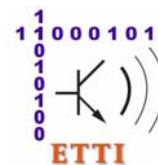




18/12/2007

## Catedra de Telecomunicatii



## Programare Orientata spre Obiecte (POO)

## Laborator 5

Interfete grafice cu utilizatorul (GUI). Tratarea evenimentelor.  
Aplicatii bazate pe socket-uri flux (TCP) Java

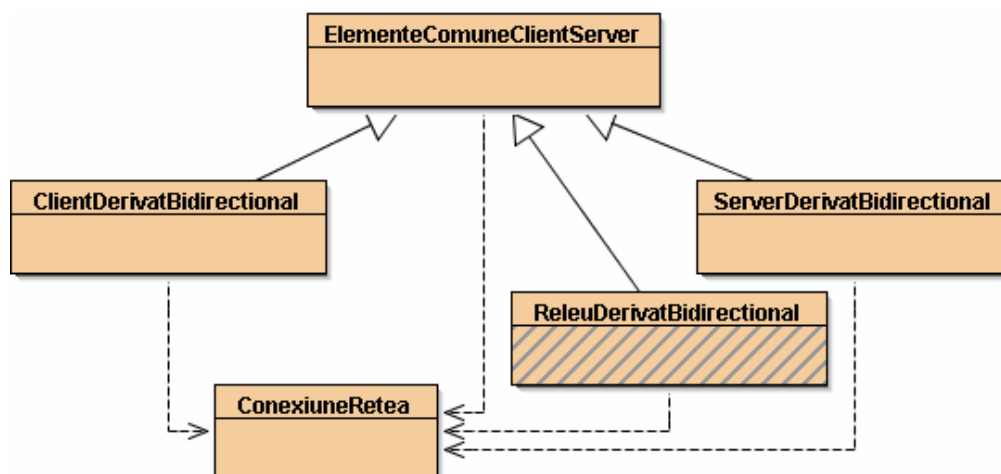
## 5.1. Descrierea laboratorului

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:

- [Aplicatii client-server bazate pe socket-uri TCP care folosesc mostenirea](#) (II)
- [Interfete grafice cu utilizatorul](#) (GUI), [Interfete swing](#) (pdf local), [Interfete swing](#) (pagina externa)
- [Precizari privind colocviul final](#)

## 5.2. Programe de lucru cu socket-uri – clienti si servere (II)

## 5.2.1. Clasele ClientDerivatBidirectional si ServerDerivatBidirectional

Superclasa ElementeComuneClientServer:

```
1 import java.net.*;
2 import java.io.*;
3 import javax.swing.JOptionPane;
4
5 public class ElementeComuneClientServer {
6     protected ConexiuneRetea conexiune;
7     protected Socket socketTCP;
8     protected int portTCP;
9
10    public ElementeComuneClientServer(String tip) throws IOException {
11        portTCP = Integer.parseInt(JOptionPane.showInputDialog(tip +
12            ": introduceti numarul de port al serverului"));
13    }
14 }
```

**Clasa utilitara ConexiuneRetea care incapsuleaza tratarea conexiunilor TCP:**

```
1 import java.net.*;
2 import java.io.*;
3 import java.util.Scanner;
4
5 public class ConexiuneRetea {
6     private Socket conexiune;
7     private Scanner scannerTCP;
8     private PrintStream printerTCP;
9     public ConexiuneRetea(Socket conexiune) throws IOException {
10        this.conexiune = conexiune;
11        this.scannerTCP = new Scanner(conexiune.getInputStream());
12        this.printerTCP = new PrintStream(conexiune.getOutputStream());
13    }
14    public String nextLine() {
15        return this.scannerTCP.nextLine();
16    }
17    public int nextInt() {
18        return this.scannerTCP.nextInt();
19    }
20    public void printLine(String text) {
21        this.printerTCP.println(text);
22        this.printerTCP.flush();
23    }
24 }
```

**Clasa ClientDerivatBidirectional mosteneste si extinde clasa ElementeComuneClientServer:**


```
1 import java.net.*;
2 import java.io.*;
3 import javax.swing.JOptionPane;
4
5 public class ClientDerivatBidirectional extends ElementeComuneClientServer {
6     private InetAddress adresaIP;
7
8     public ClientDerivatBidirectional() throws IOException {
9         super("Client");
10        adresaIP = InetAddress.getByName(JOptionPane.showInputDialog(
11            "Client: introduceti adresa serverului"));
12        socketTCP = new Socket(adresaIP, portTCP); // Creare socket
13        conexiune = new ConexiuneRetea(socketTCP);
14    }
15
16    public static void main (String args[]) throws IOException {
17        ClientDerivatBidirectional client = new ClientDerivatBidirectional();
18        String mesaj;
19
20        while(true) {
21            mesaj = JOptionPane.showInputDialog(
22                "Client: introduceti mesajul de trimis");
23            client.conexiune.printLine(mesaj);
24            mesaj = client.conexiune.nextLine();
25            JOptionPane.showMessageDialog(null, "Client: s-a primit "+mesaj);
26            if (mesaj.equals(".")) break; // Testarea conditiei de oprire
27        }
28        client.socketTCP.close(); // Inchiderea socketului si a fluxurilor
29        JOptionPane.showMessageDialog(null, "Client: Bye!");
30    }
31 }
```

**Clasa ServerDerivatBidirectional mosteneste si extinde clasa ElementeComuneClientServer:**

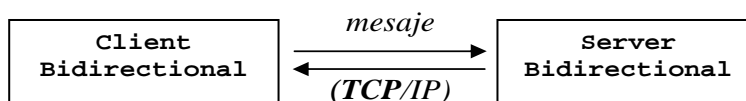
```

1  import java.net.*;
2  import java.io.*;
3  import javax.swing.JOptionPane;
4
5  public class ServerDerivatBidirectional extends ElementeComuneClientServer {
6      private ServerSocket serverTCP;
7
8      public ServerDerivatBidirectional() throws IOException {
9          super("Server");
10         serverTCP = new ServerSocket(portTCP);           // Creare socket server
11         socketTCP = serverTCP.accept();                 // Creare socket
12         conexiune = new ConexiuneRetea(socketTCP);
13     }
14
15     public static void main (String args[]) throws IOException {
16         ServerDerivatBidirectional server = new ServerDerivatBidirectional();
17         String mesaj;
18
19         while(true) {
20             mesaj = server.conexiune.nextLine();
21             JOptionPane.showMessageDialog(null, "Server: s-a primit mesajul "+mesaj);
22             mesaj = JOptionPane.showInputDialog(
23                 "Server: introduceti mesajul de trimis");
24             server.conexiune.println(mesaj);
25             if (mesaj.equals(".")) break;           // Testarea conditiei de oprire
26         }
27         server.socketTCP.close();           // Inchiderea socketului si a fluxurilor
28         JOptionPane.showMessageDialog(null, "Server: Bye!");
29     }
30 }

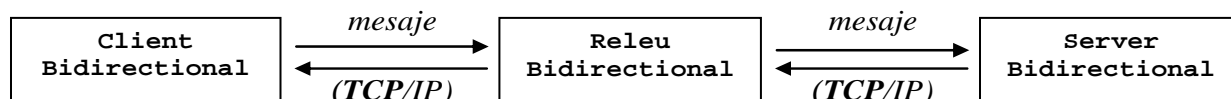
```

**Nu uitati:** Daca bara de stare a executiei este activa (  ) **verificati cu Alt+Tab** daca a aparut o fereastra Java (in spatele ferestrelor vizibile).

**Nu uitati:** **Pentru a opri executia**, **right click** pe  si **Reset Machine** (sau **Ctrl+Shift+Tab**).

**In laborator:**

1. Lansati in executie BlueJ. **Inchideti** proiectele anterioare (Ctrl+W). **Creati** un proiect *socket* (Project->New Project..., selectati **D:/**, **POO2007**, **numarul grupei**, si scrieti **socket**).
2. In proiectul *socket* **creati** clasele **ConexiuneRetea**, **ClientDerivatBidirectional**, **ServerDerivatBidirectional** si **ElementeComuneClientServer** folosind codurile date mai sus.
3. **Compilati** codurile.

**5.2.2. Clasa releu intre client si server**

**Adaugarea unei clase ReleuDerivatBidirectional pe post de retransmitator al mesajelor intre ClientDerivatBidirectional si ServerDerivatBidirectional:**

```
1 import java.net.*;
2 import java.io.*;
3 import javax.swing.JOptionPane;
4
5 public class ReleuDerivatBidirectional extends ElementeComuneClientServer {
6     private ServerSocket serverTCPReleu;
7
8     private InetAddress adresaIPServerFinal;
9     private ElementeComuneClientServer parteClient =
10         new ElementeComuneClientServer("Releu/server");
11
12     public ReleuDerivatBidirectional() throws IOException {
13         super("Releu/client");
14         adresaIPServerFinal = InetAddress.getByName(JOptionPane.showInputDialog(
15             "Releu/client: introduceti adresa serverului local"));
16         parteClient.socketTCP = new Socket(adresaIPServerFinal,
17             parteClient.portTCP); // Socket spre server
18         parteClient.conexiune = new ConexiuneRetea(parteClient.socketTCP);
19
20         serverTCPReleu = new ServerSocket(portTCP); // Creare socket server
21         socketTCP = serverTCPReleu.accept(); // Socket spre client
22         conexiune = new ConexiuneRetea(socketTCP);
23     }
24
25     public static void main (String args[]) throws IOException {
26         ReleuDerivatBidirectional releu = new ReleuDerivatBidirectional();
27         String mesaj;
28
29         while(true) {
30             mesaj = releu.conexiune.nextLine();
31             releu.parteClient.conexiune.println(mesaj);
32             JOptionPane.showMessageDialog(null, "Releu/client: s-a primit " +mesaj);
33
34             mesaj = releu.parteClient.conexiune.nextLine();
35             releu.conexiune.println(mesaj);
36             JOptionPane.showMessageDialog(null, "Releu/server: s-a primit " +mesaj);
37
38             if (mesaj.equals(".")) break; // Testarea conditiei de oprire
39         }
40         releu.socketTCP.close(); // Inchiderea socketurilor si a fluxurilor
41         releu.parteClient.socketTCP.close();
42         JOptionPane.showMessageDialog(null, "Releu: Bye!");
43     }
44 }
```

### In laborator:

1. In proiectul *socket* creati clasa **ReleuBidirectional** folosind codul dat mai sus. Compilati codul.
2. La un calculator *right-click* pe **ServerBidirectional**. Executati **main()**. Folositi portul **6000**.
3. La un alt calculator (daca nu aveti la dispozitie un alt calculator in retea, deschideti inca o sesiune *BlueJ*) *right-click* pe clasa **ReleuDerivatBidirectional**, selectati si executati **main()**.
4. Folositi **adresa primului calculator**, pe care se executa **ServerBidirectional** (adresa "**localhost**" in cazul in care folositi 3 sesiuni *BlueJ* pe acelasi calculator), port **7000** (spre client) si port **6000** (spre server).
5. La un alt calculator (daca nu aveti la dispozitie un alt calculator in retea, deschideti inca o sesiune *BlueJ*) *right-click* pe clasa **ClientBidirectional**, selectati si executati **main()**.
6. Folositi **adresa celui de-al doilea calculator**, pe care se executa **ReleuDerivatBidirectional** (adresa "**localhost**" in cazul in care folositi 3 sesiuni *BlueJ* pe acelasi calculator), si port **7000**.
7. Urmarii efectul in **Terminal Window** pe cele trei calculatoare.

## 5.3. Interfete grafice cu utilizatorul (GUI)

### 5.3.1. Modalitati de a crea containerul de nivel maxim

Pentru a crea containere de nivel maxim ([detalii privind interfetele grafice swing in Java](#)) exista mai multe modalitati, printre care:

#### 1. Utilizarea unui obiect de tip `JFrame`,

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

public class TestUtilizareFrame_Swing { // Test grafic cu JTextField si JTextArea
    private static JTextField inTextGrafic; // Intrare -linie de text grafica (JtextField)
    private static JTextArea outTextGrafic; // Iesire - zona de text grafica (JtextArea)

    public TestUtilizareFrame_Swing() { // Initializari grafice
        // Crearea obiectului cadru (frame), cu titlu specificat
        JFrame frame = new JFrame("Test utilizare Frame");
        Container containerCurent = frame.getContentPane();
        containerCurent.setLayout(new BorderLayout());

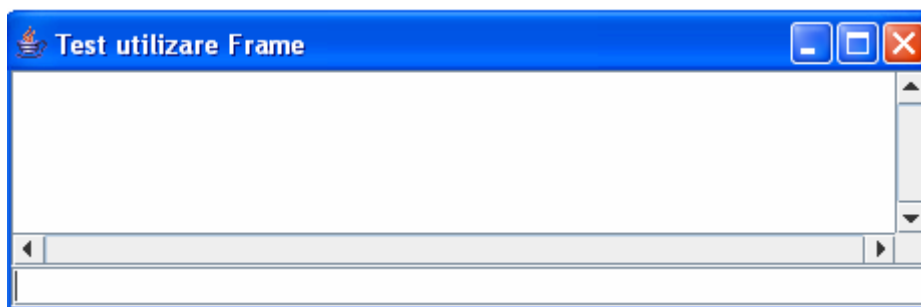
        outTextGrafic = new JTextArea(5, 40); // Zona de text non-editabila de iesire
        JScrollPane scrollPane = new JScrollPane(outTextGrafic,
            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);

        containerCurent.add("Center", scrollPane);
        outTextGrafic.setEditable(false);

        inTextGrafic = new JTextField(40); // Camp de text editabil de intrare
        containerCurent.add("South", inTextGrafic);

        frame.pack(); // Impachetarea (compactarea) componentelor
        frame.setVisible(true); // Fereastra devine vizibila
        inTextGrafic.requestFocus(); // Cerere focus pe intrarea text din fereastra curenta
    }

    public static void main (String args[]) {
        TestUtilizareFrame_Swing testUtilizareFrame = new TestUtilizareFrame_Swing();
    }
}
```



#### **In laborator:**

1. **Inchideti** proiectele anterioare (Ctrl+W). **Creati** un nou proiect *gui* (Project -> New Project..., selectati **D:**, apoi **Software2006**, apoi **numarul grupei**, si scrieti **gui**).
2. **Creati** o noua clasa, numita **TestUtilizareFrame\_Swing**, folosind codul dat mai sus.
3. **Compilati** codul, apoi *right-click* pe clasa, selectati si executati **main()**. Urmariti efectul.

## 2. Extinderea prin mostenire a clasei `JFrame`,

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

public class TestExtindereFrame_Swing extends JFrame { // Test cu JTextField si JTextArea
    private static JTextField inTextGrafic; // Intrare -linie de text grafica (JtextField)
    private static JTextArea outTextGrafic; // Iesire - zona de text grafica (JTextArea)

    public TestExtindereFrame_Swing() { // Initializari grafice
        super("Test extindere Frame"); // Stabilire titlu fereastră (JFrame)
        Container containerCurent = this.getContentPane();
        containerCurent.setLayout(new BorderLayout());

        outTextGrafic = new JTextArea(5, 40); // Zona de text non-editabila de iesire
        JScrollPane scrollPane = new JScrollPane(outTextGrafic,
            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);

        containerCurent.add("Center", scrollPane);
        outTextGrafic.setEditable(false);

        inTextGrafic = new JTextField(40); // Camp de text editabil de intrare
        containerCurent.add("South", inTextGrafic);

        pack(); // Impachetarea (compactarea) componentelor in container
        setVisible(true); // Fereastră devine vizibila
        inTextGrafic.requestFocus(); // Cerere focus pe intrarea de text din fereastră curenta
    }
    public static void main (String args[]) {
        TestExtindereFrame_Swing testFrame = new TestExtindereFrame_Swing();
    }
}
```

Optional, in laborator:

1. Tot in proiectul *gui* **creati** o noua clasa, numita `TestExtindereFrame_Swing`, folosind codul dat.
2. **Compilati** codul, apoi *right-click* pe clasa, selectati si executati `main()`. Urmariti efectul.

### 5.3.2. Crearea interactivitatii in interfetele grafice

Pentru introducerea interactivitatii, ([detalii privind interfetele grafice swing in Java](#)) trebuie tratate evenimentele din interfata grafica. In Java exista mai multe moduri de tratare a evenimentelor.

Inceput cu versiunea initiala, JDK 1.0, interfetele grafice realizate cu biblioteca AWT au 2 moduri de tratare a evenimentelor:

#### 1. Implementand metoda `action()`, care:

- primeste ca parametri un obiect de tip `Event` care incapsuleaza evenimentul produs, si un obiect de tip `Object` care incapsuleaza parametrii acestuia,
- testeaza atributele `target` si `id` ale obiectului de tip `Event` pentru a identifica obiectul tinta (in care s-a produs evenimentul) si tipul de actiune produsa, si trateaza apoi evenimentul respectiv

#### 2. Implementand metoda `handleEvent()`, care:

- primeste ca parametru un obiect de tip `Event` care incapsuleaza evenimentul produs,
- testeaza atributele `target` si `id` ale obiectului de tip `Event` pentru a identifica obiectul tinta (in care s-a produs evenimentul) si tipul de actiune produsa, si trateaza apoi evenimentul respectiv

Inceand cu versiunea JDK 1.1, interfetele grafice realizate cu biblioteca AWT au **un nou mod de tratare a evenimentelor**, utilizat si de interfetele grafice Swing, prin:

I. (a) **declararea unei clase care implementeaza o interfata « ascultator de evenimente »**, (care contine metode ce trebuie implementate de utilizator pentru tratarea evenimentului respectiv), sau (b) **declararea unei clase care extinde o clasa predefinita care implementeaza o interfata « ascultator de evenimente »**

II. (a) **implementarea tuturor metodelor definite in interfata « ascultator de evenimente »**, sau (b) **re-implementarea metodelor dorite din clasa care implementeaza interfata**

III. **inregistrarea unui obiect din clasa « ascultator de evenimente » de catre fiecare dintre componentele grafice (numite tinta sau sursa) pentru care se doreste tratarea evenimentului respectiv**

Programul `EcouGrafic_Swing` exemplifica extinderea clasei `JFrame` si tratarea evenimentelor.

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

public class EcouGrafic_Swing extends JFrame { // Ecou grafic cu JTextField si JTextArea

    private static JTextField inTextGrafic; // Intrare - linie text grafica (JtextField)
    private static JTextArea outTextGrafic; // Iesire - zona text grafica (JTextArea)
    private static JScrollBar vertical;

    public EcouGrafic_Swing() { // Initializari grafice
        super("Ecou grafic simplu Swing"); // Stabilire titlu fereastră (JFrame)
        Container containerCurent = this.getContentPane();
        containerCurent.setLayout(new BorderLayout());

        outTextGrafic = new JTextArea(5, 40); // Zona text non-editabila de iesire
        JScrollPane scrollPane = new JScrollPane(outTextGrafic,
            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        vertical = scrollPane.getVerticalScrollBar();
        containerCurent.add("Center", scrollPane);
        outTextGrafic.setEditable(false);
        outTextGrafic.append("Pentru oprire introduceti '.' si <Enter>\n\n");

        inTextGrafic = new JTextField(40); // Camp de text editabil de intrare
        containerCurent.add("South", inTextGrafic);

        // Inregistrare obiect "ascultator" de "evenimente actionare" la "obiectul sursa"
        inTextGrafic.addActionListener(new AscultatorInText());

        // Inregistrare obiect "ascultator" de "evenimente fereastră" la "sursa" fereastră
        this.addWindowListener(new AscultatorInchidere());

        pack(); // Impachetarea (compactarea) componentelor in container
        setVisible(true); // Fereastră devine vizibila

        inTextGrafic.requestFocus(); // Cerere focus pe intrarea text din fereastră curenta
    }
}
```

```

// Clasa interna "ascultator" de "evenimente actionare"
// implementeaza interfata ActionListener
class AscultatorInText implements ActionListener {

    // Tratarea actionarii intrarii de text (introducerii unui "Enter")
    public void actionPerformed(ActionEvent ev) {

        String sirCitit = inTextGrafic.getText(); // Citirea unei linii din intrarea text
        inTextGrafic.setText(""); // Golirea intrarii text
        outTextGrafic.append("S-a introdus: "+sirCitit+"\n"); // Scriere linie in zona text

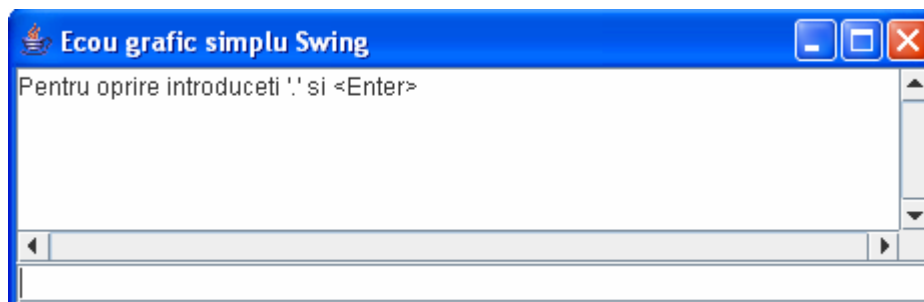
        vertical.setValue(vertical.getMaximum() - vertical.getVisibleAmount());
        validate(); repaint();
        if (sirCitit.equals(new String("."))) System.exit(0); // Conditie terminare
    }
}

// Metoda de test. Punct de intrare in program.

public static void main (String args[]) {
    EcouGrafic_Swing ecouGrafic = new EcouGrafic_Swing();
}

// Clasa "adaptor pentru ascultator" de "evenimente fereastră" extinde WindowAdapter
class AscultatorInchidere extends WindowAdapter {
    // Tratarea inchiderii ferestrei curente
    public void windowClosing(WindowEvent ev) { System.exit(0); } // Terminarea programului
}

```



#### In laborator:

1. Tot in proiectul *gui* **creati** o noua clasa, numita **EcouGrafic\_Swing**, folosind codul dat.
2. **Compilati** codul, apoi **right-click** pe clasa, selectati si **executati main()**. Urmariti efectul.

## 5.4. Precizari privind colocviul final

La ultima lucrare va fi sustinut si colocviul de laborator la care vor fi aduse toate temele de casa de la lucrarile 1, 2, 3 si 4. Colocviul va fi consta in intrebări din temele de casa.

Nota finala la laborator (30% din nota finala) va fi stabilita in functie de

- prezenta la laborator,
- activitatea la laborator (care va include datele la care au fost predate temele) si de
- cunostintele de POO dovedite la colocviul final.