第1章　イベント処理と GUI部品

アプレットのようなグラフィカルなユーザインタフェース（GUI）を持つプログラムは、ユーザがマウスのボタンを押した時、キーボードのキーを押した時などに何らかの反応を示さなければいけないことが多い。この節では、このような何らかの _____に反応するプログラムの書き方について学習する。

また、GUI 部品のボタンや、ユーザがデータを入力するためのテキストフィールドをユーザインタフェースに付け加える方法についても学ぶ。

1.1　イベント処理

ユーザがマウスボタンをクリックした、キーボードのキーを押した、などのイベントに対応するためには、_____と呼ばれるメソッドを定義する必要がある。

例題 1.1.1 マウスボタン

ファイル MouseTest.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;  /* 1 */

public class MouseTest extends JApplet implements MouseListener /* 2 */ {
  int x=50, y=20;

  @Override
  public void init() {
    addMouseListener(this); /* 3 */
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g); /* 4 */
    g.drawString("HELLO WORLD!", x, y);
  }

  public void mouseClicked(MouseEvent e) { /* 5 */
    x = e.getX();  y = e.getY();
    repaint();
    return;
  }
  public void mousePressed(MouseEvent e) {}  /* 6 */
  public void mouseReleased(MouseEvent e) {} /* 6 */
  public void mouseEntered(MouseEvent e) {}  /* 6 */
  public void mouseExited(MouseEvent e) {}   /* 6 */
}
```

このプログラムは、文字列を表示し、マウスがクリックされると、その場所に文字列を移動する。いくつか注目すべき点がある。

- まずイベントを扱うために *java.awt.event.** を *import* している。(*/* 1 */*)
  イベントを扱うプログラムは大抵この *import* 文が必要になる。
- _____というマウスボタンのクリックに対応するイベントハンドラを定義している。(*/* 5 */*) *mouseClicked* メソッドは *MouseEvent* 型の引数を受け取る。
- さらにこのクラスが、*mouseClicked* というメソッドを持っていることを示すために、*MouseListener* という _____を *implement* していることを宣言している。(*/* 2 */*)
- *MouseListener* インタフェースの他のメソッド(*mousePressed*, *mouseReleased*, *mouseEntered*, *mouseExited*)は何もしないメソッドとして、いちおう定義しておく。(*/* 6 */*)
- *init* メソッドで、*addMouseListener(this)* を呼んで、マウスのイベントを、*mouseClicked* メソッドに結び付けている。(*/* 3 */*)(*this* は一般にメソッドを実行中のオブジェクト自身を指す。*this* はアプレットオブジェクトである。ここでは実質的には、*mouseClicked* メソッドを指す。)
- このクラスの *paint* メソッドは、その中で *super.paint(g)* を呼び出している。 *super.*〜はスーパークラスで定義されているメソッドを呼び出すための書き方である。このプログラムの場合、背景を再描画している。(*/* 4 */*)

このクラスは、文字列を表示する位置をインスタンス変数 *x, y* として保持している。*mouseClicked* メソッドは、これらのインスタンス変数を、マウスの押された位置にしたがって変更する。マウスの押された位置を知るには、*MouseEvent* クラスの *getX, getY* というメソッドを使う。そのあと *repaint()* を呼び出して再描画を要求する。*repaint* は、*JApplet* クラスで定義済のメソッドで、その中で *paint* メソッドを呼び出す、

インタフェース　インタフェース（interface）というのは、C++などにはない、Java 特有のメカニズムである。一言でいえば、_____のことである。Java は多重継承（複数のスーパークラスを継承すること）を許さない代わりに、このインタフェースという仕組みを提供している。

MouseListener インタフェースの定義（ただし一部簡略化）

```
public interface MouseListener {
  public void mouseClicked(MouseEvent e);
  public void mousePressed(MouseEvent e);
  public void mouseReleased(MouseEvent e);
  public void mouseEntered(MouseEvent e);
  public void mouseExited(MouseEvent e);
}
```

例えば、インタフェース MouseListener の定義は、上のように MouseClicked などのメソッドの引数、戻り値の型を宣言している。(このインタフェースの定義は、もともと用意されているので、自分でする必要はない。) クラスと異なり、このメソッドの具体的な定義はインタフェース内のどこにもない。

あるクラスが、あるインタフェースを実装していることを宣言するためには _____ というキーワードを用いる。例えば、addMouseListener の引数は MouseListener インタフェースを実装していなければならない。

問 1.1.2 *repaint()* がなければ、*MouseTest.java* はどのような振舞いをするか?

問 1.1.3 *Othello.java* を改良して、マウスで指示をすると石を置けるようにせよ。

次の例題はマウスではなく、キーボードからのイベントを扱う。

例題 1.1.4 （参考） キーボード
U(p), D(own) の各キーが押されると文字列が移動する。
ファイル KeyTest.java

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

public class KeyTest extends JApplet implements KeyListener {
  int x=50, y=20;

  @Override
  public void init() {
    addKeyListener(this);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString("HELLO WORLD!", x, y);
  }

  public void keyPressed(KeyEvent e) {
    int k = e.getKeyCode();
    if (k=='u' || k=='U') {
      y-=10;
    } else if (k=='d' || k=='D') {
      y+=10;
    }
    repaint();
  }
  public void keyReleased(KeyEvent e) {}
  public void keyTyped(KeyEvent e) {}
}
```

メソッド名やクラス名の *Mouse* が *Key* に変わるだけで、大部分は *MouseTest.java* に似ている。*KeyListner* インタフェースは *keyPressed, keyReleased, keyTyped* の 3 つのメソッドからなる。このうちの *keyPressed* がキーが押されたときに対応するイベントハンドラである。実際に押されたキーに対応する文字を知るには *KeyEvent* クラスの、*getKeyCode* というメソッドを用いる。

問 1.1.5 *KeyTest.java* を拡張して、上下・左右・斜めにも文字列を動かせるようにせよ。
さらにカーソルキーを利用する方法を調べよ。*KeyTest.java* を改良して、カーソルキーで文字を動かせるようにせよ。

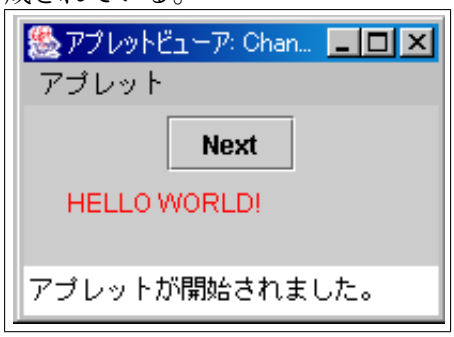
さらに、*Shift, Ctrl, Alt* キーなどの、モディファイヤキーの検出方法を調べよ。*KeyTest.java* を改良して、*Shift* キーを押しながらキーを押すと、動きが大きくなるようにせよ。
参考: *(JDKDIR)/docs/ja/api/java.awt.event.KeyEvent.html*

問 1.1.6 まず正多角形を描画し、マウスボタンを押すと、その場所から多角形の各頂点へ直線を描画するプログラムを書け。

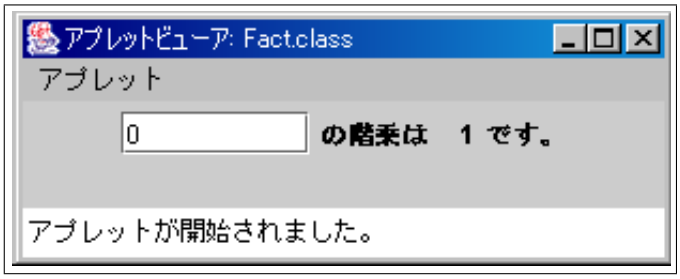
問 1.1.7 マウスを押した場所を順に結んで、折れ線を描画するプログラムを書け。
アプレットを最小化したり、他のウインドウで隠したりしても、再描画のときに折れ線もちゃんと再描画されるようにせよ。また、*java.util.ArrayList* クラスを使用して、頂点がいくつに増えても対応できるようにすること。

1.2　GUI 部品

大抵の GUI は、ボタン、テキストフィールド、ラベル、チェックボックスなどの GUI 部品から構成されている。



ボタンの例（ChangeColor.java）



テキストフィールドの例（Fact.java）

プログラムは、ボタンが押された、テキストフィールドが書き換えられた、などのイベントにも反応しなければならない。このようなイベントに対して、mouseClicked や keyPressed のような低レベルなイベントハンドラで対応するのは、不可能ではないにしても困難である。そこで GUI 部品に対するイベントには _____ というイベントハンドラが用意されている。

このイベントハンドラは _____ 型の引数を受け取る。ActionEvent 型の引数は、イベントが発生した場所・時間に関する情報などを持っていて、この引数から、どの部品でイベントが発生したかを特定することができる。

例題 1.2.1 ボタンを押すとテキストの色が変わる。

ファイル ChangeColor.java

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

public class ChangeColor extends JApplet implements ActionListener {
  Color[] cs = {Color.red, Color.blue, Color.green, Color.orange};
  String str = "Hello World!";
  int i=0;

  @Override
  public void init() {
    JButton b = new JButton("Next");
    b.addActionListener(this); /* 1 */
    setLayout(new FlowLayout()); /* 2 */
    add(b); /* 3 */
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.setColor(cs[i]);
    g.drawString("HELLO WORLD!", 20, 50);
  }

  public void actionPerformed(ActionEvent e) {
    i=(i+1)%cs.length;
    repaint();
  }
}
```

新しいボタンを作成するのに、*JButton* クラスのコンストラクタを用いる。このコンストラクタは、ボタンに表示する文字列を引数に取る。生成した部品をアプレットの画面に加えるには ____というメソッドを用いる。(*/* 3 */*)

その直前の行 (*/* 2 */*) では、*add* された部品を配置する方法を指定している。ここでは *FlowLayout* という単純な配置方法を選択している。

actionPerformed は _____インタフェースのメソッドである。このインタフェースには、他のメソッドはない。ボタン *b* が押されたときに *actionPerformed* が呼ばれるように、ボタン *b* の *addActionListener* メソッドを読んでいることに注意する。(*/* 1 */*)

この例題では、*GUI* 部品を *1* つしか使用していないので、 *actionPerformed* メソッドの引数 (*e*) は調べる必要がない。 *actionPerformed* が呼び出される度に、インスタンス変数 *i* の値が変更される。(*GUI* 部品を *2* つ以上使う例はあとで紹介する。)

例題 1.2.2 テキストフィールドに数字を入力して、その階乗を計算する。

ファイル Fact.java

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

public class Fact extends JApplet implements ActionListener {
  JTextField input;
  JLabel output;

  @Override
  public void init() {
    input=new JTextField("0", 8);
    output=new JLabel("   1");
    input.addActionListener(this);
    setLayout(new FlowLayout());
    add(input);  add(new JLabel("の階乗は"));
    add(output); add(new JLabel("です。"));
  }

  static int fact(int n) {
    int r = 1;
    for (; n>0; n--) {
      r *= n;
    }
    return r;
  }

  public void actionPerformed(ActionEvent e) {
    int n = Integer.parseInt(input.getText());
    output.setText("  "+fact(n));
  }
}
```

JTextField のコンストラクタは、最初に表示する文字列（*String* 型）と、表示できる文字数（*int* 型）の *2* つの引数を取る。 *JLabel* は単に文字を表示するためだけの *GUI* 部品である。

ユーザがテキストフィールドに文字を書き込み、リターンキーを押した時点でイベントが発生する。テキストフィールドの場合もボタンと同じく *actionPerformed* メソッドで処理する。入力された文字列は *actionPerformed* の中で *input*（*JTextField* クラス）の *getText* メソッドを使って知ることができる。

このあと output（JLabel クラス）の setText というメソッドを呼び出して、ラベルに表示されている文字列を変更している。

例題 1.2.3 ボタン *2* つを使ってテキストを左右に移動する。

ファイル UpDownButton.java

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

public class UpDownButton extends JApplet implements ActionListener {
  String str = "Hello World!";
  int x=20;
  JButton left, right;

  @Override
  public void init() {
    left  = new JButton("Left");
    right = new JButton("Right");
    left.addActionListener(this);
    right.addActionListener(this);
    setLayout(new FlowLayout());
    add(left); add(right);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString("HELLO WORLD!", x, 55);
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == left) { // Left が押された
      x-=10;
    }
    else if (e.getSource() == right) { // Right が押された
      x+=10;
    }
    repaint();
  }
}
```

このプログラムでは *GUI* 部品（ボタン）を *2* つ使用しているので、どのボタンが押されたかを *actionPerformed* メソッド中で調べる必要がある。そのために *ActionEvent* クラスの *getSource()* というメソッドを用いて、比較演算子（==）で比べることによって、イベントの起こったボタンを特定している。

問 1.2.4 摂氏の温度をテキストフィールドに入力して、これを華氏の温度に変換するアプレットを *Fact.java* にならって書け。

さらに、*2* つのテキストフィールドを用いて、摂氏と華氏の変換を双方向に行なえる（片方のテキストフィールドの値を変えると、もう片方のテキストフィールドの値が変わる）ようにせよ。

（参考）（華氏の温度）$= \dfrac{(\text{摂氏の温度}) \times 9}{5} + 32$

例えば摂氏 *0* 度は華氏で *32* 度、摂氏 *100* 度は *212* 度になる。

（参考） *String* 型を *double* 型（実数の型）に変換するには、＿＿＿＿＿＿＿＿＿＿＿＿というクラスメソッドを使う。

（円とドルの変換、センチメートルとインチの変換、ラジアンと度数法の変換など他の単位の変換

をするアプレットなどでも良い。）

一方、GUI 部品が多くなってきた時は、getSource() メソッドではなく、次の例のように内部クラス（innner class）を用いる方が効率が良い。

例題 1.2.5 *UpDownButton.java* を内部クラスを用いて書き換える

ファイル UpDownButton2.java

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

public class UpDownButton2 extends JApplet {
  String str = "Hello World!";
  int x=20;
  JButton left, right;

  public class LeftListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      x-=10;
      repaint();
    }
  }

  public class RightListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      x+=10;
      repaint();
    }
  }

  @Override
  public void init() {
    left  = new JButton("Left");
    right = new JButton("Right");
    left.addActionListener(new LeftListener());
    right.addActionListener(new RightListener());
    setLayout(new FlowLayout());
    add(left); add(right);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString("HELLO WORLD!", x, 55);
  }
}
```

Java ではクラスの中にクラスを定義することができる。（＿＿＿＿＿＿＿＿）これも関数の中に関数を定義できない *C* との大きな違いである。上の例は、この内部クラス（*LeftListener* と *RightListener*）を用いて、*ActionPerformed* メソッドを与えている。内部クラスの中では、その外側のクラスのメンバ（上の例の場合 *x*）やメソッドなど（上の例の場合 *repaint* メソッド）を参照することができる。

このように内部クラスを用いると、addActionListner の時に、コンポーネントとメソッドを関連づけることができるので、コンポーネントの数が多いときは getSource を用いるよりも効率が良い。

また、次の例のように、内部クラスに名前をつけずに（＿＿＿＿＿＿＿＿, ＿＿＿＿＿＿＿＿）定義することができる。名前のないクラスのオブジェクトは次のようにして作成する。

```
new スーパークラス名 (引数) {
  クラス本体の定義
}
```

スーパークラス名のところは ActionListener のようなインタフェース名でも良い。その場合は下の例のように引数はとらない。

例題 1.2.6 *UpDownButton.java* を無名クラス *(anonymous class)* を用いて書き換える、

ファイル UpDownButton3.java

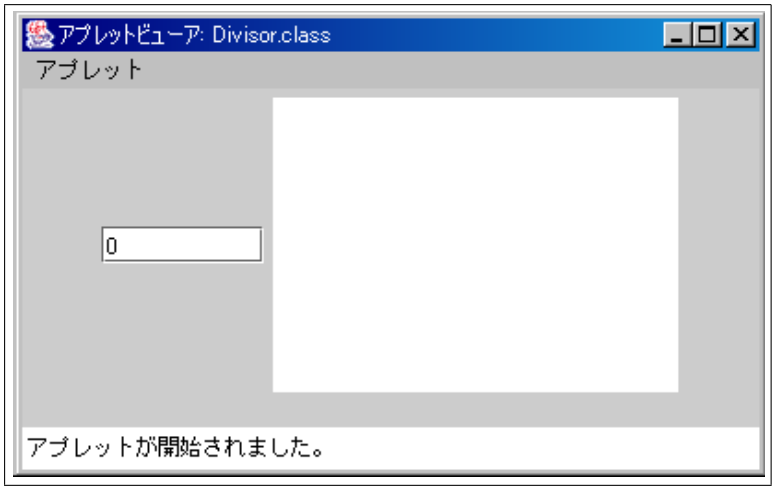
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class UpDownButton3 extends JApplet {
  String str = "Hello World!";
  int x=20;
  JButton left, right;

  @Override
  public void init() {
    left  = new JButton("Left");
    right = new JButton("Right");
    left.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        x-=10;
        repaint();
      }
    });
    right.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        x+=10;
        repaint();
      }
    });
    setLayout(new FlowLayout());
    add(left); add(right);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString("HELLO WORLD!", x, 55);
  }
}
```

例題 1.2.7 *JTextArea* — 約数の表示
整数を入力してもらって、その約数をすべて表示する。



　このアプレットのように多くのメッセージを表示する場合には＿＿＿＿＿＿という部品が便利である。

ファイル Divisor.java

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

public class Divisor extends JApplet implements ActionListener {
  JTextField input;
  JTextArea output;

  @Override
  public void init() {
    input = new JTextField("0", 8);
    output = new JTextArea(10, 20);
    input.addActionListener(this);
    setLayout(new FlowLayout());
    add(input); add(output);

  }

  public void actionPerformed(ActionEvent e) {
    int n = Integer.parseInt(input.getText());
    int i;

    for(i=1; i<=n; i++) {
      if (n%i==0) {
        output.append(i+"は "+n+"の約数です。¥n");
      }
    }
    output.append("以上¥n¥n");
  }
}
```

　JTextArea のコンストラクタの引数は、テキストエリアの＿＿＿と＿＿＿である。テキストエリアに文字列を表示するには、＿＿＿＿＿というメソッドを *C* の *printf* のように用いることができる。

問 1.2.8 *JPanel*, *JCheckBox*, *JComboBox*, *JList*, *JTable*, *JTree* など、他の *GUI* 部品の使用法を調べよ。またこれらのクラスの部品を使ってプログラムを作れ。

問 1.2.9 これまで紹介したプログラムは、*FlowLayout* を用いていて、*GUI* 部品がどのように配置されるかについては無関心だった。部品を自分の好みの位置に配置する方法（〜*Layout* という名前のクラス）を調べよ。

キーワード イベント、イベントハンドラ、keyPressed メソッド, mouseClicked メソッド, actionPerformed メソッド, インタフェース（interface）, MouseListener インタフェース, KeyListener インタフェース, ActionListener インタフェース, this, MouseEvent クラス, KeyEvent クラス, ActionEvent クラス, add メソッド, JButton クラス, JLabel クラス, JTextField クラス, JTextArea クラス