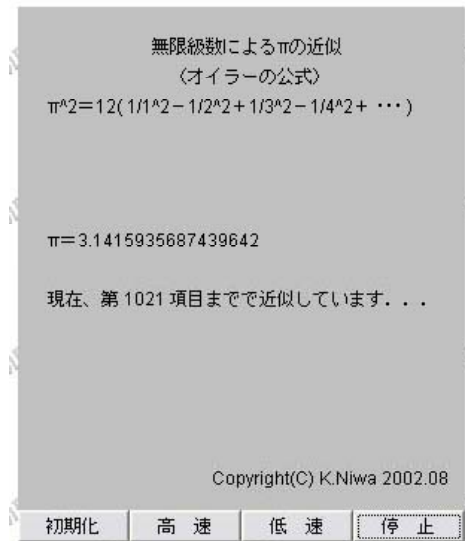
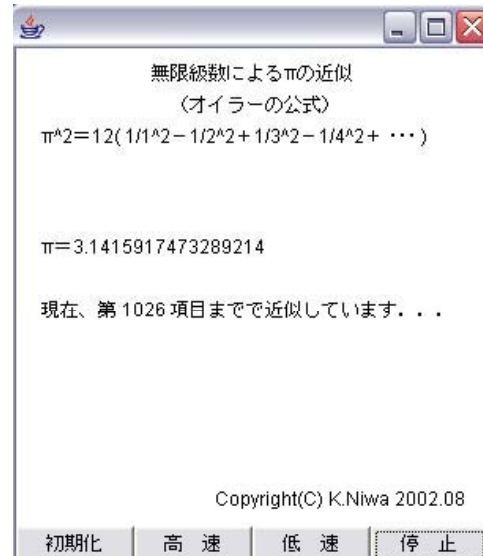


【オイラーの公式】



[J a v a アプレット]



[J a v a アプリケーション]

1. はじめに

次のオイラーの公式を用いて π の近似値を求めてみましょう。

[オイラーの公式]

$$\pi^2 = 12 \left(\frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \frac{1}{5^2} - \dots \right)$$

シミュレーションソフト「オイラーの公式による π の近似」を使って、 π の近似値が求まる様子を観察してみてください。

2. J a v a アプレット

(1) J a v a プログラムリスト

```

////////////////////////////////////
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
//                                                                //
////////////////////////////////////

//クラスの読み込み
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

/*****/ public class Oirer extends Applet implements Runnable { *****/
public class Oirer extends Applet implements Runnable { //スレッドをさせるようにする

//変数とオブジェクトの型宣言
    Thread myTh; //スレッド型で宣言する
    Button[] myBtn; //ボタン型配列で宣言する

```

```

Panel myPanel;           //パネル型で宣言する
int flag=0;
int Speed=100;          //速度
int ct=0;               //実験回数
int count;              //ループカウンター
double pai;             //πの近似値
double s;               //π/4の近似値を求める過程での無限級数

/*****/
public void init() {
    setBackground(Color.lightGray); //背景色をグレーにする
    myTh=null;                       //スレッドの初期化
    myBtn=new Button[4];             //ボタンの実体化
    myBtn[0]=new Button("初期化");
    myBtn[1]=new Button("高速");
    myBtn[2]=new Button("低速");
    myBtn[3]=new Button("停止");

    myPanel=new Panel();             //パネルの実体化
    myPanel.setLayout(new GridLayout(1,4)); //パネルをグリッドレイアウトにする
    for (count=0;count<=3;count++) {
        myPanel.add(myBtn[count]); //パネルにボタンを貼り付ける
    }
    setLayout(new BorderLayout());    //全体をボーダーレイアウトにする
    add("South",myPanel);           //パネルを南に貼り付ける

    myBtn[0].addActionListener(new ActionListener() { //初期化ボタンの定義
        public void actionPerformed(ActionEvent e) {
            flag=0; //ボタンの識別子
            repaint();
        }
    });

    myBtn[1].addActionListener(new ActionListener() { //高速ボタンの定義
        public void actionPerformed(ActionEvent e) {
            flag=1; //ボタンの識別子
            Speed=20; //速度
            repaint();
        }
    });

    myBtn[2].addActionListener(new ActionListener() { //低速ボタンの定義
        public void actionPerformed(ActionEvent e) {
            flag=2; //ボタンの識別子
            Speed=200; //速度
            repaint();
        }
    });

    myBtn[3].addActionListener(new ActionListener() { //停止ボタンの定義
        public void actionPerformed(ActionEvent e) {
            flag=3; //ボタンの識別子
            repaint();
        }
    });

} //public void init()

/*****/
public void start() {
    public void start() {
        if (myTh==null) {
            myTh=new Thread(this); //スレッドの実体化
            myTh.start();
        }
    }
}

```

```

    }

    /**** public void run()メソッド *****/
    public void run() {
        while (true) {
            try {
                myTh.sleep (Speed);
            }
            catch (InterruptedException e) {}
            if (flag==1 || flag==2) {
                repaint();
            }
        }
    }

    /**** public void paint(Graphics g)メソッド *****/
    public void paint(Graphics g) {

        //初期状態または初期化ボタンを押したときのイベント処理
        if(flag==0) {

            //g.clearRect(0,0,300,360);           //全体のクリア
            ct=0;                               //第何項目までの和であるかの初期化
            s=0;                                //π^2 の近似値を求める過程での無限級数の初期化
            pai=0;                              //π の近似値の初期化

            g.drawString("無限級数による π の近似",70+20,20+10);
            g.drawString(" (オイラーの公式) ",70+20,40+10);
            g.drawString("π ^2 = 12( 1/1^2 - 1/2^2 + 1/3^2 - 1/4^2 + ... )"
                ,20,60+10);

            g.drawString("π =",30-10,160);
            g.drawString("現在、第 "+" "+" 項目までで近似しています... "
                ,30-10,300-100); //第何項目までの和であることを表示
            g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10); //作者表示

        } //if(flag==0)

        //高速ボタンまたは低速ボタンを押したときのイベント処理
        else if (flag==1 || flag==2) {
            if (ct<65000) {
                ct=ct+1;
            }
            else {
                flag=3;
            }

            if (ct%2==1) {
                s=s+(double) 1/(ct*ct);
            }
            else if (ct%2==0) {
                s=s-(double) 1/(ct*ct);
            }
            pai=(double) Math.sqrt(12*s);
            g.drawString("無限級数による π の近似",70+20,20+10);
            g.drawString(" (オイラーの公式) ",70+20,40+10);
            g.drawString("π ^2 = 12( 1/1^2 - 1/2^2 + 1/3^2 - 1/4^2 + ... )"
                ,20,60+10);

            g.drawString("π =" +pai,30-10,160);
            g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                ,30-10,300-100); //第何項目までの和であることを表示
            g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10); //作者表示
        } //else if (flag==1 || flag==2)
    }

```

```
//停止ボタンを押したときのイベント処理
    if(flag==3) {
        g.drawString("無限級数による  $\pi$  の近似",70+20,20+10);
        g.drawString("    (オイラーの公式) ",70+20,40+10);
        g.drawString("  $\pi^2 = 12( 1/1^2 - 1/2^2 + 1/3^2 - 1/4^2 + \dots )$ "
            ,20,60+10) ;
        g.drawString("  $\pi =$  "+pai,30-10,160);
        g.drawString("現在、第 "+ct+" 項目までで近似しています... "
            ,30-10,300-100); //第何項目までの和であることを表示
        g.drawString("Copyright (C) K.Niwa 2002.08",130,330-10); //作者表示
    } //if(flag==3)
} //public void paint(Graphics g)
} //public class Gregory extends Applet implements Runnable
```

(2) HTML リスト

```
<HTML>
  <HEAD>
< !-----
               「オイラーの公式による  $\pi$  の近似」
               Copyright (C) K.Niwa 2002.08.05
----->
  </HEAD>
  <BODY>
    <CENTER>
      <B> 「オイラーの公式による  $\pi$  の近似」 </B>
      <BR><BR>
      <APPLET CODE="Oirer.class" WIDTH="300" HEIGHT="360"></APPLET>
    </CENTER>
  </BODY>
</HTML>
```

3. J a v a アプリケーション・プログラムリスト

```
//////////////////////////////////////
//
//                               「F オイラーの公式による  $\pi$  の近似」
//                               Copyright (C) K.Niwa 2002.08.16
//                               ( J a v a アプリケーション )
//
//
//////////////////////////////////////
```

```
//クラスの読み込み
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;
```

```
/* public class FOirer extends Frame implements Runnable *****/
public class FOirer extends Frame implements Runnable { //スレッドを使えるようにする
```

```
//変数とオブジェクトの型宣言
```

```
  Thread myTh; //スレッド型で宣言する
  Button[] myBtn; //ボタン型配列で宣言する
  Panel myPanel; //パネル型で宣言する
  int flag=0;
  int Speed=100; //速度
  int ct=0; //実験回数
  int count; //ループカウンター
  double pai; //  $\pi$  の近似値
  double s; //  $\pi / 4$  の近似値を求める過程での無限級数
```

*****フレームとイベント処理の定義*****

```
public FOirer() {  
  
    setSize(310,360); //フレームの大きさ  
    addWindowListener(new WindowAdapter() { //閉じるボタンのイベント処理  
        public void windowClosing(WindowEvent e) {  
            System.exit(0);  
        }  
    });  
  
    myTh=null; //スレッドの初期化  
    if (myTh==null) {  
        myTh=new Thread(this); //スレッドの実体化  
        myTh.start(); //スレッドを開始する  
    }  
  
    myBtn=new Button[4]; //ボタンの実体化  
    myBtn[0]=new Button("初期化");  
    myBtn[1]=new Button("高 速");  
    myBtn[2]=new Button("低 速");  
    myBtn[3]=new Button("停 止");  
  
    myPanel=new Panel(); //パネルの実体化  
    myPanel.setLayout(new GridLayout(1,4));  
    for (count=0;count<=3;count++) {  
        myPanel.add(myBtn[count]); //パネルにボタンを貼り付ける  
    }  
    setLayout(new BorderLayout()); //全体をボーダーレイアウトにする  
    add("South",myPanel); //パネルを南に貼り付ける  
  
    myBtn[0].addActionListener(new ActionListener() { //初期化ボタンの定義  
        public void actionPerformed(ActionEvent e) {  
            flag=0; //ボタンの識別子  
            repaint();  
        }  
    });  
  
    myBtn[1].addActionListener(new ActionListener() { //高速ボタンの定義  
        public void actionPerformed(ActionEvent e) {  
            flag=1; //ボタンの識別子  
            Speed=20; //速度  
            repaint();  
        }  
    });  
  
    myBtn[2].addActionListener(new ActionListener() { //低速ボタンの定義  
        public void actionPerformed(ActionEvent e) {  
            flag=2; //ボタンの識別子  
            Speed=200; //速度  
            repaint();  
        }  
    });  
  
    myBtn[3].addActionListener(new ActionListener() { //停止ボタンの定義  
        public void actionPerformed(ActionEvent e) {  
            flag=3; //ボタンの識別子  
            repaint();  
        }  
    });  
  
} //public FOirer()
```

```

/***** public void run()メソッド *****/ イベントがなくても動作する *****/
public void run() {
    while (true) {
        try {
            myTh.sleep (Speed);
        }
        catch (InterruptedException e) {}
        if (flag==1 || flag==2) {
            repaint();
        }
    }
} //public void run ()

/***** public void paint(Graphics g)メソッド *****/
public void paint(Graphics g) {
    //初期状態または初期化ボタンを押したときのイベント処理
    if(flag==0) {
        //g.clearRect(0,0,300+10,360); //全体のクリア
        ct=0; //第何項目までの和であるかの初期化
        s=0; //π^2 の近似値を求める過程での無限級数の初期化
        pai=0; //πの近似値の初期化

        g.drawString("無限級数によるπの近似",70+20,20+10+20);
        g.drawString(" (オイラーの公式)",70+20,40+10+20);
        g.drawString("π^2 = 12( 1/1^2 - 1/2^2 + 1/3^2 - 1/4^2 + ... )"
            ,20,60+10+20);

        g.drawString("π =",30-10,160);
        g.drawString("現在、第 "+" "+" 項目までで近似しています..."
            ,30-10,300-100); //第何項目までの和であることを表示
        g.drawString("Copyright (C) K.Niwa 2002.08",130,330-10); //作者表示
    } //if (flag==0)

    //高速ボタンまたは低速ボタンを押したときのイベント処理
    else if (flag==1 || flag==2) {
        if (ct<65000) {
            ct=ct+1;
        }
        else {
            flag=3;
        }

        if (ct%2==1) {
            s=s+(double) 1/(ct*ct);
        }
        else if (ct%2==0) {
            s=s-(double) 1/(ct*ct);
        }

        pai=(double) Math.sqrt (12*s);
        g.drawString("無限級数によるπの近似",70+20,20+10+20);
        g.drawString(" (オイラーの公式)",70+20,40+10+20);
        g.drawString("π^2 = 12( 1/1^2 - 1/2^2 + 1/3^2 - 1/4^2 + ... )"
            ,20,60+10+20);

        g.drawString("π =" +pai,30-10,160);
        g.drawString("現在、第 "+ct+" 項目までで近似しています..."
            ,30-10,300-100); //第何項目までの和であることを表示
        g.drawString("Copyright (C) K.Niwa 2002.08",130,330-10); //作者表示
    } //else if (flag==1 || flag==2)

```

```

//停止ボタンを押したときのイベント処理
    if(flag==3) {
        g.drawString("無限級数による  $\pi$  の近似",70+20,20+10+20);
        g.drawString("    (オイラーの公式)",70+20,40+10+20);
        g.drawString("  $\pi^2 = 12 ( 1/1^2 - 1/2^2 + 1/3^2 - 1/4^2 + \dots )$ "
                    ,20,60+10+20);

        g.drawString("  $\pi =$ " +pai,30-10,160);
        g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                    ,30-10,300-100); //第何項目までの和であるかを表示
        g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10); //作者表示
    } //if(flag==3)

} //public void paint(Graphics g)

//*****main メソッドで J a v a アプリケーションには必ず必要である*****
public static void main(String[] args) {
    Frame w=new FOirer();
    w.show();
} //public static void main(String[] args)

} //public class FOirer extends Frame implements Runnable

```