2018 年 7 月 5 日（木）実施

GUI プログラミング

今回の授業では，Java 言語での GUI（Graphical User Interface）プログラミングの基礎について学習する。

GUI ツールキット

Java 言語では，GUI プログラミング用のツールキットとして，次のものが用意されている。

1) AWT（Abstract Window Toolkit）
2) Swing

AWT は Java 言語の当初から実装されていた標準の GUI ツールキットで，OS のネイティブな GUI コンポーネント（GUI を構成するパーツ，ウィジェット）を利用するため，扱いが容易で動作が速い半面，実行環境による制約がある。

Swing は Java 言語で実装された GUI ツールキットであるので，実行環境には依存しないが，動作が遅くなる。

Eclipse では両者の長所を生かした SWT（Standard Widget Toolkit）が提供されている。これは Java 言語の標準ではないが，Eclipse をインストールしていない環境にも SWT のみを導入することが可能である。

今回は，AWT を用いた GUI プログラミングについて学ぶ。

インタフェース

AWT には，**インタフェース**という，**実装**の無いクラスが多用されている。ここで，**実装とは，枠組みだけが定義されているものに，中身を与えること**である。インタフェースはクラスの階層にとらわれずに実装が可能であるが，記述出来る内容は定数フィールドと**抽象メソッド**に限られる。ここで，**抽象メソッドとは，インタフェースを実装したクラスの中で必ずオーバーライドしなければならないメソッド**である。

インタフェースの定義の一般形は次のようになる。

```
修飾子 interface インタフェース名 {
    型 フィールド名;
    戻り値のデータ型 メソッド名( 引数 );
}
```

インタフェースでは明示しなくても，全てのフィールドは public static final の修飾子を付けた扱いとなる。また，全てのメソッドは public abstract の修飾子を付けた扱いとなり，その処理の定義は記述しないので，ブロックにはならず，セミコロンを必要とする。

なお，インタフェースは直接インスタンス化することが出来ないので，コンストラクタを宣言

することは出来ない。（インタフェースを実装するクラスには，コンストラクタを宣言出来る）

インタフェースを実装するクラスは，implements（実装）により，次の様に定義する。（コンストラクタを用いる場合）

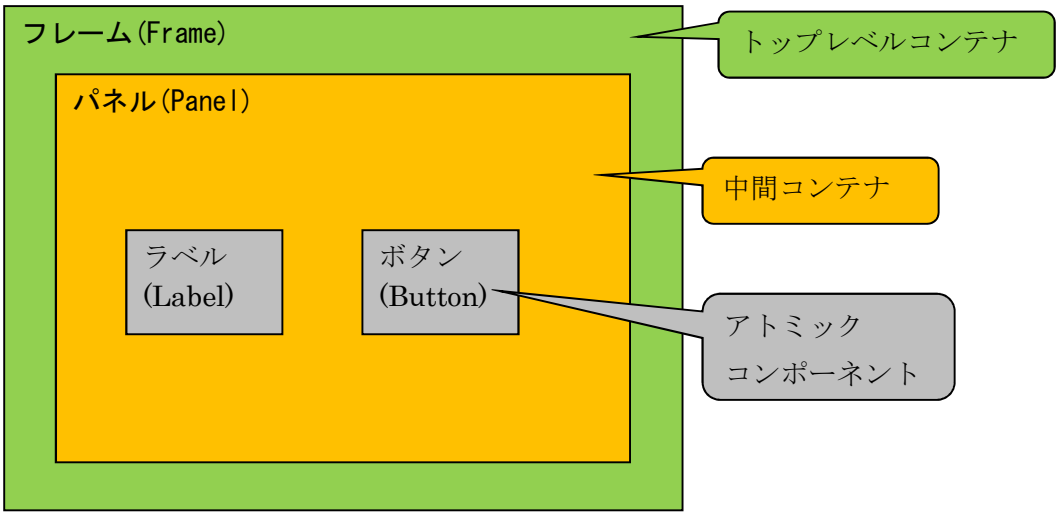
```
修飾子 class クラス名 implements インタフェース名 1 [,インタフェース名 2, ・・・]{
  型 フィールド名;
  修飾子 コンストラクタ名( 引数 ) {
    コンストラクタの中身
  }
  修飾子 戻り値のデータ型 メソッド名( 引数 ) {
    メソッドの中身
  }
}
```

＊ クラスの継承とインタフェースの実装とを同時に行うクラスの作成が可能である。

＊＊ インタフェースを extends で継承したサブインタフェースの作成も可能である。

GUI コンポーネント

GUI コンポーネントには 3 つの階層があり，これらを重ね合わせて用いる。



TestAWT クラス・TestAWT2 クラス・TestAWT3 クラス

Eclipse で jimbo の様に『自分の名前』のパッケージを指定して，次の TestAWT, TestAWT2, TestAWT3 の 3 つのクラスを作成せよ。(これ等は直接実行出来ないクラス ⇒ 後の例題に利用する)

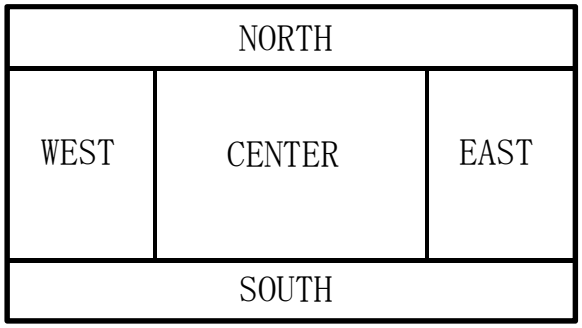
```
package jimbo;

import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.Frame;
import java.awt.Label;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class TestAWT extends Frame implements ActionListener {
    Label l1 = new Label();

    public TestAWT(){
        this.setSize(400,300);

        Button b1 = new Button("Go!");
        b1.addActionListener(this);

        this.add(b1, BorderLayout.CENTER);
        this.add(l1, BorderLayout.SOUTH);

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent wevent) {
                System.exit(0);
            }
        });
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO 自動生成されたメソッド・スタブ
        l1.setText("Hello GUI World!");
    }
}
```

メソッド宣言がスーパークラスのメソッド宣言をオーバーライドすることを示す。(オーバーライドに誤りがあればエラーメッセージを出す。)

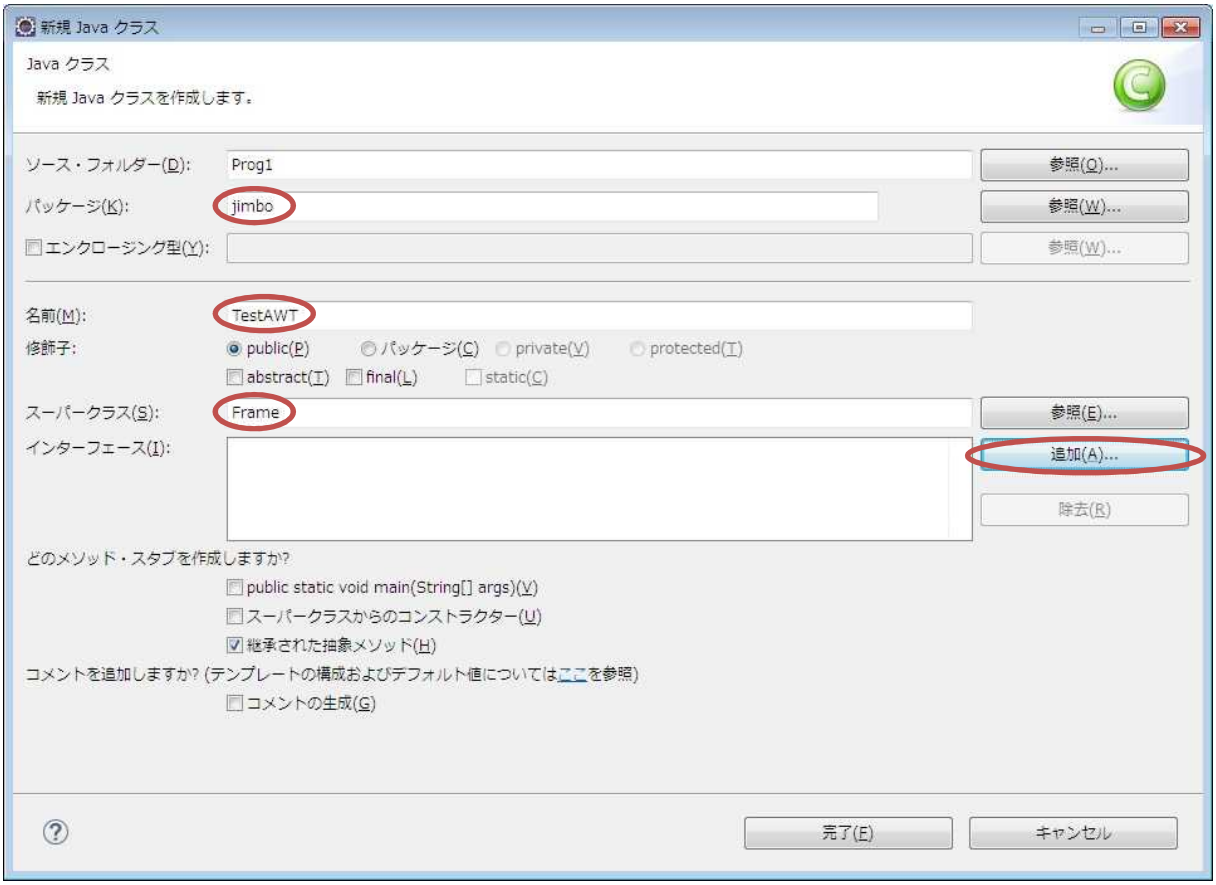
【解説】

1. Frame クラスはフレームの作成に用いられる。
2. actionPerformed は，インタフェース ActionListener のメソッドをオーバーライドしている。
3. setSize(400,300)は，幅を 400 ピクセル，高さを 300 ピクセルに設定する。
4. addActionListener は，このボタンからアクションイベントを受け取るために，指定されたアクションリスナーを追加する。
5. BorderLayout は，NORTH（上端），SOUTH（下端），EAST（右端），WEST（左端）及び CENTER（中央）という 5 つの領域に収まるように，コンポーネントを整列，サイズ変更して，コンテナに配置する。

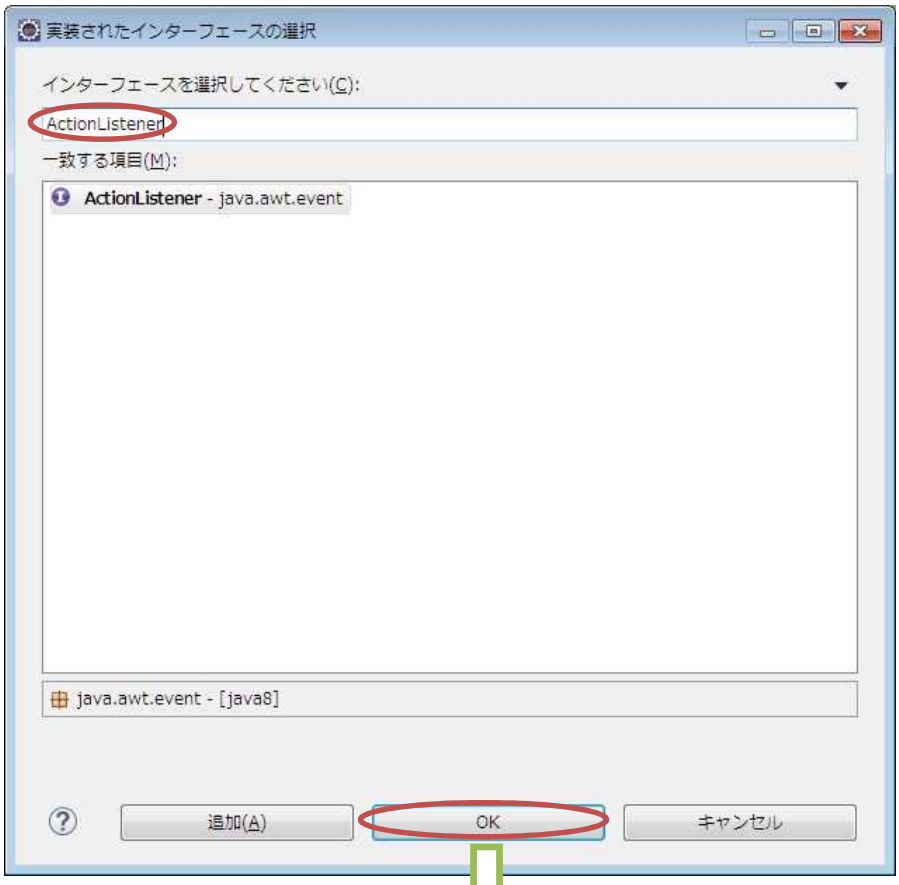


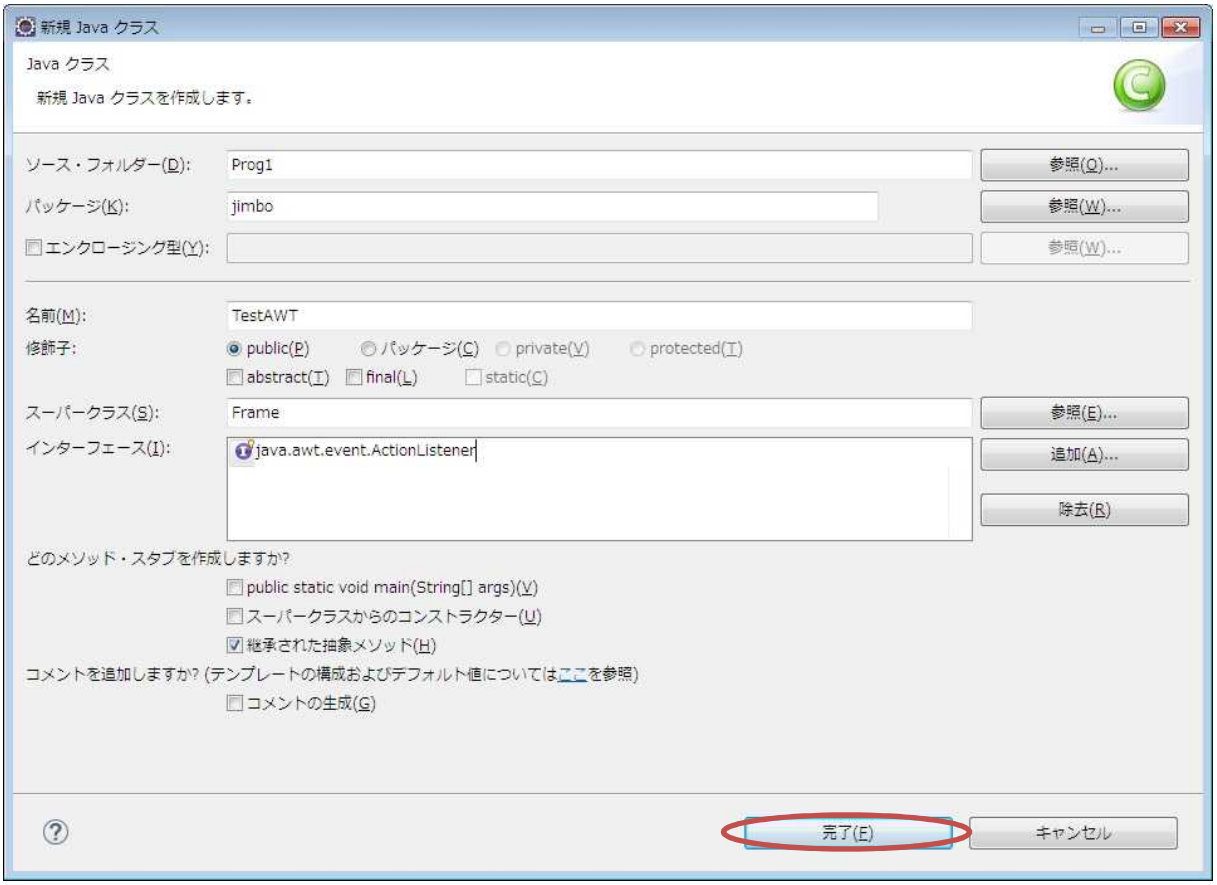
6. addWindowListener は，このウィンドウからウィンドウイベントを受け取るために，指定されたウィンドウリスナーを追加する。
7. WindowAdapter は，ウィンドウイベントを受け取るための**抽象アダプタクラス**と呼ばれるものである。このクラスのメソッドは全て空となっているので，WindowEvent リスナーを作成するには，このクラスを拡張して関係のあるイベント用のメソッドをオーバーライドする。ここでは，ユーザがウィンドウのシステムメニューでウィンドウを閉じようとしたときに呼び出される windowClosing メソッドを『System.exit(0);』でオーバーライドしている。
8. addWindowListener メソッドの引数では，メソッド内に new でインスタンスを作成する際に，メソッドを定義している。これを**無名クラス**と呼ぶ。無名クラスにはフィールドも定義出来る。

なお，eclipse では新規のクラスを作成する際に，第 9 回の教材で学んだ様に，スーパークラスを指定することが出来るが，更に次の様にして，実装するインタフェースも指定することが出来る。



(続きは次のページ)

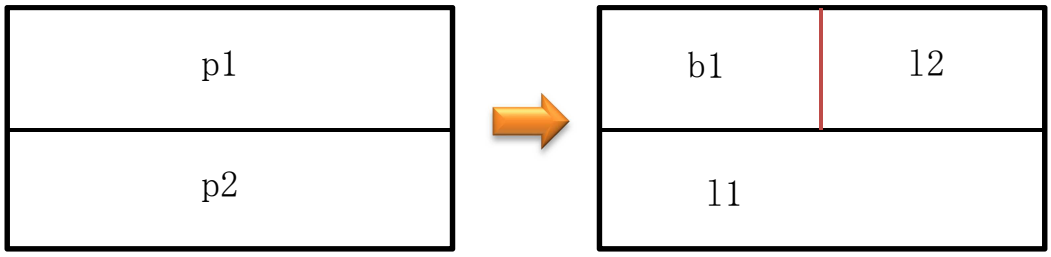

実装されたインターフェースの選択
インターフェースを選択してください(C):
ActionListener
一致する項目(M):
ActionListener - java.awt.event
java.awt.event - [java8]
追加(A)
OK
キャンセル


新規 Java クラス
Java クラス
新規 Java クラスを作成します。
ソース・フォルダー(D):
Prog1
参照(O)...
パッケージ(K):
jimbo
参照(W)...
エンクロージング型(Y):
参照(W)...
名前(M):
TestAWT
修飾子:
public(P)
パッケージ(C)
private(V)
protected(T)
abstract(T)
final(L)
static(C)
スーパークラス(S):
Frame
参照(E)...
インターフェース(I):
java.awt.event.ActionListener
追加(A)...
除去(R)
どのメソッド・スタブを作成しますか?
public static void main(String[] args)(V)
スーパークラスからのコンストラクター(U)
継承された抽象メソッド(H)
コメントを追加しますか? (テンプレートの構成およびデフォルト値についてはここを参照)
コメントの生成(G)
完了(F)
キャンセル


```
package jimbo;

import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;

public class TestAWT2 extends TestAWT {
    public TestAWT2(){
        Panel p1 = new Panel();
        Panel p2 = new Panel();
        Label l2 = new Label("Click Button!");

        setLayout(new GridLayout(2, 1));
        add(p1);
        add(p2);

        p1.setLayout(new GridLayout(1, 2));
        p2.setLayout(new GridLayout(1, 1));
        p1.add(l2);
        p2.add(l1);
    }
}
```

【解説】

1. TestAWT2 クラスは TestAWT クラスを継承しているので，インスタンス化する際に TestAWT クラスのコンストラクタが先ず呼び出される。
2. setLayout は Container クラスのメソッドで，この**コンテナ**のレイアウトマネージャを設定する。ここでコンテナとは，他の AWT コンポーネントを含むことが出来るコンポーネントである。
3. GridLayout クラスは，コンテナのコンポーネントを矩形グリッドで配置するレイアウトマネージャであり，コンテナは等しいサイズの矩形に分割され，各矩形にコンポーネントが 1 つずつ配置される。



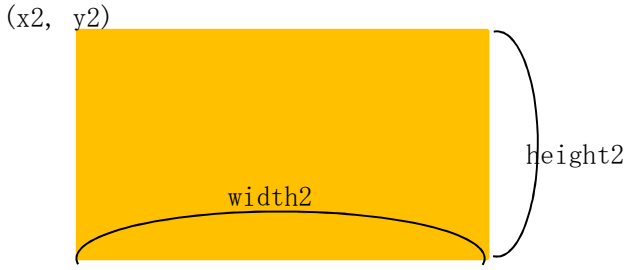

```
package jimbo;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Label;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class TestAWT3 extends Frame {
    static final int x1 = 145, y1 = 120, width1 = 60, height1 = 60;
    static final int x2 = 100, y2 = 100, width2 = 150, height2 = 100;

    public TestAWT3(){
        this.setSize(400,300);
        Label l1 = new Label("Click Window!");
        this.add(l1,BorderLayout.NORTH);

        this.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent mevent) {
                Graphics gr = getGraphics();

                gr.setColor(Color.orange);
                gr.fillRect(x2, y2, width2, height2);
                gr.setColor(Color.red);
                gr.fillOval(x1, y1, width1, height1);
            }
        });

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent wevent) {
                System.exit(0);
            }
        });
    }
}
```

【解説】

1. addMouseListener は，このコンポーネントからマウスイベントを受け取るために，指定されたマウスリスナーを追加する。

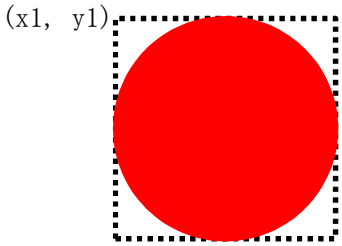
2. MouseAdapter クラスは，マウスイベントを受け取る抽象アダプタクラスである。
3. それぞれのコンポーネントは Graphics クラスのインスタンスを持っているので，getGraphics メソッドで描画オブジェクトを取得出来る。
4. setColor は，このグラフィックスコンテキスト(描画情報を格納しておくためのもの)の現在の色を指定された色に設定する。
5. fillRect(x2, y2, width2, height2)は，指定された矩形を塗りつぶす。矩形の左端，右端はそれぞれ x2，x2+width2-1 で，上端，下端はそれぞれ y2，y2+height2-1 である。これで指定される矩形は，幅 150 ピクセル，高さ 100 ピクセルの領域である。パラメータの意味は次の通りである。

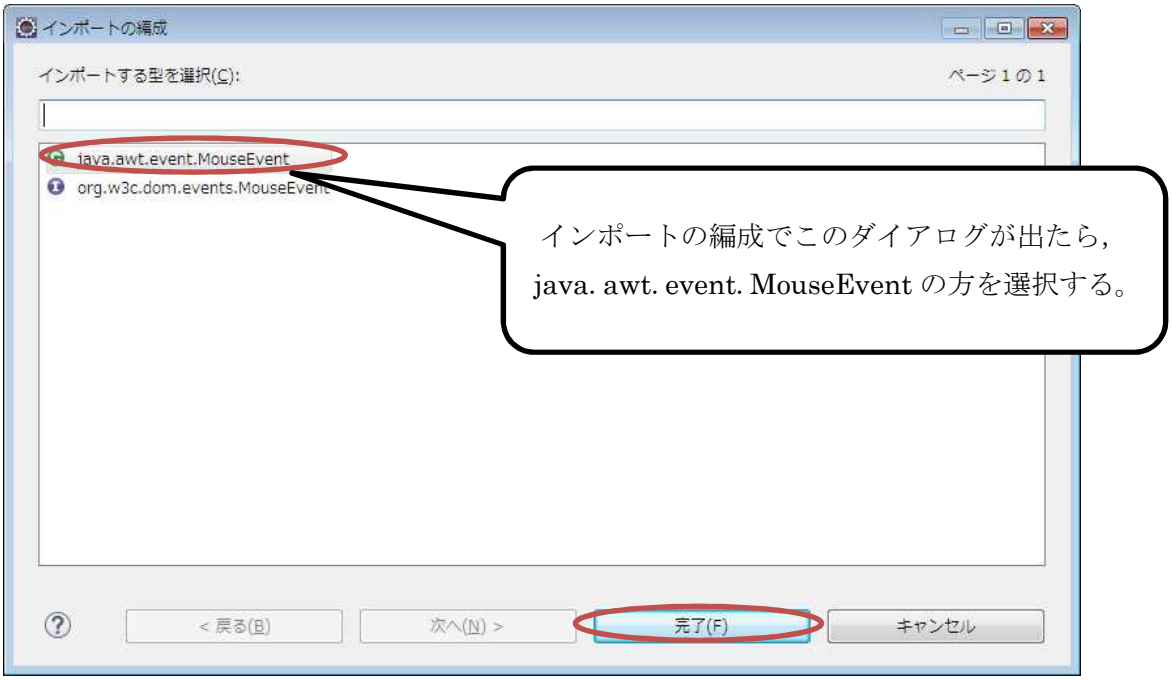
x2 - 塗りつぶされる矩形の x 座標
y2 - 塗りつぶされる矩形の y 座標
width2 - 塗りつぶされる矩形の幅
height2 - 塗りつぶされる矩形の高さ



6. fillOval(x1, y1, width1, height1)は，指定された矩形の中の楕円形を現在の色で塗りつぶす。パラメータの意味は次の通りである。

x1 - 描画される楕円の左上隅の x 座標
y1 - 描画される楕円の左上隅の y 座標
width1 - 塗りつぶされる楕円の幅
height1 - 塗りつぶされる楕円の高さ





例題 1 （**ボタンとラベルの表示**）

次のプログラムを入力し，ビルドして，実行せよ。ここで，クラス名は Sample12_1，ソースファイル名は Sample12_1.java とする。なお，プログラムを終了するには，ウィンドウを閉じる。

```
package jimbo;


public class Sample12_1 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        TestAWT testAWT = new TestAWT();

        testAWT.setVisible(true);
    }

}
```

【解説】setVisible は引数が true の場合にコンポーネントを表示し，そうでない場合はコンポーネントを隠す。

例題 2（**パネルを使ったレイアウト**）

次のプログラムを入力し，ビルドして，実行せよ。ここで，クラス名は Sample12_2，ソースファイル名は Sample12_2.java とする。

```
package jimbo;


public class Sample12_2 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        TestAWT2 testAWT2 = new TestAWT2();

        testAWT2.setVisible(true);
    }

}
```

例題 3（**描画**）

次のプログラムを入力し，ビルドして，実行せよ。ここで，クラス名は Sample12_3，ソースファイル名は Sample12_3. java とする。

```
package jimbo;

public class Sample12_3 {
```

```
    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        TestAWT3 testAWT3 = new TestAWT3();

        testAWT3.setVisible(true);
    }

}
```

演習

“Graphics java”をキーワードとして Web 上で検索を行い，docs.oracle.com のサイトにある Java Platform SE 7 のドキュメントを探して読み，図形の描画について調べ，ウィンドウをクリックすると，自分のデザインした図形の描画を行うプログラムを作成し，ビルドして実行せよ。ここでは，Frame クラスを継承したサブクラス TestAWT4 を新規に作成して，自分のデザインした図形の描画を行うコンストラクタを作成する（ソースプログラム名は TestAWT4.java）ものとし，更にそのインスタンスを生成する main メソッドを含むプログラム（クラス名は Ex12，ソースプログラム名は Ex12.java とする）を作成して，ビルド・実行する。

提出物：

1）演習のソースプログラムのファイル TestAWT4.java 及び Ex12.java をメールに添付する。

2）第 11 回の授業の復習の内容を埋めたファイル Review_11th.txt をメールに添付する。