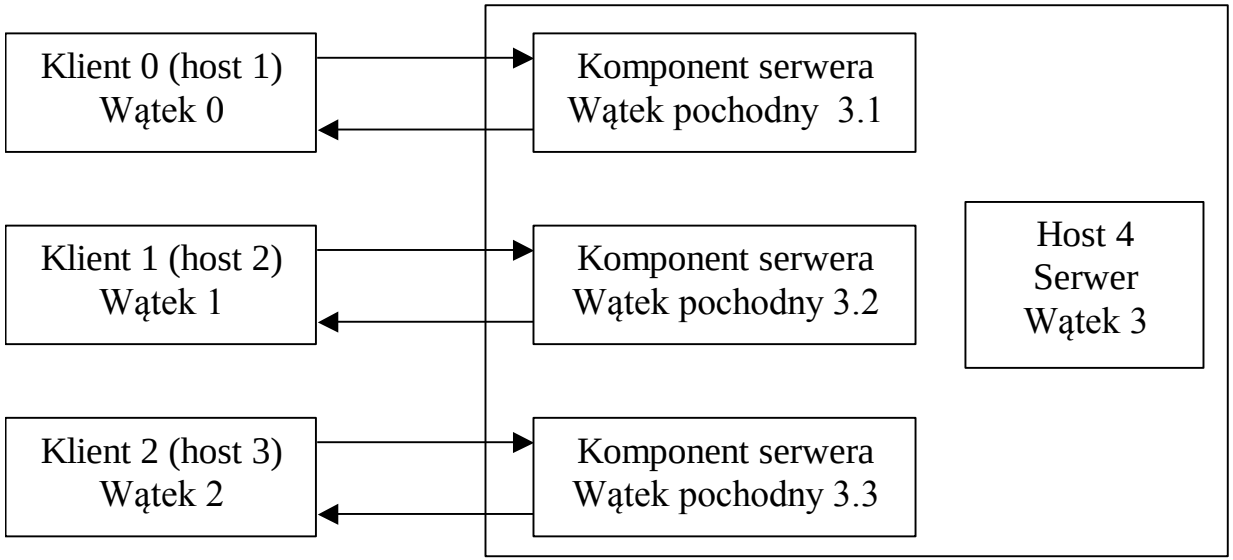
Wykład 2 – część 2

Gniazda
Custom Networking

https://docs.oracle.com/javase/tutorial/networking/index.html

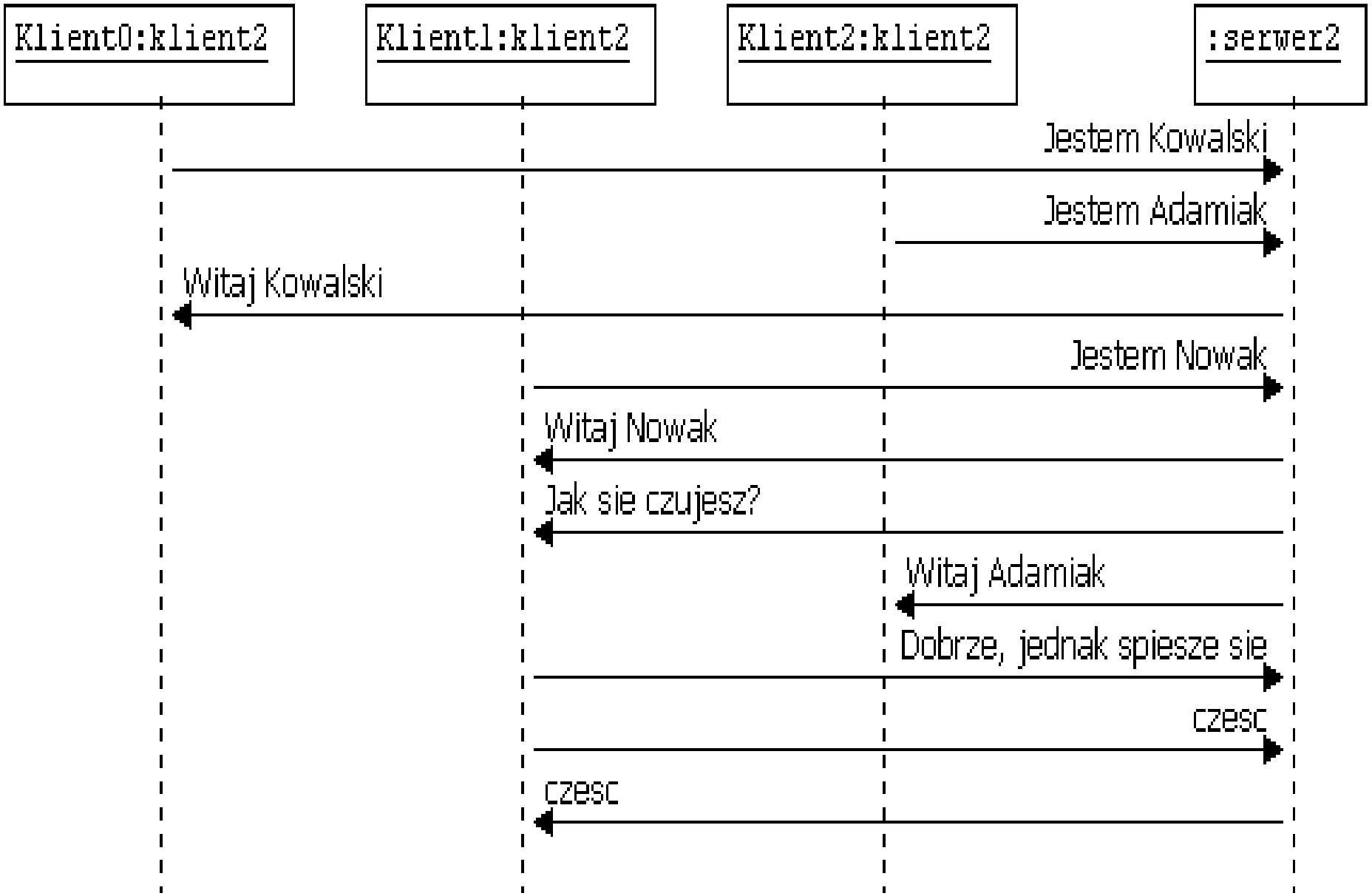
Aplikacja wielowątkowa – prosty komunikator

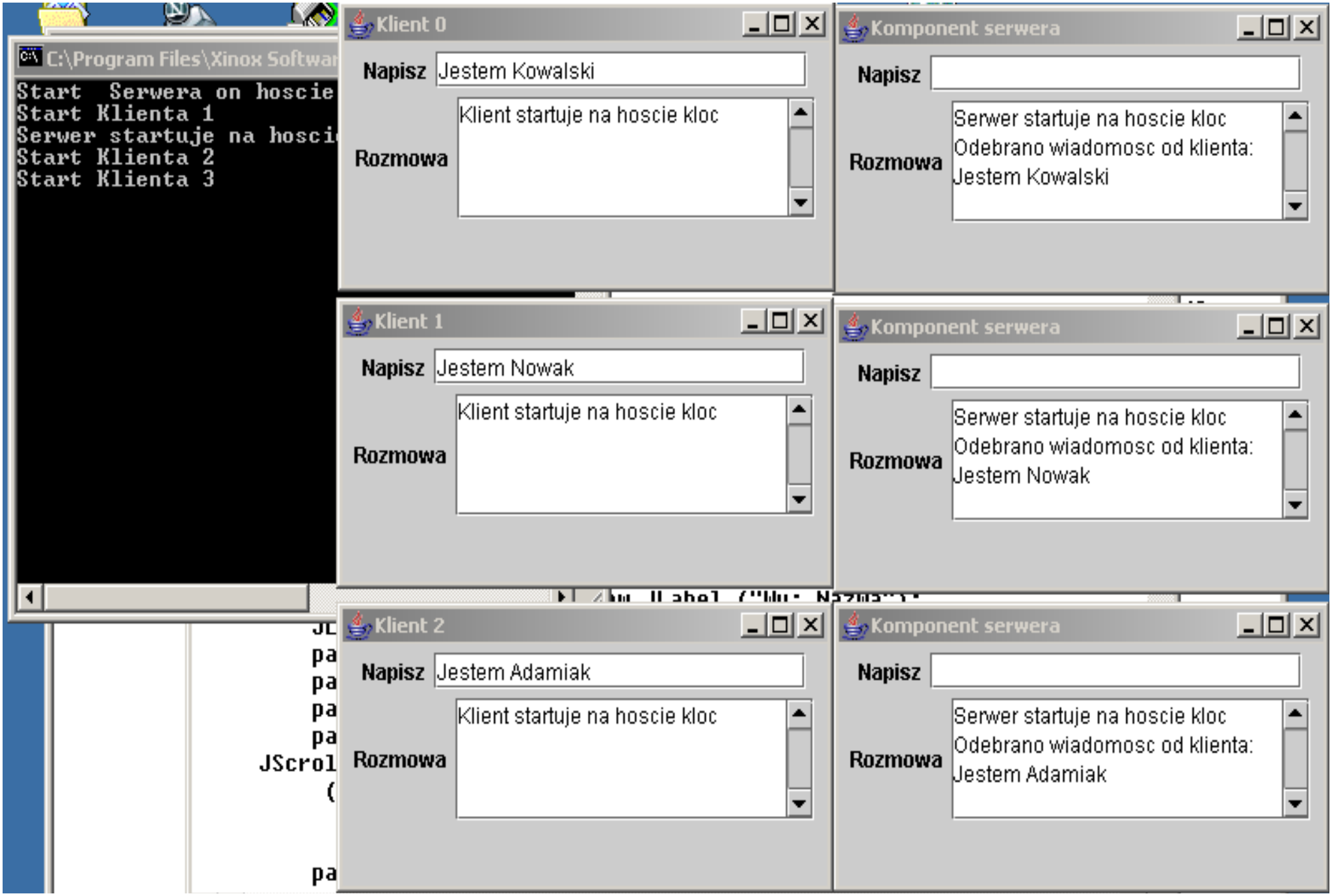


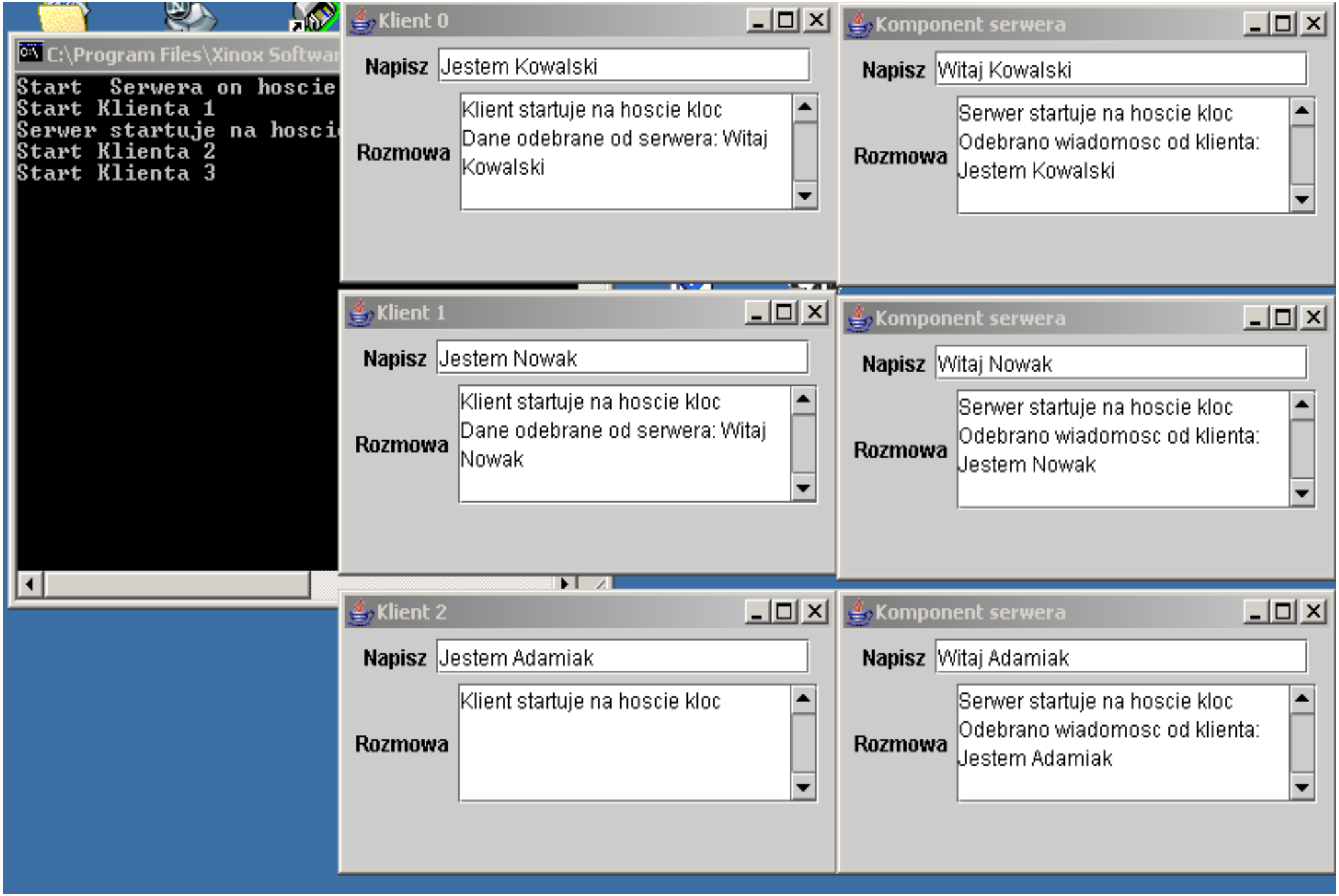
Klient: **private int** sPort *//port serwera*
private String host *//nazwa hosta serwera*
private Socket s *//gniazdo klienta do komunikacji z serwerem, który znajduje*
//się na porcie sPort i na komputerze host
private ObjectOutputStream output
private ObjectInputStream input

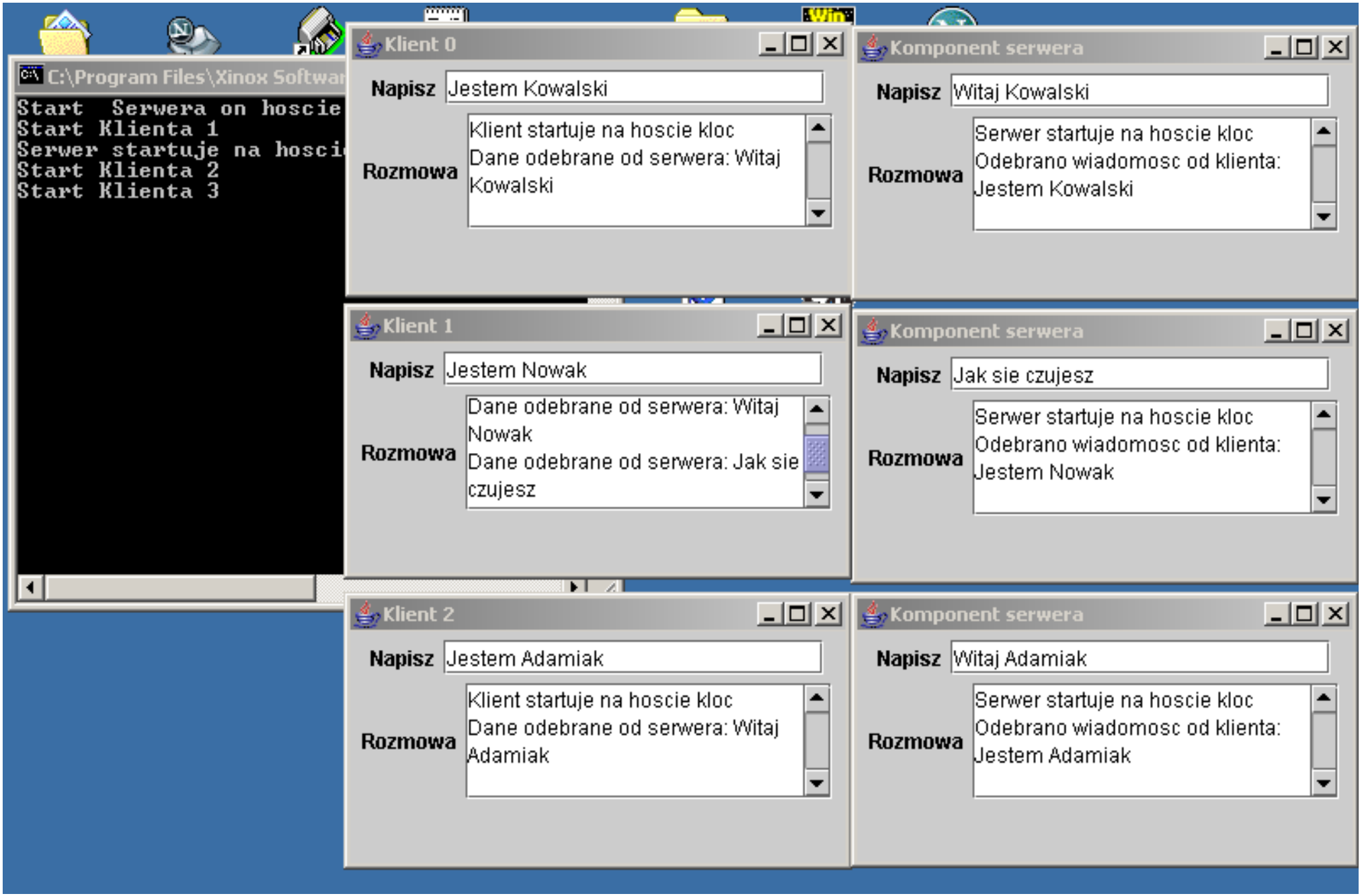
Serwer: **private int** sPort
private String host
private ServerSocket serwer *//gniazdo do wykrywania połączeń z klientem*

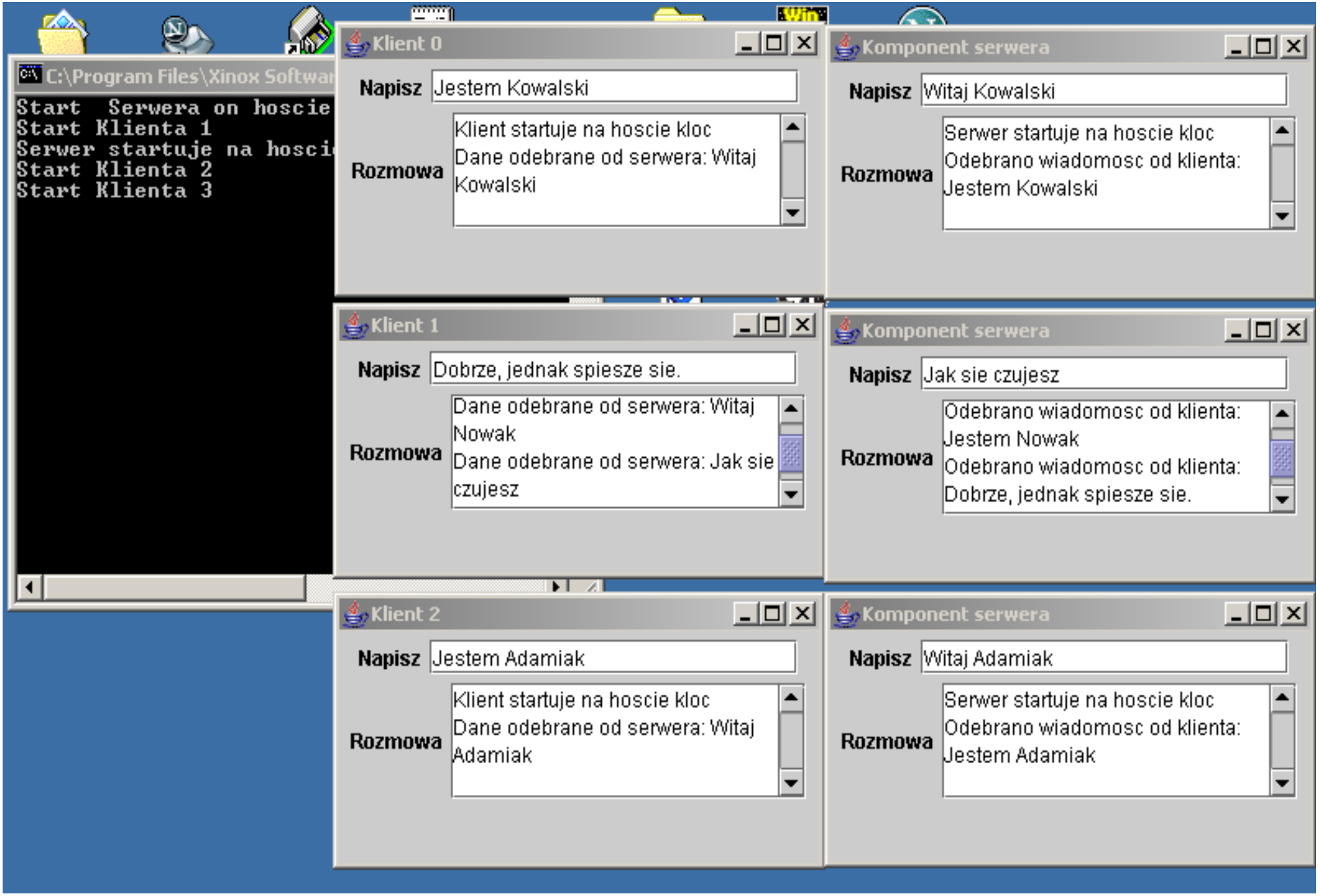
Komponent serwera: **private** Socket s *//gniazdo do komunikacji z klientem*
private ObjectOutputStream output
private ObjectInputStream input

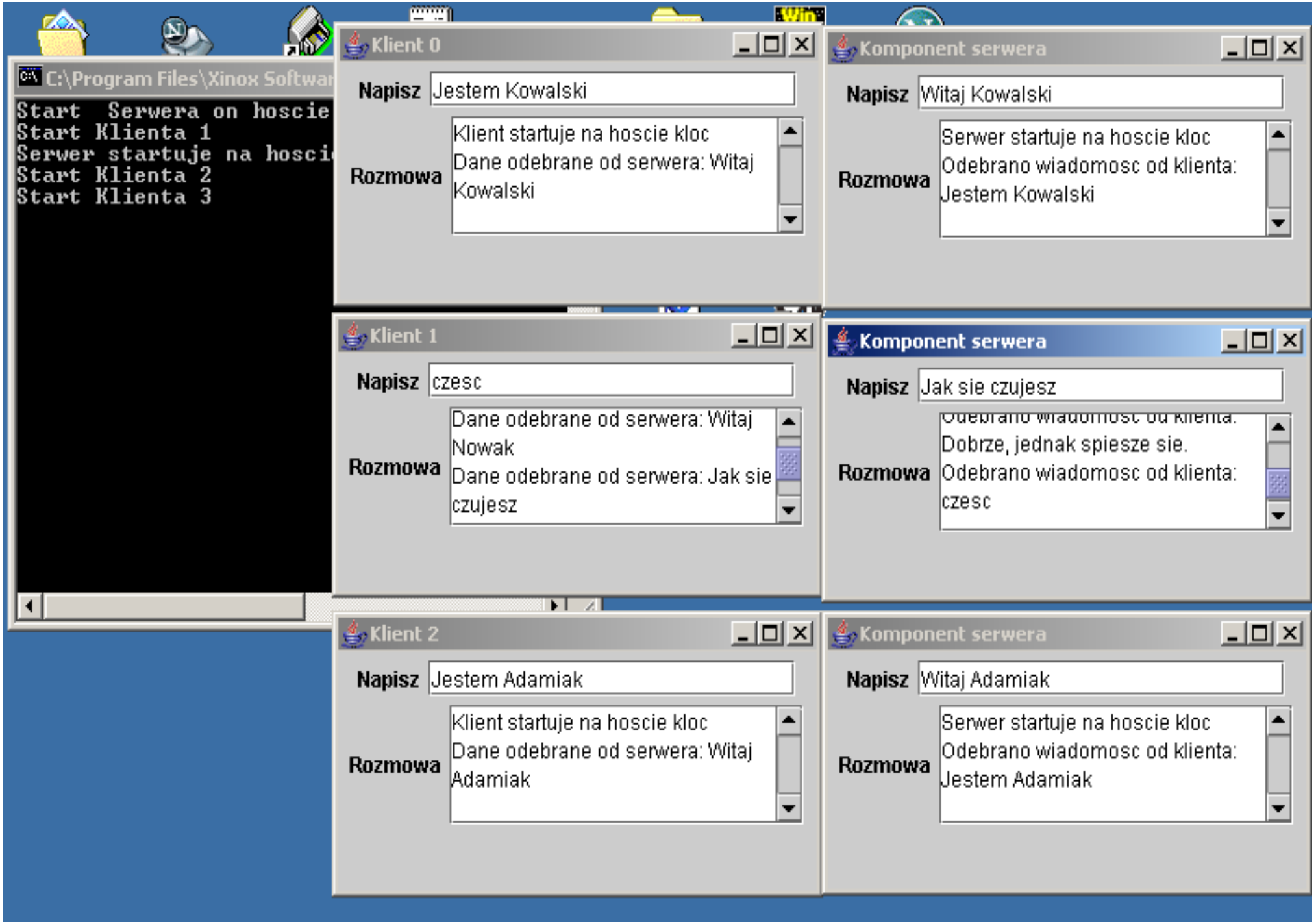

Klient0:klient2
Klient1:klient2
Klient2:klient2
:serwer2
Jestem Kowalski
Jestem Adamiak
Witaj Kowalski
Jestem Nowak
Witaj Nowak
Jak sie czujesz?
Witaj Adamiak
Dobrze, jednak spiesze sie
czesc
czesc

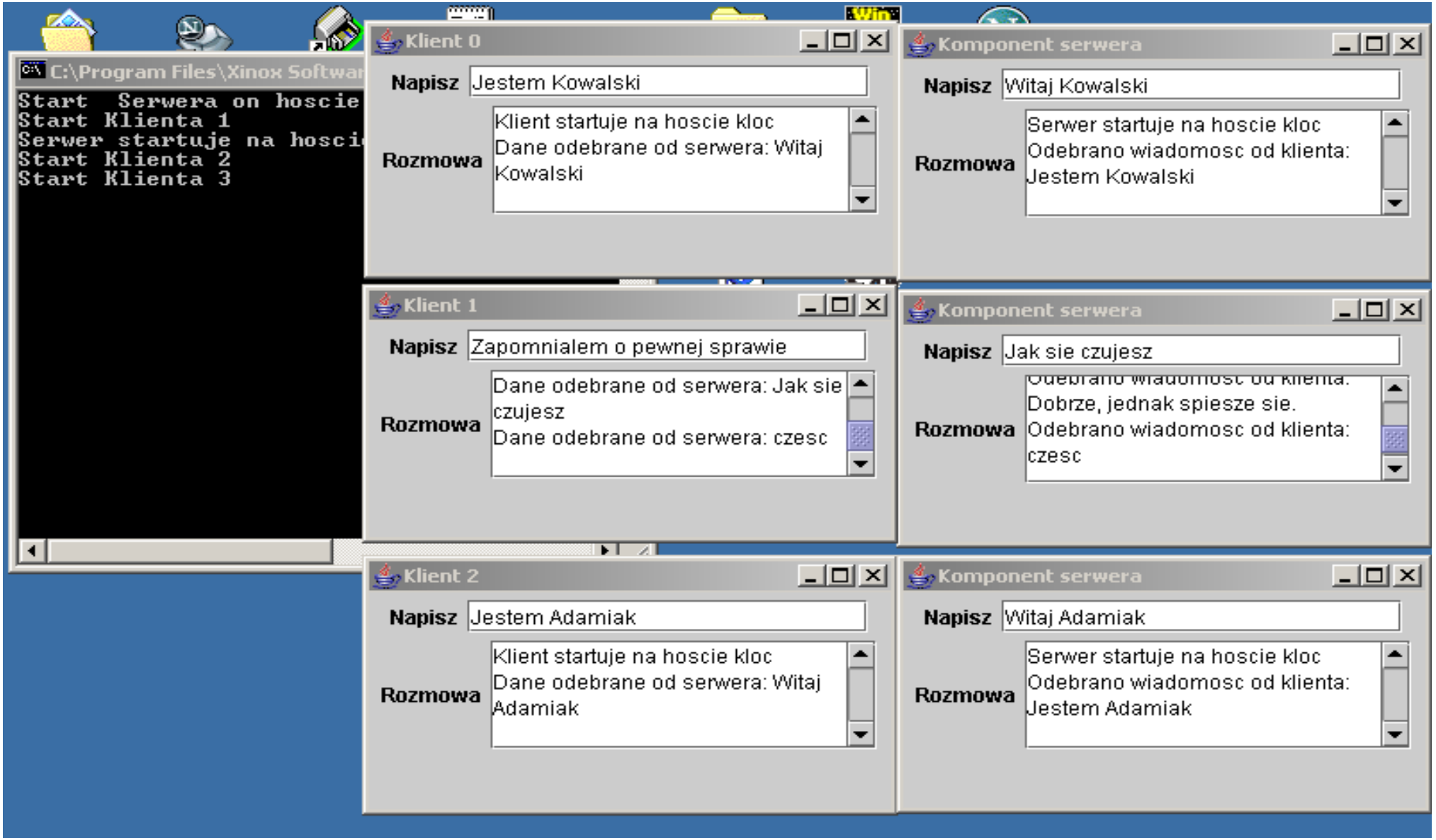

C:\Program Files\Xinox Softwar
Start Serwera on hoscie
Start Klienta 1
Serwer startuje na hosci
Start Klienta 2
Start Klienta 3
Klient 0
Napisz
Jestem Kowalski
Rozmowa
Klient startuje na hoscie kloc
Komponent serwera
Napisz
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta:
Jestem Kowalski
Klient 1
Napisz
Jestem Nowak
Rozmowa
Klient startuje na hoscie kloc
Komponent serwera
Napisz
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta:
Jestem Nowak
Klient 2
Napisz
Jestem Adamiak
Rozmowa
Klient startuje na hoscie kloc
Komponent serwera
Napisz
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta:
Jestem Adamiak


C:\Program Files\Xinox Softwar
Start Serwera on hoscie
Start Klienta 1
Serwer startuje na hosci
Start Klienta 2
Start Klienta 3
Klient 0
Napisz
Jestem Kowalski
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Kowalski
Komponent serwera
Napisz
Witaj Kowalski
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Kowalski
Klient 1
Napisz
Jestem Nowak
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Nowak
Komponent serwera
Napisz
Witaj Nowak
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Nowak
Klient 2
Napisz
Jestem Adamiak
Rozmowa
Klient startuje na hoscie kloc
Komponent serwera
Napisz
Witaj Adamiak
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Adamiak


C:\Program Files\Xinox Softwar
Start Serwera on hoscie
Start Klienta 1
Serwer startuje na hosci
Start Klienta 2
Start Klienta 3
Klient 0
Napisz
Jestem Kowalski
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Kowalski
Komponent serwera
Napisz
Witaj Kowalski
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Kowalski
Klient 1
Napisz
Jestem Nowak
Rozmowa
Dane odebrane od serwera: Witaj Nowak
Dane odebrane od serwera: Jak sie czujesz
Komponent serwera
Napisz
Jak sie czujesz
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Nowak
Klient 2
Napisz
Jestem Adamiak
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Adamiak
Komponent serwera
Napisz
Witaj Adamiak
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Adamiak


C:\Program Files\Xinox Softwar
Start Serwera on hoscie
Start Klienta 1
Serwer startuje na hosci
Start Klienta 2
Start Klienta 3
Klient 0
Napisz
Jestem Kowalski
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Kowalski
Komponent serwera
Napisz
Witaj Kowalski
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Kowalski
Klient 1
Napisz
Dobrze, jednak spiesze sie.
Rozmowa
Dane odebrane od serwera: Witaj Nowak
Dane odebrane od serwera: Jak sie czujesz
Komponent serwera
Napisz
Jak sie czujesz
Rozmowa
Odebrano wiadomosc od klienta: Jestem Nowak
Odebrano wiadomosc od klienta: Dobrze, jednak spiesze sie.
Klient 2
Napisz
Jestem Adamiak
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Adamiak
Komponent serwera
Napisz
Witaj Adamiak
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Adamiak


C:\Program Files\Xinox Softwar
Start Serwera on hoscie
Start Klienta 1
Serwer startuje na hosci
Start Klienta 2
Start Klienta 3
Klient 0
Napisz
Jestem Kowalski
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Kowalski
Komponent serwera
Napisz
Witaj Kowalski
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Kowalski
Klient 1
Napisz
czesc
Rozmowa
Dane odebrane od serwera: Witaj Nowak
Dane odebrane od serwera: Jak sie czujesz
Komponent serwera
Napisz
Jak sie czujesz
Rozmowa
Dobrze, jednak spiesze sie.
Odebrano wiadomosc od klienta: czesc
Klient 2
Napisz
Jestem Adamiak
Rozmowa
Klient startuje na hoscie kloc
Dane odebrane od serwera: Witaj Adamiak
Komponent serwera
Napisz
Witaj Adamiak
Rozmowa
Serwer startuje na hoscie kloc
Odebrano wiadomosc od klienta: Jestem Adamiak



Koniec komunikacji między klientem 1 a komponentem serwera po otrzymaniu słowa cześć komponent serwera potwierdza odebranie „czesc” wysłaniem słowa „czesc” i kończy połączenie. Komunikat „Zapomnialem o pewnej sprawie” nie zostanie wysłany przez klienta.

```
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;

public class Ramka extends JPanel implements ActionListener {
    JTextField nazwa = new JTextField(20);
    JTextArea komentarz = new JTextArea(4, 18);
    Komponent_nowegoklienta3 komp1 = null;
    nowyKlient3 komp2 = null;

    public void panel() {
        setSize(300, 160);
        nazwa.addActionListener(this);
        JLabel etykieta_nazwy = new JLabel("Napisz");
        JLabel etykieta_komentarza = new JLabel("Rozmowa");
        komentarz.setLinewrap(true);
        komentarz.setWrapStyleWord(true);
        add(etykieta_nazwy);
        add(nazwa);
        add(etykieta_komentarza);
        add(komentarz);
        JScrollPane obszar_przewijany1 = new JScrollPane(komentarz,
                ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
                ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        add(obszar_przewijany1);
    }
```

```
public Ramka(Komponent_nowegoklienta3 komp) {
    this.komp1 = komp;
    panel(); }

public Ramka(nowyKlient3 komp) {
    this.komp2 = komp;
    panel(); }

public void actionPerformed(ActionEvent evt) {
    Object zrodlo = evt.getSource();
    if (zrodlo == nazwa) {
        if (komp1 != null) {
            komp1.wyslij(nazwa.getText());
        } else {
            komp2.wyslij(nazwa.getText());   }
    }
    repaint();
}

public void GUI(String tytul) {
    JFrame okno = new JFrame(tytul);
    okno.setSize(300, 200);
    okno.setContentPane(this);
    okno.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    okno.setVisible(true);
}}
```

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;

public class Komponent_nowegoklienta3 implements Runnable {
    private Socket gniazdo_klienta;
    private ObjectOutputStream wyjscie;
    private ObjectInputStream wejscie;
    private Ramka panel = new Ramka(this);          // panel okienka graficznego
    String m1="",m="";
    public Komponent_nowegoklienta3(Socket gniazdo_klienta,
                        ObjectInputStream wejscie, ObjectOutputStream wyjscie)  {
       this.gniazdo_klienta = gniazdo_klienta;
       this.wejscie = wejscie;
       this.wyjscie = wyjscie;
       panel.GUI("Komponent klienta po stronie serwera");/*tworzenie okienka graficznego*/
     }
//wysyłanie wiadomośći - metoda wywoływana w metodzie actionPerformed panela
    public void wyslij(String lan) {
       m1=lan;
       if (!m1.equals("czesc")&&gniazdo_klienta!=null) {
          try {
             wyjscie.writeObject(m1);
          } catch (Exception e) {
              System.out.println("Wyjatek serwera " + e); }
       } }
```

```
//odbior wiadomosci
    public void run() {
        String pom;
        try {
            panel.komentarz.setText("Serwer startuje na hoscie " +
                    InetAddress.getLocalHost().getHostName() + "\n");
            while (true) {
                m = (String) wejscie.readObject();
                pom = panel.komentarz.getText();
                panel.komentarz.setText(pom + "Odebrano wiadomosc od klienta: " + m + "\n");
                if (m.equals("czesc")) {
                    m1="czesc";
                    wyjscie.writeObject(m1);   //wysyla "czesc" po odbiorze "czesc"
                    break;                     //i przerywa odbior- przechodzi do bloku zamykania polaczenia
                }
            }
            //zamykanie polaczenia
            wejscie.close();
            wyjscie.close();
            gniazdo_klienta.close();
            gniazdo_klienta = null;
        } catch (Exception e) {
            System.out.println("Wyjatek serwera1 " + e);
        }
    }
}
```

```
public class Serwer3 implements Runnable
{
public void run()                    //metoda serwera wykonywana wątku - rozpoznaje połączenia
 {   Socket gniazdo_klienta;          //z kolejnymi klientami
     ObjectOutputStream output;
     ObjectInputStream input;
     while (true) {
          try {
              gniazdo_klienta = serwer.accept();        //oczekiwanie na klienta
             if (gniazdo_klienta == null) {
                 System.out.println("Minal czas akceptacji");
                 continue;
              }
              output = new ObjectOutputStream(gniazdo_klienta.getOutputStream());
              output.flush();
              input = new ObjectInputStream(gniazdo_klienta.getInputStream());
                                  //tworzenie watku do obslugi klienta
              Komponent_nowegoklienta3 komp_klienta =
                    new komponent_nowegoklienta3(gniazdo_klienta, input, output);
              Thread watek_komponentu_klienta = new Thread(komp_klienta);
             watek_komponentu_klienta.start();
          } catch (IOException e) {
              System.out.println("Nie mozna polaczyc sie z klientem " + e);
              System.exit(1); }        // przerwanie pracy serwera nie jest zalecane w praktyce
       } }
```

```
  private int  sPort;
  private ServerSocket serwer;

 public Serwer3(int port_)
  {  sPort = port_;
     try
       {                                                  //serwer tworzy gniazdo do wykrywania
         serwer = new ServerSocket(sPort);     //połączeń z klientami
         System.out.println("Serwer startuje na hoscie " +
                 InetAddress.getLocalHost().getHostName());
        } catch(IOException e)
            { System.out.println(e);  }
    }

//ten program należy uruchomić jako pierwszy
  public static void main(String args[]) throws Exception
   {
     int Port = 15000;
     Serwer3 s2 = new serwer3(Port);
     Thread t = new Thread(s2);
     t.start();
   }
 }
```

```
import java.io.ObjectInputStream; //ten program naley uruchomi jako drugi
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;

public class Klient3 implements Runnable {
    private int port;
    private Socket gniazdo_klienta;
    private ObjectOutputStream wyjscie;
    private ObjectInputStream wejscie;
    private String host;
    private Ramka panel = new Ramka(this);          //panel okienka graficznego
    private String m1="",m="";

    Klient3(String host_, int port_) {
        host = host_;
        port = port_;
        try {
            gniazdo_klienta = new Socket(host, port);
            wejscie = new ObjectInputStream(gniazdo_klienta.getInputStream());
            wyjscie = new ObjectOutputStream(gniazdo_klienta.getOutputStream());
            wyjscie.flush();
             panel.GUI("Klient");           /*tworzenie okienka graficznego*/
        } catch (Exception e) {    }
    }
```

```
//odbior wiadomosci

    public void run() {
        String pom;
        try {
            panel.komentarz.setText("Klient startuje na hoscie " +
                    InetAddress.getLocalHost().getHostName() + "\n");
            do {
                if(!m1.equals("czesc")&&!m.equals("czesc")) {
                m = (String) wejscie.readObject();
                pom = panel.komentarz.getText();
                panel.komentarz.setText(pom + "Dane odebrane od serwera: " + m + "\n");}
            } while(!m.equals("czesc"));
             //zamykanie polaczenia
            wyjscie.close();
            wejscie.close();
            gniazdo_klienta.close();
            gniazdo_klienta = null;
        } catch (Exception e) {
            System.out.println("Wyjatek klienta " + e);
        }
    }
```

```
//wysyłanie wiadomośći - metoda wywoływana w metodzie actionPerformed panela
    public void wyslij(String lan) {
        m1=lan;
         if (gniazdo_klienta != null) {
            try {
             wyjscie.writeObject(lan);
            } catch (Exception e) {
                System.out.println("Wyjatek klienta " + e);
            }
        }
    }

public static void main(String args[]) throws Exception {
        String s = InetAddress.getLocalHost().getHostName();
        //String s = InetAddress.getByName("kloc").getHostName();
                        //poszukiwania serwera w sieci
        Klient3 k2 = new Klient3(s, 15000);
        Thread t = new Thread(k2);
        t.start();
    }
}
```

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

public class Tester
 {
    public Tester()
     {
       super();
     }
```

/ Program Testera startuje najpierw tworząc obiekt serwera* ***server*** *i wstawia go do wątku i następnie tworzy tablicę* ***clients*** *zawierającą trzy wątki, każdy z klientem. Po starcie serwer tworzy gniazdo* ***ServerSocket*** *o nazwie* ***serwer*** *i jego metodą* ***accept*** *oczekuje na zgłoszenie klienta. Każdy z klientów po wystartowaniu w niezależnym wątku tworzy gniazdo typu* ***Socket*** *znając port i nazwę hosta, na którym znajduje się serwer oraz tworzy strumienie wejścia/wyjścia typu* ***ObjectOutputStream*** *o nazwie* ***output*** *oraz typu* ***ObjectInputStream*** *o nazwie* ***input*** *i wysyła do serwera komunikat (np. Jestem Kowalski). Kiedy metoda* ***accept*** *wykryje połączenie z klientem, zwraca powiązany z klientem obiekt typu* ***Socket*** *o nazwie* ***s****. Serwer tworzy strumienie wejścia/wyjścia typu* ***ObjectOutputStream*** *o nazwie* ***output*** *oraz typu* ***ObjectInputStream*** *o nazwie* ***input****. Następnie tworzy obiekt typu* ***server_komp*** *i wstawia go wątku pochodnego przekazując mu gniazdo* ***s*** *oraz strumienie* ***input*** *i* ***output****. Za jego pośrednictwem może serwer porozumiewać się z klientem i działać jednocześnie niezależnie tzn. identyfikować za pomocą metody* ***accept*** *gniazda* ***ServerSocket*** *połączenia z nowymi klientami, tworząc nowe wątki pochodne z obiektami typu* ***server_komp****. /*

```
public static void main(String args[])
    {
       int NUMCLIENTS = 3;
       int sPort=5000;
       String host;
       Thread server;
       Thread clients[]= new Thread[NUMCLIENTS];
       try
         {
            host = InetAddress.getLocalHost().getHostName();
          System.out.println("Start  Serwera on hoscie "+ host);
          server = new Thread(new serwer3(sPort), host);
          server.start();
          for (int i=0;i<clients.length;i++)
            {
               System.out.println("Start Klienta " + (i +1));
             clients[i]=new Thread(new  Klient3(host,sPort));
             clients[i].start();
            }
       } catch (UnknownHostException e)
         {
          System.out.println("Nieznany wyjątek podczas startu klienta");   }
    }
  }
```