# 情報処理演習９

鎌原淳三 kamahara@port.kobe-u.ac.jp

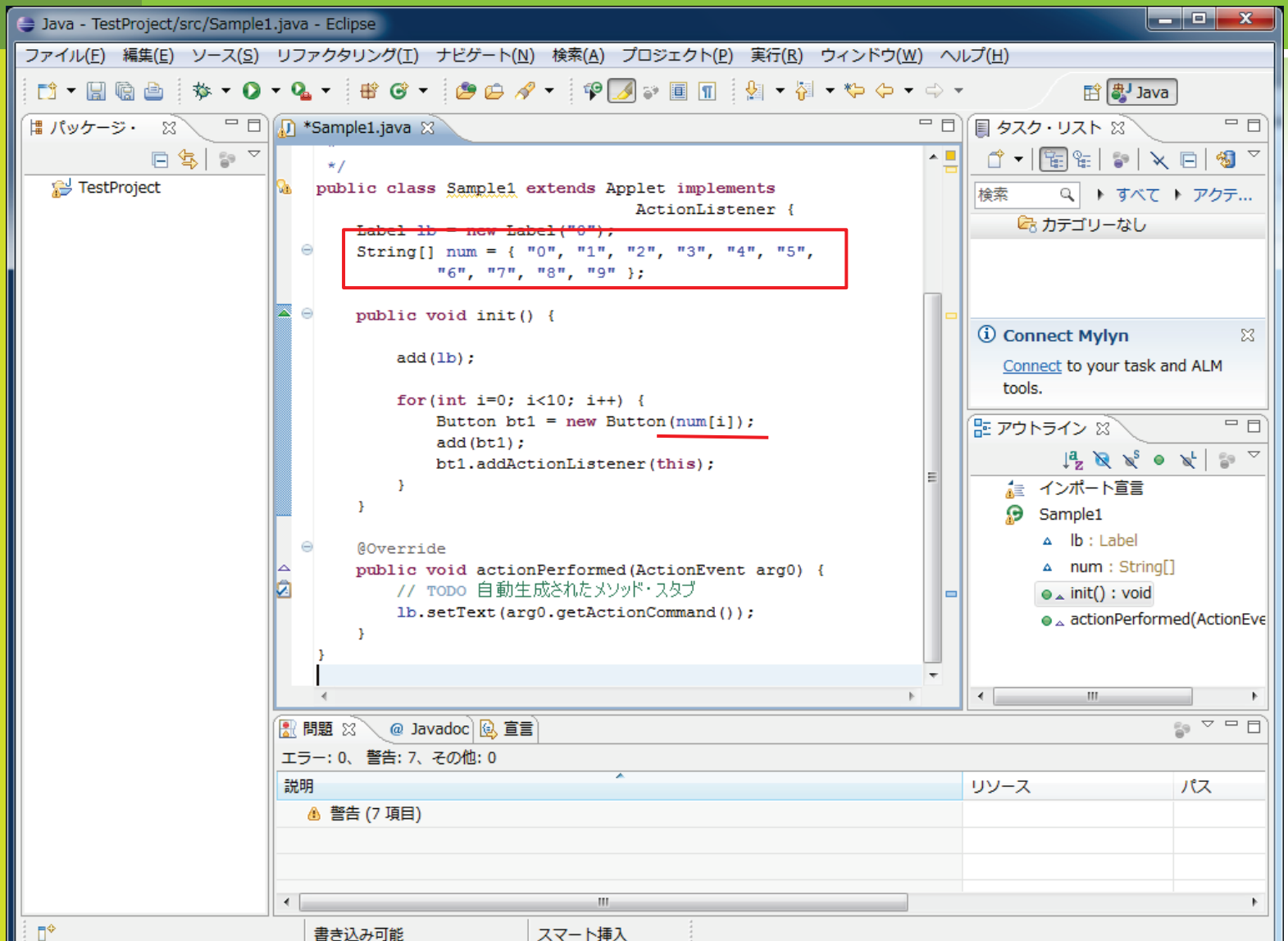
# 前回の課題

- 押すとラベルが0から9まで書き変わる10個のボタンをfor文を使って生成し、アプレット上に表示されるようにしなさい

String.valueOf(i)でもよい

```
for( int i=0; i<10; i++) {
        Button bt1 = new Button(“”+i);
        add(bt1);
        bt1.addActionListener(this);
}
```

# 別解


Java - TestProject/src/Sample1.java - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
Java
パッケージ・
TestProject
*Sample1.java
*/
public class Sample1 extends Applet implements
ActionListener {
Label lb = new Label("0");
String[] num = { "0", "1", "2", "3", "4", "5",
"6", "7", "8", "9" };
public void init() {
add(lb);
for(int i=0; i<10; i++) {
Button bt1 = new Button(num[i]);
add(bt1);
bt1.addActionListener(this);
}
}
@Override
public void actionPerformed(ActionEvent arg0) {
// TODO 自動生成されたメソッド・スタブ
lb.setText(arg0.getActionCommand());
}
}
タスク・リスト
検索
すべて
アクテ...
カテゴリーなし
Connect Mylyn
Connect to your task and ALM tools.
アウトライン
インポート宣言
Sample1
lb : Label
num : String[]
init() : void
actionPerformed(ActionEve
問題
Javadoc
宣言
エラー: 0、 警告: 7、その他: 0
説明
リソース
パス
警告 (7 項目)
書き込み可能
スマート挿入

# ボタンの配置を自由に

- setLayout(null);でボタンの配置を自由に設定できるようになる
- ただし、ボタン毎にsetSizeで大きさ、setLocationで位置を指定する必要がある

ボタンの大きさ、位置を指定する例

```
Button bt = new Button(num[i]);
bt.setSize( 20, 20 );
bt.setLocation( 40, 50 );
add(bt);
```

# ボタンを置く座標を配列に格納

- 二次元配列を用いる

pos

| | 0: X | 1: Y |
|---|---|---|
| 0 | 10 | 30 |
| 1 | 30 | 30 |
| 2 | 50 | 30 |

- int[][]  pos = new int[10][2];　10行2列の例
- int[][] pos = { { 10, 30},  { 30, 30 }, {50, 30 } };
  3行2列で初期値で生成する例

# 二次元配列を使って位置を指定

```
for( int i=0; i<num.length; i++) {
        Button bt1 = new Button(num[i]);
        bt1.setSize( 19, 19 );
        bt1.setLocation(pos[i][0], pos[i][1] );
        add(bt1);
        bt1.addActionListener(this);
}
```

# その他の部分を指定

```
public class Sample1 extends Applet implements
                                    ActionListener {
    Label lb = new Label("0");
    String[] num = { "0", "1", "2", "3", "4", "5",
            "6", "7", "8", "9" };
    int[][] pos = {
            { 10, 30 }, { 30, 30 }, {50, 30 },
            { 10, 50 }, { 30, 50 }, {50, 50 },
            { 10, 70 }, { 30, 70 }, {50, 70 },
            { 10, 90 }
    };

    public void init() {
        setLayout(null);

        lb.setSize(180,16);
        lb.setLocation(30, 10);
        add(lb);

        for(int i=0; i<num.length; i++) {
            Button bt1 = new Button(num[i]);
            bt1.setSize(19,19);
            bt1.setLocation(pos[i][0], pos[i][1]);
            add(bt1);
            bt1.addActionListener(this);
        }
    }

    @Override
    public void actionPerformed(ActionEvent arg0) {
        // TODO 自動生成されたメソッド・スタブ
        lb.setText(arg0.getActionCommand());
    }
}
```

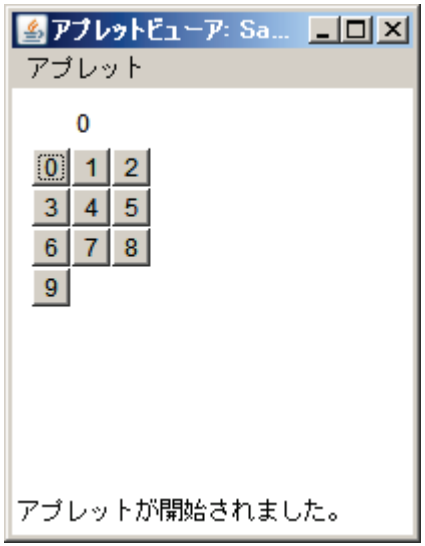
# 関数化しておく

```
public void init() {
    setLayout(null);

    lb.setSize(180,16);
    lb.setLocation(30, 10);
    add(lb);

    for(int i=0; i<num.length; i++) {
        addButton(i);
    }
}
public void addButton(int i) {
    Button bt1 = new Button(num[i]);
    bt1.setSize(19,19);
    bt1.setLocation(pos[i][0], pos[i][1]);
    add(bt1);
    bt1.addActionListener(this);
}
```

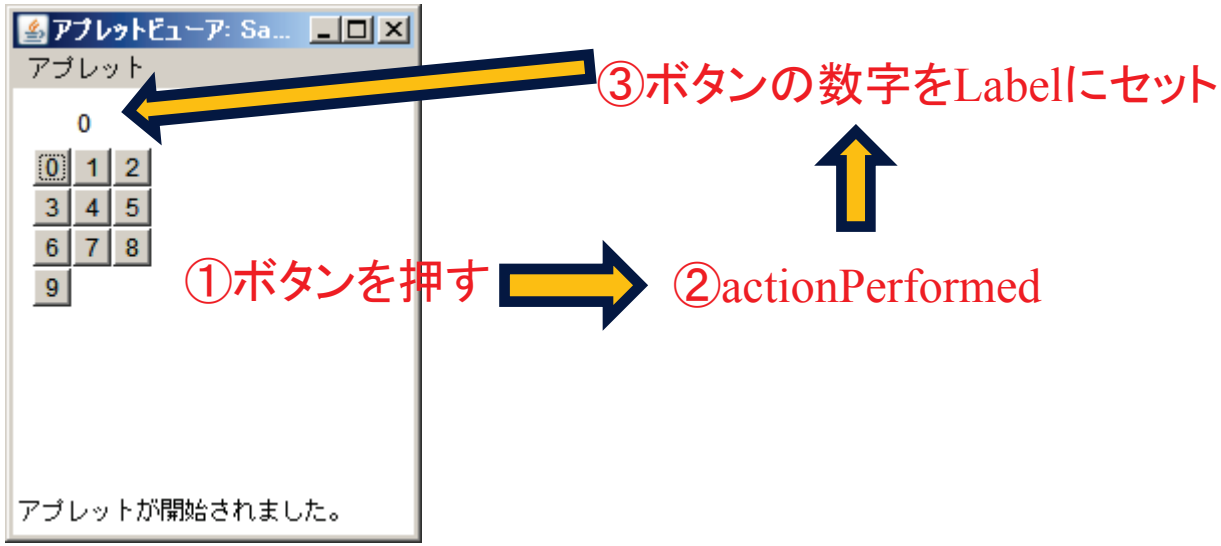
# 実行例



# 電卓を作ってみよう

- 押すと数字の桁が増えていく
- 足し算ができるようにする
- ボタンとして「+」と「=」が必要

今の処理の流れ



# 数字の桁が増える

- 現在は「③ボタンの数字をLabelにセット」
- これを「現在のラベルの数字にボタンの数字を追加する」にする
- 現在のラベルは、lb.getText()で取得
- 前の行は // でコメントとしておく

```
public void actionPerformed(ActionEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ
    String numstr = lb.getText();
    //lb.setText(arg0.getActionCommand());
    lb.setText(numstr+arg0.getActionCommand());
}
```

# 実行してみよう

- 問題はあるか？
- 最初の０がそのまま残る
- 「現在のラベル」が０の時だけ、処理を変える→if文

```
public void actionPerformed(ActionEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ
    String numstr = lb.getText();
    if(numstr.equals("0")) {
        lb.setText(arg0.getActionCommand());
    } else {
        lb.setText(numstr+arg0.getActionCommand());
    }
}
```

現在のラベルが“0”だったら
数字と置き換える

それ以外の時は後ろに数字を追加する

# 足し算ができるようにする

- ラベルは実は単なる文字列
- 文字列を数字にする必要がある
- また、足し算の時には１つ前の数字を電卓が覚えている
- 現在表示している数字をdispNumber、覚えている数字をremNumberとする
- 演算も覚えておく operatorに記憶

# dispNumber追加

```
public class Sample1 extends Applet implements
                                ActionListener {
    Label lb = new Label("0");
    String[] num = { "0", "1", "2", "3", "4", "5",
            "6", "7", "8", "9" };
    int[][] pos = {
            { 10, 30 }, { 30, 30 }, {50, 30 },
            { 10, 50 }, { 30, 50 }, {50, 50 },
            { 10, 70 }, { 30, 70 }, {50, 70 },
            { 10, 90 }
    };
    double dispNumber=0;

    public void init() {
        setLayout(null);

        lb.setSize(180,16);
        lb.setLocation(30, 10);
        add(lb);

        for(int i=0; i<num.length; i++) {
            addButton(i);
        }
    }
    public void addButton(int i) {
        Button bt1 = new Button(num[i]);
        bt1.setSize(19,19);
        bt1.setLocation(pos[i][0], pos[i][1]);
        add(bt1);
        bt1.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent arg0) {
        // TODO 自動生成されたメソッド・スタブ
        String numstr = lb.getText();
        if(numstr.equals("0")) {
            lb.setText(arg0.getActionCommand());
        } else {
            lb.setText(numstr+arg0.getActionCommand());
        }
        dispNumber = Double.parseDouble(lb.getText());
    }
}
```

```
            { 10, 70 }, { 30, 7
            { 10, 90 }
    };
    double dispNumber=0;

    public void init() {
        setLayout(null);
```

```
        lb.setText(numstr+arg0.getActionCommand());
    }
    dispNumber = Double.parseDouble(lb.getText());
}
```

# “+”ボタンを追加する

- 配列の要素を増やすだけでボタン自体は追加できる（処理は別）

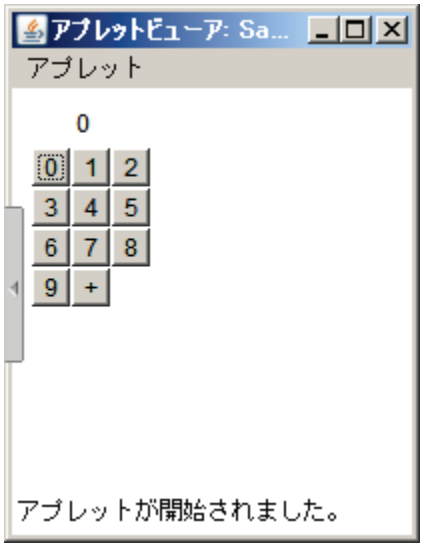
```
String[] num = { "0", "1", "2", "3", "4", "5",
        "6", "7", "8", "9", "+" };
int[][] pos = {
        { 10, 30 }, { 30, 30 }, {50, 30 },
        { 10, 50 }, { 30, 50 }, {50, 50 },
        { 10, 70 }, { 30, 70 }, {50, 70 },
        { 10, 90 }, { 30, 90 }
};
double dispNumber=0;
```

# dispNumberが0の時の処理も追加

- “+”を押した後にdispNumberを0にするのでその時は、数字が置き換わるようにする

```
String numstr = lb.getText();
if(numstr.equals("0")||(dispNumber==0)) {
    lb.setText(arg0.getActionCommand());
} else {
    lb.setText(numstr+arg0.getActionCommand());
}
dispNumber = Double.parseDouble(lb.getText());
```

# こまめに実行して動作しているか確認



# “=“ボタンやその他の変数を定義

```
String[] num = { "0", "1", "2", "3", "4", "5",
        "6", "7", "8", "9", "+", "=" };
int[][] pos = {
        { 10, 30 }, { 30, 30 }, {50, 30 },
        { 10, 50 }, { 30, 50 }, {50, 50 },
        { 10, 70 }, { 30, 70 }, {50, 70 },
        { 10, 90 }, { 30, 90 }, {50, 90 }
};
double dispNumber=0, remNumber=0;
String operator=null;
```

# ACTIONPERFORMEDで場合分けする

```
public void actionPerformed(ActionEvent arg0) {
    if(arg0.getActionCommand().equals("+")) {
        ＋が押された時の処理
    } else if (arg0.getActionCommand().equals("=")) {
        ＝が押された時の処理
    } else {
        String numstr = lb.getText();
        if(numstr.equals("0")||(dispNumber==0)) {
            lb.setText(arg0.getActionCommand());
        } else {
            lb.setText(numstr+arg0.getActionCommand());
        }
        dispNumber = Double.parseDouble(lb.getText());
    }
}
```

それ以外の時の処理

# “+”が押された時の処理を考える

- ＋が押された時は、すぐに計算するのではなく、現在表示している値(dispNumber)を、記憶している値(remNumber)とする
- また、押された演算をoperatorに入れる
- dispNumberは0にする

```
if(arg0.getActionCommand().equals("+")) {
    remNumber = dispNumber;
    operator = arg0.getActionCommand();
    dispNumber = 0;
```

# 足し算の処理

- answerに答えを入れる
- lb.setText(asnwer.toString());で答えをラベルに表示する

```
} else if (arg0.getActionCommand().equals("=")) {
    Double answer =
        lb.setText(answer.toString());
    remNumber = answer;
    dispNumber = 0;
} else {
```

# 課題

- ①足し算が動作するプログラムとせよ
- ②ボタンの配置を電卓みたいに並べよ
- 追加課題：引き算や掛け算・割り算もできるようにする

# 課題の提出

- Z:¥workspace¥TestProject¥src¥Sample1.javaを添付ファイルとして提出すること。ファイル中にコメントして学籍番号が入っていること
- 件名：情報処理演習９ 学籍番号　名前
- 宛先：kamahara@port.kobe-u.ac.jp