情報処理技法
(Javaプログラミング)2

第11回
ちょっと高度なGUIの部品, レイアウトマネージャ

人間科学科コミュニケーション専攻
白銀 純子

第11回の内容

- ちょっと高度なGUIの部品
  - メニュー
  - リストボックス
- レイアウトマネージャ

前回の出席課題の解答

- コンビニの商品管理プログラムを考えるとき、下記のクラスを作る場合に、どのような継承関係にすれば良いか、考えて答えなさい。
  - 商品, パン, アイス, ドリンク, コーヒー, ジュース, お茶

解答
- 親クラス: 商品, 子クラス: パン, アイス, ドリンク
- 親クラス: ドリンク, 子クラス: コーヒー, ジュース, お茶

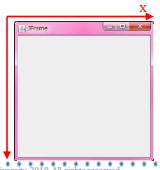
前回の復習

GUIとは

- GUI: Graphical User Interface
  - 人間がソフトウェアとのやりとりの接点を視覚的に表現したもの
    - ボタンや入力フィールドでソフトウェアを操作
    - ソフトウェアからの処理結果の提示
    - etc.
  - Graphical: 視覚的
  - User: 人間の利用者
  - Interface: ものとものとの接点(人間とソフトウェアとの間に限らず)

ウィンドウの作り方の基本

1. ウィンドウ内の構成(ボタンや入力フィールドなどの配置を決める)
2. 「JFrame」というウィンドウの土台を作る
3. 土台の上に、ボタンなどのGUIの部品を配置していく
   - 配置は、座標で指定する(前回は)
   - x座標は右方向
   - y座標は下方向(グラフの座標軸とは逆)



部品の作り方

- 部品は、一種のデータ型(クラス)
  - 部品: ボタンや入力フィールドなどの1つ1つのGUIの要素
  - 1つ1つの部品の変数を宣言する
  - 部品の作成 = 部品のオブジェクト作成
    - *部品名 変数名*=new *部品名*();
  - 作成した部品に、いろいろな情報を設定する(部品を置く位置や大きさ、部品の見た目上の名前など)
  - 作成した部品をJFrameに貼り付ける

※JFrameも部品の1つ

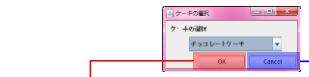
GUIプログラムの基本形(1)

```
import java.io.*;
import java.lang.*;
import javax.swing.*;

public class クラス名 extends JFrame {
  部品の変数の宣言

  public クラス名() {
   getContentPane().setLayout(null);
     JFrame以外の部品を作成したり情報を設定する領域
   setTitle(タイトルバーに表示する名前);
   setSize(横の長さ,縦の長さ);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setVisible(true);
  }
  public static void main(String[] args) {
    new クラス名();
  }
}
```


部品の扱い方

- 部品に対しては、様々な操作をすることができる
  - 見た目の名前をつける
  - 貼り付ける座標や大きさを決める
  - etc.
- 部品に対して様々な操作を行うために、メソッドが用意されている
- ウィンドウ中に1つ部品を作るにあたり、その部品のオブジェクトを1つ作成する
  - 部品はJavaでクラスとしてあらかじめ提供されている
- 「*部品の変数名.メソッド名(引数)*」で部品に対する設定を行う

1部品につき1オブジェクト




```
JButton okBut;
....
okBut = new JButton();
okBut.setText("OK");
...
```

```
JButton cancelBut;
....
cancelBut = new JButton();
cancelBut.setText("Cancel");
...
```

1つの部品につき、1つ変数を宣言をしてオブジェクトを作成、設定が必要

部品作成・ウィンドウ表示処理の順序

```
public class CakeWindow extends JFrame {
   JLabel label;
   JComboBox combo;
   JButton ok, cancel;

   public CakeWindow() {
     getContentPane().setLayout(null);

     label = new JLabel(); /* JLabelの作成 */
     label.setText("ケーキの選択");
     label.setBounds(10, 5, 280, 25);
     getContentPane().add(label);

     .... 略 ....

     setTitle("ケーキの選択");
     setSize(300, 140);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     setVisible(true);
   }
```

個々の部品作成とウィンドウへの貼り付け

ウィンドウの表示処理

- ウィンドウ上への部品の配置は、コンピュータの処理では、部品の絵を描くようなもの
- ウィンドウを表示したときに、描画処理

ウィンドウを表示後に部品を配置しても、描画処理が行われない

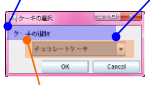
配置した部品が表示されない

「setVisible(true);」の処理は一番最後に!

座標計算(1)

このJLabelの左上隅の座標は x座標: 10, y座標: 5

幅: 280, 高さ: 25なので、このJLabelの右下端の座標はx座標: 290, y座標: 30



上のJLabelが、y座標が30まであるので、このJComboBoxの左上端のy座標は30以上にしないと、上のJLabelと重なる

座標計算(2)

左のJButtonが、x座標が170まであるので、このJButtonの左上端のx座標は170以上にしないと、左のJButtonと重なる

幅: 100, 高さ25なので、このJButtonの右下端の座標はx座標: 170, y座標: 100

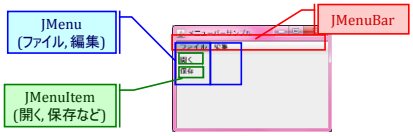
このJButtonの左上隅の座標はx座標: 70, y座標; 75

部品の座標は、上下左右に隣接する部品の四隅の座標を計算した上で決定すること(でないと、隣接する部品に重なってしまうので注意)

ちょっと高度なGUIの部品

メニューバー(1)

- クリックすることでメニューが表示され、選択すると何か処理を行うための部品
- JMenuBar, JMenu, JMenuItemという3種類の部品で構成



メニューバー(2)

- メニューバー作成の手順:
  1. JMenuBarを作成し、フレームに貼り付ける
     - 1つのフレームにJMenuBarは1つしかつけられない
  2. JMenuを作成し、JMenuBarに貼り付ける
     - JMenuはいくつでもよい
     - プログラム中に書いた順に、メニューバーの左から並んで貼りつけられる
  3. JMenuItemを作成し、JMenuに貼り付ける
     - JMenuItemはいくつでもよい
     - プログラム中に書いた順に、上から並んで貼りつけられる

メニューバー～1. JMenuBarを作る～

- JMenuBarの作り方は、ボタンなど他の部品と同じ
- JMenuBarのフレームへの貼り付け方:
  setJMenuBar(*JMenuBarの変数名*)

例

```
public class MenuSample extends JFrame {
  JMenuBar bar;
  public MenuSample() {
    ……
    bar = new JMenuBar();
    setJMenuBar(bar);
    ……
  }
}
```


メニューバー～2. JMenuを作成する～

- JMenuの作り方は、ボタンなどの他の部品と同じ
- JMenuの文字列の設定: setText("*文字列*")
- JMenuBarへのJMenuの貼り付け方:
  JMenuBarの変数名.add(*JMenuの変数名*)

例

```
public class MenuSample extends JFrame {
  JMenuBar bar;
  JMenu file, edit;
  public MenuSample() {
    ……
    file = new JMenu();
    file.setText("ファイル");
    bar.add(file);
    ……
  }
}
```

メニューバー～3. JMenuItemを作成する～

- JMenuItemの作り方は、ボタン等の他の部品と同じ
- JMenuItemの文字列の設定:
  JMenuItemの変数名.setText("*文字列*")
- JMenuへのJMenuItemの貼り付け方:
  JMenuの変数名.add(*JMenuItemの変数名*)

例
```
public class MenuSample extends JFrame {
    JMenu file;
    JMenuItem open, save;
    public MenuSample() {
        ……
        open = new JMenuItem();
        open.setText("開く");
        file.add(open);
        ……
    }
}
```


リストボックス

- 項目を縦に並べるためのボックス

- 部品の名前: JList
- JListの作り方(JFrameへの貼り付け方は、他と同じ):
  1. JListで表示させる項目の一覧を、配列として用意する
     - 配列はString型
  2. JListのコンストラクタの引数として、項目一覧の配列を与える

例
```
public class ListSample extends JFrame {
    JList colorList;
    public ListSample() {
        String[] colors = new String[100];
        colors[0] = "red";
        colors[1] = " blue";
        ……
        colorList = new JList(colors);
        ……
    }
}
```


スクロールバー(1)

- 部品の名前: JScrollPane(スクロールバーを持った敷物)

- スクロールバーのつけ方:
  - スクロールバーをつけたい部品(JList, JTextAreaなど)を作成する
  - JScrollPaneのコンストラクタの引数として、スクロールバーをつけたい部品を与える
  - JScrollPaneをフレームに貼り付ける

※JList, JTextAreaなどの、スクロールバーをつけたい部品は、JScrollPaneに貼り付けるので、フレームには貼り付けない

スクロールバー(2)

例
```
public class ListSample extends JFrame {
    JList colorList;
    JScrollPane scroll;
    public ListSample() {
        ……
        String[] colors = new String[100];
        ……
        colorList = new JList(colors);
        scroll = new JScrollPane(colorList);
        scroll.setBounds(5, 5, 200, 100);
        getContentPane().add(scroll)
        ……
    }
}
```

位置と大きさを設定し、フレームに貼り付けるのは、JScrollPaneのみ

レイアウトマネージャ

レイアウトマネージャって?

- GUIの部品の位置や大きさを管理してくれる機能
  - 部品の位置や大きさを決める枠組みを作り、それぞれの部品がどの枠に収まるかを指定することで位置や大きさを決定する

    JFrameやJPanelなどの部品を配置するための部品に対して、枠を設定する

    ※JFrameに関しては、大きさの設定は必要
  - 枠の並べ方でいくつか種類あり
- 利点: 部品の座標や大きさを計算しなくてよい
- 欠点: 慣れなければ使いこなすのが難しく、細かい設定はできない

レイアウトマネージャの設定方法

レイアウトマネージャを使うためのパッケージ

```
import java.awt.*;
import javax.swing.*;

public class クラス名 extends JFrame {
 public クラス名() {
  getContentPane().setLayout( レイアウトマネージャのオブジェクト );

  ……

  setTitle( タイトルバーに表示する名前 );
  setSize( 横の長さ, 縦の長さ );
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setVisible(true);
 }
```

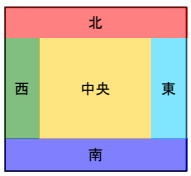
「null」と書くと、レイアウトマネージャを使わないこととなり、部品の座標と大きさを指定する

よく使われるレイアウトマネージャ

- BorderLayout
- GridLayout
- FlowLayout

BorderLayout(1)

- 部品を東、西、南、北、中央の5つの領域に配置するレイアウトマネージャ



BorderLayout(2)

- 部品を配置する枠を、東, 西, 北, 南, 中央で指定する
  - 部品の大きさは、レイアウトマネージャが決定する
  - 通常、中央が一番大きい
- ただし、BorderLayoutで配置する部品は5つより少ない、つまり
  - WestとCenterに配置する部品だけ
  - North, Center, Southに配置する部品だけ

などでもかまわない

BorderLayout(使い方)

- JFrameやJPanelに対してレイアウトマネージャを設定:
  setLayout(new BorderLayout())
- JFrameやJPanelの上に部品置くとき: add(*部品の変数名*, *位置*)

位置は、
- 東: BorderLayout.EAST
- 西: BorderLayout.WEST
- 南: BorderLayout.SOUTH
- 北: BorderLayout.NORTH
- 中央: BorderLayout.CENTER

で指定

BorderLayout(例)


```
getContentPane().setLayout(new BorderLayout());

b1 = new JButton("北");
getContentPane().add(b1, BorderLayout.NORTH);

b2 = new JButton("南");
getContentPane().add(b2, BorderLayout.SOUTH);

b3 = new JButton("東");
getContentPane().add(b3, BorderLayout.EAST);

b4 = new JButton("西");
getContentPane().add(b4, BorderLayout.WEST);

b5 = new JButton("中央");
getContentPane().add(b5, BorderLayout.CENTER);

setSize(300, 300);
```

※b1～b5はJButtonの変数

GridLayout

- 部品を縦横に配置するレイアウトマネージャ
  - 部品はすべて同じ大きさで配置される
- 配置する部品の縦と横の数を指定

縦4つ、横2つ配置する場合

1	2
3	4
5	6
7	8

※番号は、配置していく順序
プログラムの上に書かれているものから順に配置される)

GridLayout(使い方)

- JFrameやJPanelに対してレイアウトマネージャを設定:
  setLayout(new GridLayout(*縦の数, 横の数*))

縦に並べる部品の数と横に並べる
部品の数を引数(int型)として書く

- JFrameやJPanelの上に部品置くとき: add(*部品の変数名*)

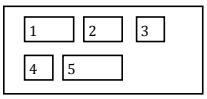
GridLayout(例)

```
getContentPane().setLayout(new GridLayout(2,3));

g1 = new JButton("1");
getContentPane().add(g1);
g2 = new JButton("2");
getContentPane().add(g2);
g3 = new JButton("3");
getContentPane().add(g3);
g4 = new JButton("4");
getContentPane().add(g4);
g5 = new JButton("5");
getContentPane().add(g5);
g6 = new JButton("6");
getContentPane().add(g6);

setSize(300,150);
```

※g1～g6はJButtonの変数

FlowLayout

- 左から右に向かって部品を配置するレイアウトマネージャ
  - 部品の大きさは、部品のラベル名などに応じてレイアウトマネージャが決定する
- 右端まで部品が埋まったら、次の行に配置

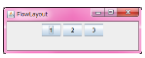


※番号は、配置していく順序
(プログラムの上に書かれているものから順に配置される)

FlowLayout(使い方)

- JFrameやJPanelに対してレイアウトマネージャを設定:
  setLayout(new FlowLayout())
- JFrameやJPanelの上に部品置くとき: add(*部品の変数名*)

FlowLayout(例)



```
getContentPane().setLayout(new FlowLayout());

fr1 = new JButton("1");
getContentPane().add(fr1);
fr2 = new JButton("2");
getContentPane().add(fr2);
fr3 = new JButton("3");
getContentPane().add(fr3);

setSize(300,100);
```

※fr1～fr3はJButtonの変数

ちょっと複雑な配置は?

- レイアウトマネージャは、決められた枠に部品を置くことしかできない
- 決められた枠だけでは配置できないような、ちょっと複雑な配置は?

➡ JPanelを利用する

JPanel

- GUIの部品を置くための敷物
- GUIの部品を置いてしまったJPanelは、1つの部品として扱うことができる

➡ 複数の部品をまとめて1つの部品として扱い、レイアウトマネージャの枠にはめることもできる

物を箱に入れて整理するとき、箱の中にさらに箱を入れるというイメージ

- JPanelの中にさらにJPanelを入れ込むことも可能
- JPanel自体にもレイアウトマネージャを設定し、GUIの部品を配置する

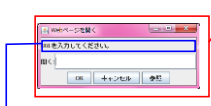
JPanelの使い方

- 「JPanel 変数名 = new JPanel();」でオブジェクトを作成
- 「JPanelの変数名.setLayout(レイアウトマネージャ);」で
  レイアウトマネージャを設定
- 「JPanelの変数名.add(部品の変数名);」で部品をJPanelに貼り付け

JPanel利用の基本形

```
JPanel 変数名;
……
変数名 = new JPanel();
変数名.setLayout(レイアウトマネージャ);
……
変数名.add(panelの上に配置する部品の変数名)
```


ちょっと複雑な配置の例(1)

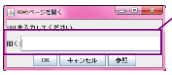


JFrame: GridLayout(縦3, 横1)

```
getContentPane().setLayout(new GridLayout(3, 1));

label = new JLabel();
label.setText("URLを入力してください。");
getContentPane().add(label);
```

JLabel: JFrameに直接貼り付け

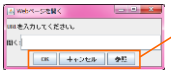
ちょっと複雑な配置の例(2)



- 「開く:」のJLabelとJTextFieldをJPanelに貼り付け
- JPanelをJFrameに貼り付け
  (JPanelはBorderLayout)

```
panel1 = new JPanel();
panel1.setLayout(new BorderLayout());
openLabel = new JLabel();
openLabel.setText("開く:");
panel1.add(openLabel, BorderLayout.WEST);

url = new JTextField();
panel1.add(url, BorderLayout.CENTER);
getContentPane().add(panel1);
```


ちょっと複雑な配置の例(3)



- JButtonをJPanelに貼り付け
- JPanelをJFrameに貼り付け
  (JPanelはFlowLayout)

```
panel2 = new JPanel();
panel2.setLayout(new FlowLayout());

ok = new JButton();
ok.setText("OK");
panel2.add(ok);
cancel = new JButton();
cancel.setText("キャンセル");
panel2.add(cancel);
ref = new JButton();
ref.setText("参照");
panel2.add(ref);
getContentPane().add(panel2);

setSize(350,140);
```