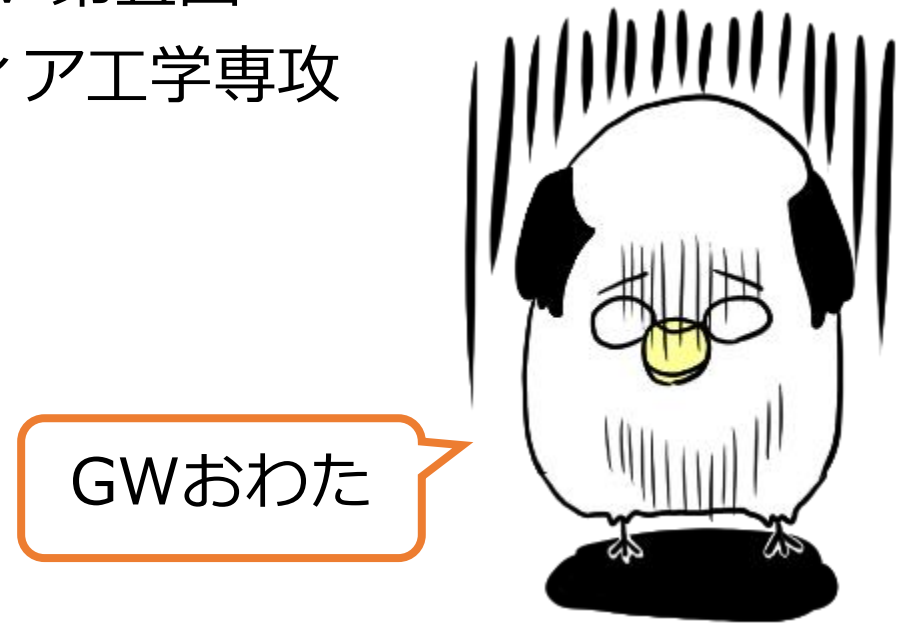
GUI 1

フレーム　レイアウト　イベント

2018/5/8(火)　プログラミングIV 第五回

福井大学　工学研究科　情報・メディア工学専攻

長谷川達人



演習問題の解説

- HP上にアップロードした.
- 難所？はコメントしてあるので詳細は各自確認すること.
- 質問回答後，前回の課題の解説を軽く実施する.

- 学籍番号を入力すると，成績を表示する仕組みを開発してみましたので使ってみてください.

演習問題の配点

- どこが間違えたのか知りたいという感想が多く見られた.
  - 間違い箇所を表示できるよう拡張予定だが忙しくて開発できていない.（もう少し待たれし）
- 自動採点に予想外の回答があったため，一部採点し直しを行った.
- 演習問題得点アップのコツ
  - メソッドのpublicを忘れない
  - フィールド修飾子(privateやprotected)を忘れない
  - 例示された通りにコンソール出力する

ポリモーフィズム

復習

ザックリ捉えると**Monster**だよね，**Slimi**って

```
Monster m = new Slimi();
```

箱の型　　**中身の型**

extendsかimplementsの関係にある場合に限る

箱の型→どのメソッドが使えるのかを決める
中身の型→メソッドが呼ばれた際の動作を決める

キャストで強引に箱の型を変更することができる

ポリモーフィズムができることで，インスタンス配列で
上手く楽をしたり，汎用的な引数のメソッドを実現できる

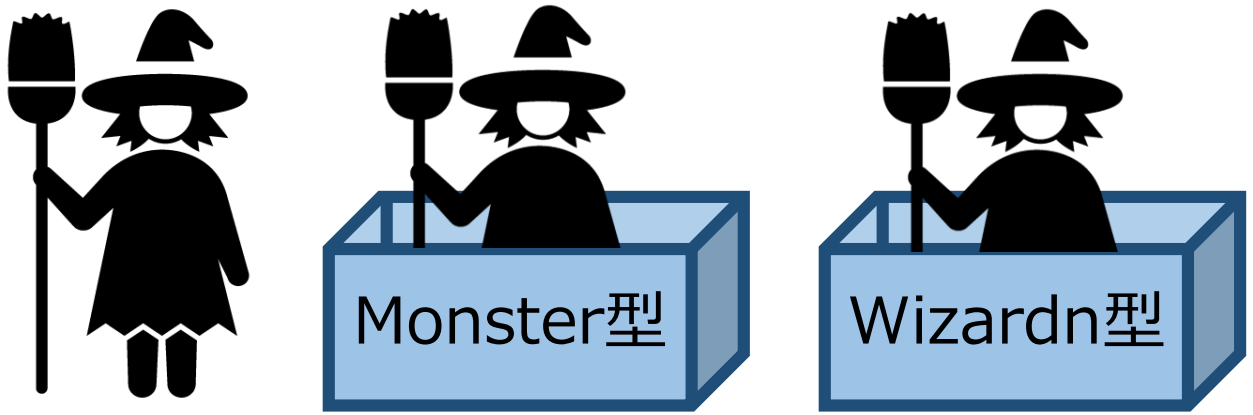
ポリモーフィズム

練習問題３：どうやって逃げるのか

```java
public class Monster {

  public void run(){
    System.out.println("逃げ出した");
  }



}
```

```java
public class Wizardn extends Monster{
  @Override
  public void run(){
    System.out.println("箒で飛んで逃げた");
  }
  public void magic(){
    System.out.println(“魔法を使った");
  }
}
```

```java
public class Main {
  public static void main(String[] args){
    Wizardn w =  new Wizardn();
    Monster m =  new Wizardn();
    w.run();     // 1.
    m.run();     // 2.
    w.magic();   // 3.
    m.magic();   // 4.
  }
}
```



質問への回答

- implementsやextendsで繋がっていれば，間に何個継承があってもポリモーフィズムできるのか.
  - できる．ただし叔父に当たる箱の型で甥に当たる中身の型を受け取ることはできない.
- instanceofの使いみちはif文で使うだけか.
  - 基本的にはif文でキャスト可能かを検査するために使われる.
- instanceof以外に安全確認を行う術はあるのか.
  - ない．ClassCastExceptionを例外キャッチして対応することはできる（例外の章で説明）.

質問への回答

- いまそのクラスがどの箱に入っているか確認するにはどうすればよいか.
  - 箱の型を宣言したソースコードを読む.

正確には**インタフェース**

- **インターフェイス**名の型でオブジェクトを生成するのはどのようなときに行うのか？
  - コールバックを送信する側の実装時などなど.
- 型を変えずにoverride前のメソッドを使えないか.
  - override前のメソッドは基本的には使えない.
  - 使うためにはoverride時にsuper.method()とする.

質問への回答

- キャストのメリットは？あまりしない方が良い？
  - 今後，超汎用的なインスタンス等が登場する．実際のメソッドを利用するためにはキャストして使用しなければならない．必要な時に適切にキャストし，エラー落ちしないようにすれば全然使って良い．
- どうしたらプログラミングができるようになるか．
  - 楽しむ！！授業のプログラミングは言語を習得するための「課題」を解くものだが，プログラミングは本来作りたいものを開発するツールである．作りたいものを見つけ，完成させる楽しみを知ってほしい．

質問への回答

- ダウンキャストの逆はできるのか.
  - ダウンキャストの逆はポリモーフィズム.
- 実行の際に不具合がなければ，ダウンキャストはそのまま実行できるのか，実行に失敗するのか。
  - 違う型（SlimiをMonsterの箱に入れてMaouにキャストした時等）の場合は例外なくエラー落ちする.
- ダウンキャストするくらいなら初めから箱と中身を統一すればいいのになと思った.
  - メソッドで送られてきた型をダウンキャストしたい時などがある.

質問への回答

- 自動採点を学生側からも使えるようにしてほしい.
  - してあげたいが，開発している余裕がない．．．
- 課題提出時の文字コードは？
  - Shift-JIS（eclipseの初期設定）
- 自由課題はAndroidStudioで作ってもよいのか.
  - Javaなら何でも良い.
- 自由課題は全て自作しなければならないのか.
  - 画像等フリー素材使用可．一部であれば他者のコード流用可．ただし，自身の貢献を明示すること.

前回の演習課題の解説

- これまでに比べて，前回の平均点が低かった
- ポリモーフィズムの具体例やメリットに関する質問が多かった
- ポリモーフィズムの動作がよくわからなかった

などなどの質問が多かったので解説を行う.

本講義の概要

前半		後半	
第1回	基本文法の復習	第9回	標準ライブラリ
第2回	クラス～カプセル化の復習	第10回	ファイル入出力
第3回	抽象クラス，インタフェース	第11回	デバッグ，インポート，高速化
第4回	ポリモーフィズム	第12回	オブジェクト指向
第5回	**GUI 1 ~~: Canvas~~**	第13回	自由開発演習 1
第6回	GUI 2 ~~: Layout，イベントリスナ~~	第14回	自由開発演習 2
第7回	スレッド，例外処理	第15回	自由開発演習発表会
第8回	ジェネリクス，コレクション	第16回	期末試験

何を開発するか，少しずつ考えておくこと

本日の目標

概要

はじめてのGUIプログラミングの基本について学習する.

目標

JavaのSwingを用いたプログラミングに慣れる.

本日の提出課題

講義パート

課題を意識しながら
講義を聞くと良い.

課題 1

本日の授業を聞いて,
よくわかったと思う内容を2点簡潔に述べよ.

課題 2

本日の授業を聞いて,
質問事項または**気になった点**を1点以上簡潔に述べよ.

課題 3

感想（あれば）

CUIとGUI

Character-based User Interface & Graphical User Interface

- **CUI**: Character-based User Interface
  - C言語～これまで使ってきたコンソールと文字列を用いてやり取りを行うユーザインタフェース
- **GUI**: Graphical User Interface
  - Desktopのアイコンやフォルダ選択等のように，グラフィカルな表示とマウス操作等の操作でやり取りを行うユーザインタフェース

JavaにおけるGUI

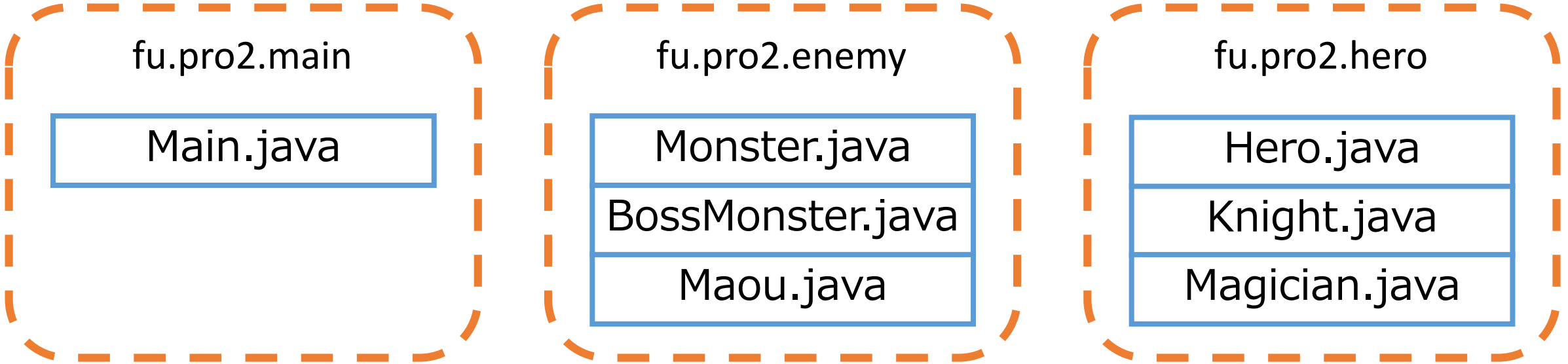
JavaではGUIを実装するためのAPIが準備されている.

- AWT
  - 初期からあるライブラリだがOSにより外観が異なることがある.
- **Swing**
  - AWTの欠点を取り除き，かつ細かい設定を実現したライブラリである．AWTの進化版のようなもの．AWTより多少重い.
- Java FX
  - AWTやSwingより更に新しいライブラリである.
- Android

時代の流れに従って使うライブラリが変わり，そのたびに書き方が変わる.
今回はそのうちひとつを体験しておくことで，今後新しいライブラリを学ぶ際の基礎となってくれればと思う.

前準備
パッケージとimport

パッケージ：複数のクラスファイルのグループ
→フォルダみたいなもの



パッケージ名はxxx.yyy.packagenameとすることが
多く，xxxやyyyには所属やアプリ名が入ることが多い.

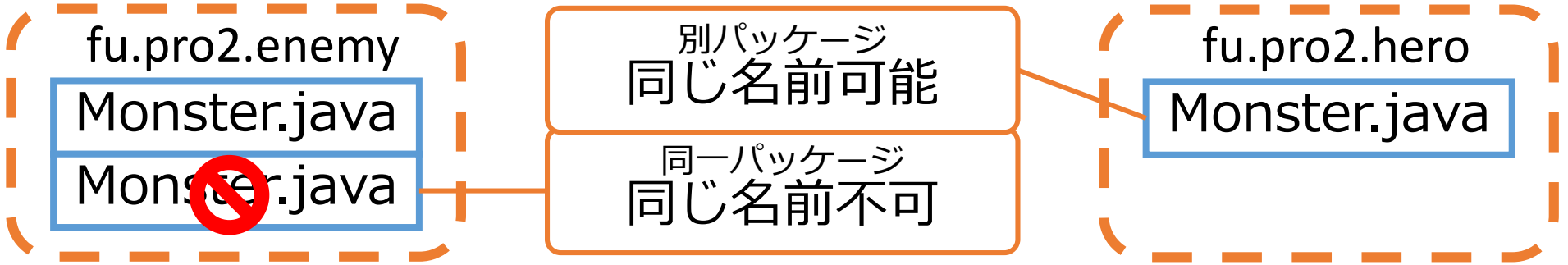
前準備
パッケージとimport

パッケージ化すると，他パッケージのクラスに
容易にアクセスできなくなる.

```
Monster m = new Monster(100, "スライミα", 5);
  ↓  このように書かなければ，別パッケージのクラスが使えない.
fu.pro2.enemy.Monster m = new fu.pro2.enemy.Monster(100, "スライミα", 5);
```

前準備
パッケージとimport

- 関係のないクラスに容易にアクセスできなくなる.
  →影響範囲の切り分けができる.
  →予期しないクラスの使用を防止できる.
- クラスファイルの分類や整理ができる.
  →パッケージの形に添ってフォルダ分けされる.
- クラス名の衝突を防ぐことができる.
  →パッケージが異なれば，同じクラス名が使用できる.



前準備
パッケージとimport

- 自分が所属するパッケージは以下のように記述する.

```
package [所属したいパッケージ名];
```

- クラス使用時に毎回長いパッケージ名を書くのは面倒なので，import文がある．（C言語でいう#include）

```
import [パッケージ名].[クラス名];
```

```
package fu.pro2.main;
import fu.pro2.enemy.Monster;

public class Main {
	public static void main(String args[]){
		Monster m = new Monster(100, "スライミα", 20);
	}
}
```

Main.javaはfu.pro2.mainパッケージに所属

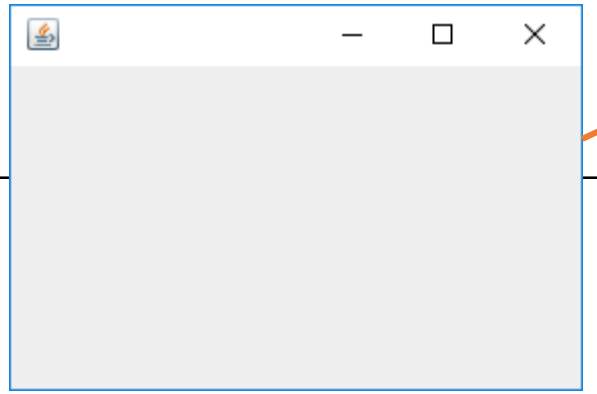
fu.pro2.enemyパッケージに所属のMonsterをimportし，簡単に使用できるようにした.

GUIプログラミング

Swingを使ってみよう

- 次のプログラムをMain.javaに入力し実行してみよう.

```
import javax.swing.JFrame;

public class Main {
  public static void main(String[] args){
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 200);
    frame.setVisible(true);
  }
}
```

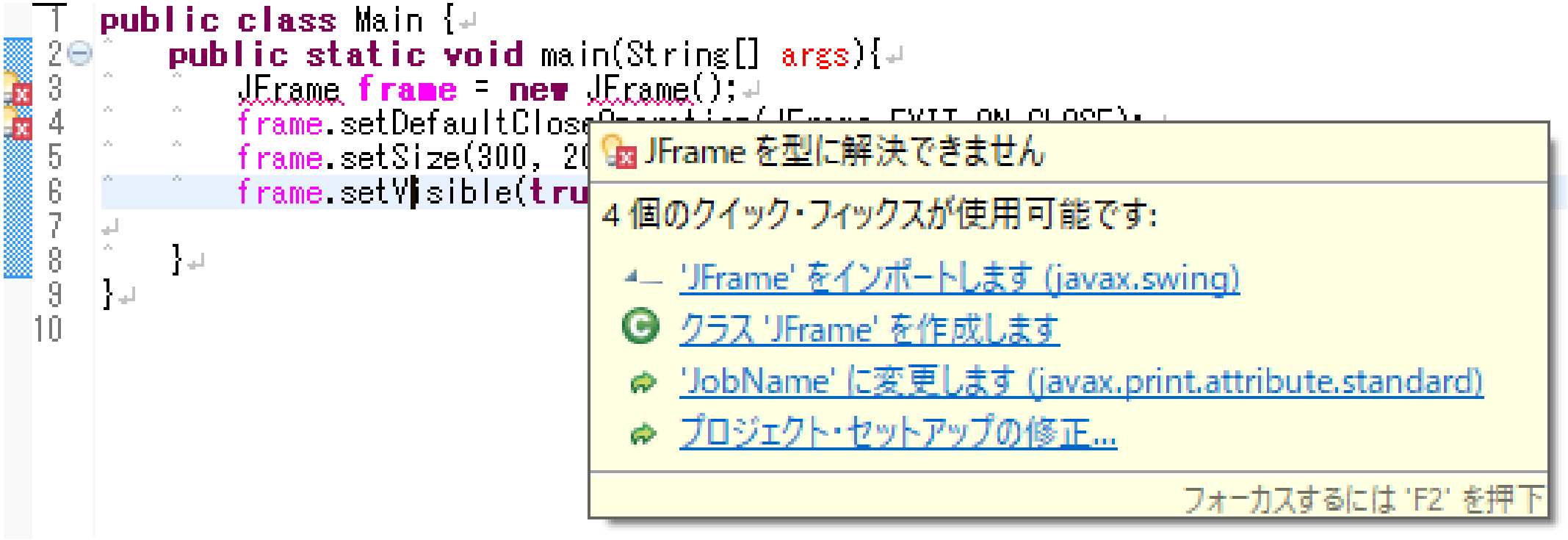


これがでる

GUIプログラミング

Swingを使ってみよう

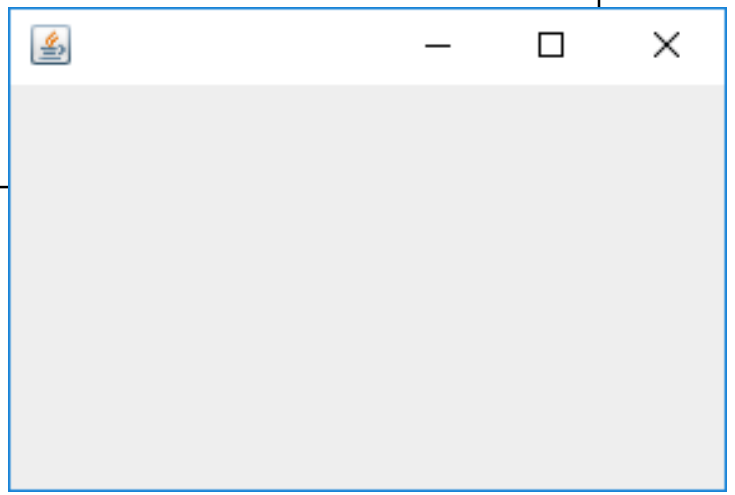
- import文を書かなかった場合以下のようなエラーが出るので，「JFrameをインポートします（javax.swing）」を選ぶと，import文を自動で挿入してくれる（Eclipseの機能）．



GUIプログラミング

Swingを使ってみよう

```
// JFrame型のインスタンスを生成する
JFrame frame = new JFrame();

// このフレームの×ボタンを押したときの動作を設定する
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// このフレームのサイズ（Width[px], Height[px]）を設定する
frame.setSize(300, 200);

// このフレームの表示をtrueにする
frame.setVisible(true);
```



GUIプログラミング

Swingを使ってみよう

JavaではSwingライブラリとしてGUIを簡単に実装する仕組みが準備されている.

→JFrameをインスタンス化して各種設定等を行うだけで簡単にWindowを表示できる.

→JFrameを継承することで，独自のWindowを開発することができる.

このように，様々なプログラムを容易に実装するためのライブラリが豊富＋継承でそれを拡充できるのはJavaの醍醐味の一つである.

GUIプログラミング

練習問題１：Swingを使ってみよう

JFrameを継承したクラスMyFrameを開発せよ.

先のコードではmain()メソッドで各種設定処理を実装していた．しかし，main()で細かい処理を毎回書くのは面倒なので，コンストラクタに内蔵させ，main()では

```
MyFrame frame = new MyFrame();
```

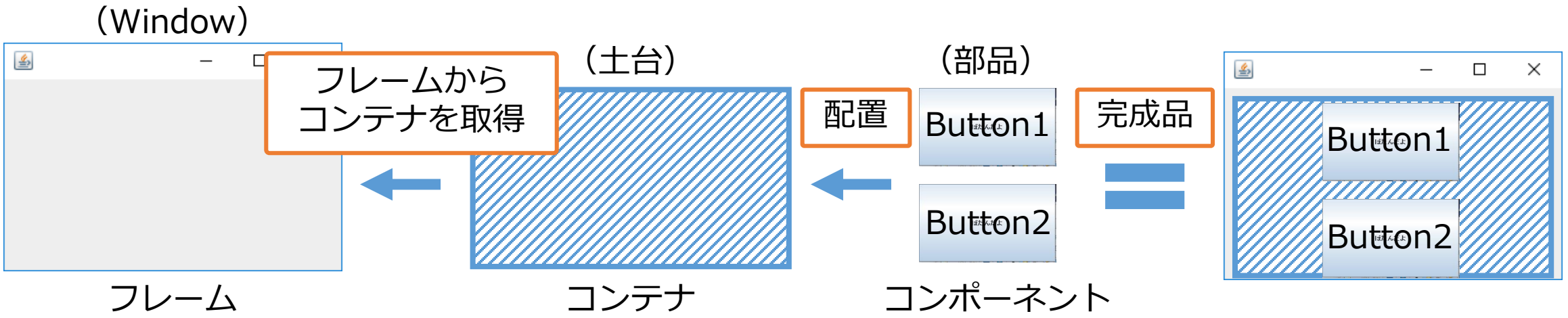
と書くだけで表示までできるようにせよ.

GUIプログラミング

コンポーネントを追加しよう

ボタンやチェックボックスのようなGUI部品のことを
コンポーネントと呼ぶ.

JFrameで生成したWindowのことをフレームと呼び,
フレームの中のコンテナという領域にコンポーネントを
配置していく.



GUIプログラミング

コンポーネントを追加しよう

例えばボタンを追加する場合，以下の手順となる.

1. フレームからコンテナを取得する.
2. ボタンインスタンスを生成する.
3. ボタンをコンテナに配置する.

これを一行で書くと次のようになる.

```
frame.getContentPane().add(new JButton("ぼたんだよ"));
```

1.フレームからコンテナを取得する　2.ボタンインスタンスを生成する

3.「コンテナ.add(ボタンインスタンス)」でボタンをコンテナに配置する

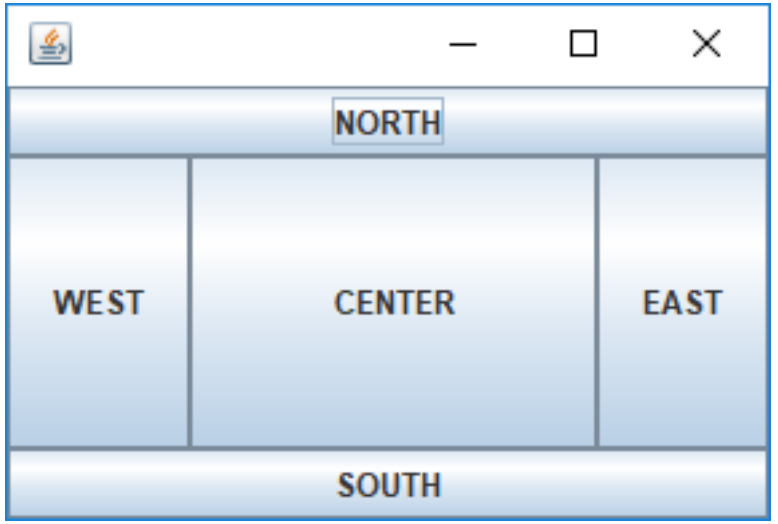
```
import javax.swing.JButton;
```
が必要だがeclipseが自動補完してくれるので以降省略する.

実行してみよう

GUIプログラミング
レイアウト

コンポーネントを追加する際に，レイアウトを指定することが可能である．

例えば，ボーダーレイアウトはレイアウトを東西南北で指定することができる．



第一引数として追記

```
frame.getContentPane().add(BorderLayout.NORTH, new JButton("NORTH"));
frame.getContentPane().add(BorderLayout.WEST, new JButton("WEST"));
frame.getContentPane().add(BorderLayout.EAST, new JButton("EAST"));
frame.getContentPane().add(BorderLayout.SOUTH, new JButton("SOUTH"));
frame.getContentPane().add(BorderLayout.CENTER, new JButton("CENTER"));
```

省略された方角は付近の方角と結合されたものとして表示される．

GUIプログラミング
レイアウト

レイアウトは色々な種類がある.

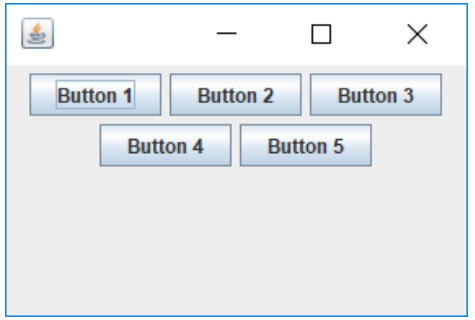
- BorderLayout　　：東西南北に配置する
- FlowLayout　　：左上から順番に配置する
- BoxLayout　　：縦か横一列に順に配置する
- GridLayout　　：格子状の区画で配置する

使用するには，コンテナに対してレイアウトを指定してから配置を行う.

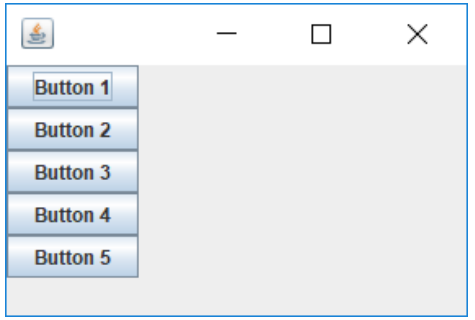
```
frame.getContentPane().setLayout(new FlowLayout());
frame.getContentPane().setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
frame.getContentPane().setLayout(new GridLayout(2,3));
```

GUIプログラミング
レイアウト

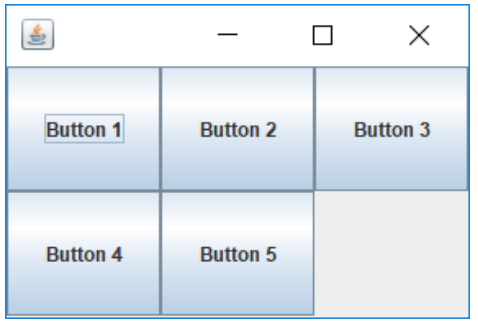
```
frame.getContentPane().setLayout(new FlowLayout());
frame.getContentPane().add(new JButton("Button 1"));
frame.getContentPane().add(new JButton("Button 2"));
frame.getContentPane().add(new JButton("Button 3"));
frame.getContentPane().add(new JButton("Button 4"));
frame.getContentPane().add(new JButton("Button 5"));
```




```
frame.getContentPane().setLayout(
      new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
frame.getContentPane().add(new JButton("Button 1"));
frame.getContentPane().add(new JButton("Button 2"));
frame.getContentPane().add(new JButton("Button 3"));
frame.getContentPane().add(new JButton("Button 4"));
frame.getContentPane().add(new JButton("Button 5"));
```



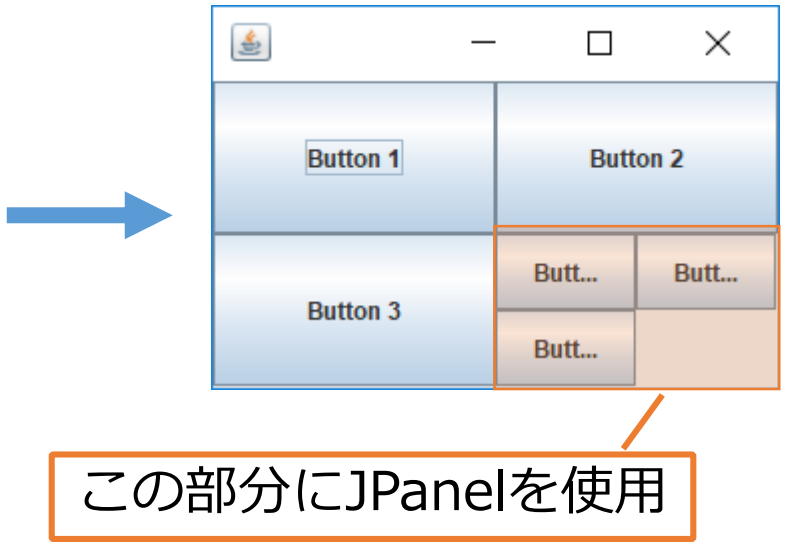

```
frame.getContentPane().setLayout(new GridLayout(2,3));
frame.getContentPane().add(new JButton("Button 1"));
frame.getContentPane().add(new JButton("Button 2"));
frame.getContentPane().add(new JButton("Button 3"));
frame.getContentPane().add(new JButton("Button 4"));
frame.getContentPane().add(new JButton("Button 5"));
```



GUIプログラミング
レイアウト

コンポーネントを乗せることができる（コンテナのような）コンポーネントを**パネル**と呼ぶ.

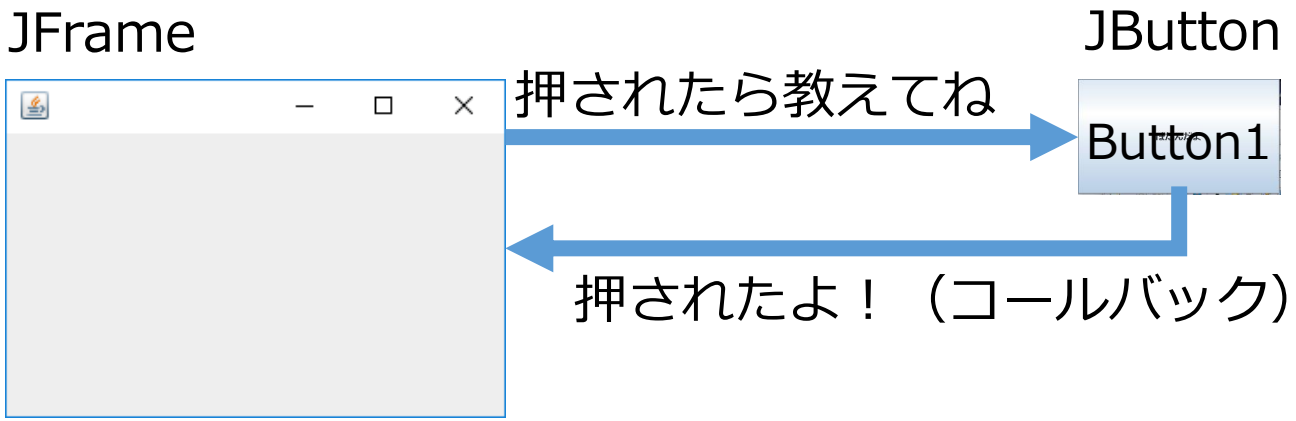
```
frame.getContentPane().setLayout(new GridLayout(2,2));
frame.getContentPane().add(new JButton("Button 1"));
frame.getContentPane().add(new JButton("Button 2"));
frame.getContentPane().add(new JButton("Button 3"));
JPanel panel = new JPanel();
panel.setLayout(new GridLayout(2,2));
panel.add(new JButton("Button 4"));
panel.add(new JButton("Button 5"));
panel.add(new JButton("Button 6"));
frame.getContentPane().add(panel);
```



GUIプログラミング
イベント

ボタンが押された動作など，アプリに対して何か操作が行われた際に発生する通知のことをイベントと呼ぶ.

コンポーネントに対して，**イベントが発生した際に実施する処理を明示しておく**ことで，利用者の操作に応じて処理を変えるといったインタラクションを実装できる.



GUIプログラミング
イベント検出の実装（ボタンクリックの場合）

使うだけなら
超シンプル

1. ActionListenerインタフェースを実装したクラスをインスタンス化する.
(JFrame自身が実装するのが手っ取り早い)

```
MyFrame frame = new MyFrame();
```

2. JButtonのインスタンスに対して「１で生成したインスタンスにコールバックしてください」と登録する.

```
public class MyFrame extends JFrame implements ActionListener{
  public MyFrame(){
    // フレームの設定関連は省略
    JButton button = new JButton("ぼたん");   // ボタンのインスタンス化
    this.getContentPane().add(button);        // コンテナにボタンを配置
    button.addActionListener(this);           // 自身をボタンに登録（ココ！）
  }
  ...(略)
```

このframeとthisは同じものを示す

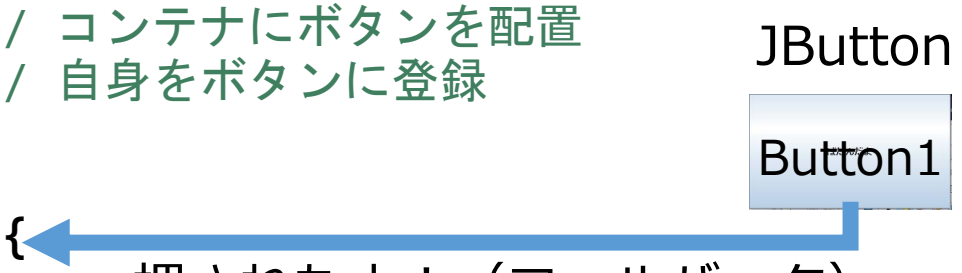
buttonにthisを渡すので，buttonの内部処理でthisのメソッドが利用できる

GUIプログラミング
イベント検出の実装（ボタンクリックの場合）

使うだけなら
超シンプル

3. ActionListenerインタフェースに必要なコールバック先**void actionPerformed(ActionEvent e)**に対して，ボタンクリックイベント発生時にコールバックを行う．

```
public class MyFrame extends JFrame implements ActionListener{
  public MyFrame(){
    // フレームの設定関連は省略
    JButton button = new JButton(“ぼたん”);  // ボタンのインスタンス化
    this.getContentPane().add(button);      // コンテナにボタンを配置
    button.addActionListener(this);         // 自身をボタンに登録
  }
  @Override
  public void actionPerformed(ActionEvent e) {
    System.out.println("ボタンが押されました");
  }
}
```



GUIプログラミング
イベント

複数コンポーネントからイベントを受け取る場合も同じ

```
public class MyFrame extends JFrame implements ActionListener{
  private JButton button1, button2;  // フィールドにしておく
  public MyFrame(){
    // フレームの設定関連（省略する）
    // ボタンを作成する
    this.button1 = new JButton("ぼたん1");
    this.button2 = new JButton("ぼたん2");
    // ボタンをコンテナに配置する
    this.getContentPane().add(this.button1);
    this.getContentPane().add(this.button2);
    // リスナをボタンに登録する
    this.button1.addActionListener(this);
    this.button2.addActionListener(this);
}
  @Override
  public void actionPerformed(ActionEvent e) {
    System.out.println("ボタンが押されました");
  }
}
```

両方とも下のactionPerformed()でイベントを受け取るように，両方ともにthisを登録する

button1とbutton2どっちからきたイベントなの！？

GUIプログラミング
イベント

複数コンポーネントからイベントを受け取る場合も同じ

```
@Override
public void actionPerformed(ActionEvent e) {
  if(e.getSource() == button1){
    System.out.println(“ぼたん1が押されました");
  }else if(e.getSource() == button2){
    System.out.println(“ぼたん2が押されました");
  }
}
```

actionPerformed()の引数はイベントのソースに関する情報を管理している

e.getSource()でイベント発生源を得ることができる

```
@Override
public void actionPerformed(ActionEvent e) {
  System.out.println(((JButton)e.getSource()).getText());
}
```

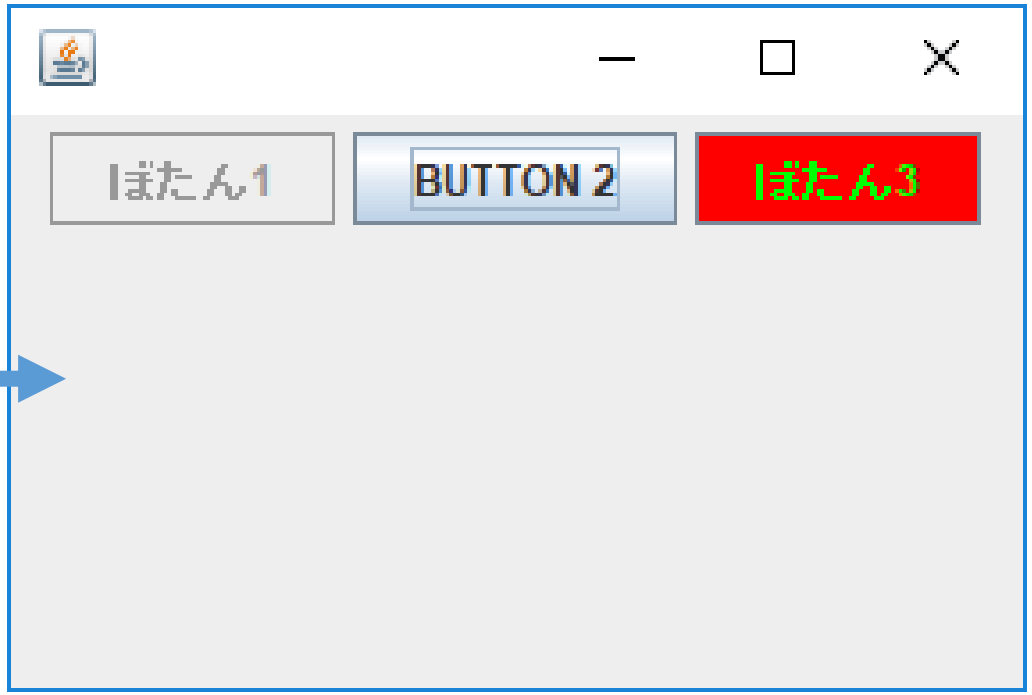
ボタン名を表示したいだけなら，ダウンキャストを用いる手法も．．．

GUIプログラミング

ボタンコンポーネントのメソッド

コンポーネントも様々なメソッドを持つ．ボタンの例を以下に示す．これらのメソッドはコンポーネント共通で，以降説明するチェックボックス等に対しても使える．

```
// ボタンが押せるかどうかを設定する
button1.setEnabled(false);
// ボタンが押せるかどうかを取得する
button1.isEnabled(); // 戻り値はboolean型
// ボタンのテキストを取得する
button2.getText();
// ボタンのテキストを設定する
button2.setText("BUTTON 2");
// ボタンの前景色を変更する
button3.setForeground(Color.GREEN);
// ボタンの背景色を変更する
button3.setBackground(Color.RED);
```



GUIプログラミング

練習問題２：ボタンのイベントリスナ

練習問題１のMyFrameに対して，次の機能を加えよ.

1. BoxLayout（縦方向に追加）でボタンを３つ追加する.
2. ボタン名は「ボタン１～３」とする.
3. ボタン１を押すと，ボタン１の背景色が青になる.
4. ボタン２を押すとボタン２のボタン名を「ぼたん２」にする.
5. ボタン３を押すとボタン１が押せなくなる.

GUIプログラミング

様々なコンポーネント

GUIプログラミング

様々なコンポーネント

様々なコンポーネントを紹介する前に前準備を行う.

Main.javaのmain()より，MyFrameをインスタンス化してフレームを表示していたが，下記のおまじないを書く.

```
public static void main(String[] args){
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            // MyFrameのインスタンスを生成する
            MyFrame frame = new MyFrame();
        }
    });
}
```

GUIのスレッド処理に関する理由により，main()から描画に関する処理を行う際にはこのようなおまじないを書くとだけ知っておいてくれれば良い.

GUIプログラミング

様々なコンポーネント：チェックボックス

インスタンスの作成

チェックの初期値（省略可）

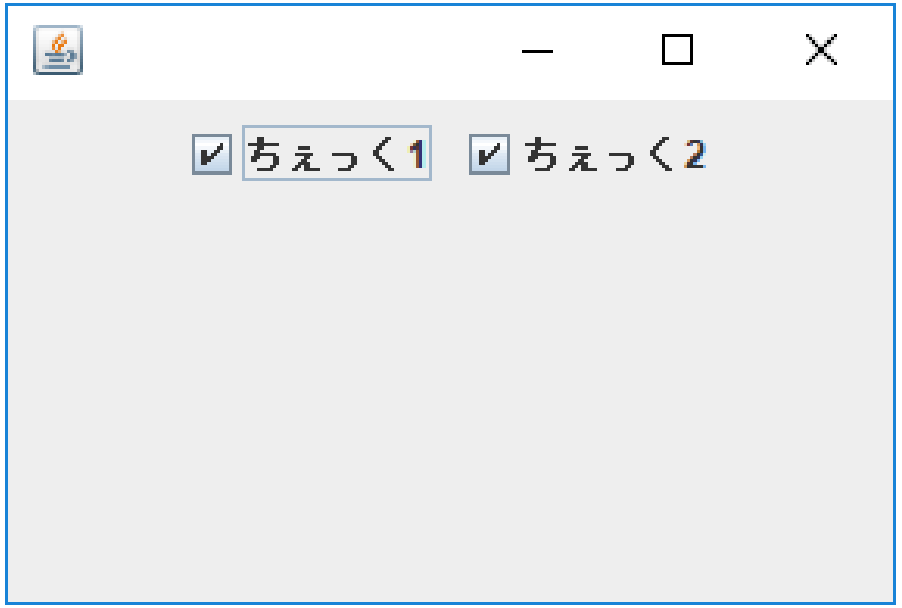
```
JCheckBox check1 = new JCheckBox("ちぇっく1", true);
JCheckBox [変数名] = new JCheckBox([表示テキスト], [初期値]);
```

CheckBox関連メソッド

```
// チェックをつける/はずす
check1.setSelected(true);
// チェックされているのかを返す
check1.isSelected();
```

CheckBox関連イベント

```
// 押された時の動作を定義
check1.addActionListener(this);
```



GUIプログラミング

様々なコンポーネント：ラジオボタン

インスタンスの作成

チェックの初期値（省略可）

```
JRadioButton radio1 = new JRadioButton("らじお1", true);
JRadioButton [変数名] = new JRadioButton([表示テキスト], [初期値]);
```

RadioButton関連メソッド

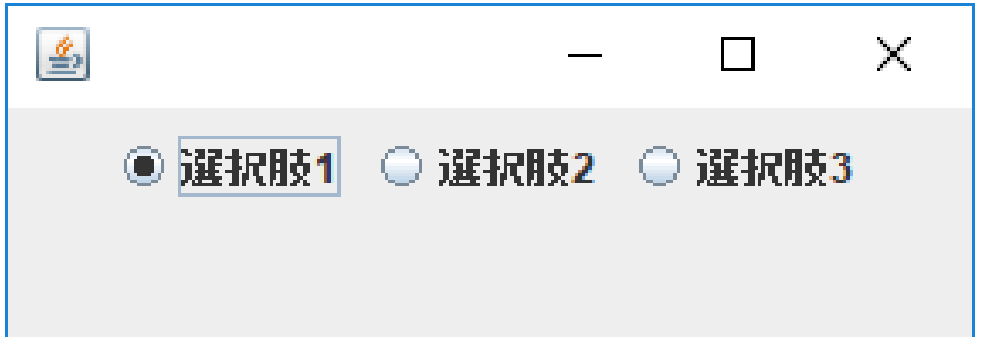
```
// チェックをつける/はずす
radio1.setSelected(true);
// チェックされているのかを返す
radio1.isSelected();
```

```
ButtonGroup bg = new ButtonGroup();
bg.add(radio1);
bg.add(radio2);
bg.add(radio3);
```

グループ化しておくことで，グループ内で1つだけ選択できる

RadioButton関連イベント

```
// 押された時の動作を定義
radio1.addActionListener(this);
```



GUIプログラミング

様々なコンポーネント：ラベル

インスタンスの作成

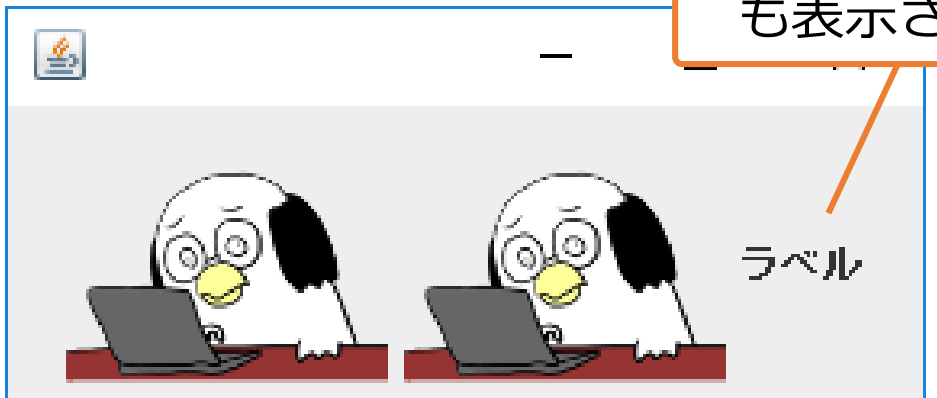
```
JLabel label = new JLabel("ラベル");
JLabel [変数名] = new JLabel([表示テキスト]);
```

Labelを用いた画像の表示

```
// 1つはJLabelインスタンス生成時に画像をセットする方法
JLabel imglabel = new JLabel(new ImageIcon("./image.png"));
// もう1つは既存のJLabelに画像をセットする方法
label.setIcon(new ImageIcon("./image.png"));
```

プロジェクトフォルダ直下に画像を置く場合

後者はテキストも表示される



Label関連イベント

```
// なし
```

GUIプログラミング

様々なコンポーネント：テキストフィールド

インスタンスの作成

引数両方省略可

```
JTextField text = new JTextField("てきすと", 20);
JTextField [変数名] = new JTextField([表示テキスト], [列数]);
```

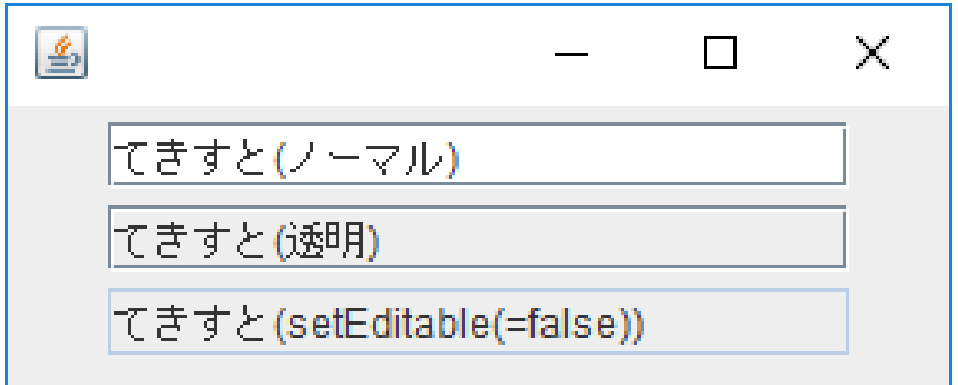
TextField関連メソッド

```
// 背景不透明(true)/透明(false)
text.setOpaque(false);
// 選択されている部分文字列を取得
text.getSelectedText();
```

```
// 2-3文字目のテキストを選択する
text.getSelectedText(2,3);
// 編入可能不可の切り替え
text.setEditable(false);
```

TextField関連イベント

```
// Enterが押された時の動作を定義
text.addActionListener(this);
```



GUIプログラミング

様々なコンポーネント：コンボボックス

インスタンスの作成

```
String[] sarr = {"項目1", "項目2"};
JComboBox combo = new JComboBox(sarr);
JComboBox [変数名] = new JComboBox([表示テキスト配列]);
```

ComboBox関連メソッド

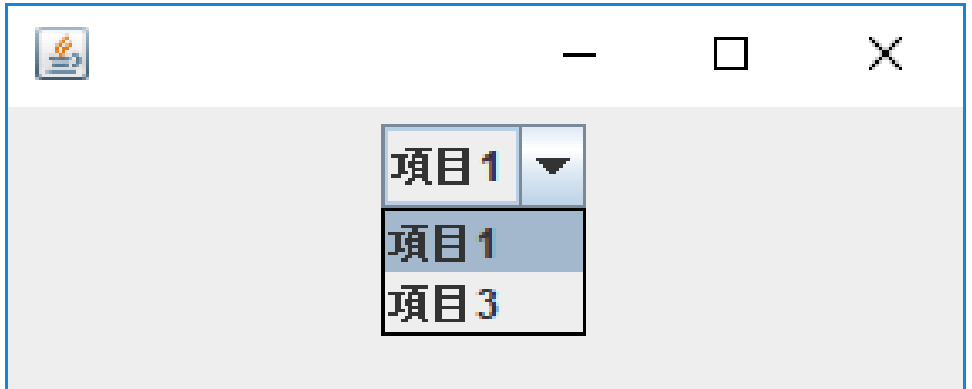
```
// 選択された項目を取得
combo.getSelectedItem();
// 項目を追加
combo.addItem("項目3");
```

```
// 選択できる項目数を取得
combo.getItemCount();
// 2番目の項目の削除
combo.removeItemAt(1);
```

ComboBox関連イベント

```
// 項目が選択された時の動作を定義
text.addItemListener(this);
```

thisはItemListenerを実装しているとする



GUIプログラミング

様々なコンポーネント：メニュー

インスタンスの作成

```
JMenuBar menubar = new JMenuBar();
JMenu menu = new JMenu("ファイル");
JMenuItem menuOpen = new JMenuItem("開く");
JMenuItem menuExit = new JMenuItem("終了");
```

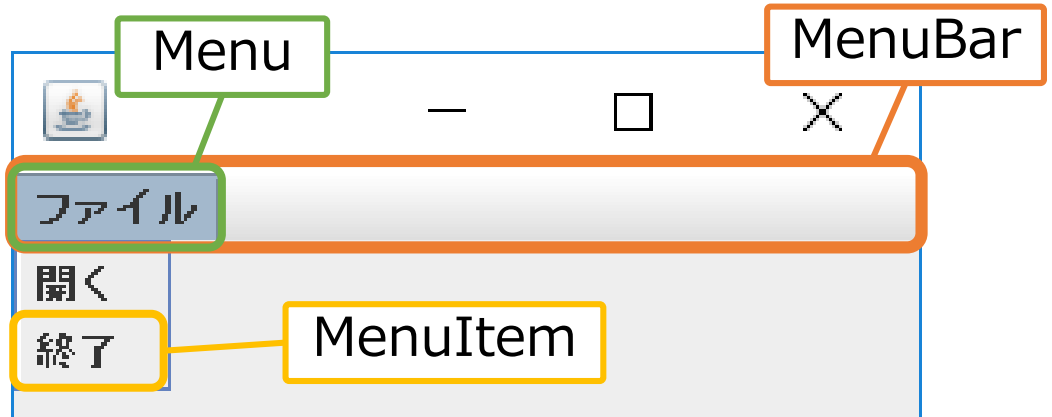
と準備

```
this.setJMenuBar(menubar);
menubar.add(menu);
menu.add(menuOpen);
menu.add(menuExit);
```

メニューはMenuBarにMenuを追加し，MenuにMenuItemを追加する形で構成される．各MenuItemにActionListenerを登録することで，JButtonのようにクリックを検出することができる．

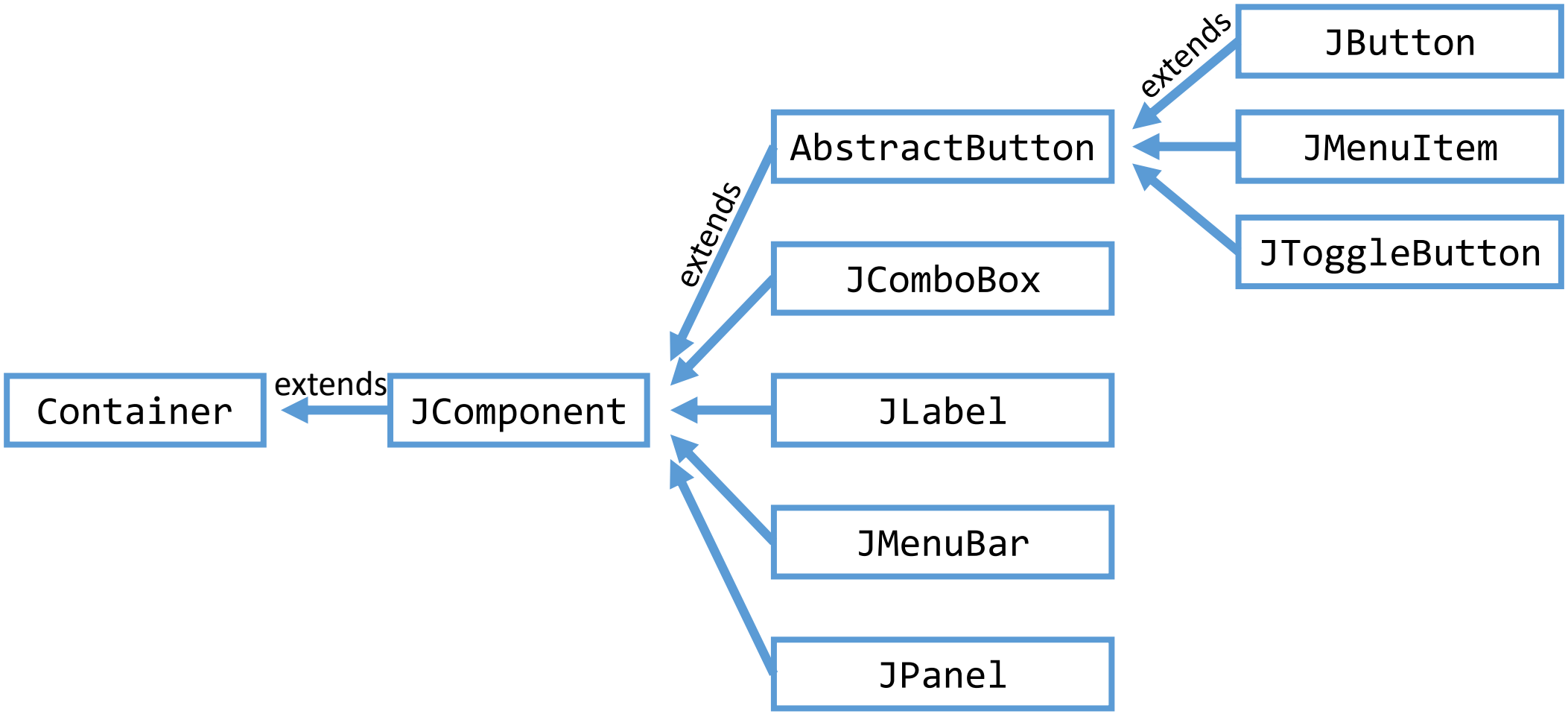
MenuItem関連イベント

```
// 開くと終了が押された時の動作を定義
menuOpen.addActionListener(this);
menuExit.addActionListener(this);
```



GUIプログラミング

参考：GUI関連の継承関係



次週予告

※次週以降も計算機室

前半

図形を描画するようなGUIプログラミングを学ぶ.

後半

講義内容に関するプログラミング演習課題に取り組む.

本日の提出課題

講義パート

課題 1

本日の授業を聞いて，
よくわかったと思う内容を
2点簡潔に述べよ.

課題 2

本日の授業を聞いて，
質問事項または**気になった点**
を1点以上簡潔に述べよ.

課題 3

感想（あれば）

課題 4

なし

演習

- 昼休み，いつものWebページに演習問題をPDFで演習問題をアップロードする．各自実施してプロII同様のWebページから提出すること.
- 質問は３人体制で受け付けるので遠慮なく申し出る．質問の際は，どこまでわかっていて何がわからないのかを申し出ること.
- （ないとは思うが）コピペは発覚次第両成敗する.
  - ✓ {コピペ，カンニング} ∈不正行為
- つまらないミスも今回は問答無用で×とするので，最終チェックを怠らないこと．（昨年は目視で甘めに採点していたが，自動採点を開発している意味がないので．．．）