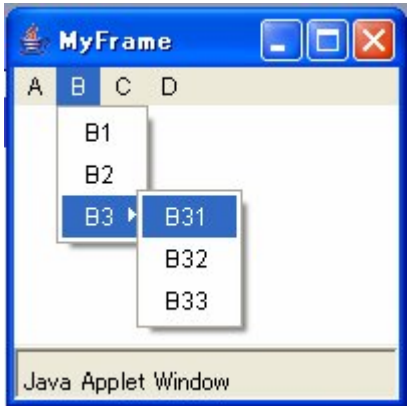
独習Java第3版

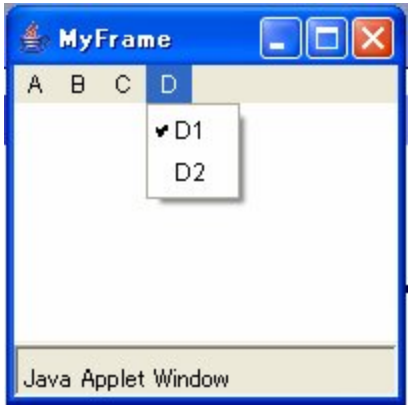
15.16 メニューとメニューバー
15.17 ダイアログとファイルダイアログ

7月25日(火)

発表者
佐々木研　田島勇樹

１５．１６メニューとメニューバー（1）

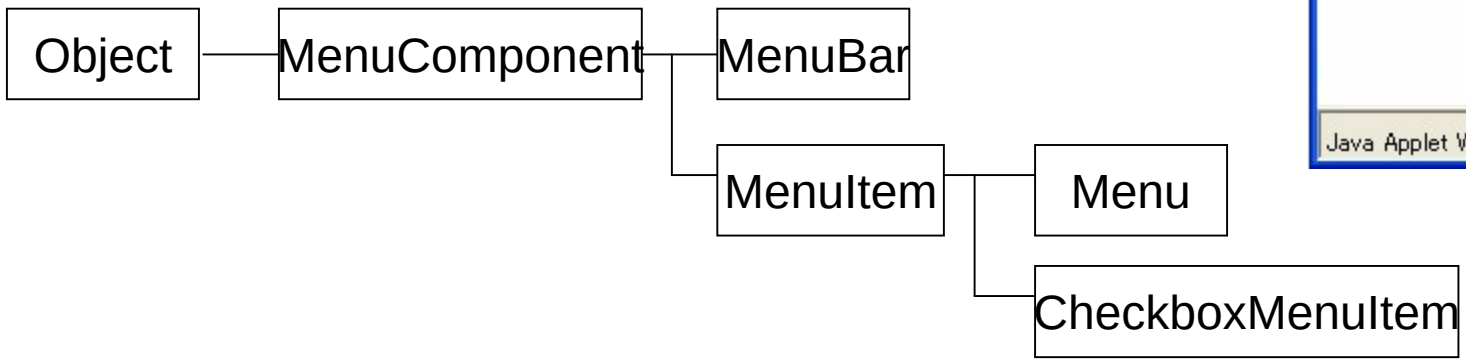




- メニューバー

・フレームにメニューバーを入れることが出来る。
・メニューバーにはオプションのセットが表示
・サブオプションにサブオプションを追加できる。
・サブオプションを選択するとチェックマークが表示される

この機能を提供するクラス間の関係



15．16メニューとメニューバー (2)

- MenuComponentクラスは図で示した全てのクラスのスーパークラス

- MenuComponentクラスで定義されているメソッド

setFont()メソッド

void setFont(Font f)

メニュー関連の文字列に使用するフォント

- MenuBarクラスはメニューバーの機能をカプセル化

MenuBarコンストラクタ

MenuBar()

- MenuBarクラスで定義されているメソッド

add()メソッド

Menu add(Menu m)

メニューバーにMenuオブジェクトを追加

１５．１６メニューとメニューバー（３）

- Menuコンストラクタ

Menu（String str）
strはメニューに表示する文字列

- Menuで定義されているメソッド

add()メソッド
MenuItem add(MenuItem mi)　メニューにMenuItemを追加する
void add(String str)　メニューに追加する文字列

- MenuItemコンストラクタ

MenuItem(String str)
strはメニュー項目用の表示文字列

１５．１６メニューとメニューバー（４）

- MenuItemで定義されているメソッド

- アクションイベントを受け取るための登録と解除を行うメソッド

void addActionListener(ActionListener al)　登録
void removeActionListener(ActionListener al)　解除
al：アクションリスナ

- メニュー項目を有効、無効にするメソッド

void setEnabled(boolean flag)

flagが真の場合、有効になり、偽の場合無効になる。

１５．１６メニューとメニューバー (5)

- ChackboxMenuItemクラスはチェックボックスメニュー項目をカプセル化

ChackboxMenuItemコンストラクタ

ChackboxMenuItem (String str)
ChackboxMenuItem (String str,boolean flag)

strはチェックボックスメニュー項目用の表示文字列
flagが真の場合チェックされ、偽の場合チェックされない。

１５．１６メニューとメニューバー（6）

- ChackboxMenuItemクラスで定義されているメソッド

- 項目イベントを受け取るための登録、解除を行うメソッド

void addItemListener(ItemListener il)　登録
void removeItemListener(ItemListener il)　解除
il:項目リスナ

- チェックボックスメニュー項目の読み取り、書き取りを行うメソッド

boolean getState()　　メニュー項目の状態を返す
void setState(boolean flag)

flagが真の場合チェックボックスメニュー項目は設定される。
偽の場合は消去される。

15．16プログラム(1)

```
メニューとメニューバーを使ったプログラム
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*
  <applet code="MenuItemEvents" width=400 height=200>
  </applet>
*/

class MenuFrame extends Frame
implements ActionListener, ItemListener, WindowListener {
  MenuItemEvents menuItemEvents;

 MenuFrame(String title, MenuItemEvents menuItemEvents) {
    super(title);                  //タイトルバーに文字列表示
 this.menuItemEvents = menuItemEvents;
 addWindowListener(this);//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を追加
    // メニューバーを作成し、設定する
 MenuBar mb = new MenuBar();
 setMenuBar(mb);
```

１５．１６プログラム（２）

```
// メニューAを作成する
   Menu a = new Menu("A");
    mb.add(a);                    //メニューバーに追加
    MenuItem a1 = new MenuItem("A1");
    a1.addActionListener(this);//メニュー項目からアクションイベントを受け取るアクションリスナ
      を追加
    a.add(a1);                    //メニューAに追加
    MenuItem a2 = new MenuItem("A2");
    a2.addActionListener(this);
    a.add(a2);
    MenuItem a3 = new MenuItem("A3");
    a3.addActionListener(this);
    a.add(a3);

    // メニューBを作成する
    Menu b = new Menu("B");
    mb.add(b);
    MenuItem b1 = new MenuItem("B1");
    b1.addActionListener(this);
    b.add(b1);
    MenuItem b2 = new MenuItem("B2");
    b2.addActionListener(this);
    b.add(b2);
```

１５．１６プログラム (3)

```
// B3のサブメニューを作成する
    Menu b3 = new Menu("B3");
    b.add(b3);
    MenuItem b31 = new MenuItem("B31");
    b31.addActionListener(this);
    b3.add(b31);
    MenuItem b32 = new MenuItem("B32");
    b32.addActionListener(this);
    b3.add(b32);
    MenuItem b33 = new MenuItem("B33");
    b33.addActionListener(this);
    b3.add(b33);// メニューCを作成する
```

15．16プログラム (4)

```
  // メニューCを作成する
   Menu c = new Menu("C");
  mb.add(c);
  MenuItem c1 = new MenuItem("C1");
  c1.addActionListener(this);
  c.add(c1);
  MenuItem c2 = new MenuItem("C2");
  c2.addActionListener(this);
  c.add(c2);

  // メニューDを作成する
  Menu d = new Menu("D");
  mb.add(d);
  CheckboxMenuItem d1 = new CheckboxMenuItem("D1");
  d1.addItemListener(this);//チェックボックスからの項目イベントを受け取れるように、指定さ
                           //れた項目リスナを追加
  d.add(d1);
  CheckboxMenuItem d2 = new CheckboxMenuItem("D2");
  d2.addItemListener(this);
  d.add(d2);
}
```

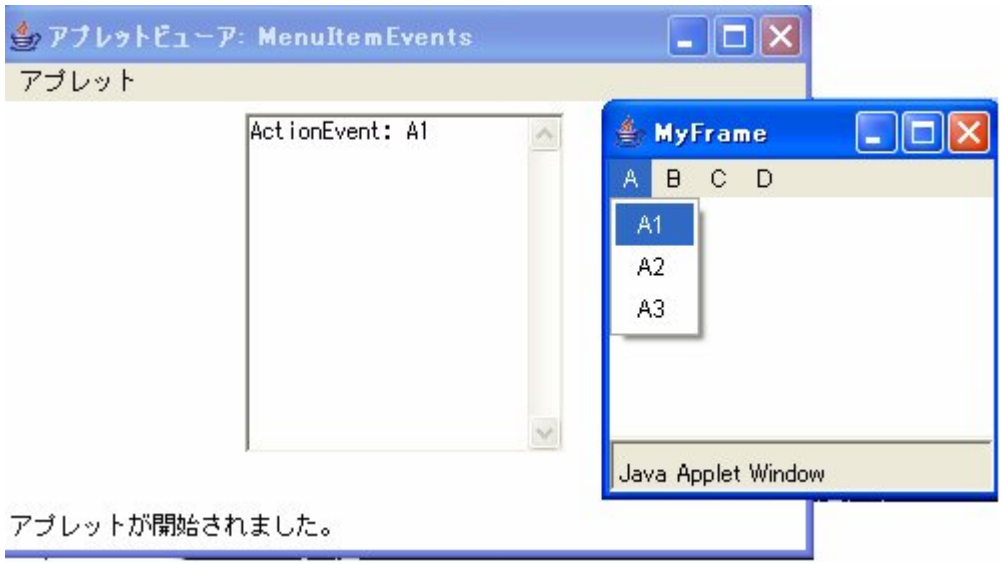
15．16プログラム(5)

```
public void actionPerformed(ActionEvent ae) {  //アクション発生時に呼び出される。
    menuItemEvents.ta.append("ActionEvent: " +
      ae.getActionCommand() + "\n");
  }
public void itemStateChanged(ItemEvent ie) {//ユーザによって項目が選択または選択解除
                                                    //されたときに呼び出されます
    CheckboxMenuItem cbmi = (CheckboxMenuItem)ie.getSource();//イベント生成
    menuItemEvents.ta.append("ItemEvent: " +
      cbmi.getLabel() + "\n");
  }

 public void windowActivated(WindowEvent we) {
 }
  public void windowClosed(WindowEvent we) {
  }
  public void windowClosing(WindowEvent we) {
    dispose();                                      //リソースを放棄
  }
  public void windowDeactivated(WindowEvent we) {
  }
  public void windowDeiconified(WindowEvent we) {
  }
```

１５．１６プログラム（6）

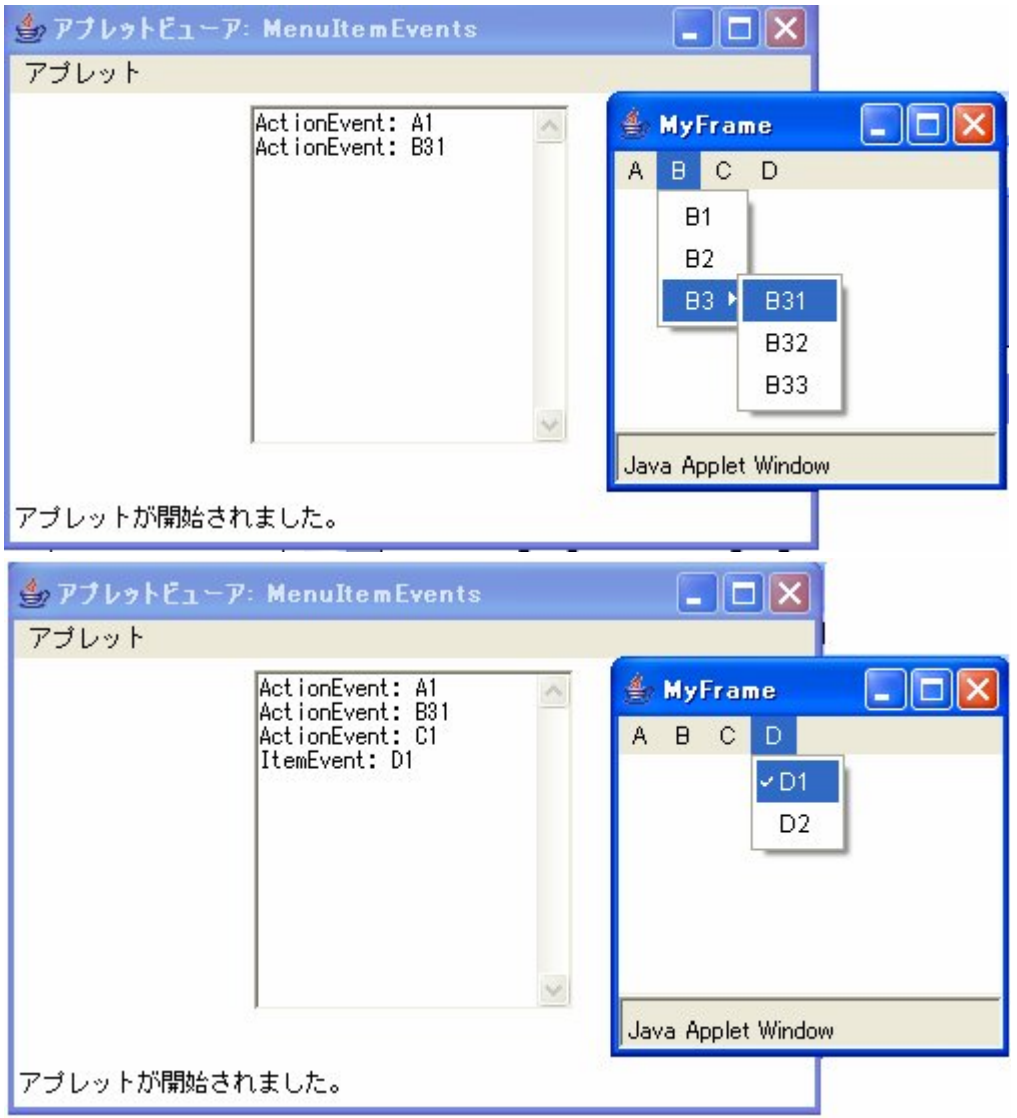
```
public void windowIconified(WindowEvent we) {
  }
  public void windowOpened(WindowEvent we) {
  }
}

public class MenuItemEvents extends Applet {//アプレットの中身
  TextArea ta;

 public void init() {
    MenuFrame mf = new MenuFrame("MyFrame", this);
    mf.setSize(200, 200);//ウインドウサイズ設定
    mf.setVisible(true);//フレーム表示
    ta = new TextArea(10, 20);
//10行20列のテキスト領域作成
    add(ta);
//アプレットにテキスト領域追加
  }
}
```

実行結果



１５．１６プログラム (7)

- 実行結果2



１５．７ダイアログとファイルダイアログ（1）

- ダイアログ

GUI内で情報を表示したり入力を取得するためのもの

- モーダルダイアログ

一度開いたダイアログボックスを閉じるまで、他の操作をできなくするタイプのダイアログボックス

- モードレスダイアログ

一度開いたダイアログボックスを閉じなくても、ダイアログボックスを開いたアプリケーションソフトに他の操作を支持できるタイプのダイアログボックス

１５．７ダイアログとファイルダイアログ（2）

- Dialog

→Windowを拡張したクラス。
ユーザーインターフェイスコンポーネントを全て使用できるウインドウを提供

- Dialogコンストラクタ

Dialog(Frame parent)
Dialog(Frame parent,boolean flag)
Dialog(Frame parent,String title)
Dialog(Frame parent,String title,boolean flag)

parent:ダイアログの所有者の参照
flag:真の場合ダイアログはモーダル、偽の場合はモードレス
title:初期設定でタイトルバーにtitleが表示

１５．７ダイアログとファイルダイアログ（３）

- setVisible()メソッド・・ダイアログを可視状態にする

void setVisible(Boolean b)

- FileDialog

→Dialogを拡張したクラス。
読み取りまたは書き込み対象のファイルを選択できるダイアログを提供
このクラスにはLOADとSAVEというint型の変数が定義されている。

- FileDialogコンストラクタ

FileDialog(Frame parent)　parent:ダイアログを作成するフレーム
FileDialog(Frame parent,String str)　str:タイトルバーの初期設定
FileDialog(Frame parent,String str,int rw)
rw:FileDialog.LOADの場合ファイルは読み取り用
　FileDialog.SAVEの場合ファイルは書き込み用

最初の２つの形式はファイルを読み取るダイアログが作成される

１５．７ダイアログとファイルダイアログ（４）

- FileDialogで定義されているメソッド

- getFile()メソッド

String getFile()・・このファイルダイアログの選択されているファイルを取得

- setFile()メソッド

void setFile(String str)・・このファイルダイアログの選択されているファイル
を指定されたファイルに設定

15．7ダイアログ（1）

- Dialogクラスを使ったプログラム

```
import java.awt.*;
import java.awt.event.*;

public class MessageDialogDemo extends Frame
implements ActionListener {
  Button b;

  public static void main(String args[]) {
    MessageDialogDemo mdd = new MessageDialogDemo();
    mdd.setSize(200, 100);
    mdd.setVisible(true);
  }

  MessageDialogDemo() {
    super("Messager Dialog Demo");

// レイアウトマネージャを設定する
    setLayout(new FlowLayout());
```

15．7ダイアログ（2）

```
// ボタンを作成する
    b = new Button("Message Dialog");
    b.addActionListener(this);
    add(b);

    // 無名インナクラスがウィンドウイベントを処理する
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        System.exit(0);
      }
    });
  }

  public void actionPerformed(ActionEvent ae) {
    String message = "This is the message";
    MessageDialog md =
      new MessageDialog(this, "Message Dialog", true, message);
    md.setVisible(true);
  }
}
```

15．7ダイアログ（3）

```
class MessageDialog extends Dialog
implements ActionListener {
  Button ok;

  MessageDialog(Frame parent, String title,
  boolean mode, String message) {
    super(parent, title, mode);

    // Centerパネルを作成し、追加する
    Panel pc = new Panel();
    Label label = new Label(message);
    pc.add(label);
    add(pc, BorderLayout.CENTER);

    // Southパネルを作成し、追加する
    Panel ps = new Panel();
    ok = new Button("OK");
    ok.addActionListener(this);
    ps.add(ok);
    add(ps, BorderLayout.SOUTH);
```

15．7ダイアログ（4）

```
// コンポーネントをレイアウトして、このダイアログボックスの
    // 初期サイズを設定する
    pack();

    // 無名インナクラスがウィンドウイベントを処理する
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        System.exit(0);
      }
    });
  }

  public Insets getInsets() {
    return new Insets(40, 20, 20, 20);   //コンテナの境界の周りの上下左右のマージンに関す
                                         //る情報をカプセル化
  }

  public void actionPerformed(ActionEvent ae) {
    dispose();
  }
}
```

15．7ダイアログ (5)

注意：アプレットでないのでそのまま実行する

- 実行結果

１５．７ファイルダイアログ（1）

- FileDialogクラスを使ったプログラム

```
import java.awt.*;
import java.awt.event.*;

public class DialogApplication extends Frame
implements ActionListener, WindowListener {
  Button l, s;
  TextField tf;

  public static void main(String args[]) {
    DialogApplication da = new DialogApplication();
    da.setSize(400, 100);
    da.setVisible(true);//フレーム表示
  }
```

15．7ファイルダイアログ（2）

```
DialogApplication() {
   super("Dialog Application");
   setLayout(new FlowLayout());
   l = new Button("Load");
   l.addActionListener(this);
   add(l);
   s = new Button("Save");
   s.addActionListener(this);
   add(s);
   tf = new TextField(20);
   add(tf);
   addWindowListener(this);
 }
```

１５．７ファイルダイアログ（３）

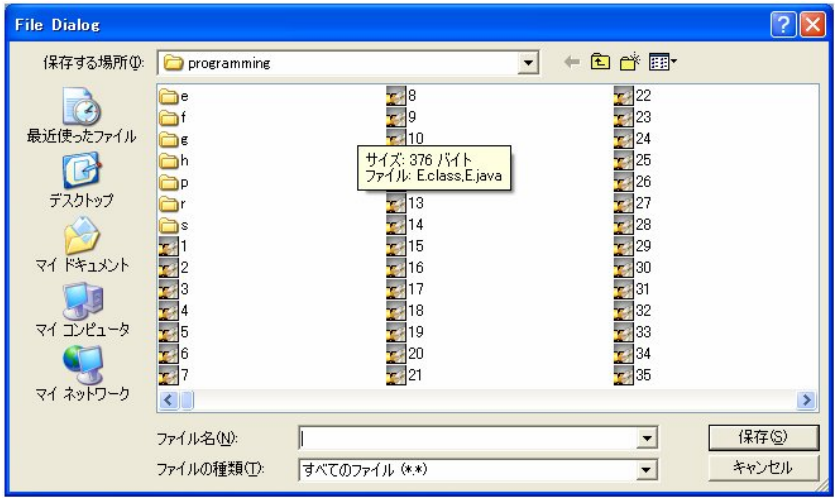
```
public void actionPerformed(ActionEvent ae) {
    FileDialog fd;
    if (ae.getSource() == l) {//ボタンLOADが押されたら
      fd = new FileDialog(this, "File Dialog", FileDialog.LOAD);
    }
    else {//ボタンSAVEが押されたら
      fd = new FileDialog(this, "File Dialog", FileDialog.SAVE);
    }
    fd.setVisible(true);//ダイアログを可視状態にする
    String filename = fd.getFile();
    if (filename != null) {
      tf.setText(filename);
    }
  }
```

15．7ファイルダイアログ（4）

```
public void windowActivated(WindowEvent we) {
 }

 public void windowClosed(WindowEvent we) {
 }

 public void windowClosing(WindowEvent we) {
   System.exit(0);
 }

 public void windowDeactivated(WindowEvent we) {
 }

 public void windowDeiconified(WindowEvent we) {
 }

 public void windowIconified(WindowEvent we) {
 }

 public void windowOpened(WindowEvent we) {
 }
}
```

15．7ファイルダイアログ (5)

- 実行結果



練習問題

1. メニューバーを持つフレームを表示するアプリケーションを作成しなさい。ユーザーがメニューまたはメニュー項目を選択すると、フレームの中心のテキストエリアにその選択項目が表示される。メニューは前述と同じ。

練習問題

2．1つのボタンを持つフレームを作成するアプリケーションを作成せよ。ボタンを押すたびに、現在の日付を表示するモーダルダイアログボックスが表示される。