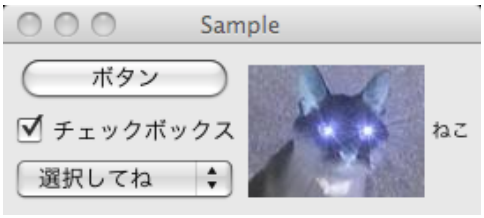
- GUI クラス
- レイアウト方式
- Swing コンポーネント

★8　GUI クラス

★8.1　GUI クラス (p.111) & Swing 入門 (p.115)

★8.1.1　GUI を備えたプログラムの構成



GUI（グラフィカルユーザインターフェース）とは，右図のように，グラフィックスを用いた表示に対してユーザがマウスなどのポインティングデバイスを用いた直感的な操作（ボタンを押したりメニューを選択したり）を行うことでプログラムとやりとりできるようにした操作環境のことをいう．

教科書に記されているように，GUI を備えたプログラムは次のような動きをする必要がある．

- ボタンやメニューなどの部品（**コンポーネント**と呼ぶ）をウィンドウ上に配置して表示する
- マウスやキーボードを通したユーザからの入力を感知して，それに応じたふるまいをする（**イベント**を処理するという）

今回は，このうちの一つ目について学ぶ．二つ目については，次々回以降に学ぶ予定である．

★8.1.2　Java の GUI クラス

Java で GUI を備えたプログラムを開発するには，標準で提供されている **AWT パッケージ**や **Swing パッケージ**を利用するのが一般的である．これらのパッケージには，ボタンやメニューなどのコンポーネントを表示するためのクラスや，マウスクリックやキー入力などのイベントを処理するためのクラスなどが含まれている．

- AWT(Abstract Window Toolkit): 基本的なグラフィックスやコンポーネントの描画や配置，イベント処理などを行うためのクラスの集まり
- Swing: AWT を拡張して作られた GUI 用のクラスの集まり．

この授業でもこれらを利用する．

★8.1.3　Swing のコンポーネント

Swing のコンポーネントを用いる例として，`Hello.java` というサンプルプログラム（ソースは最後の頁にあります）を説明する．実際にそのソースを見る前に，まずはその実行結果をよく観察しよう．また，このウィンドウがどのようなコンポーネントから成り立っているかを先にみておこう (☆ 1)．

☆ 1) トップレベルコンテナのクラスは `JFrame` 以外にもある (p.113 参照)．コンテナのクラスには `JPanel` 以外にもある (p.115 上参照)．コンテナの中にコンテナをおくこともできる．

Hello クラスの実行画面

観察
・タイトルバー（Hello と書いてある）やアイコン化／ウィンドウ最大化などのボタンのついたウィンドウが表示されている（注）
・「こんにちは」と表示されている．
・ 二つのボタンがあり，押している間だけ見た目が変わる．

注: ウインドウそのものの画面上での配置，タイトルバーやウインドウ枠の見た目を制御するのは，個々の GUI プログラムの役割ではなく，ウインドウマネージャと呼ばれるソフトウェアの役割である

コンポーネントの構成
(1) このプログラムの表示領域（いわば額縁）となる**トップレベルコンテナ**（JFrame クラスのオブジェクト）を用意
(2) その中に，コンポーネントを配置するための**コンテナ**（JPanel クラスのオブジェクト）をおく
(3) その中に，3 つのコンポーネント（ラベル (JLabel) が 1 つ，ボタン (JButton) が 2 つ）を配置

★8.2 レイアウト方式 (p.119)

コンポーネントの大きさや位置をプログラム作成者が全て指定するのは大変なので，AWT&Swing では，コンポーネントの大まかな配置方針を指定したらあとは自動的に各コンポーネントの大きさや位置を制御してくれる仕組みが用意されている．それが**レイアウトマネージャ**である．さきほどのソースの該当箇所を抜き出して，その使い方を説明する．

Hello.java の一部

```
 8     public Hello(){
 :
12         setLayout(new BorderLayout());    //パネルの配置方式を設定


13         add(label, BorderLayout.NORTH);  //パネルにラベルを置く
14         add(b1, BorderLayout.CENTER);    //パネルにボタンを置く
15         add(b2, BorderLayout.EAST);      //パネルにボタンを置く
16     }
```

レイアウトマネージャには配置方式の違いによって様々なものがあるが，教科書では，FlowLayout, BorderLayout, GridLayout, BoxLayout の 4 つを説明している．これらの詳しい使い方については，教科書 p.119–125 参照．

ところで，Hello.java の 22 行目では，BorderLayout を利用して Hello クラスの（したがって JPanel の）コンポーネントをフレームに add している．それなのに，このフレームに対する setLayout メソッドが見当たらない．それでも正常に動作するのは，JFrame のデフォルトのレイアウトマネージャが BorderLayout だからである．もしもこのフレームに対して BorderLayout 以外の配置方式を用いたければ，ちゃんと setLayout する必要がある．一方，JPanel のデフォルトのレイアウトマネージャは FlowLayout である．

Q1. Hello.java の 12 行目を削除し，13–15 行目を add(label); add(b1); add(b2); に変更するとどうなるか．

★ 8.3 Swing コンポーネント (p.126)

Swing のコンポーネントのごく一部を紹介する．ここに出てくるクラスの詳しい使い方は教科書 p.126–131 参照．それ以外のものについては，Java の API を参照（この科目のウェブページからたどれる）．

★ 8.3.1 ラベル: JLabel (p.126)

文字列やアイコン画像を表示する．ボタンなどと違って入力は受け付けない．

★ 8.3.2 ボタン抽象クラス: AbstractButton (p.127)

ボタンやチェックボックスなどの，マウスクリックによるユーザインタフェースを実現するコンポーネントに共通の機能を定義したクラス．以下で説明している JButton, JCheckBox や，JRadioButton などはこのクラスを継承している（第 13 章にでてくるイベント処理の際にこのクラスの機能を利用する）．

Q2. Java API の中から JRadioButton のコンストラクタの説明を探しなさい．

★ 8.3.3 JButton (p.127)

文字列とアイコンを貼付けられるボタン．

★ 8.3.4 JCheckBox (p.128)

選択されている (true)，いない (false)，の 2 状態をもつボタン．

JCheckBoxExample.java の一部（p.128 のものとはちょっと違う）

```
 :
 4   public class JCheckBoxExample extends JPanel{
 5     JCheckBox cbox1, cbox2;

 6     public JCheckBoxExample(){
 7        setLayout(new GridLayout(2, 0));

 8        cbox1 = new JCheckBox("チェックボックス");

 9        Icon icon1 = new ImageIcon("blackuni.jpg");
10        Icon icon2 = new ImageIcon("whiteuni.jpg");

11        cbox2 = new JCheckBox("ねこ", icon1);
12        cbox2.setSelectedIcon(icon2);

13        add(cbox1);
14        add(cbox2);
15     }
 :   (main メソッドの部分は Hello.java とほとんど共通である．p.128 参照)
```

Q3. 8 行目を `cbox1 = new JCheckBox("チェックボックス", true);` とするとどうなるか．

JCheckBoxExample

JComboBoxExample

★ 8.3.5 JComboBox (p.129)

複数の項目の中から一つを選択するのに用いる.

JComboBoxExample.java の一部 (p.129 のものとはちょっと違う)

```
 :
 4  public class JComboBoxExample extends JPanel{
 5      JCheckBox cbox1, cbox2;
 6      JComboBox<String> select;

 7      public JComboBoxExample(){
 8          String[] list = {"寝る", "昼寝する", "ふて寝する", "暴れる",};
 9          select = new JComboBox<String>(list);
10          add(select);
11      }
 :  (main メソッドの部分はやはり Hello.java とほとんど共通である. p.129 参照)
```

注意: JComboBox の <hoge> という書き方は,「ジェネリクス」という機能を使ったものである. hoge の部分で, リストに並べるオブジェクトのクラスを指定する. 上記の例では String クラスのオブジェクトを並べている.「ジェネリクス」については, 教科書第 17 章参照.

宿題？

- 授業中に解説されなかった Q を解いてみよう.
- 次回は第 11 章のつづきと第 12 章です. あらかじめ読んでおくこと. 例題のプログラムを作成し実行してみておくこと.

最初のプログラム Hello.java

Hello.java

```
 1  import java.awt.*;
 2  import javax.swing.*;
 3
 4  public class Hello extends JPanel{



 5      JLabel label; //ラベル用の変数
 6      JButton b1, b2;//ボタン用の変数
 7

 8      public Hello(){


 9          label = new JLabel("こんにちは");  //JLabel オブジェクトの生成
10          b1 = new JButton("ボタン１");  //JButton オブジェクトの生成
11          b2 = new JButton("ボタン２");  //JButton オブジェクトの生成
12          setLayout(new BorderLayout());   //パネルの配置方式を設定


13          add(label, BorderLayout.NORTH);  //パネルにラベルを置く
14          add(b1, BorderLayout.CENTER);    //パネルにボタンを置く
15          add(b2, BorderLayout.EAST);      //パネルにボタンを置く
16      }



17      public static void main(String[] args){


18          JFrame frame = new JFrame("Hello");//JFrame オブジェクトを生成
19          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20                                  //ウィンドウを閉じるとプログラムが終了


21          Hello h = new Hello();  //Hello のオブジェクトを生成
22          frame.add(h, BorderLayout.CENTER);

23          frame.pack();               //フレームを必要最小の大きさにする
24          frame.setVisible(true);     //フレームを画面に見えるようにする
25      }

26  }
```