独習Java

１３．１　アプレットの概要

１３．２　最初のJavaアプレット

１３．３　アプレットのライフサイクル

１３．４　Graphicsクラス

発表者

田島　勇樹

１３．１ アプレットの概要（1）

- アプレットとは‥

WebページのHTMLソースコードから参照されるプログラム

・アプレットはWebサーバからブラウザに動的にダウンロードされる。
・ダウンロードされたアプレットはブラウザの環境で実行される。

アプレットの操作できる範囲はサンドボックスの中に限られる。

→信頼されないコードは一定の境界の外では動作できない
例　ローカルディスクに対して読み書きできない

→ディスクに記録されているデータを誤ってまたは故意に破壊する危険をはらんでいるため

１３．１ アプレットの概要 (2)

・アプレットではネイティブコードを実行できない。
→サンドボックスによる制限の裏をかくことができるため

・ダウンロード元のホストにソケット接続を開くことができるが他のホストには接続できない
→クラッカーの侵入を阻止するため

デジタル署名をアプレットに関連付けることでアプレットの制限を緩和

１３．２　最初のJavaアプレット(1)

- アプレットプログラムを作るときのルール

・java.applet.Applet クラスのサブクラスとして作る
・public なクラスとして作る

クラスの定義

```
import java.applet.Applet;
public class MyApplet extends Applet { ...
```

１３．２　最初のJavaアプレット(2)

・アプレットウィンドウに文字列を表示するプログラム

```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="FirstApplet" width=200 height=200>
  </applet>
*/

public class FirstApplet extends Applet {
  public void paint(Graphics g) {
    g.drawString("This is my first applet!", 20, 100);
  }
}
```

１３．２　最初のJavaアプレット(3)

- アプレットの表示

コンパイルした後
appletviewer FirstApplet.java で実行

実行結果

アプレットビューアはJDKに付属されているツール
Java のアプレットをブラウザを使わずに実行することができる

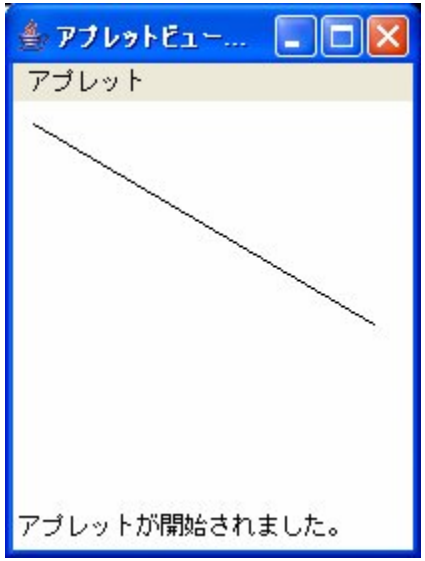
Javaソースファイルの先頭にあるコメント内のHTMLソースコードにしたがってアプレットを実行する

１３．２　最初のJavaアプレット(４)

- アプレットウィンドウ上の2点を結ぶ線を描画するプログラム

```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="DrawLine" width=200 height=200>
  </applet>
*/

public class DrawLine extends Applet {
  public void paint(Graphics g) {
    g.drawLine(10, 10, 180, 110);
  }
}
```

実行結果



１３．３　アプレットのライフサイクル（1）

- アプレットはWebブラウザまたはアプレットビューアなどのツールが用意された環境で実行
  →main()メソッドは存在しない

→その代わりに
  init(),start(),stop(),destroy()の４つのメソッドが呼び出される

これらのメソッドは
  java.applet.Appletクラスに定義されている

メソッド名	説明
init()	アプレットの実行が開始されるときだけ呼び出される。
start()	init()メソッドの実行が終わった後で実行される。 また、アプレットの実行を再開するとき呼び出される。 アプレットビューアを最小化してから最大化するとき、 Webページに別のページが表示され、元のページに戻るときに呼び出される。
stop()	アプレットの実行を中断するとき呼び出される アプレットビューアを最小化するとき、 Webページに別のページが表示されたとき 呼び出される。 destory()が呼び出される前に呼び出される
destory()	アプレット終了直前に呼び出される

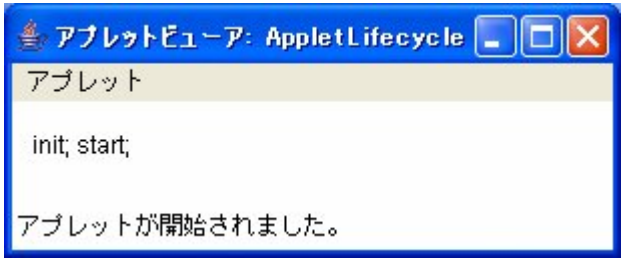
１３．３　アプレットのライフサイクル（２）

各メソッドの呼び出し

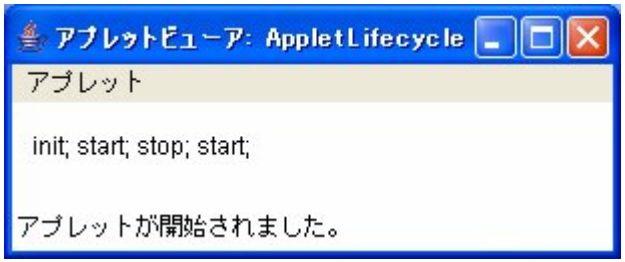
```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="AppletLifecycle"
     width=300 height=50>
  </applet>
*/

public class AppletLifecycle extends
     Applet {
  String str = "";

 public void init() {
    str += "init; ";
  }

public void start() {
    str += "start; ";
  }

 public void stop() {
    str += "stop; ";
  }

  public void destroy() {
    System.out.println("destroy");
  }

  public void paint(Graphics g) {
    g.drawString(str, 10, 25);
  }
}
```

13．3　アプレットのライフサイクル (3)

- アプレットの開始



- アプレットウインドウを最小化、最大化する



- アプレットを終了する

>appletviewer AppletLifecycle.java
destroy

13．4　Graphicsクラス（1）

- Graphicsオブジェクト

→グラフィックを出力するメソッド群がカプセル化されている

線、円、四角形、文字列、イメージ、文字、弧を描画することができる

Graphicsクラスに定義されている主なインスタンスメソッド（1）

メソッド	説明
abstract void drawArc(int x,int y,int w, int h,int degrees0,int degrees1)	degrees0とdegrees1の角度で弧を描画する。弧の中心は、座標x,yに左上隅が配置される幅w,高さhの四角形の中心となる。角度は、逆時計周りに増加し、0度は時計の午後3時に相当する。
abstract boolean drawImage(Image img, int x,int y,ImageObserver io)	座標x,yに左上隅が配置されるようにイメージimgを描画する。描画処理の進行状況は、ioに送られる。
abstract void drawLine(int x0,int y0,int x1,int y1)	座標x0,y0とx1,y1を結ぶ線を描画する
abstract void drawOval(int x,int y,int w,int h)	円を描画する。円の中心は、座標x,yに左上隅が配置される幅w,高さhの四角形の中心となる
abstract void drawPolygon(int x[],int y[],int n)	n個の頂点を持つ多角形を描画する。頂点の座標は、配列xとyの要素として引き渡す。最初の頂点と最後の頂点は、自動的に接続される。
abstract void drawPolyline(int x[],int y[],int n)	n個の頂点を持つ多角線（ポリライン）を描画する。頂点の座標は配列xとyの要素として引き渡す。
void drawRect(int x,int y,int w,int n)	座標x,yに左上隅が配置される幅w,高さhの四角形を描画する。

13．4 Graphicsクラス（2）

- Graphicsクラスに定義されている主なインスタンスメソッド（2）

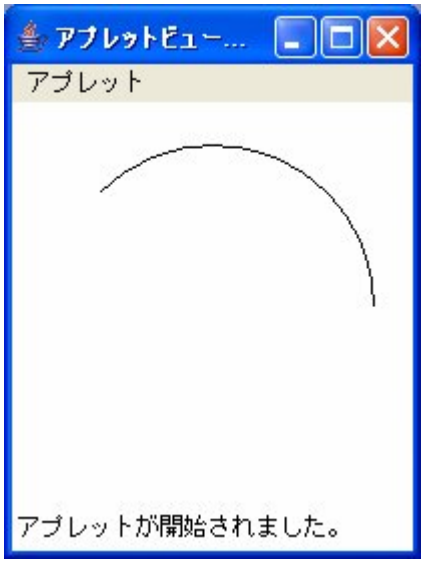
メソッド	説明
abstract void drawString(String str,int x, int y)	strを座標x,yに描画する
abstract void fillArc(int x,int y,int w,int h, int degree0,int degree1)	degree0とdegree1の角度で弧を塗りつぶして描画する。 弧の中心は、座標x,yに左上隅が配置される幅w,高さhの四角形の中心となる。0度は時計の午後3時に相当する。
abstract void fillOval(int x,int y,int w, int h)	円を塗りつぶして描画する。円の中心は、座標x,yに左上隅が配置される幅w,高さhの四角形の中心となる。
abstract void fillPolygon(int x[],int y[], int n)	n個の頂点を持つ多角形を塗りつぶして描画する。頂点の座標は、配列xとyの要素として引き渡す。
void fillRect(int x,int y,int w,int h)	座標x,yに左上隅が配置される幅w,高さhの四角形を塗りつぶして描画する。
abstract Color getColor()	現在のオブジェクトのカラーを取得する。
abstract Font getFont()	現在のオブジェクトのフォントを取得する。
abstract FontMetrics getFontMetrics()	現在のオブジェクトのフォントメトリックスを取得する。
abstract void setColor(Color c)	グラフィックコンテキストの現在のカラーとしてcを設定する。
abstract void getFont(Font f)	現在のオブジェクトのフォントとしてfを設定する。

１３．４　Graphicsクラス（3）

・アプレットウィンドウに弧を描くプログラム

```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="DrawArc" width=200 height=200>
  </applet>
*/

public class DrawArc extends Applet {
  public void paint(Graphics g) {
    g.drawArc(20, 20, 160, 160, 0, 135);
  }
}
```

実行結果



１３．４　Graphicsクラス（４）

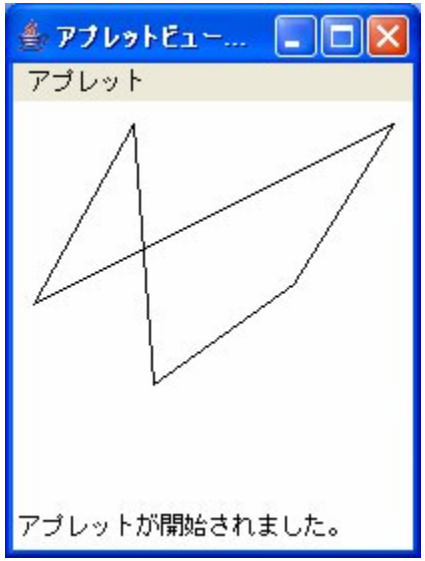
・アプレットウィンドウに多角形を描くプログラム

```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="DrawPolygon" width=200 height=200>
  </applet>
*/
public class DrawPolygon extends Applet {
  public void paint(Graphics g) {
    int n = 5;
    int xdata[] = new int[n];
    int ydata[] = new int[n];
    xdata[0] = 10;
    ydata[0] = 100;
```

１３．４　Graphicsクラス（5）

```
    xdata[1] = 60;
    ydata[1] = 10;
    xdata[2] = 70;
    ydata[2] = 140;
    xdata[3] = 140;
    ydata[3] = 90;
    xdata[4] = 190;
    ydata[4] = 10;
    g.drawPolygon(xdata, ydata, n);
  }
}
```

実行結果



練習問題1

- アプレットウィンドウに日時を表示せよ。

ただし日時に色を着けること。

練習問題2

- 下のアプレットウインドウのように表示するプログラムを作成せよ。(ヒント:Randomクラスを使用して、その乱数によって円と文字列の表示される座標位置、円の半径を決める)