをまず実行してあげる

Graphics g を使うとprocessing のように描画ができる

マウス操作に反応するようにする

- 今回は interface MouseListener に対応させる

```
public class MyPanel extends JPanel implements MouseListener {

    private boolean fill;

    public MyPanel() {
        this.addMouseListener(this);
    }
```

```
@Override
public void mouseClicked(MouseEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ
    fill = !fill;
    repaint();
}

@Override
public void mouseEntered(MouseEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ

}
```

※必ずしも別のクラスを
作る必要はない

（つづきは省略）

最終課題の予告

- Swing API を使った GUI アプリを何か作る
  - 〆切は8月8日（月）
- 来週は「何か」のネタになりそうな
  サンプルプログラムをいくつか紹介します
  - 自分で作りたいものがある人はそれでOK