1 目的

- Java Applet で簡単なプログラムを作る.
- オブジェクト指向の初歩を理解する.
- アップレット製作のための知識や技術面での準備をする.

2 原理

2.1 Java Applet とは

「アップレット (Applet)」には,「アプリケーション (Application)」よりも規模が小さいというニュアンスがある.

実際,Java Applet は,前回までに実験したような形のプログラムに比べると,ウェブ上で使用される場合が多いために規模が小さい.

アップレットを作るときには,中心てとなるクラスは必ず `java.applet.Applet` というクラスを継承しなければならない.これによって,最低限の機能などがそろいアップレットとして動作するようになる.また,実際に動作させるときには HTML ファイルのなかで,`<APPLET>`タグを使ってブラウザにアップレットを実行させるか,AppletViewer というプログラムを使用する.

2.2 GUI のつけかた

ボタンやテキストフィールドなど,ユーザーがプログラムを操作したり情報を入力するために必要な部品は,AWT(Abstract Window Toolkit) というクラスの集まり (クラスライブラリ) になっていて,`java.awt` 以下にある.この様にライブラリになっているものを使う場合は `import` 文を使い,プログラムのその他の部分では `java.awt` や `java.appket` などの部分を省略して書く.

AWT のほかにも Swing など,ウインドウ・プログラムのための色々なライブラリがあるので,これらを利用すれば比較的簡単に GUI のあるプログラムが製作できる.

2.3 イベント・ドリブン (event driven) について

GUI を持つプログラムでは，処理の流れの大部分はユーザーがそのときした操作によって決まっていく．これに対応するために，「イベント (event)」という概念を使う．

「マウスが動いた」や「キーボードが押された」,「値が変更された」など，あるクラスに対して操作などが加えられることを，そのクラスにイベントが発生するという．イベントとそれに応じた処理だけをプログラムに書き，イベントを起こす順番はある程度ユーザーに任せてしまう，この様なやり方を「イベント・ドリブン」という．

Java では，`Listener`(リスナー) という種類のインターフェイス (`interface`) を，イベントが起きたときにメッセージを受け取りたいクラスに `implements` しておき，メッセージを出す側からメッセージが出るように `add()` メソッドで設定し，出す側に受け取る側のクラスを `add<イベントの種類>Listener()` メソッドで登録しておけば，イベントが発生すると受け取る側のクラスにあるイベントに応じた名前のメソッドが実行される．つまり，プログラマーは，イベントに対応したメソッドに処理を書きこんでおけば良い．

3 実験内容

実験では次の 3 個のプログラムを実際に入力し実行した．各プログラムのプログラムリストと実行結果を添付する．なお，各リストの冒頭の`//<APPLET… ></APPLET>`の部分は AppletViewer を使用する場合に必要となるものである．

3.1 Graphic.java

```
1  //<applet code="Graphic.class" width=900 height=700></applet>
2  import java.applet.Applet;
3  import java.awt.*;
4
5  public class Graphic extends Applet{
6  public void paint(Graphics g){
7  setBackground(Color.yellow);//背景は黄色。
8  drwFigures(g);
9  testFonts(g);
10 Colors(g);
11 Images(g);
12 }
13
14 private void drwFigures(Graphics g){
15 int xl=40, wdl=150, htl=30;
16 g.drawLine(xl,10,xl+wdl,40);//線分 (xl,10)->(xl+wdl,40)
17 g.drawRect(xl,50,wdl,htl);
18 g.fillRect(xl,90,wdl,htl);
19 g.drawOval(xl,130,wdl,htl);
20 g.fillOval(xl,170,wdl,htl);
21 g.drawArc(xl,210,50,50,0,270);
22 g.fillArc(xl+100,210,50,50,0,270);
23 }
24
25 private void testFonts(Graphics g){
26 String s = "こんにちは、アップレット。Hello, Applet !!";
27 //
28 g.drawString(s, 210, 20);//デフォルト
29 //
30 g.setFont(new Font("Serif", Font.PLAIN, 24));
```

```
31  g.drawString(s, 210, 50);
32  //
33  g.setFont(new Font("ＨＧ教科書体", Font.BOLD, 36));
34  g.drawString(s, 210, 90);
35  //
36  g.setFont(new Font("ＨＧＰ創英角ＵＢ", Font.ITALIC, 48));
37  g.drawString(s, 210, 140);
38  }
39
40  private void Colors(Graphics g){
41  g.setFont(new Font("Serif", Font.PLAIN, 24));
42  g.setColor(Color.blue);
43  g.drawString("これは青。", 210, 180);
44  g.fillRect(360, 155, 200, 28);
45  //
46  g.setColor(new Color(0,255,0));
47  g.drawString("こっちは緑。", 210, 210);
48  g.fillRect(360, 185, 200, 28);
49  }
50
51  private void Images(Graphics g){
52  Image Img;
53  Img=getImage(getDocumentBase(),"jyuria1-mono.gif");
54  g.drawImage(Img, 230, 230, this);
55  }
56  }
```

3.2 Adev.java

```
1  //<applet code="Adev.class" width=700 height=280></applet>
2  import java.applet.Applet;
3  import java.awt.*;
4  import java.awt.event.*;
5
6  public class Adev extends Applet implements ActionListener{
7  TextField txt1, txt2;//テキストフィールド２つ。
8  Label lbl1, lbl2;//ラベル２つ。
9
10 public void init(){
11 setLayout(null);//レイアウト手動設定？
12 //
13 lbl1=new Label("素因数分解プログラム"); add(lbl1);
14 lbl1.setBounds(160, 10, 150, 20);
15 txt1=new TextField(20);
16 txt1.addActionListener(this); add(txt1);
17 txt1.setBounds(45, 35, 150, 20);
18 lbl2=new Label("="); add(lbl2);
19 lbl2.setBounds(200, 40, 10, 10);
20 txt2=new TextField(20);
21 txt2.addActionListener(this); add(txt2);
22 txt2.setBounds(215, 35, 150, 20);
23 }
24
25
26 public void actionPerformed(ActionEvent e){
27 String input;
28 if(e.getSource()==txt1){//txt1 で Enter 入力
29 input=txt1.getText();
30 input=devide(input);
31 txt2.setText(input);
```

```
32  }else{
33  input=txt2.getText();
34  input=devide(input);
35  txt1.setText(input);
36  }
37  }
38
39  public static String devide(String s){
40  String ss="1";
41  int x=Integer.parseInt(s), o;
42  //
43  for(int k=2;k<=x;k++){
44  while(x%k==0){
45  x=x/k;
46  ss=ss+"×"+k;
47  }
48  }
49  return ss;
50  }
51  }
52
53
54
```

3.3 Pict.java

```
 1  //<applet code="Pict.class" width=600 height=400></applet>
 2  import java.applet.Applet;
 3  import java.awt.*;
 4  import java.awt.event.*;
 5
 6  public class Pict extends Applet
 7  implements MouseListener, MouseMotionListener, ActionListener{
 8  int x=-10, y=-10, wd=10, ht=10, status=0, clr=2;
 9  int[] rgb={0,225,0};
10  Button btn1, btn2;
11
12  public void init(){
13  addMouseListener(this);
14  addMouseMotionListener(this);
15  setLayout(null);
16  //
17  btn1=new Button("全消去");
18  btn1.addActionListener(this);
19  add(btn1);
20  btn1.setBounds(50, 10, 100, 20);
21  //
22  btn2=new Button("色を変える");
23  btn2.addActionListener(this);
24  add(btn2);
25  btn2.setBounds(200, 10, 100, 20);
26  //
27  }
28
29  public void start(){}//スレッド用？
30
31  public void mouseClicked(MouseEvent e){}//MouseListener
```

```
32  public void mousePressed(MouseEvent e){//MouseListener
33  x=e.getX();
34  y=e.getY();
35  repaint();
36  }
37  public void mouseReleased(MouseEvent e){}//MouseListener
38  public void mouseEntered(MouseEvent e){}//MouseListener
39  public void mouseExited(MouseEvent e){}//MouseListener
40  public void mouseDragged(MouseEvent e){//MouseMotionListener
41  x=e.getX();
42  y=e.getY();
43  repaint();
44  }
45  public void mouseMoved(MouseEvent e){}//MouseMotionListener
46
47  public void paint(Graphics g){
48  for(int i=0;i<=2;i++){
49  if(i==clr%3){ rgb[i]=225; }else{ rgb[i]=0; }
50  }
51  g.setColor(new Color(rgb[0],rgb[1],rgb[2]));
52
53  if(status==0){
54  g.clearRect(0,0,600,400);
55  g.fillRect(350,10,100,20);
56  status=1;
57  }else{
58  if(status==2){
59  g.fillRect(350,10,100,20);
60  status=1;
61  }else{
62  g.fillOval(x,y,wd,ht);
63  }
64  }
```

```
65  }
66
67  public void update(Graphics g){ paint(g); }
68
69  public void actionPerformed(ActionEvent e){
70  if(e.getSource()==btn1){
71  status=0;
72  repaint();
73  }
74  if(e.getSource()==btn2){
75  clr=clr+1;
76  status=2;
77  repaint();
78  }
79  }
80  }
```

4 考察

4.1 Graphic.java について

`Graphics g` に，`g.<メソッド>()` という処理をしている場所がたくさんある．この部分では，`paint()` に引数として渡された，アップレットの表示領域全体を表すインスタンス `g` に対して文字や図形の表示などの操作を行っている．

4.2 Adev.java

プログラムの一番重要な部分である因数分解の処理は前回 Java の実習で出てきたアルゴリズムを流用しているので，どちらかというと，イベントリスナーを使用していることの方が重要である．

メッセージを受け取る `Adev` クラスに，まず `implements ActionListener` でインターフェイスを付け，その中で，メッセージ源の `TextField txt1,txt2` に `txt1.addActionListener(this);` で `Adev` 自身を受け取る側として登録している．また，イベントが起こる場所として `TextField txt1,txt2` を，`add(txt1);` と `add(txt2);` で登録している．

4.3 Pict.java

マウスが押されるイベントや，マウスでドラッグされるイベントのときに，対応するメソッドには引数の一部としてそのときのポインタの座標も渡されるので，この座標をもとにその周辺に小さい円を描いている．

参考文献

[1]『3.Java Applet プログラミングの基礎』(配布プリント)，東京都立工業高等専門学校電子情報工学科，2002/12/18

[2] http://yougo.ascii24.com/，ASCII Corporation，1999-2002