

# ĆWICZENIE

TERMIN ODDANIA ĆWICZENIA

**29 KWIECIEŃ 2010**

Stworzyć przy pomocy CORB-y w języku JAVA serwer i klient dla komunikacji typu „Chat”. Polegać będzie na tym, że klienci będą się rejestrować na serwerze, będą mogli wysyłać wiadomości i wiadomość wysłana od jednego klienta będzie rozsyłana przez serwer do wszystkich pozostałych klientów.

Do tego celu, zarówno serwer jak i klient muszą być „serwerami” w rozumieniu CORB-y. Posłużyć się można następującymi definicjami interfejsów:

#### module chat {

##### module Client{

```
interface ChatClient {
    // metoda wywoływana przez serwer czata,
    //nakazująca klientowi wyświetlenie tekstu w oknie rozmowy
    void printString(in string str);
};
```

```
};
```

##### module Server{

```
interface ChatServer {
    // rejestracja klienta
    void register(in Object client);

    // metoda wywoływana przez klienta czata, nakazująca
    //serwerowi wysłanie tekstu do wszystkich klientów
    void sendString(in string str);
};
```

```
};
```

```
};
```

Po stronie serwera, należy zachować referencje do wszystkich zarejestrowanych klientów (np. w formie wektora). Przy przesyłaniu wiadomości dalej, konieczny jest blok „try...catch” aby wychwycić te referencje, które nie są już aktualne (np. klient zakończył działanie) i usunąć je z listy.

Serwer powinien rejestrować się w usłudze nazwowej i poprzez nią powinien być znajdowany przez klientów.

Klient powinien być wykonany wraz z GUI umożliwiającym wpisanie i wysłanie wiadomości, oraz listę otrzymanych wiadomości. Można skorzystać z klasy *ChatClientWindow*, której kod znajduje się na ostatniej stronie. Możliwy schemat klienta powinien odpowiadać poniższemu diagramowi:

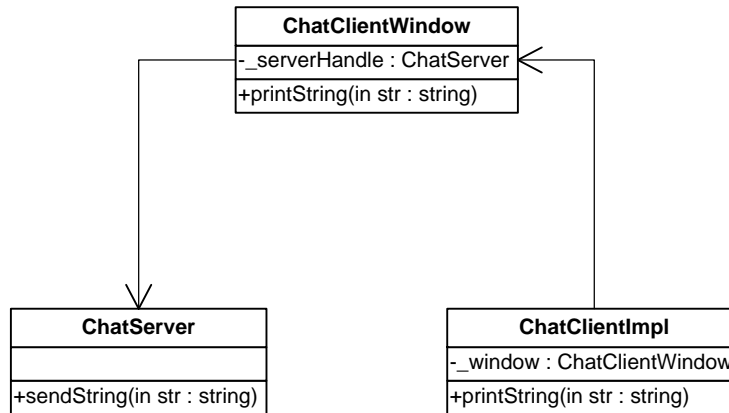


DIAGRAM 1

Obrazuje on następujące zależności: Serwant klienta (ChatClientImpl) musi znać powołane okno ChatClientWindow by móc zawołać na nim wyświetlenie nowej wiadomości. Okno natomiast musi znać referencję do proxy do serwera, by móc zawołać wysłanie wpisanej w oknie wiadomości do innych użytkowników.

Możliwe jest wykonanie aplikacji na zasadzie listenera lub w dowolny inny sposób.

**Powyższe wymagania pozwalają uzyskać 15pkt.**

**By uzyskać 20pkt należy** przerobić interfejsy i kod tak, by każdy klient rejestrował się w serwerze podając nazwę użytkownika („Basia”, „Marek”) i były one używane przy rozsyłaniu wiadomości (aby klienci wiedzieli, od kogo pochodzi wiadomość).

## ChatClientWindow.java

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

public class ChatClientWindow extends JFrame {
    private BorderLayout borderLayout = new BorderLayout();
    private JTextField inputText = new JTextField();
    private JScrollPane outputTextScroller = new JScrollPane();
    private JTextArea outputTextArea = new JTextArea();

    public ChatClientWindow() {
        try {
            jbInit();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        this.getContentPane().setLayout(borderLayout);
        this.setTitle("Chat window");
        inputText.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                inputText_actionPerformed(e);
            }
        });
        this.getContentPane().add(inputText, BorderLayout.SOUTH);
        outputTextScroller.getViewport().add(outputTextArea, null);
        this.getContentPane().add(outputTextScroller, BorderLayout.CENTER);
        outputTextArea.setEditable(false);
        this.setSize(400, 200);
    }

    public void printString(String str) {
        outputTextArea.append(str+ "\n");
    }

    private void inputText_actionPerformed(ActionEvent e) {
        //TODO: wyślij tekst pobrany "inputText.getText();" na serwer.
        inputText.setText("");
    }
}
```