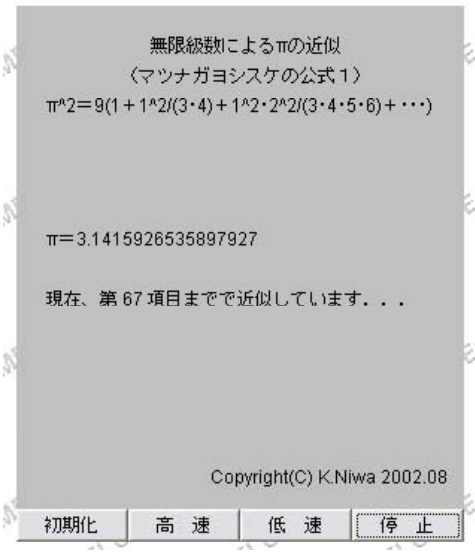
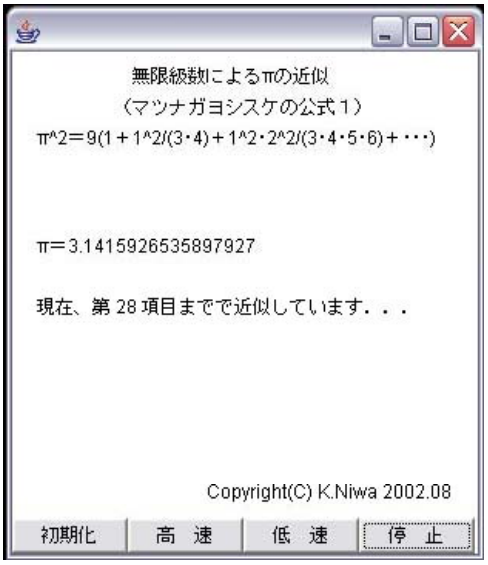
【マツナガ・ヨシスケの公式１】



［Ｊａｖａアプレット］



［Ｊａｖａアプリケーション］

１．はじめに

次のマツナガ・ヨシスケの公式１を用いてπの近似値を求めてみましょう。

［マツナガ・ヨシスケの公式１］

$$\pi^2 = 9\left(1+\frac{1^2}{3\cdot 4}+\frac{1^2\cdot 2^2}{3\cdot 4\cdot 5\cdot 6}+\frac{1^2\cdot 2^2\cdot 3^2}{3\cdot 4\cdot 5\cdot 6\cdot 7\cdot 8}+\cdots\right)$$

シミュレーションソフト「マツナガ・ヨシスケの公式１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．Ｊａｖａアプレット

（1）Ｊａｖａプログラムリスト

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                                                    //
//                                        「マツナガ・ヨシスケの公式１によるπの近似」                                                 //
//                                              Copyright（C）K.Niwa  2002.08.05                                                      //
//                                                  （Ｊａｖａアプレット）                                                            //
//                                                                                                                                    //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// ライブラリーからのクラスの読み込み
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

//*****public class Matunaga1 extends Applet implements Runnable ***********************************
public class Matunaga1 extends Applet 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=1000;                         //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 sa;                              //πの近似値を求める過程で使用
        double sb;                              //πの近似値を求める過程で使用

/****** public void init()ﾒｿｯﾄﾞ ************************************************************/
        public void init(){
                setBackground(Color.lightGray);
                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));         //ﾊﾟﾈﾙをｸﾞﾘｯﾄﾞﾚｲｱｳﾄにする
                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;                                 //ボタン識別子
                                Speed=500;                              //速度
                                repaint();
                        }
                });

                myBtn[2].addActionListener(new ActionListener(){        //低速ボタンの定義
                        public void actionPerformed(ActionEvent e){
                                flag=2;                                 //ボタン識別子
                                Speed=1000;                             //速度
                                repaint();
                        }
                });

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

        }//public void init()
```

```
/******  public  void  start()ﾒｿｯﾄﾞ  ************************************************************/
        public void start() {
                if (myTh==null) {
                        myTh=new Thread(this);                  //スレッドの実体化
                        myTh.start();                           //スレッドの開始
                }
        }

/******  public  void  run()ﾒｿｯﾄﾞ  ************ イベントなしで動作する  ************************/
        public void run() {
                while (true) {
                        try{
                                myTh.sleep(Speed);
                        }
                        catch (InterruptedException e)  {}
                        if (flag==1 || flag==2) {
                                repaint();
                        }
                }
        }

/******  public  void  paint(Graphics  g)ﾒｿｯﾄﾞ  ********************************************************/
        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
                        sa=1;                         //πの近似値を求める過程での初期化
                        sb=1;                         //πの近似値を求める過程での初期化
                        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("（マツナガヨシスケの公式１） ",70,40+10);
                        g.drawString("π^2 ＝ 9(1 ＋ 1^2/(3･4) ＋ 1^2･2^2/(3･4･5･6) ＋･･･)"
                                                                                ,20,60+10);
                        g.drawString("π＝",30-10,160);
                //第何項目までの和であるかを表示
                        g.drawString("現在、第 "+"  "+" 項目までで近似しています. . .  "
                                                                        ,30-10,300-100);
                        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;
                        }

                        sb=sb*(double)(ct*ct)/(double)((2*ct+1)*(2*ct+2));
                        sa=sa+sb;

                        pai=(double)Math.sqrt((double)9*sa);

                        g.drawString("無限級数によるπの近似",70+20,20+10);
                        g.drawString("（マツナガヨシスケの公式１） ",70,40+10);
                        g.drawString("π^2 ＝ 9(1 ＋ 1^2/(3･4) ＋ 1^2･2^2/(3･4･5･6) ＋･･･)"
                                                                                ,20,60+10);
                        g.drawString("π＝"+pai,30-10,160);
```

```
                //第何項目までの和であるかを表示
                        g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                                                                        ,30-10,300-100);
                        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πの近似",70+20,20+10);
                        g.drawString("（マツナガヨシスケの公式１） ",70,40+10);
                        g.drawString("π^2 ＝ 9(1 ＋ 1^2/(3･4) ＋ 1^2･2^2/(3･4･5･6)＋･･･)"
                                                                                ,20,60+10);
                        g.drawString("π＝"+pai,30-10,160);
                //第何項目までの和であるかを表示
                        g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                                                                        ,30-10,300-100);
                        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>
<!-----------------------------------------------------------------------------------------------------------------------
                        「マツナガヨシスケの公式１によるπの近似」
                           Copyright（C）K.Niwa 2002.08.05
----------------------------------------------------------------------------------------------------------------------->
        </HEAD>
        <BODY>
                <CENTER>
                        <B>「マツナガヨシスケの公式によるπの近似」</B>
                        <BR><BR>
                        <APPLET CODE="Matunaga1.class" WIDTH="300" HEIGHT="360"></APPLET>
                </CENTER>
        </BODY>
</HTML>
```

３．Ｊａｖａアプリケーション・プログラムリスト

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                                                //
//                                「Ｆマツナガヨシスケの公式１によるπの近似」                                                    //
//                                        Copyright（C）K.Niwa  2002.08.17                                                        //
//                                          （Ｊａｖａアプリケーション）                                                          //
//                                                                                                                                //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//ライブラリーからのクラスの読み込み
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

/****** public class FMatunaga1 extends Frame implements Runnable ********************************/
public class FMatunaga1 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=1000;                     //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 sa;                          //πの近似値を求める過程で使用
    double sb;                          //πの近似値を求める過程で使用

/******  フレームとイベント処理の定義 ****************************************************/
    public FMatunaga1(){

        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();                               //スレッドの開始
        }

        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));         //ﾊﾟﾈﾙをｸﾞﾘｯﾄﾞﾚｲｱｳﾄにする
        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;                                             //ボタン識別子
                Speed=500;                                          //速度
                repaint();
            }
        });

        myBtn[2].addActionListener(new ActionListener(){      //低速ボタンの定義
            public void actionPerformed(ActionEvent e){
                flag=2;                                             //ボタン識別子
                Speed=1000;                                         //速度
                repaint();
            }
        });
```

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

	}//public FMatunaga1()

/****** public void run()ﾒｿｯﾄﾞ **************イベントなしでアクションする********************/
	public void run(){
		while (true){
			try{
				myTh.sleep(Speed);
			}
			catch (InterruptedException e) {}
			if (flag==1 || flag==2){
				repaint();
			}
		}
	}

/****** public void paint(Graphics g)ﾒｿｯﾄﾞ ************************************************/
	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(0,0,310,360);	//全体のクリア
			ct=0;				//第何項目までの和であるかの初期化
			sa=1;				//πの近似値を求める過程での初期化
			sb=1;				//πの近似値を求める過程での初期化
			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+10+20);
			g.drawString("（マツナガヨシスケの公式１）",70,40+10+20);
			g.drawString("π^2 ＝ 9(1 ＋ 1^2/(3･4) ＋ 1^2･2^2/(3･4･5･6)＋･･･)"
								,20,60+10+20);
			g.drawString("π＝",30-10,160);
		//第何項目までの和であるかを表示
			g.drawString("現在、第 "+"  "+" 項目までで近似しています... "
								,30-10,300-100);
			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;
			}

			sb=sb*(double)(ct*ct)/(double)((2*ct+1)*(2*ct+2));
			sa=sa+sb;

			pai=(double)Math.sqrt((double)9*sa);

			g.drawString("　無限級数によるπの近似",70,20+10+20);
			g.drawString("（マツナガヨシスケの公式１）",70,40+10+20);
			g.drawString("π^2 ＝ 9(1 ＋ 1^2/(3･4) ＋ 1^2･2^2/(3･4･5･6)＋･･･)"
								,20,60+10+20);
			g.drawString("π＝"+pai,30-10,160);
```

```
                //第何項目までの和であるかを表示
                        g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                                                                        ,30-10,300-100);
                        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πの近似",70,20+10+20);
                        g.drawString("（マツナガヨシスケの公式１） ",70,40+10+20);
                        g.drawString("π^2 = 9(1 + 1^2/(3･4) + 1^2･2^2/(3･4･5･6)+･･･)"
                                                                        ,20,60+10+20);
                        g.drawString("π ="+pai,30-10,160);
                  //第何項目までの和であるかを表示
                        g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                                                                        ,30-10,300-100);
                        g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);       //作者表示

                }//if(flag==3)

        }//public void paint(Graphics g)

/******  public static void main ﾒｿｯﾄﾞ  ***********************************************************/
        public static void main(String[] args){
                Frame w=new FMatunaga1();
                w.show();
        }//public static void main(String[] args)

}//public class FMatunaga1 extends Frame implements Runnable
```