PRIEDAI
Programų pavyzdžiai ir savarankiško darbo užduotys

1–2 programos

Savarankiškos aplikacijos pavyzdys

```
class FirstApp {
        public static void main ( String args[ ] ) {
                System.out.println( "My first application." );
          }
}
```

Klientinės programos pavyzdys

```
import java.awt.*;
import java.applet.*;

public class FirstApplet extends Applet{
        Image NewImage;
        public void init( ){
                resize( 400, 400 );
                NewImage = getImage( getCodeBase( ),"New.gif" );
        }
        public void paint( Graphics g ){
                g.drawImage( NewImage, 100, 350, this );
                play( getCodeBase( ), "New.au" );
        }
}
```

3 programa

Aritmetiniai reiškiniai. Pasakykite, kokius rezultatus turi gauti programa. Pasitikrinkite, vykdydami programą.

```
import java.applet.*;

public class ArithmTest extends Applet{
        int x = 2;
        int y = 3;
        int z = 0;
        int tripleAndAdd( int a ) {
                x = x + a;
                return 3 * a;      }
        public void init( ) {
                x ++;
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
                x += y;
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
                z = x + tripleAndAdd( y );
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
                z = tripleAndAdd( y ) + x;
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
                x += tripleAndAdd( y );
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
        }
}
```

HTML dokumentas programai:

```
<HTML>
<HEAD>
<TITLE>First Java Applet</TITLE>
</HEAD>
<BODY>
<APPLET CODE="ArithmTest" WIDTH=500 HEIGHT=100></APPLET>
</BODY>
</HTML>
```

4 programa

Matomumo sritys. Pasakykite, kokius rezultatus turi gauti programa. Pasitikrinkite, vykdydami programą.

```
import java.applet.*;
public class ArithmTest2 extends Applet{
        int x = 2;
        int y = 3;
        int z = 0;
        int tripleAndAdd( int a ) {
                System.out.println( "Beginning of tripleAndAdd:" +
                                        "x =" + x + " a = " + a );
                x = x + a;
                System.out.println( "End of tripleAndAdd:" +
                                        "x =" + x + " a = " + a );
                return x * a;
         }
         public void init( ) {
                int x = 5;
                x ++;
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
                x += y;
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
                z = x + tripleAndAdd( y );
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
                x += tripleAndAdd( y );
                        System.out.println( "x=" + x + " y=" + y + " z=" + z );
        }
}
```

5–6 programos

Loginiai reiškiniai. Raktažodis *this*. Pasakykite, ką spausdins programos. Pasitikrinkite, vykdydami programą.

```
import java.applet.*;

class InventoryItem extends Object{
        public int InStock = 0;
        public boolean getItem(){
                if( this.InStock > 0 ) {
                        this.InStock--;
                        return true; }
                else { return false; }
        }
}

public class Store extends Applet{
        InventoryItem Chicken, Egg;
        public void init( ){
                Chicken = new InventoryItem( );
                Chicken.InStock = 2;
                Egg = new InventoryItem( );
                Egg.InStock = 2;
                if(Chicken.getItem( ) || Egg.getItem( ) ){
                        System.out.println( "First order filled" );  }
                else {  System.out.println( "First order not filled" );  }
                        System.out.println( "Chickens left:" + Chicken.InStock );
                        System.out.println( "Eggs left:" + Egg.InStock );
                        if( Chicken.getItem( ) && Egg.getItem( ) ){
                                System.out.println( "Second order filled" );  }
                        else {  System.out.println( "Second order not filled" );  }
                                System.out.println( "Chickens left:" + Chicken.InStock);
                                System.out.println( "Eggs left:" + Egg.InStock );
                                if( Chicken.getItem( ) && Egg.getItem( ) ){
                                        System.out.println( "Third order filled" );  }
                                else {  System.out.println( "Third order not filled" );  }
                                        System.out.println( "Chickens left:" +
                                                Chicken.InStock );
                                        System.out.println( "Eggs left:" + Egg.InStock );
                         }
}
```

```
import java.applet.*;

public class Kids extends Applet {
        int boredomFactor=5;       //how long until they quit
        String message="";         //what they normaly say
        String quitMessage="";     //what they say until they quit
         public void MyTurn ( Kids WhozNext ){
                if ( boredomFactor-- <=0 ){
                        System.out.println( quitMessage );}
                else{
                        System.out.println( message );
                       WhozNext.MyTurn( this );
                }
        }
         public void init( ){
                Kids Bobby,Kenny;
                Bobby =  new Kids( );
                Bobby.message = "kenny, you did it";
                Bobby.boredomFactor = 4;
                Bobby.quitMessage = "fine";
                 Kenny = new Kids( );
                 Kenny.message = "BOBBY, YOU DID IT";
                        //Kenny.boredomFactor = 3;
                Kenny.quitMessage = "FINE";
                Kenny.MyTurn( Bobby );
         }
}
```

7–8 programos

Metodų-konstruktorių vykdymo tvarka. Metodų perkrovimas. Ką spausdins programos? Pasitikrinkite, vykdydami programą.

```
// Initialization: variables and methods
//
class Tag{
        Tag( int i ){
                System.out.println( "Tag("+i+")" );
         }
}

class Card{
        Tag t1 = new Tag( 1 ); // before constructor !
         Card( ){
                System.out.println( "Inside constructor Card" );
                 t3 = new Tag (33 );
         }
        Tag t2 = new Tag( 2 );
         void m( ){
                 System.out.println( "Finish" );
         }
         Tag t3 = new Tag( 3 ); // after constructor !
}

public class Initialization{
        public static void main(String args[ ]) {
                Card c= new Card( );
                 c.m( );
         }
}
```

```
// Overloading
//

public class PrimitiveOverloading{
  static void prt( String s ){
    System.out.println( s );
  }

  void m1(char x){ prt("m1(char)" ); }
  void m1(byte x){ prt("m1(byte)" ); }
  void m1(short x){ prt("m1(short)" ); }
  void m1(int x){ prt("m1(int)" ); }
  void m1(long x){ prt("m1(long)" );}
  void m1(float x){ prt("m1(float)" ); }
  void m1(double x){ prt("m1(double)" ); }

  void m2(byte x){ prt("m2(byte)" ); }
  void m2(short x){ prt("m2(short)" ); }
  void m2(int x){ prt("m2(int)" ); }
  void m2(long x){ prt("m2(long)" );}
  void m2(float x){ prt("m2(float)" ); }
  void m2(double x){ prt("m2(double)" ); }

  void m3(short x){ prt("m3(short)" ); }
  void m3(int x){ prt("m3(int)" ); }
  void m3(long x){ prt("m3(long)" );}
  void m3(float x){ prt("m3(float)" ); }
  void m3(double x){ prt("m3(double)" ); }

  void m4(int x){ prt("m4(int)" ); }
  void m4(long x){ prt("m4(long)" );}
  void m4(float x){ prt("m4(float)" ); }
  void m4(double x){ prt("m4(double)" ); }

  void m5(long x){ prt("m5(long)" );}
  void m5(float x){ prt("m5(float)" ); }
  void m5(double x){ prt("m5(double)" ); }

  void m6(float x){ prt("m6(float)" ); }
  void m6(double x){ prt("m6(double)" ); }

  void m7(double x){ prt("m7(double)" ); }


   void testChar( ){
     char x ='x';
     prt( "char argument" );
     m1( x ); m2( x ); m3( x ); m4( x ); m5( x ); m6( x ); m7( x );
    }
```

```
  void testByte( ){
    byte x =1;
    prt( "byte argument" );
    m1( x ); m2( x ); m3( x ); m4( x ); m5( x ); m6( x ); m7( x );
  }

  void testShort( ){
    short x =1;
    prt( "short argument" );
    m1( x ); m2( x ); m3( x ); m4( x ); m5( x ); m6( x ); m7( x );
  }

  void testInt( ){
    int x =1;
    prt( "int argument" );
    m1( x ); m2( x ); m3( x ); m4( x ); m5( x ); m6( x ); m7( x );
  }

  void testLong( ){
    long x =1L;
    prt( "long argument" );
    m1( x ); m2( x ); m3( x ); m4( x ); m5( x ); m6( x ); m7( x );
  }

  void testFloat( ){
    float x =1F;
    prt( "float argument" );
    m1( x ); m2( x ); m3( x ); m4( x ); m5( x ); m6( x ); m7( x );
  }

  void testDouble( ){
    double x =1;
    prt( "double argument" );
    m1( x ); m2( x ); m3( x ); m4( x ); m5( x ); m6( x ); m7( x );
  }


 public static void main( String args[ ] ){
   PrimitiveOverloading pl = new PrimitiveOverloading( );
   pl.testChar( );
   pl.testByte( );
   pl.testShort( );
   pl.testInt( );
   pl.testLong( );
   pl.testFloat( );
   pl.testDouble( );
 }

}
```

9 programa

Paveldimumas. Ką spausdins programa? Pasitikrinkite, vykdydami programą.

```
// Inheritance
//
class Class1 {
        private String s = "Class1:  ";
        public void append( String ss ) { s += ss; }
        public void m1( ) { append( "m1( )  " );}
        public void m2( ) { append( "m2( )  " );}
        public void m3( ) { append( "m3( )  " );}
        public void print( ) { System.out.println( s );}
        public static void main( String [ ] args ) {
                Class1 c1 = new Class1( );
                c1.m1( ); c1.m2( ); c1.m3( ); c1.print( );
        }
}

public class Class2 extends Class1 {
        public void m3( ){   // overriding method m3
                append( "Class2.m3()  " );
                super.m3( );       // calling m3 of superclass
        }
        public void m4( ) { append( "   m4( )" ); } // extension of interface
        public static void main( String [ ] args ) {
                Class2 c2 = new Class2( );
                c2.m1( ); c2.m2( ); c2.m3( ); c2.m4( ); c2.print( );
                System.out.println( "Test for the superclass Class1:" );
                Class1.main( args ); // args already defined
        }
}
```

10 programa

Paveldimumas. Paveldimumu susietų klasių konstruktorių vykdymo tvarka. Ką spausdins programa? Pasitikrinkite, vykdydami programą.

```
// Inheritance: constructors
//
class Class3 {
        Class3( int i ) {
                System.out.println( "Constructor of Class3  "+i );
        }
}

class Class4 extends Class3 {
         Class4( int i ) {
                super( i ); // calling constructor of superclass Class3
                i++;  // scope of "i" is limited with constructor area!
                System.out.println( "Constructor of Class4  "  +i );
          }
}

public class Class5 extends Class4 {
        Class5( int i ) {
                super( i ); // calling constructor of superclass Class4
                i++;
                System.out.println( "Constructor of Class5  "  +i );
          }
        public static void main ( String [ ] args ) {
                Class5 c5 = new Class5( 123 );
         }
}
```

11 programa

Polimorfizmas. Geresnei polimorfizmo iliustracijai perrašykite programą, kad į masyvą *Shape* elementai būtų sudedami atsitiktine tvarka (naudokite *Math.random* metodą).

```
// Polymorphous methods: 1th example
//
class Shape {
        void draw( ){}
        void erase( ){}
}
class Circle extends Shape {
        void draw( ) {
                System.out.println( "Circle.draw( )" );
         }
        void erase( ) {
                System.out.println( "Circle.erase( )" );
         }
}
class Square extends Shape {
         void draw( ) {
                System.out.println( "Square.draw( )" );
         }
        void erase( ) {
                System.out.println( "Square.erase( )" );
        }
}
class Triangle extends Shape {
        void draw( ) {
                 System.out.println( "Triangle.draw( )" );
         }
         void erase( ) {
                System.out.println( "Triangle.erase( )" );
        }
}

public class Shapes {
         public static void main ( String [ ] args ) {
                Shape[ ] s = {
                        new Circle( ),
                        new Square( ),
                        new Triangle( )  };  // upcasting toward Shape
                for( int i=0; i<s.length; i++ ) {
                         s[ i ].draw( );  // polymorphous calling
                         s[ i ].erase( );
                }
         }
}
```

12 programa

11 programa, perrašyta naudojant abstrakčias klases.

```
// Polymorphous methods: 2nd example
//
abstract class ShapeAbstract {
        abstract void draw( );
        abstract void erase( );
        void print( int i ) {
                System.out.println( "Real method print( "+i+" )" );
         }
}
class Circle extends ShapeAbstract {
         void draw( ) {
                System.out.println( "Circle.draw( )" );
        }
         void erase( ) {
                System.out.println( "Circle.erase( )" );
        }
}
class Square extends ShapeAbstract {
        void draw( ) {
                System.out.println( "Square.draw( )" );
         }
         void erase( ) {
                System.out.println( "Square.erase( )" );
        }
}
class Triangle extends ShapeAbstract {
         void draw( ) {
                System.out.println( "Triangle.draw( )" );
         }
         void erase( ) {
                System.out.println( "Triangle.erase( )" );
         }
}
public class ShapesAbstract {
         public static void main ( String [ ] args ) {
         ShapeAbstract[ ] s = {
                        new Circle( ),
                        new Square( ),
                        new Triangle( )  };  // upcasting
         for( int i=0; i<s.length; i++ ) {
                s[ i ].draw( );  // polymorphous calling!
                s[ i ].erase( );
                s[ i ].print( i );
         }
         }
}
```

13 programa

11 programa, perrašyta naudojant sąsajų klases.

```
// Polymorphous methods: 3rd example
//
interface ShapeInterface {
        int i = 123;
          void draw( );
         void erase( );
}
class Circle implements ShapeInterface {
         public void draw( ) {
                 System.out.println( "Circle.draw( )" );
        }
          public void erase( ) {
                 System.out.println("Circle.erase( )" );
         }
}
class Square implements ShapeInterface {
         public void draw( ) {
                 System.out.println( "Square.draw( )" );
         }
         public void erase( ) {
                 System.out.println( "Square.erase( )" );
        }
}
class Triangle implements ShapeInterface {
         public void draw( ) {
                 System.out.println( "Triangle.draw( )" );
         }
        public void erase( ) {
                 System.out.println( "Triangle.erase( )" );
         }
        static void print( ) {
                 System.out.println( "Triangle.print( "+i+" )" );
          }
}
class Polygon extends Triangle {  }

public class ShapesInterface {
        public static void main ( String [ ] args ) {
                 ShapeInterface[ ] s = {
                                  new Circle( ),
                                  new Square( ),
                                 new Triangle( ),
                                 new Polygon( ),
                 };
                 for( int i=0; i<s.length; i++ ) {
                         s[ i ].draw( );  // polymorphous calling!
```

```
                    s[ i ].erase( );
                }
                Polygon.print(  );
            }
        }
```

14 programa

Klientinė programa. Grafinių komponentų išdėstymo konteineryje tvarkyklė. Pabandykite komponentus išdėstyti savo nuožiūra. Išbandykite kitas išdėstymo tvarkykles (*BorderLayout, GridLayout* ir kt.).

```
// 1th example: buttons
//
//<applet code = Button1 width = 300 height = 200>
//</applet>
//
import javax.swing.*;
import java.awt.*;

public class Button1 extends JApplet{
	Jbutton	b1 = new JButton( "Button 1" ),
			b2 = new JButton( "Button 2" );
	public void init( ){
			Container c = getContentPane( );
			c.setLayout( new FlowLayout( ) );
			 c.add( b1 );
			c.add( b2 );
	}
}
```

15–17 programos

Klientinė programa. Skirtingos įvykių realizavimo sintaksės. Pabandykite perrašyti programas, naudodami skirtingas įvykių registravimo sintakses.

```
// 2nd example: buttons and event, inner class
//
//<applet code = Button2 width = 300 height = 200>
//</applet>
//
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class Button2 extends JApplet{
        JButton b1 = new JButton( "Button 1" ),
                 b2 = new JButton( "Button 2" );
        JTextField tf = new JTextField( 20 );
         class ALC implements ActionListener{
                public void actionPerformed( ActionEvent e ){
                        String buttonName = ( ( JButton )e.getSource( ) ).getText( );
                         tf.setText( buttonName );
                }
         }
        ALC al = new ALC( );
         public void init( ){
                 b1.addActionListener( al );
                 b2.addActionListener( al );
                 Container c = getContentPane( );
                 c.setLayout( new FlowLayout( ) );
                c.add( b1 );
                c.add( b2 );
                 c.add( tf );
          }
}
```

```
// 3rd example: buttons and event, inner classes, other layouts
//
//<applet code = Button3 width = 300 height = 200>
//</applet>
//
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class Button3 extends JApplet{
        JButton b1 = new JButton( "Button 1" ),
                b2 = new JButton( "Button 2" );
        JTextField tf = new JTextField( 20 );
        ActionListener al = new ActionListener( ){       // 2nd syntax' alternative;
                public void actionPerformed( ActionEvent e ){  // to be preferred
                        String buttonName = ( ( JButton )e.getSource( ) ).getText( );
                        tf.setText( buttonName );
                }
        }; // ;!
        public void init( ){
                b1.addActionListener( al );
                b2.addActionListener( new ActionListener( ){
                                                // 3rd syntax' alternative; to be
                                                // preferred for a single addition
                        public void actionPerformed( ActionEvent e ){
                            String buttonName = ( ( JButton )e.getSource( ) ).getText( );
                            tf.setText( buttonName );
                        }
                } );
                Container c = getContentPane( );
                c.add( BorderLayout.EAST, b1 );
                c.add( BorderLayout.WEST, b2 );
                c.add( BorderLayout.CENTER, tf );
        }
}
```

```
// 4th example: buttons and event, alternative listening to the events
//
//<applet code = Button31 width = 300 height = 200>
//</applet>
//
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class Button31 extends JApplet implements ActionListener{
        JButton b1 = new JButton( "Button 1" ),
                b2 = new JButton( "Button 2" );
        JTextField tf = new JTextField( 20 );
        public void init( ){
                b1.addActionListener( this ); // ie, adding applet as a listener
                b2.addActionListener( this );
                Container c = getContentPane( );
                c.add( BorderLayout.NORTH, b1 );
                c.add( BorderLayout.SOUTH, b2 );
                c.add( BorderLayout.CENTER, tf );
        }
        public void actionPerformed( ActionEvent e ){
                String buttonName = ( ( JButton )e.getSource( ) ).getText( );
                tf.setText( buttonName );
        }
}
```

18 programa

Klientinė programa. Reagavimas į kelis pelės įvykius – pranešimų apie įvykusį įvykį (ir įvykio koordinates) spausdinimas klientinės programos lange.

Po kelių įvykių langas prisipildo pranešimų; pelės pėdsakas (spausdinamas tempiant pelę nuspaustu klavišu) taip pat gali pripildyti langą. Pakeiskite programą taip, kad būtų spausdinami tik kelių paskutinių įvykių pranešimai (sakykime, 10-ies įvykių) ir būtų spausdinamas riboto ilgio (tarkime, iš 10-ies simbolių) pelės pėdsakas. Galimas sprendimo kelias: sutalpinti pranešimus ir pėdsaką į masyvus; kai masyvai prisipildo – naujus elementus užrašyti vėl nuo masyvo pradžios.

Atkreipkite dėmesį į metodo *repaint( )* naudojimą.

```
// Applet example: mouse and multiple events
//
//<applet code = Mouse1 width = 800 height = 500>
//</applet>
//
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class Mouse1 extends JApplet
      implements MouseListener, MouseMotionListener{  // applet listens to all
                                                       // mouse events
 String message = "";
  int x=0, y=0;          // where the message is printed
  int xa=0, ya=0;      // where the mouse enters and leaves window
  int i=0;                // counts events "entered" and "exited"

  public void init( ){
    addMouseListener( this );     // ie, adding applet as a listener
    addMouseMotionListener( this );
  }

  public void mouseClicked( MouseEvent me ){
    x = me.getX( )+100;        // to separate messages "clicked" and "pressed"
    y = me.getY( );
    message = "Mouse clicked";
    repaint( );
  }

  public void mouseEntered( MouseEvent me ){
    i++;
    x = 10;
    y = i*10;
    xa = me.getX( );
    ya = me.getY( );
    message = "Mouse entered"+xa+ya;
    repaint( );
```

```
 }

 public void mouseExited( MouseEvent me ){
  i++;
  x = 10;
  y = i*10;
  xa = me.getX( );
  ya = me.getY( );
  message = "Mouse exited"+xa+" "+ya;
  repaint();
 }

 public void mousePressed( MouseEvent me ){
  x = me.getX( );
  y = me.getY( );
  message = "Mouse pressed";
  repaint( );
 }

 public void mouseReleased( MouseEvent me ){
  x = me.getX( )+100;
  y = me.getY( );
  message = "Mouse released";
  repaint( );
 }

 public void mouseDragged( MouseEvent me ){
  x = me.getX( );
  y = me.getY( );
  message = "*";
  showStatus( "Dragging at "+x+", "+y );  // message at the bottom of window
  repaint( );
  }

 public void mouseMoved( MouseEvent me ){
  x = me.getX( );
  y = me.getY( );
  message = " ";
  showStatus( "Moving at "+x+", "+y );
  repaint( );
 }

 public void paint( Graphics g ) {
  g.drawString( message, x, y );
 }
}
```

19 programa

Klientinė programa. Kolekcijos pavyzdys: *HashMap* lentelė. Į teksto lauką perduodami lentelėje saugomi duomenys. Kitu mygtuku tekstinis laukas išvalomas.

1. Į grafinę sąsają įdėkite du naujus mygtukus: vieną nuspaudus, į teksto lauką turi būti išvestos visos lentelės raktų reikšmės natūralia didėjimo tvarka; kitą – visos reikšmės natūralia didėjimo tvarka.
2. Pabandykite klientinės programos atliekamiems veiksmams pritaikyti kitas kolekcijų klases: rinkinius ir sąrašus. Parašykite bent du naujus programos variantus.

```
// Example for JTextArea,collection HashMap
//
//<applet code = TextArea width = 500 height = 500>
//</applet>
//
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*;  //all collections are saved here

public class TextArea extends JApplet {
  JButton b1       = new JButton( "Add data" ),
          b2       = new JButton( "Delete data" );
  JTextArea ta     = new JTextArea( 20,40 );
  HashMap hm   = new HashMap( );
  public void init( ){
    hm.put( "a",new Double( 1. ) );
    hm.put( "b",new Double( 2. ) );
    hm.put( "c",new Double( 3. ) );
    hm.put( "d",new Double( 4. ) );
    hm.put( "e",new Double( 5. ) );
    hm.put( "f",new Double( 6. ) );
    b1.addActionListener( new ActionListener( ){
      public void actionPerformed( ActionEvent ae ){
        Set s = hm.entrySet( ); //all entries (pairs key/value) of hashtable
                        // transmitted to a  set; iterator is available for a set
        Iterator i = s.iterator( ); //method returns iterator object i
        while( i.hasNext( ) ){      //hasNext - method of class Iterator
          Map.Entry me = ( Map.Entry )i.next( ); //method next returns object of
                                                 //type Object; downcasting.
                                                //Entry - inner class of Map
          ta.append( me.getKey( )+" : "
                  +me.getValue( )+"\n"); //method append (JTextArea) appends
                                         //current text to the existing text
        }
      }
    });
    b2.addActionListener( new ActionListener( ){
```

```
      public void actionPerformed( ActionEvent ae ){
        ta.setText( "" ); //ie, deletes text
      }
    } );
    Container c = getContentPane( );
    c.setLayout( new FlowLayout( ) );
    c.add( new JScrollPane( ta ) );    //text area wrapped into a scrollbar
    c.add( b1 );
    c.add( b2 );
  }
}
```

20 programa

Klientinė programa. Duomenų skaitymas iš rinkmenos aplanke, esančiame ten pat, iš kur įkraunamas programos kodas. Išnagrinėkite kodą ir parenkite skaitomą duomenų rinkmeną pagal kode reikalaujamas specifikacijas. Paleiskite programą. Papildykite programą išvestimi į pultą. Palyginkite savo programą su 21 programa.

```
// Example for data input from file on the  directory of
// code location
//
//<applet code = ReadingFile width = 500 height = 500>
//</applet>
//
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.net.*;  // for class URL
import java.io.*;   // for all input classes

public class ReadingFile extends JApplet {
  JButton b1      = new JButton( "Add data" ),
          b2      = new JButton( "Delete data" );
  JTextField tf  = new JTextField( "inputFile.java",20 );
  JTextArea ta  = new JTextArea( 20,40 );
  public class ReadData implements ActionListener{
    public void actionPerformed( ActionEvent ae ){
      try{
        URL url = new URL( getCodeBase( ), tf.getText( ) ); // location,
        ta.setText( url + "\n" );                           // added to text area
        InputStream is = url.openStream( );  // input stream
        BufferedReader in = new BufferedReader( new InputStreamReader( is ) );
        String line;
        while( (line=in.readLine() ) != null )
          ta.append( line + "\n" );
      } catch( Exception e) {
         ta.append( e.toString( ) );
      }
    }
  }
  public class DeleteData implements ActionListener{
    public void actionPerformed( ActionEvent ae ){
      ta.setText( "" );
    }
  }
  public void init( ){
    Container c = getContentPane( );
    c.setLayout( new FlowLayout( ) );
    b1.addActionListener( new ReadData( ) );
    b2.addActionListener( new DeleteData( ) );
    c.add( new JScrollPane( ta ) );
```

```
    c.add( b1 );
    c.add( tf  );
    c.add( b2 );
  }
}
```

21 programa

20 programa, papildyta išvestimi į pultą.

```
// Example for data input from file on the  directory of
// code location. Printing to the client console
//
//<applet code = RWFile width = 500 height = 500>
//</applet>
//
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.net.*;  // for class URL
import java.io.*;   // for all input classes

public class RWFile extends JApplet {
  JButton b1     = new JButton( "Add data" ),
          b2     = new JButton( "Delete data" ),
          b3     = new JButton( "Print data" );
  JTextField tf  = new JTextField( "inputFile.java",20 );
  JTextArea ta   = new JTextArea( 20,40 );
  String s       = "";

  public class ReadData implements ActionListener{
    public void actionPerformed( ActionEvent ae ){
      try{
        URL url = new URL( getCodeBase( ), tf.getText( ) ); // location,
        ta.setText( url + "\n" );                           // added to text area
        InputStream is = url.openStream( );  // input stream
        BufferedReader in = new BufferedReader( new InputStreamReader( is ) );
        String line;
        while( (line=in.readLine( ) ) != null ){
          ta.append( line + "\n" );
          s += line + "\n";
        }
      } catch( Exception e) {
          ta.append( e.toString( ) );
      }
    }
  }

  public class DeleteData implements ActionListener{
    public void actionPerformed( ActionEvent ae ){
      ta.setText( "" );
      s = "";
    }
  }

  public class PrintData implements ActionListener{
```

```
    public void actionPerformed( ActionEvent ae ){
      PrintWriter out = new PrintWriter( System.out, true );
      out.println( s );
    }
  }

  public void init( ){
    Container c = getContentPane( );
    c.setLayout( new FlowLayout( ) );
    b1.addActionListener( new ReadData( ) );
    b2.addActionListener( new DeleteData( ) );
    b3.addActionListener( new PrintData( ) );
    c.add( new JScrollPane( ta ) );
    c.add( b1 );
    c.add( tf  );
    c.add( b2 );
    c.add( b3 );
  }
}
```

22 programa

Programa, nustatanti klasei prieinamų metodų ir konstruktorių sąrašą. Paketo vardas metodams ir konstruktoriams iš *java.lang* paketo į sąrašus neperduodamas. Pabandykite panaikinti paketo vardą klasės *StringTokenizer* metodais. Panaikinkite ne tik *java.lang*, bet ir visų kitų paketų vardus.

```
//
// Utility: list of methods for the indicated class
//         [ possessing indicated substring ]
//         to the screen    and
//         to the file "List.out" in the same directory
//

import java.lang.reflect.*; //for class Class
import java.io.*;

public class ML{

  static final String usage =
    "Use: java ML packageName.ClassName     or\n"+
    "     java ML packageName.ClassName subnameOfMethod";
  static String [ ] ls;  //array for the list
  static int count = 0; //counts the methods found

  static String removeName( String s ) {  //removes "java.lang." from
    int sl    = s.length( );                //the name of method
    int first = s.indexOf( "java.lang." );  //see class String
    if( first == -1 ) return s;
      int last  = first + 10;
      return removeName(
        s.substring( 0,first ) + s.substring( last,sl ) );  //see class String
  }

  public static void main( String args[ ]){
    if( args.length < 1 ) {
      System.out.println( usage );
      System.exit( 1 );   //erroneous exit; see class System
    }
    try {
      Class c           = Class.forName( args[ 0 ] );
      Method [ ] m      = c.getMethods( );   //see class Class
      Constructor [ ] cr = c.getConstructors( ); //same
      ls = new String[ m.length+cr.length ];

      if( args.length == 1 ) { //ie, full list of methods and constructors
        for ( int i = 0; i < m.length; i++ ) //for class given in args[0]
            ls[ count++ ] = "method  "+ removeName( m[ i ].toString( ) );
        for ( int i = 0; i < cr.length; i++ )
            ls[ count++ ]= "constructor  "+ removeName( cr[ i ].toString( ) );
```

```
    }else { //ie, length==2; list of methods containing substring given
      count = 0;            //in args[1]
      for ( int i = 0; i < m.length; i++ ) {
        String ms = m[ i ].toString( );
        if( ms.indexOf( args[ 1 ] )!= -1 )
            ls[ count++ ]= "method  "+ removeName( ms );
      }
      for ( int i = 0; i < cr.length; i++ ) {
        String ms =  cr[ i ].toString( );
        if( ms.indexOf( args[ 1 ] )!= -1 )
            ls[ count++ ]= "constructor  "+ removeName( ms );
      }
    }
   } catch( ClassNotFoundException e ) {
      System.err.println( "Class "+e+" does not exist" ); //errors to the screen
    }
   try{  //output to the screen and file
    PrintWriter out = new PrintWriter( new BufferedWriter( new FileWriter(
                                          "List.out" ))),
              con = new PrintWriter( System.out, true ); //true: empty
    for( int i=0; i<count; i++)  {                       //the buffer
      out.println( i + "  " + ls[ i ] );
      con.println( i + "  " + ls[ i ] );
    }
    out.close( );
   } catch( Exception e ){
      System.err.println( e.toString( ) ); //errors to the screen
    }
  }
}
```

23 programa

Klientinė programa. Kai kurie kiti *Swing* elementai: lentelė, panelis. Perdarykite programą, kad lentelės pavidalą nustatytų „lentelės modelis“.

```
/*Applet with some new components:
  JTable, JPanel, tool-tip.
  Input file located  on the same directory as code:
  collection of lines, each of 3 String-type
  elements divided by space. Later on first two elements
  of each line are put to the array1 while the third one to array2.
  array1 makes the model for the table; array2 plays role of
  explanatory text.
  Name of file may be supplied in the text field.
  All the error stream is put to the text area. */

//<applet code = Table width =500 height=550>
//</applet>

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;

public class Table extends JApplet{
  String array1[ ][ ] = new String[ 20 ][ 2 ],
         array2[ ]    = new String[ 20 ],
         colHeads[ ] = { "Number","Name" };
  JButton b1    = new JButton( "Add data" ),
          b2    = new JButton( "Delete data" );
  JTextField tf = new JTextField( "Input.txt",20 );
  JTextArea ta  = new JTextArea( 3,40 );
  JTable table  = new JTable( array1, colHeads ){
    public String getToolTipText( MouseEvent me ){ //overriding method
       String tip = "";                             //that sets "tip-text"
       Point p = me.getPoint( );                    //to the table
       int row = table.rowAtPoint( p );
       tip = array2[ row ];
       return tip;
    }
  };

  class ReadingFile implements ActionListener{
   public void actionPerformed( ActionEvent ae ){
    try{
     URL url = new URL( getCodeBase( ), tf.getText( ) );
     InputStream is = url.openStream( );
     BufferedReader in = new BufferedReader( new InputStreamReader( is ));
     String line = "";
```

```
   int count = 0;
     while( ( line=in.readLine( ))! = null ){
       array2[ count ] = line.substring( line.lastIndexOf(" ")+1,line.length( ) );
       array1[ count ][ 0 ] = line.substring( 0, line.indexOf(" ") );
       array1[ count ][ 1 ] = line.substring( line.indexOf(" ")+1,
                                  line.lastIndexOf(" ") );
       count++;
     }
   } catch( ArrayIndexOutOfBoundsException e ){
      ta.append( "Dimensions of table only until 20 rows allowed" );
   } catch( Exception e ) {
      ta.append( "Error reading data: "+ e );
   }
   table  = new JTable( array1, colHeads );
   showStatus( "Data added" );
   repaint( );
  }
 }

 class DeletingData implements ActionListener{
  public void actionPerformed( ActionEvent ae ){
   for( int i=0; i<20; i++ ){
     array1[i][0] = "";
     array1[i][1] = "";
     array2[i]    = "";
   }
   table  = new JTable( array1, colHeads ); //modifying table
   showStatus( "Data deleted" );
   repaint( );
  }
 }

 public void init( ){
   b1.addActionListener( new ReadingFile( ) );
   b2.addActionListener( new DeletingData( ) );
     Container c = getContentPane();
     JScrollPane spt = new JScrollPane( table );
       JPanel p1 = new JPanel( );
       p1.setLayout( new FlowLayout( ) );
       p1.add( tf  );
       p1.add( b1 );
       p1.add( b2 );
         JPanel p2 = new JPanel( );
         JScrollPane spta = new JScrollPane( ta );
         p2.add( spta );
           c.add( spt,BorderLayout.NORTH );
           c.add( p1,BorderLayout.CENTER );
           c.add( p2,BorderLayout.SOUTH );
  }
}
```

24 programa

23 programa su spausdinimo galimybėmis.

```
//
//<applet code = Tabpr width =600 height=500>
//</applet>

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.awt.print.*;
import javax.swing.table.*;

public class Tabpr extends JApplet{
  String array1[ ][ ] = new String[ 20 ][ 2 ],
           array2[ ]   = new String[ 20 ],
         colHeads[ ] = { "Number","Name" };
  JButton b1    = new JButton( "Add data" ),
          b2    = new JButton( "Delete data" ),
          b3    = new JButton( "Print data" );
  JTextField tf = new JTextField( "Input.txt",20 );
  JTable table  = new JTable( array1, colHeads ){
    public String getToolTipText( MouseEvent me ){  //overriding method
       String tip = "";                              //that sets "tip-text"
       Point p = me.getPoint( );                     //to the table
       int row = table.rowAtPoint( p );
       tip = array2[ row ];
       return tip;
    }
  };

  class ReadingFile implements ActionListener{
   public void actionPerformed( ActionEvent ae ){
    try{
     URL url = new URL( getCodeBase( ), tf.getText( ) );
     InputStream is = url.openStream( );
     BufferedReader in = new BufferedReader( new InputStreamReader( is ));
     String line = "";
     int count = 0;
       while( (line = in.readLine( ))! = null ){
         array2[ count ] = line.substring( line.lastIndexOf(" ")+1,line.length( ) );
         array1[ count ] [ 0 ] = line.substring( 0, line.indexOf(" ") );
         array1[ count ][ 1 ] = line.substring( line.indexOf(" ") + 1,
                                line.lastIndexOf(" ") );
         count++;
       }
    } catch( ArrayIndexOutOfBoundsException e ){
```

```
   System.err.println( "Dimensions of table only until 20 rows allowed" );
  } catch( Exception e ) {
   System.err.println( "Error reading data: " +  e );
  }
  table  = new JTable( array1, colHeads );
  showStatus( "Data added" );
  repaint();
 }
}

class DeletingData implements ActionListener{
 public void actionPerformed( ActionEvent ae ){
  for( int i=0; i<20; i++ ){
   array1[ i ][ 0 ] = "";
   array1[ i ][ 1 ] = "";
   array2[ i ]    = "";
  }
  table  = new JTable( array1, colHeads ); //modifying table
  showStatus( "Data deleted" );
  repaint( );
 }
}

class PrintingData implements ActionListener, Printable{
  public void actionPerformed( ActionEvent ae ) {
    PrinterJob printJob = PrinterJob.getPrinterJob( );
    printJob.setPrintable( this );
      if ( printJob.printDialog( ) ) {
         try {
            printJob.print( );
         } catch ( Exception e ) {
           System.err.println( "Error printing data: " +  e );
         }
      }
  }
  public int print( Graphics g, PageFormat pf, int pi )
                                        throws PrinterException {
    Graphics2D g2 = ( Graphics2D )g;
    table.paint( g2 );
    return Printable.PAGE_EXISTS;
  }
}



public void init( ){
  b1.addActionListener( new ReadingFile( ) );
  b2.addActionListener( new DeletingData( ) );
  b3.addActionListener( new PrintingData( ) );
    Container c = getContentPane( );
```

```
    JScrollPane sp = new JScrollPane( table );
      JPanel p = new JPanel( );
      p.setLayout( new FlowLayout( ) );
      p.add( tf  );
      p.add( b1 );
      p.add( b2 );
      p.add( b3 );
        c.add( sp,BorderLayout.NORTH );
        c.add( p,BorderLayout.SOUTH );
  }
}
```

25 programa

Klientinė programa. Judanti antraštė atskirame nuo programos pagrindinio vykdymo srauto sraute. Perrašykite programą, naudodami alternatyvią srauto realizavimo sintaksę.

```
//Applet with moving banner
//
//<applet code = Banner width =500 height=100>
//</applet>

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

public class Banner extends JApplet{

  JTextField tf   = new JTextField( 20 );
  JButton   b1    = new JButton( "Start moving" ),
            b2    = new JButton( "Stop  moving" );
  MovingBanner mb;


  class MovingBanner implements Runnable{

    String message = "   A simple moving banner   ";
    Thread t;
    volatile boolean moving;

    public void start( ){
      moving = true;
      t = new Thread( this );
      t.start( );
    }

    public void stop( ){
      moving = false;
    }

    public void run( ){
      char c;
      while( moving ){
        try{
          repaint( );
          Thread.sleep( 100 );
          c = message.charAt( 0 );
          message  = message.substring( 1, message.length( ) );
          message += c;
          tf.setText( message );
        } catch( InterruptedException ie ){}
```

```
    }
   }
  }


  public void init( ){
   b1.addActionListener( new ActionListener( ){
    public void actionPerformed( ActionEvent ae ){
     mb = new MovingBanner( );
     mb.start( );
    }
   });
   b2.addActionListener( new ActionListener( ){
    public void actionPerformed( ActionEvent ae ){
     mb.stop( );
    }
   });
   Container c = getContentPane( );
   c.setLayout( new FlowLayout( ) );
   c.add( tf  );
   c.add( b1 );
   c.add( b2 );
  }

}
```

26 programa

Eilė. Programa su srautų sinchronizavimu. Išjunkite srautų sinchronizaciją, pabandykite įvairias pauzes ir įsitikinkite, kad eilei realizuoti sinchronizavimas yra būtinas.

```
//Model of queue Q: Producer produces number n, Consumer takes it
//Synchronization. Inter-thread collaboration
//
class Q{
  int n;
  boolean go = false;
  synchronized int get( ){
    if( !go )
      try{
        wait( );
      } catch( InterruptedException ie ){}
    System.out.println( "Got: "+ n );
    go = false;
    notify( );
    return n;
  }
  synchronized void put( int n ){
    if( go )
      try{
        wait( );
      } catch( InterruptedException ie ){}
    this.n = n;
    System.out.println( "Put: "+ n );
    go = true;
    notify( );
  }
}

class Producer implements Runnable{
  Q q;
  Producer( Q q ){
    this.q = q;
    new Thread( this,"Producer" ).start( );
  }
  public void run( ){
    int i = 0;
    while( true ){
      q.put( i++ );
      try{
        Thread.sleep( 1000 );
      } catch( InterruptedException ie ){}
    }
  }
}
```

```
class Consumer implements Runnable{
  Q q;
  Consumer( Q q ){
    this.q = q;
    new Thread( this,"Consumer" ).start( );
  }
  public void run( ){
    while( true ) q.get( );
  }
}

class InfiniteQueue{
  public static void main( String [ ] args ){
    System.out.println( "Ctrl-C for finish" );
      Q q = new Q( );
      new Producer( q );
      new Consumer( q );
  }
}
```

27 programa

Klientinė programa, paleidžianti norimą kiekį nepriklausomų vykdymo srautų. Pertvarkykite programą, kad viename sraute būtų valdomos kelių laukelių spalvos. Pažiūrėkite, kiek srautų pajėgia paleisti jūsų kompiuteris.

Pakeiskite klientinę programą taip, kad vienu metu būtų keičiama spalva visos eilutės laukeliams.

```
//Blinking random-colored boxes. <grid*grid> boxes in all.
//Blinking rate: <pause>, in milisec.
//Applet launches individual thread for each box.
//Applet illustrates the efficiency of multitasking.
//
//<applet code = MThreads width = 500 height = 500>
//</applet>
//

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

class Box extends JPanel implements Runnable{ //separate colored box.
  private Thread t;                    //Rendered in separate thread
  private int pause;
  boolean running = true;
  private Color clr = gettingColor( );

  private Color gettingColor( ){     //random color for the box
    int   red   = ( int ) ( Math.random( ) * 255 ),
        green = ( int ) ( Math.random( ) * 255 ),
         blue  = ( int ) ( Math.random( ) * 255 );
    return new Color( red,green,blue ); //one of Color's constructors
  }

  public void paintComponent( Graphics g ){  //painting the colored box
    super.paintComponent( g );
    g.setColor( clr );
    Dimension d = getSize( ); //class has two fields: "width, height"
    g.fillRect( 0,0,d.width-1,d.height-1 );
  }

  public Box( int pause ){
    this.pause = pause;
    t = new Thread( this );
    t.start( );
  }

  public void run( ){
    while( running ){ //will be stopped by closing frame-window for all boxes
      clr = gettingColor( );  //re-coloring the box
```

```
      repaint( );
      try{
        t.sleep( pause );    //<pause> milisec
      } catch( InterruptedException ie ){}
    }
  }
}


class Boxes extends JFrame{  //panel for <grid*grid> boxes
  int grid, pause;
  public Boxes( int grid, int pause ){
    this.grid = grid;
    this.pause = pause;
    setSize( 300,300 );
    Container cf = getContentPane( );
    cf.setLayout( new GridLayout( grid,grid ) );
    for( int i=0; i<grid*grid; i++ )
       cf.add( new Box( pause ) );
    addWindowListener( new WindowAdapter( ){
      public void windowClosing( WindowEvent we ) {
        dispose( );      //response to the frame-window buttons: close the window
      }
    });
  }
}


public class MThreads extends JApplet{  //monitor program
  private int grid = 5;
  private int pause = 100;
  JLabel l1 = new JLabel( "Enter number of rows and columns" ),
         l2 = new JLabel( "Enter delay time (milisec)" ),
         l3 = new JLabel( "Messages" );
  JTextField t1 = new JTextField( 10 ),
             t2 = new JTextField( 10 );
  JTextArea  ta = new JTextArea( 3,20 );
  JButton b1 = new JButton( "Start" ),
          b2 = new JButton( "Re-enter" );
  JFrame boxes;

  public void init( ){
    Container c = getContentPane( );
    JPanel p11 = new JPanel( ),
           p12 = new JPanel( );
    p11.setLayout( new GridLayout( 3,3 ) );
    p12.setLayout( new FlowLayout( ) );
    t1.addActionListener( new ActionListener( ){
      public void actionPerformed( ActionEvent ae ){
        ta.setText("");
```

```
    String s = t1.getText( );
    try{
      grid = Integer.parseInt( s );
    } catch( NumberFormatException nfe ){
      ta.append( "Erroneous input for number of rows."+
                  " Re-enter or use default: 5 \n" );
      grid = 5;
    }
    if( grid<1 ) {
      ta.append( "Erroneous input for number of rows \n"+
                  "Using default: 5  \n" );
      grid = 5;
    }
    ta.append( "Entered: number of rows  "+grid+"\n" );
  }
});
t2.addActionListener( new ActionListener( ){
  public void actionPerformed( ActionEvent ae ){
    ta.setText("");
    String s = t2.getText( );
    try{
      pause = Integer.parseInt( s );
    } catch( NumberFormatException nfe ){
      ta.append( "Erroneous input for pause."+
                  " Re-enter or use default: 100 milisec \n" );
      pause = 100;
    }
    if( pause<1 ) {
      ta.append( "Erroneous input for pause \n"+
                  "Using default: 100 milisec \n" );
       pause = 100;
    }
    ta.append( "Entered: pause (milisec)  "+pause+"\n" );
  }
});
p11.add( l1 ); p11.add( t1 ); p11.add( l2 ); p11.add( t2 );
p11.add( l3 ); p11.add( ta );

b1.addActionListener( new ActionListener( ){
  public void actionPerformed( ActionEvent ae ){
    boxes = new Boxes( grid,pause );
    boxes.setVisible( true );
  }
});
b2.addActionListener( new ActionListener( ){
  public void actionPerformed( ActionEvent ae ){
    t1.setText( "" );
    t2.setText( "" );
    ta.setText( "" );
    boxes.dispose( );
```

```
        repaint( );
      }
    });
    p12.add( b1 );
    p12.add( b2 );

     c.add( p11,BorderLayout.NORTH );
     c.add( p12,BorderLayout.SOUTH );
  }
}
```

28 programa

Klientinė programa, iliustruojanti prioritetų įtaką vykdant srautus. Pabandykite įvairius prioritetus įvairiam skaitiklių auginimo laikui (tik atsiminkite, kad skaitiklio formatas – *int*).

```
//Creating threads with different priorities
//in applet context
//Multiply re-evaluation of counters enabled.
//
//<applet code = Threads4 width =600 height=550>
//</applet>

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


class Counter implements Runnable{

  int count;
  Thread t;
  volatile boolean running;
  public Counter( int priority ){
    count = 0;
    t = new Thread( this );
    t.setPriority( priority );
  }

  public void run( ){
    while( running ) count++;
  }

  public void start( ){
    running = true;
    t.start( );
  }

  public void stop( ){
    running = false;
  }

}


public class Threads4 extends JApplet{

  Counter c1,c2;
  JButton b1 = new JButton( "Start threads" ),
          b2 = new JButton( "Stop threads " );
```

```
 JTextArea ta = new JTextArea( 3,50 );

 public void init( ) {
  Thread.currentThread().setPriority( Thread.MAX_PRIORITY );
  b1.addActionListener( new ActionListener( ){
    public void actionPerformed( ActionEvent ae ){
     c1 = new Counter( Thread.NORM_PRIORITY+2 );
     c2 = new Counter( Thread.NORM_PRIORITY-2 );
      c1.start( );
      c2.start( );
      ta.append("   Threads started \n" );
    }
  });
  b2.addActionListener( new ActionListener( ){
    public void actionPerformed( ActionEvent ae ){
      c1.stop( );
      c2.stop( );
      ta.append( "Low-priority  counter: "+c2.count+"\n" );
      ta.append( "High-priority counter: "+c1.count+"\n" );
    }
  });
  try{
    Thread.sleep( 1000 );
  } catch( InterruptedException ie ){}

  Container c = getContentPane( );
  c.setLayout( new FlowLayout( ) );
  c.add( b1 );
  c.add( b2 );
  c.add( ta );
 }
}
```

29–30 programos

Serverio ir kliento programos. Serveris aptarnauja tik vieną klientą. Paleiskite programų ūkį.

```
//The simplest server: getting messages from
//client and printing them
//

import java.io.*;
import java.net.*;

public class SimpleServer{
  public static final int PORT = 2000;
  public static void main( String args[ ] ) throws IOException{
    ServerSocket ss = new ServerSocket( PORT );
    System.out.println( "Launched: "+ss );
    try{
      Socket s = ss.accept(); //waiting for connection
      try{
        System.out.println( "Connection got: "+s );
        BufferedReader in = new BufferedReader(
                        new InputStreamReader(
                        s.getInputStream( ) ) );
        PrintWriter out =   new PrintWriter(
                        new BufferedWriter(
                        new OutputStreamWriter(
                        s.getOutputStream( ) ) ),true );
        while( true ){                      //server will be closed after
          String str = in.readLine( ); //client's message "END"
          if( str.equals( "END" )) break;
          System.out.println( "Got from client: "+str );
          out.println( str );
        }
      } finally{  //socket must be closed!
          System.out.println( "Disconnecting client" );
          s.close( );
      }
    } finally{ //socket must be closed!
        System.out.println( "Closing server" );
        ss.close( );
    }
  }
}
```

```
//The simplest client: sending messages to
//server and getting messages from server
//

import java.io.*;
import java.net.*;

public class SimpleClient{
  public static void main( String args[ ] ) throws IOException{
    InetAddress ia = InetAddress.getByName( null ); //ie, server on the
                                                    //same machine
    System.out.println( "Address of server: "+ia );
    Socket s = new Socket( ia,SimpleServer.PORT ); //code "SimpleServer must
                                                   //be in the same package
    System.out.println( "Launched socket: "+s );
    try{
      BufferedReader in = new BufferedReader(
                      new InputStreamReader(
                      s.getInputStream( ) ) );
      PrintWriter out =   new PrintWriter(
                      new BufferedWriter(
                      new OutputStreamWriter(
                      s.getOutputStream( ) ) ),true );
      for( int i=0; i<10; i++ ){
        String msg = "Client's message "+i;
        out.println( msg );
        System.out.println( "Sent to server: "+msg );
        String str = in.readLine( );
        System.out.println( "Got from server: "+str );
      }
      out.println( "END" ); //"END" appointed for the finish
    } finally{
       System.out.println( "Closing client" );
       s.close( );
    }
  }
}
```

31–32 programos

Serverio ir kliento programos. Serveris aptarnauja kelis klientus. Kiekvienas klientas aptarnaujamas atskirame vykdymo sraute.

Pakeiskite programas taip, kad serveris gyvuotų ne amžinai, o tik tam tikrą laiką arba išsijungtų aptarnavęs nurodytą kiekį klientų.

```
//The simplest multi-client server: getting messages from
//clients and printing them
//One server-thread for one client
//

import java.io.*;
import java.net.*;

class ServeOneClient extends Thread{
  private Socket s;
  private BufferedReader in;
  private PrintWriter out;
  private static int counter = 0;
  private int id = counter++;
    public ServeOneClient( Socket s ) throws IOException{
      this.s = s;
      in = new BufferedReader(
           new InputStreamReader(
           s.getInputStream( ) ) );
      out = new PrintWriter(
            new BufferedWriter(
            new OutputStreamWriter(
            s.getOutputStream( ) ) ),true );
      start( );
    }
    public void run( ){
      try{
        while( true ){
          String str = in.readLine( );
          if( str.equals( "END" )) break;
          System.out.println( "Got from client: "+str );
          out.println( str );
        }
      } catch( IOException ioe ){
          System.err.println( "I/O error" );
      } finally{
          try{
            System.out.println( "Disconnecting client "+id );
            s.close( );
          } catch( IOException ioe ){
              System.err.println( "Error while closing socket" );
          }
      }
```

```
    }
}

public class SimpleMultiServer{
  public static final int PORT = 2000;
  public static void main( String args[ ] ) throws IOException{
    ServerSocket ss = new ServerSocket( PORT );
    System.out.println( "\n Press Ctrl/C for termination \n" );
    System.out.println( "Launched: "+ss );
    try{
      while( true ){    //endless loop!
        Socket s = ss.accept( ); //waiting for connection
        try{
          new ServeOneClient( s );
        } catch( IOException ioe ){
            s.close( );
        }
      }
    } finally{
        ss.close( );
    }
  }
}
```

```
//Code generates simplest clients: sending messages to
//server and getting messages from server.
//Until MAX_CLIENTS clients may exist simultaneously.
//One thread for one client
//
//Results of code depend exclusively on the period
//pause. Experiment on it!
//

import java.io.*;
import java.net.*;

class OneClient extends Thread{
  private Socket s;
  private BufferedReader in;
  private PrintWriter out;
  private static int counter = 0;
  private int id = counter++;     //ID-number of client
  private static int threadCount = 0; //number of "living" clients
  public static int threadCounting( ){
    return threadCount;
  }
  public OneClient( InetAddress ia,int PORT ){
    System.out.println( "Creating client No "+id );
    threadCount++;  //count in this client
    try{
      s = new Socket( ia,PORT );
    } catch( IOException ioe ){
       System.err.println( "Error while creating socket No " + id );
    }
    try{
      in =  new BufferedReader(
           new InputStreamReader(
           s.getInputStream( ) ) );
      out = new PrintWriter(
           new BufferedWriter(
           new OutputStreamWriter(
           s.getOutputStream( ) ),true );
      start( );
    } catch( IOException ioe1 ){
       try{
         s.close( );
       } catch( IOException ioe2 ){
           System.err.println( "Unable to close socket No "+id );
       }
    }
  }
  public void run( ){
    try{
      for( int i=0; i<10; i++ ){
```

```
        String msg = "Client's No "+id+" message "+i;
        out.println( msg );
        System.out.println( "Sent to server: "+msg );
        String str = in.readLine( );
        System.out.println( "Got from server: "+str );
       }
      out.println( "END" );
    } catch( IOException ioe ){
        System.out.println( "I/O error" );
    } finally{
        try{
          s.close( );
          System.out.println( "Disconnecting socket No "+id );
        } catch( IOException ioe ){
            System.out.println( "Unable to close socket No "+id );
        }
        threadCount- -; //client is closed; decrease the number of clients
    }
  }
}

public class SimpleMultiClient{
  static final int MAX_CLIENTS = 3;
  public static void main( String args[ ] ) throws IOException, InterruptedException{
    InetAddress ia = InetAddress.getByName( null );
    System.out.println( "Address of server: "+ia );
    for( int i=0; i<20; i++ ){
      if( OneClient.threadCounting() < MAX_CLIENTS )
        new OneClient( ia,SimpleMultiServer.PORT );
      Thread.currentThread().sleep( 1 );
    }
  }
}
```