Idoneità Java - Appunti Esame

Daniele Compagnoni

28 gennaio 2025

1 NorthPanel Class

```java
package ModelloEsame;
import javax.swing.*;
import javax.swing.JPanel;

public class NorthPanel extends JPanel {
    JTextField ipAddressField;
    JLabel ipAddressLabel;

    JTextField portField;
    JLabel portLabel;

    JButton connectButton;
    JButton disconnectButton;

    public NorthPanel() {
        ipAddressField = new JTextField("127.0.0.1", 20);
        ipAddressLabel = new JLabel("ServerAddress");
        ipAddressLabel.setLabelFor(ipAddressField);

        portField = new JTextField("4400", 20);
        portLabel = new JLabel("Port");
        portLabel.setLabelFor(portField);

        connectButton = new JButton("Connect");
        disconnectButton = new JButton("Disconnect");

        add(ipAddressLabel);
        add(ipAddressField);
        add(portLabel);
        add(portField);
        add(connectButton);
        add(disconnectButton);
    }
}
```

Listing 1: NorthPanel Class

2 SouthPanel Class

```java
package ModelloEsame;

import javax.swing.*;

public class SouthPanel extends JPanel {

    JLabel stringLabel;
    JTextField stringField;
    JButton getButton;
    JButton stopButton;

    public SouthPanel() {
        stringLabel = new JLabel("Film:");
        stringField = new JTextField("", 20);
        stringLabel.setLabelFor(stringField);

        getButton = new JButton("Get");
        stopButton = new JButton("Stop");

        add(stringLabel);
        add(stringField);
        add(getButton);
        add(stopButton);
    }
}
```

Listing 2: SouthPanel Class

3 Main Class

```java
package ModelloEsame;

public class Main {
    public static void main(String[] args) {
        ClientFrame frame = new ClientFrame();
    }
}
```

Listing 3: Main Class

4 ClientFrame Class

```java
package defau;

import java.awt.BorderLayout;
import java.awt.LayoutManager;
import javax.swing.JFrame;

public class ClientFrame extends JFrame {

    NorthPanel north;
    CenterPanel center;
    SouthPanel south;

    public ClientFrame() {
        super("Daniele Compagnoni 2114420");

        north = new NorthPanel();
        center = new CenterPanel();
        south = new SouthPanel();

        // Layout
        add(north, BorderLayout.NORTH);
        add(center, BorderLayout.CENTER);
        add(south, BorderLayout.SOUTH);

        // Listener
        ClientListener cl = new ClientListener(this);
        north.connectButton.setActionCommand(ClientListener.Connect);
        north.connectButton.addActionListener(cl);
        north.disconnectButton.setActionCommand(ClientListener.Disconnect);
        north.disconnectButton.addActionListener(cl);
        south.startButton.setActionCommand(ClientListener.Start);
        south.startButton.addActionListener(cl);
        south.stopButton.setActionCommand(ClientListener.Stop);
        south.stopButton.addActionListener(cl);

        // Window Setup
        updateUI(false, false);
        pack();
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public void updateUI(boolean connected, boolean
       transmitting) {
        if (connected) {
            north.connectButton.setEnabled(false);
            center.vm1.setEnabled(true);
            center.vm2.setEnabled(true);
```

```
49                center.vm3.setEnabled(true);
50                if (transmitting) {
51                    north.disconnectButton.setEnabled(false);
52                    south.startButton.setEnabled(false);
53                    south.stopButton.setEnabled(true);
54                } else {
55                    north.disconnectButton.setEnabled(true);
56                    south.startButton.setEnabled(true);
57                    south.stopButton.setEnabled(false);
58                }
59            } else {
60                north.connectButton.setEnabled(true);
61                north.disconnectButton.setEnabled(false);
62                south.startButton.setEnabled(false);
63                south.stopButton.setEnabled(false);
64                center.vm1.setEnabled(false);
65                center.vm2.setEnabled(false);
66                center.vm3.setEnabled(false);
67            }
68        }
69
70    }
```

Listing 4: ClientFrame Class

5 Primo Esempio: Bandiere

5.1 CenterPanel Class: Bandiere

```java
package Bandiere;

import java.awt.Color;
import java.awt.Dimension;

import javax.swing.JPanel;

public class CenterPanel extends JPanel {

    JPanel panel1;
    JPanel panel2;
    JPanel panel3;

    public CenterPanel() {
        this.panel1 = new JPanel();
        this.panel2 = new JPanel();
        this.panel3 = new JPanel();
        this.panel1.setPreferredSize(new Dimension(250, 500));
        this.panel1.setBackground(Color.LIGHT_GRAY);
        this.panel2.setPreferredSize(new Dimension(250, 500));
        this.panel2.setBackground(Color.LIGHT_GRAY);
        this.panel3.setPreferredSize(new Dimension(250, 500));
        this.panel3.setBackground(Color.LIGHT_GRAY);

        add(panel1);
        add(panel2);
        add(panel3);
    }

}
```

Listing 5: CenterPanel Class

5.2 ClientListener Class:Bandiere

```java
package Bandiere;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.*;
import java.util.Scanner;

import javax.swing.JOptionPane;

import java.io.*;
```

```
12
13  public class ClientListener implements ActionListener {
14
15      public static final String Start = "start";
16      public static final String Stop = "stop";
17      public static final String Connect = "connect";
18      public static final String Disconnect = "disconnect";
19
20      private ClientFrame frame;
21      private Socket sock;
22      private Scanner sockScan;
23      private PrintWriter sockPw;
24      private DownloaderThread downloadthread;
25      private boolean connected;
26      private boolean transmitting;
27
28      public ClientListener(ClientFrame frame) {
29          this.frame = frame;
30      }
31
32      private void setupConnection() throws IOException {
33          try {
34              String serverAddress =
                    frame.north.ipAddressField.getText();
35              int serverPort =
                    Integer.parseInt(frame.north.portField.getText());
36              sock = new Socket(serverAddress, serverPort);
37              sockScan = new Scanner(sock.getInputStream());
38              sockPw = new PrintWriter(new
                    OutputStreamWriter(sock.getOutputStream()));
39          } catch (NumberFormatException e) {
40              throw new IOException("Formato porta errato");
41          }
42      }
43
44      @Override
45      public void actionPerformed(ActionEvent e) {
46          String cmd = e.getActionCommand();
47          System.out.println(cmd);
48
49          // Connect sequence
50          if (cmd.equalsIgnoreCase(ClientListener.Connect)) {
51              try {
52                  setupConnection();
53                  connected = true;
54                  JOptionPane.showMessageDialog(frame,
                        "Connessione stabilita");
55                  frame.center.panel1.setBackground(Color.LIGHT_GRAY);
56                  frame.center.panel2.setBackground(Color.LIGHT_GRAY);
57                  frame.center.panel3.setBackground(Color.LIGHT_GRAY);
58              } catch (IOException ex) {
```

```
59                  JOptionPane.showMessageDialog(frame,
                        "Impossibile connettersi al server: " +
                        ex.getMessage());
60                  return;
61              }
62
63          // Download sequence
64          } else if (cmd.equalsIgnoreCase(ClientListener.Start)) {
65              frame.center.panel1.setBackground(Color.LIGHT_GRAY);
66              frame.center.panel2.setBackground(Color.LIGHT_GRAY);
67              frame.center.panel3.setBackground(Color.LIGHT_GRAY);
68              downloadthread = new DownloaderThread(frame, this,
                    sockScan);
69              transmitting = true;
70              sockPw.println(Start);
71              sockPw.flush();
72              Thread tr = new Thread(downloadthread);
73              tr.start();
74
75          // Stop sequence
76          } else if (cmd.equalsIgnoreCase(ClientListener.Stop)) {
77              sockPw.println(ClientListener.Stop);
78              sockPw.flush();
79              downloadthread.stop();
80              transmitting = false;
81
82          // Disconnect sequence
83          } else if
                (cmd.equalsIgnoreCase(ClientListener.Disconnect)) {
84              sockPw.println(ClientListener.Disconnect);
85              if (downloadthread != null) {
86                  downloadthread.stop();
87              }
88
89              transmitting = false;
90              connected = false;
91
92              try {
93                  sockPw.close();
94                  sockScan.close();
95                  sock.close();
96              } catch (IOException ex) {
97                  ex.printStackTrace();
98              }
99          }
100         frame.updateUI(connected, transmitting);
101     }
102
103     public void downloadComplete() {
104         transmitting = false;
105         frame.updateUI(connected, transmitting);
```

```
106     }
107
108     public void disconnected() {
109         frame.north.disconnectButton.doClick();
110     }
111 }
```

Listing 6: ClientListener Class

5.3 DownloaderThread Class:Bandiere

```
1  package Bandiere;
2
3  import java.awt.Color;
4  import java.util.HashMap;
5  import java.util.Hashtable;
6  import java.util.Scanner;
7
8  import javax.swing.JPanel;
9
10 public class DownloaderThread implements Runnable {
11
12     private ClientFrame frame;
13     private ClientListener listener;
14     private Scanner scanner;
15     private Hashtable<Integer, Color> colors;
16     private Hashtable<String, JPanel> panels;
17
18     private boolean running;
19     private boolean errored;
20
21     public DownloaderThread(ClientFrame frame, ClientListener
           listener, Scanner scanner) {
22         this.frame = frame;
23         this.listener = listener;
24         this.scanner = scanner;
25         this.colors = new Hashtable<Integer, Color>();
26         this.panels = new Hashtable<String, JPanel>();
27         this.colors.put(0, Color.WHITE);
28         this.colors.put(1, Color.BLACK);
29         this.colors.put(2, Color.GREEN);
30         this.colors.put(3, Color.RED);
31         this.colors.put(4, Color.YELLOW);
32         this.colors.put(5, Color.BLUE);
33         this.colors.put(6, Color.ORANGE);
34         this.panels.put("SX", frame.center.panel1);
35         this.panels.put("CX", frame.center.panel2);
36         this.panels.put("DX", frame.center.panel3);
37     }
38
39     @Override
```

```
40      public void run() {
41          if (running) {
42              return;
43          }
44
45          running = true;
46
47          while (running) {
48              // Check if the current thread has been interrupted
49              if (Thread.currentThread().isInterrupted()) {
50                  running = false;
51                  return;
52              }
53
54              // Process server data
55              String cmd = scanner.nextLine();
56              System.out.println("Ricevuto" + cmd);
57
58              // End sequence
59              if (cmd.equalsIgnoreCase("END")) {
60                  running = false;
61                  listener.downloadComplete();
62                  if (errored) {
63                      listener.disconnected();
64                  }
65              }
66
67              // Error sequence
68              else if (cmd.equalsIgnoreCase("ERROR")) {
69                  running = false;
70                  errored = true;
71                  listener.disconnected();
72              }
73
74              // Interrupt sequence
75              else if (cmd.equalsIgnoreCase("INTERRUPTED")) {
76                  running = false;
77              }
78
79              // SPECIFIC SOLUTION FOR FLAGS EXERCISE
80              else {
81                  String[] parts = cmd.split(";");
82                  if (parts[0].equals("-1") &&
                        parts[1].equals("-1")) {
83                      running = false;
84                      listener.downloadComplete();
85                      if (errored) {
86                          listener.disconnected();
87                      }
88                  } else {
```

```
89                      Color color =
                           colors.get(Integer.parseInt(parts[0]));
90                      JPanel panel = panels.get(parts[1]);
91                      panel.setBackground(color);
92                  }
93              }
94          }
95      }
96
97      public void stop() {
98          if (!running) {
99              return;
100         } else {
101             running = false;
102         }
103     }
104 }
```

Listing 7: DownloaderThread Class

6 Secondo Esempio: Bingo

6.1 CenterPanel Class : Bingo

```java
package bingo;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.LayoutManager;
import java.util.ArrayList;
import java.util.Random;

import javax.swing.BorderFactory;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class CenterPanel extends JPanel {

    ArrayList<TicketCell> cells;
    JPanel cartella;

    JTextArea logArea;
    JScrollPane logScroll;

    public CenterPanel() {
        this.setLayout(new BorderLayout());
        // Add the cells
        cartella = new JPanel(new GridLayout(3, 5));
        cartella.setBorder(BorderFactory.createTitledBorder("Cartella"));
        cells = new ArrayList<TicketCell>();
        for (int i = 0; i < 15; i++) {
            cells.add(new TicketCell());
            cartella.add(cells.get(i));
        }
        add(cartella, BorderLayout.WEST);

        // Log area
        logArea = new JTextArea(20, 30);
        logArea.setEditable(false);
        logScroll = new JScrollPane(logArea);
        logScroll.setBorder(BorderFactory.createTitledBorder("Log"));

        add(logScroll, BorderLayout.EAST);
    }

}
```

Listing 8: CenterPanel Class

6.2 ClientListener Class : Bingo

```java
package bingo;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOError;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Scanner;

import javax.swing.JOptionPane;

public class ClientListener implements ActionListener {

    public static final String Connect = "connect";
    public static final String Disconnect = "disconnect";
    public static final String Start = "start";
    public static final String Stop = "stop";

    private ClientFrame frame;
    private Socket sock;
    private Scanner sockScan;
    private PrintWriter sockPw;
    private DownloaderThread downloadthread;

    private boolean connected;
    private boolean transmitting;

    public ClientListener(ClientFrame frame) {
        this.frame = frame;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String cmd = e.getActionCommand();

        // Connection sequence
        if(cmd.equalsIgnoreCase(ClientListener.Connect)) {
            try {
                setupConnection();
                connected = true;
                JOptionPane.showMessageDialog(frame,
                   "Connessione stabilita");
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(frame,
                   "Impossibile connettersi al server:" +
```

```
                    ex.getMessage());
48                  ex.printStackTrace();
49                  return;
50              }
51          // Disconnection sequence
52          } else if
               (cmd.equalsIgnoreCase(ClientListener.Disconnect)) {
53              sockPw.println(ClientListener.Disconnect);
54              if (downloadthread != null) {
55                  downloadthread.stop();
56              }
57              transmitting = false;
58              connected = false;
59              try {
60                  sockPw.close();
61                  sockScan.close();
62                  sock.close();
63              } catch(IOException ex) {
64                  ex.printStackTrace();
65              }
66          // Download sequence
67          } else if (cmd.equalsIgnoreCase(ClientListener.Start)) {
68              // Setup della cartella
69              ArrayList<Integer> randomNumbers = new
                   ArrayList<Integer>();
70              for (int i = 0; i < frame.center.cells.size(); i++) {
71                  int randomNumber = (int)(Math.random()*90);
72                  while (randomNumbers.contains(randomNumber)) {
73                      randomNumber = (int)(Math.random()*90);
74                  }
75                  randomNumbers.add(randomNumber);
76              }
77              randomNumbers.sort(null);
78
79              for (int i = 0; i < frame.center.cells.size(); i++) {
80                  TicketCell cell = frame.center.cells.get(i);
81                  cell.setValue(randomNumbers.get(i));
82                  cell.setSelected(false);
83                  cell.setBackground(Color.WHITE);
84              }
85
86              downloadthread = new DownloaderThread(frame, this,
                   sockScan);
87              transmitting = true;
88              sockPw.println(ClientListener.Start);
89              sockPw.flush();
90              Thread tr = new Thread(downloadthread);
91              tr.start();
92          // Stop download sequence
93          } else if (cmd.equalsIgnoreCase(ClientListener.Stop)) {
94              sockPw.println(ClientListener.Stop);
```

```
95              sockPw.flush();
96              downloadthread.stop();
97              transmitting = false;
98          }
99
100         frame.updateUI(connected, transmitting);
101     }
102
103     private void setupConnection() throws IOException {
104         try {
105             String serverAddress =
                    frame.north.serverField.getText();
106             int serverPort =
                    Integer.parseInt(frame.north.portField.getText());
107             sock = new Socket(serverAddress, serverPort);
108             sockScan = new Scanner(sock.getInputStream());
109             sockPw = new PrintWriter(new
                    OutputStreamWriter(sock.getOutputStream()));
110         } catch(NumberFormatException ex) {
111             throw new IOException("Formato porta errato");
112         }
113     }
114
115     public void downloadComplete() {
116         transmitting = false;
117         frame.updateUI(connected, transmitting);
118     }
119
120     public void disconnected() {
121         frame.north.disconnectButton.doClick();
122     }
123
124 }
```

Listing 9: ClientListener Class

6.3 DownloaderThread Class: Bingo

```
1  package bingo;
2
3  import java.awt.Color;
4  import java.util.Scanner;
5
6  public class DownloaderThread implements Runnable {
7
8      private ClientFrame frame;
9      private ClientListener listener;
10     private Scanner scanner;
11
12     private boolean running;
13     private boolean errored;
```

```
14
15      public DownloaderThread(ClientFrame frame, ClientListener
            listener, Scanner scanner) {
16          this.frame = frame;
17          this.listener = listener;
18          this.scanner = scanner;
19      }
20
21      @Override
22      public void run() {
23          if (running) {
24              return;
25          }
26          running = true;
27
28          while (running) {
29              // Check if the current thread has been interrupted
30              if (Thread.currentThread().isInterrupted()) {
31                  running = false;
32                  return;
33              }
34              String cmd = scanner.nextLine();
35              System.out.println("Ricevuto: " + cmd);
36
37              // Process server data
38
39              // Terminazione: +
40              if (cmd.equals("+")) {
41                  running = false;
42                  listener.downloadComplete();
43                  if(errored) {
44                      listener.disconnected();
45                  }
46                  frame.center.logArea.append("--- Fine partita
                        ---\n\n\n");
47                  frame.center.logScroll.getVerticalScrollBar().setValue(frame
48              // Caso Base: numero estratto
49              } else {
50                  int estratto = Integer.parseInt(cmd);
51                  frame.center.logArea.append("Estratto: " + cmd +
                        "\n");
52                  frame.center.logScroll.getVerticalScrollBar().setValue(frame
53
54                  for (int i = 0; i < frame.center.cells.size();
                        i++) {
55                      TicketCell cell = frame.center.cells.get(i);
56                      if (cell.getValue() == estratto) {
57                          cell.setSelected(true);
58                          cell.setBackground(Color.GREEN);
59                      }
60                  }
```

```
61                  }
62              }
63          }
64
65          public void stop() {
66              if (!running) {
67                  return;
68              }
69              running = false;
70          }
71
72  }
```

Listing 10: DownloaderThread Class

7 Istruzioni di Consegna Esame

- Aprire la cartella **Esame** sul desktop.
- Aprire il menu **Tools**.
- Selezionare **"Open current folder in terminal"**.
- Nella finestra di terminale, digitare `java -jar ClientConsegna.jar`.
- Riempire i campi **Cognome, Nome, Matricola**.
- Premere il tasto **"Scegli file"**.
- Selezionare la voce **"esame"**.
- Selezionare **"biar"**.
- Selezionare **"eclipse-workspace"**.
- Selezionare il nome del progetto (nell'esempio, il progetto si chiama **"Shot"**).
- Selezionare **"src"**.
- Selezionare uno qualsiasi dei file Java.
- Il programma conferma che i file sono stati selezionati.
- Premere **"Invia file"**.
- Verificare che i file da inviare siano tutti. In questo caso, premere **"Yes"**.
- Il programma conferma l'invio.
- Premere **"Ok"**.
- Rimane un'ulteriore conferma dell'invio.