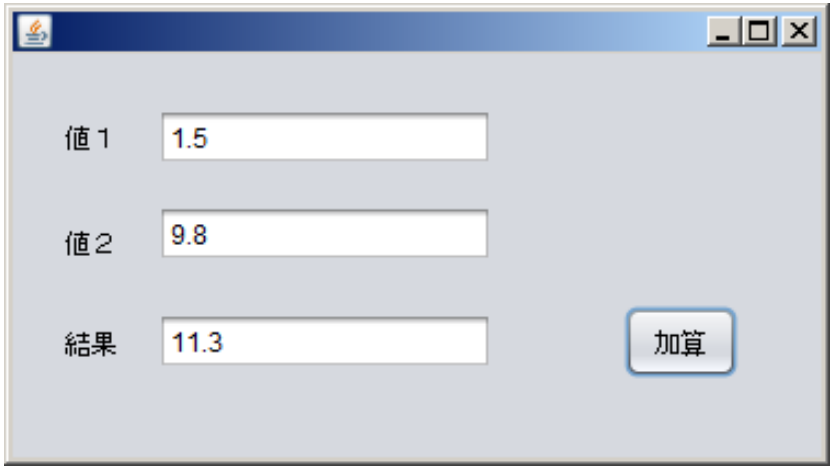
# 調査・製作ワークショップ

オブジェクト指向プログラム

# GUIを使用したJavaプログラミング

- 30分でできる計算プログラム




GUIビルトの概要

https://netbeans.org/kb/docs/java/gui-functionality_ja.html

# 使用する技術

- オブジェクト指向プログラミング
  - クラス
- イベント処理
- 例外処理

# イベント処理

- 昔々

```
while(true) {
    if(ボタンクリック) {
        クリック処理
    }
```

- イベント駆動

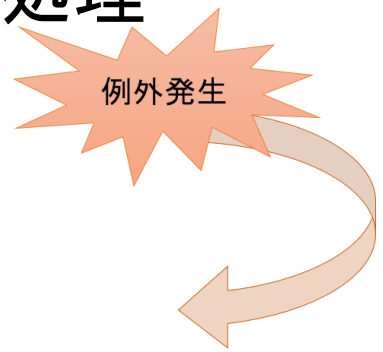
1. イベント処理関数の記述
2. 関数の登録

# 例外処理

- 文字列から数値への変換

“12.5” → 12.5　　“Hello” → ? (例外)

```
try {
    例外が発生する可能性がある処理
}
catch ( 例外変数1 ) {
    例外処理1
}
catch ( 例外処理2 ) {
}
```



# オブジェクト指向プログラミング

- オブジェクト
  - コンピュータ・メモリ上のデータと関数（手続き）の集まり
- オブジェクトを物のようにみてプログラミング
- オブジェクトにメッセージを送ると反応
  - 内部の状態（変数の値）が変わる
  - 返事が返ってくる

# オブジェクト指向

a

| 製造番号 | 20160601 |
|---|---|
| 位置 | (4.5, 2.0) |
| 向き | 90° |
| エネルギー | 140 |
| … | … |

メッセージ

```
a = new robot();
b = new robot();
  …
a.チャージ(85);
 …
b.チャージ(100);
```

b

| **製造番号** | **20131203** |
|---|---|
| 位置 | (8.0, -1.0) |
| 向き | 45° |
| エネルギー | 80 |
| … | … |

```
class robot {
   int     製造番号;
   twoD 位置;
   float   向き;
    …
    void チャージ(int x) { … }
```

# カプセル化

```
class myClass {
    public float x;
    private float y;
    …
}
```

```
a = new myClass();
a.x = 4.0;
a.y = 1.5;
```

*エラー!*

private -- 他のクラスからのアクセスは不可

# 継承(インヘリタンス)

ロボット・クラス　　魚ロボット・クラス

JTextField クラス　　MyTextField クラス

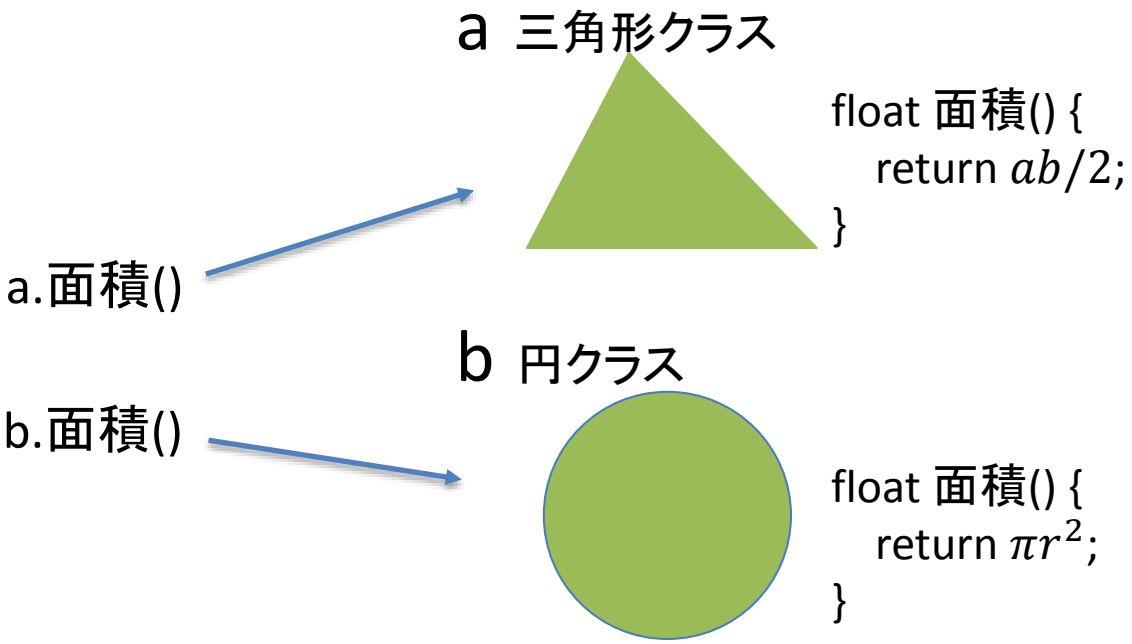
jTextFieldクラスの属性とメソッド
＋ 数値の読み取りと表示

# 多相性(ポリモフィズム)

- 多様なデータ(その組み合わせ)に対して1つの共通した関数呼び出しが可能

println(**3.5 + 2.0**);

println(**“Hello!”**);



# ソース・コード

```
1
2    import static javax.swing.JOptionPane.showMessageDIalog;
3
4   /*
5    * To change this license header, choose License Headers in Project Prop…
6    * To change this template file, choose Tools | Templates
6     * and open the template in the editor.
7    */

8    /*
9     *
10    *
11    * @author cxxxxxx
12    */
```

import文

# public class Calc　クラスの定義

```
14  public class Calc extends javax.swing.JFrame {
15      /**
16       * Creates new form Calc
17       */
18      public Calc() {
19          initComponents();
20      }
21  …


161 }
```

# public Calc()　コンストラクタ

```
18    public Calc()  {
19        initComponents();
20    }
21    …
```

18行～20行　コンストラクタ（初期化作業）

# private void initComponents()

```
 30  private void initComponents() {
 31      jLabel1 = new javax.swing.JLabel();
 32      jLabel2 = new javax.swing.JLabel();
             …
103  }
```

30行～103行　initConponents関数

部品を初期化。部品の配置

# private void initComponents() 続き

```
49    jButton1.addActionListener(new
50        java.awt.event.ActionListener() {
51            public void actionPerformed(
52                ActionEvent evt) {
53                    jButton1ActionPerformed(evt);
54                }
55        });
```

イベント処理関数を登録

# イベント処理関数

```
105   private void jButton1ActionPerformed(...) {
106       double v1, v2;
107       try {
108           v1 = myTextField1.getValue();
109           v2 = myTextField2.getVaue();
110            myTextField2.setText(v1+v2);
111       }
112       catch {
113           showMessageDialog(null,”数値ではない”);
114   }
```

例外発生

イベント処理関数

# ソース･コード(全体)

Import 文

public class Calc extends Jframe {　　クラス定義

public calc()　　コンストラクタ

private initComponents()
部品の初期化など

private void jButton1ActionPerformed
イベント処理

public static void main

変数の宣言