Fondamenti di informatica L-B: esercitazione 06

Autore: Andrea Urbini urbo83@tiscali.it

Matricola: 167064

Descrizione problema

L'esercitazione chiede la costruzione di un sistema di monitoraggio delle condizioni dei pazienti dell'ospedale. In particolare le grandezze da monitorare sono pressione e temperatura. Il sistema deve essere regolabile dal personale dell'ospedale, deve, infatti, essere possibile modificare i valori soglia delle due grandezze da monitorare. Quando temperatura, pressione o il loro prodotto superano i valori soglia il sistema deve visualizzare su un'opportuna finestra e salvare su un file di testo il messaggio relativo indicante la data, l'ora, i valori soglia, e il valore della grandezza presa in considerazione.
Siccome al momento non disponiamo dell'hardware necessario per interfacciare i veri sensori al sistema, questi vengono sostituiti da due elementi *Swing* in cui l'utente può variare i valori di temperatura e pressione.

Analisi

I componenti del sistema possono essere divisi in tre gruppi:

- **Sensori:** in questo caso rappresentano l'astrazione dei sensori Hardware. Danno la possibilità all'utente di variare manualmente la temperatura e la pressione. Ogni volta che cambia il valore di una grandezza viene generato in evento che deve essere catturato dal controllore. Come suggerito dal testo dell'esercitazione il sensore di pressione è rappresentato da un componente *JSlider*, mentre quello di temperatura da un campo di testo in cui l'utente inserisce i valori. Entrambe le finestra dei sensori sono implementate in un'unica classe chiamata *Sensor*.

- **Controllore:** confronta i valori delle grandezze fisiche provenienti dai sensori con i valori di soglia. Questi ultimi dati vanno impostati dall'utente in un apposito componente Swing. Se i valori provenienti dai sensori superano le soglie allora il controllore dovrebbe generare il tipo di evento catturato dai visualizzatori. L'utente che vuole variare le soglie, deve inserire i nuovi valori di soglia in due campi di testo e premere il bottone "Aggiorna". La classe sarà chiamata *Controll*.

- **Visualizzatori:** ascoltano gli eventi significativi per il sistema generati dal controllore che riguardano il superamento delle soglie. Esistono due visualizzatori: il primo, attraverso una finestra, visualizza sullo schermo del pc i messaggi relativi al superamento delle soglie; il secondo scrive gli stessi messaggi su file di testo. I messaggi hanno la seguente forma: *"Mercoledì 25 Giugno 2003 – 10:18:34 – Il valore della pressione ha superato la soglia (200): 230"* Le due classi saranno rispettivamente chiamate *Viewer* e *Saver*.

Oltre a questi componenti occorre progettare anche il componente software contenente il main del programma. Questa classe sarà chiamata *ACMEPatientMonitoring.*

Passiamo ora all'analisi degli eventi generati nel sistema: il componente *JSlider* genera eventi di tipo *ChangeEvent* ascoltati dal metodo dell'interfaccia *ChangeListener*. Tutti gli altri componenti (bottoni e campi di testo) generano eventi *ActionEvent* ascoltati tramite i metodi dell'interfaccia *ActionEvent*. Il problema sorge nella comunicazione tra *Controllore* e *Visualizzatori*: il testo dell'esercitazione sembra far intendere che questa comunicazione debba avvenire utilizzando gli eventi. Purtroppo non sono riuscito a generare ed inviare eventi senza che sia l'utente a deciderlo: nel caso di un bottone, l'evento viene generato quando è l'utente che preme il bottone; in questo caso l'evento deve essere creato quando si avvera una certa condizione. Per sopperire a questa lacuna, il controllore modifica la stringa messaggio dei *Visualizzatori* attraverso un opportuno metodo. In questo modo il *Controllore* deve contenere i metodi per la costituzione della stringa *messaggio* che deve essere passata ai *Visualizzatori*.

Come emerso dall'analisi occorre progettare quattro entità: una classe *Sensor*, una classe *Controll*, una classe *Viewer* e una classe *Saver*. Oltre a queste entità si deve progettare il componente software *ACMEPatientMonitoring* contente il *main*.

Progetto

- **Classe *Sensor*:** i due sensori devono essere posti in due finestre separate, per cui occorre creare due *JFrame* diversi. Il sensore di pressione è rappresentato da un componente di tipo *JSlider* che permette di scegliere un valore di pressione tra 0 e 300. Il valore iniziale è 150. Gli eventi generati dal *JSlider* vengono ascoltati dal controllore assegnato alla variabile *ascoltatore*. Il sensore di temperatura è rappresentato da un *JTextField* in cui l'utente può inserire il valore corrente della temperatura. Il valore iniziale è 36.51°C. Le fasi di creazione di questi componenti sono contenute tutte nel costruttore dell'oggetto.

- **Classe *Controll*:** I campi di questa classe sono i seguenti:

Tipo	Nome	Cosa indica
int	*pressioneSoglia*	Valore soglia della pressione.
int	*pressioneCorrente*	Valore corrente della pressione.
double	*temperaturaSoglia*	Valore soglia della temperatura.
double	*temperaturaCorrente*	Valore corrente della temperatura.
double	*prodottoSoglia*	Valore soglia del prodotto temperatura*pressione.
JButton	*aggiorna*	Pulsante "aggiorna", serve per aggiornare i valori delle soglie.
JTextField	*presSolgia, tempSoglia, prodSoglia*	Campi di testo in cui l'utente inserisce i valori delle soglie di pressione, temperatura, prodotto temperatura*pressione.
Viewer	*ascoltatore1*	Memorizza l'oggetto corrispondente.
Saver	*ascoltatore2*	Memorizza l'oggetto corrispondente.

Questa classe ha sia la funzione di output di componenti Swing sia di ascoltatore degli eventi generati dai propri componenti e di quelli della classe *Sensor*. Il *JFrame* creato contiene tre *JTextField* in cui l'utente inserisce i valori soglia di pressione (200), temperatura (38°C) e prodotto pressione temperatura (700). L'utente per variare le soglie deve poi premere il *JButton "Aggiorna"*. Gli eventi creati della classe *Sensor* sono del tipo *ChangeEvent* e *ActionEvent* ascoltabili tramite i metodi delle interfacce *ChangeListener* e *ActionListener*. Implementando queste interfacce, la classe *Controll* deve implementare anche i metodi *stateChanged* ed *actionPerformed*:

- *stateChanged*: quando si verifica un evento di tipo *ChangeEvent*, si recupera il valore corrente del JSlider e lo si confronta con il valore soglia della pressione, se la soglia viene superata allora viene invocato il metodo *addMessage* dei visualizzatori (classi *Viewer* e *Saver*). Gli oggetti corrispondenti alle due classi visualizzatrici sono assegnate a due campi privati denominati *ascoltatore1* e *ascoltatore2*.
- *actionPerformed*: quando si verifica un evento di tipo *ActionEvent*, si recupera il nome del componente che l'ha generato. Se l'evento è stato generato dal *JButton* "*Aggiorna*" significa che sono state variate le soglie e quindi devono essere recuperati i nuovi valori dai tre *JTextField* presenti nella finestra e assegnati ai rispettivi campi. Per esclusione, l'altro componente che può generare eventi di questo tipo è il *JTextField* presente nella finestra del sensore di temperatura e quindi nella classe *Sensor*. Anche in questo caso si recupera il valore contenuto nel *JTextField* e lo si confronta con il valore soglia della temperatura. Se la soglia è stata superata allora si invoca il metodo *addMessage* dei visualizzatori (classi *Viewer* e *Saver*).

In entrambi i metodi va verificato che il prodotto dei valori correnti di temperatura e pressione non superi la soglia. Quando la soglia viene superata si invoca il metodo *addMessage* dei visualizzatori (classi *Viewer* e *Saver*).
Il messaggio inviato ai visualizzatori deve contenere anche la data nel formato giorno della settimana, giorno del mese, mese, anno, ora, minuti, secondi. Per fare ciò occorre un implementare un metodo che recuperi la data della macchina su cui gira il sistema. Per fare ciò occorre implementare un metodo chiamato *data()* che crea un oggetto di tipo *java.util.Calendar* da cui si estraggono (tramite il comando *get*) degli interi da convertire in stringa. Per il giorno del mese, l'anno, l'ora, i minuti e i secondi la conversione è automatica, mentre per i giorni della settimana e i mesi occorre impostare una serie di controlli per assegnare ad ogni intero la stringa contenete il nome italiano del giorno e del mese.

- **Classe *Viewer*:** La classe deve visualizzare su schermo le stringhe di allarme generate dal sistema. Per fare ciò occorre creare un *JFrame* il cui inserire una *JTextArea* (non modificabile dall'utente) in cui il sistema visualizza i messaggi di allarme. Per garantire la visibilità anche dopo molti messaggi di allarme, la *JTextArea* deve essere incapsulata in un *JScrollPane* che aggiunge all'area di testo le barre laterali di scorrimento. La classe implementa anche il metodo *addMessage(String message)* con cui un oggetto esterno può aggiungere messaggi di allarme, infatti questo metodo non fa altro che appendere la stringa *message* alla *JTextArea*.

- **Classe *Saver*:** La classe deve salvare su un file di testo i messaggi di allarme generati dal sistema. Le operazioni di scrittura su file avvengono nel seguente modo:
  Si crea una variabile di nome *file* e di tipo *FileWriter*; successivamente si prova ad assegnargli un oggetto di tipo *FileWriter* corrispondente a *alarm.txt* a cui è stata abilitata la funzione *append* (le stringhe successive vengono inserite in coda al file). Se viene lanciata un'eccezione di tipo *IOException* allora viene stampato su standard output il messaggio d'errore "Impossibile aprire il file" e il programma termina.
  La fase di scrittura vera e propria inizia con il comando *write(messaggio)* che inserisce dentro a *file* la stringa messaggio, poi viene chiuso lo stream. Se viene lanciata una eccezione di tipo *IOException* viene stampato su standard output il messaggio d'errore "Errore di output" e il programma termina.

- **Classe *ACMEPatientMonitoring:*** Questa classe è il componente software che si deve lanciare per avviare il programma. Nel main de programma vengono creati tutti gli oggetti necessari al funzionamento del programma:

tipo	nome
Saver	*scriviFile*
Viewer	*visualizzatore*
Controll	*controllore*
Sensor	*sensori*

Ho scelto per semplicità di dare la possibilità all'utente di chiudere il programma premendo su qualsiasi pulsante apposito di ogni finestra.

Implementazione

Riporto il codice della classe ACMEPatientMonitoring:

```
package es06;

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

/**
 * Questo componente software contiene le operazioni per la creazione della
 * finestra del programma.
 * Per avviare il programma occorre lanciare questa classe.
 *
 * @author Andrea Urbini
 */

public class ACMEPatientMonitoring{

	/**
	 * Contiene le operazioni di creazione e visualizzazione dell'interfaccia
	 * grafica.
	 */

	public static void main(String[] args){
		Saver scriviFile=new Saver();
		Viewer visualizzatore=new Viewer();
		Controll controllore=new Controll(visualizzatore,scriviFile);
		Sensor sensori=new Sensor(controllore);
	}
}
```

Riporto il codice della classe Sensor:

```
package es06;

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

/**
```

```
 *Questa classe crea le due finestre corrispondenti ai due sensori.
 *@author Andrea Urbini
 */

public class Sensor{

	Controll ascoltatore;

	/**
	 *Costruisce i deu JFrame con i due sensori: un JSlider e un JTextField
	 *@param ascoltatore oggetto che ascolta gli eventi generati nella classe
	 */

	public Sensor(Controll ascoltatore){
		this.ascoltatore=ascoltatore;

		//realizzo il frame del sensore di pressione
		JFrame presFrame=new JFrame("Sensore di pressione");
		presFrame.setBounds(10,10,300,100);
		presFrame.setResizable(false);
		presFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container cp=presFrame.getContentPane();
		JPanel presPanel=new JPanel();
		JLabel presLabel=new JLabel("Valore corrente");
		presPanel.add(presLabel);
		JSlider presValore=new JSlider(0,300,150);
		presValore.setMajorTickSpacing(300);
		presValore.setMinorTickSpacing(1);
		presValore.setPaintLabels(true);
		presValore.addChangeListener(ascoltatore);
		presPanel.add(presValore);
		cp.add(presPanel);
		presFrame.show();

		//Realizzo il frame del sensore di temperatura
		JFrame tempFrame=new JFrame("Sensore di temperatura");
		tempFrame.setBounds(320,10,300,100);
		tempFrame.setResizable(false);
		tempFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container ct=tempFrame.getContentPane();
		JPanel tempPanel=new JPanel();
		JLabel tempLabel=new JLabel("Valore corrente (°C):");
		tempPanel.add(tempLabel);
		JTextField tempValore=new JTextField("36.51",5);
		tempValore.setHorizontalAlignment(JTextField.RIGHT);
		tempValore.addActionListener(ascoltatore);
		tempPanel.add(tempValore);
		ct.add(tempPanel);
		tempFrame.show();
	}
}
```

Riporto il codice della classe Controll:

```
package es06;

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

/**
 *Questa classe costruisce la finestra rappresentante il controllore. Qui
 *l'utente puo' inserire i valori soglia delle grandezze. Quesat classe ascolta
 *gli eventi generati dai componenti della classe Sensor.
 *@author Andrea Urbini
 */

public class Controll implements ActionListener,ChangeListener{

	int pressioneSoglia,pressioneCorrente;
	double temperaturaSoglia,temperaturaCorrente,prodottoSoglia;
	JButton aggiorna;
	JTextField tempSoglia,presSoglia,prodSoglia;
	Viewer ascoltatore1;
	Saver ascoltatore2;

	/**
	 *Costruisce il JFrame relativo al controllore
```

```
         *@param ascoltatore oggetto che ascolta gli eventi generati nella classe
         */

        public Controll(Viewer ascoltatore1,Saver ascoltatore2){
                this.ascoltatore1=ascoltatore1;
                this.ascoltatore2=ascoltatore2;

                //Creo il frame di controllo
                JFrame controlFrame=new JFrame("Controllore");
                controlFrame.setBounds(10,120,300,130);
                controlFrame.setResizable(false);
                controlFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                Container cc=controlFrame.getContentPane();
                JPanel controlPanel=new JPanel();
                JLabel tempLabel=new JLabel("Temperatura Soglia (°C):");
                Box colonna1=new Box(BoxLayout.Y_AXIS);
                Box colonna2=new Box(BoxLayout.Y_AXIS);
                tempSoglia=new JTextField("38",5);
                tempSoglia.setHorizontalAlignment(JTextField.RIGHT);
                JLabel presLabel=new JLabel("Pressione Soglia:");
                presSoglia=new JTextField("200",4);
                presSoglia.setHorizontalAlignment(JTextField.RIGHT);
                JLabel prodLabel=new JLabel("Pressione * Temperatura Soglia:");
                prodSoglia=new JTextField("700",5);
                prodSoglia.setHorizontalAlignment(JTextField.RIGHT);
                aggiorna=new JButton("Aggiorna");
                aggiorna.addActionListener(this);
                colonna1.add(tempLabel);
                colonna1.add(presLabel);
                colonna2.add(tempSoglia);
                colonna2.add(presSoglia);
                colonna1.add(prodLabel);
                colonna2.add(prodSoglia);
                controlPanel.add(colonna1);
                controlPanel.add(colonna2);
                controlPanel.add(aggiorna);
                cc.add(controlPanel);
                controlFrame.show();
                temperaturaSoglia=Double.parseDouble(tempSoglia.getText());
                pressioneSoglia=Integer.parseInt(presSoglia.getText());
                prodottoSoglia=Double.parseDouble(prodSoglia.getText());
        }

        /**
         * Ascoltatore degli eventi generati dai componenti JButton e JTextField
         * rispettivamente delle classi Controll e Sensor. In particolare rileva i
         * valori del JTextField di Sensor (temperatura corrente) e dei JTextField
         * di Controll (temperatura Soglia, pressione Soglia e pressione *
         * temperatura Soglia). Se i valori correnti superano le soglie manda
         * un messaggio al visualizzatore.
         * @param e evento di tipo ActionEvent
         */

        public void actionPerformed(ActionEvent e){
                String actionName=e.getActionCommand();
                //Se premo aggiorna si aggiornano i valori soglia.
                if (actionName.equals("Aggiorna")){
                        temperaturaSoglia=Double.parseDouble(tempSoglia.getText());
                        pressioneSoglia=Integer.parseInt(presSoglia.getText());
                        prodottoSoglia=Double.parseDouble(prodSoglia.getText());
                }else{
                        //Altrimenti, per esclusione, l'evento proviene dal JTextField della
temperatura corrente.
                        temperaturaCorrente=Double.parseDouble(((JTextField)e.getSource()).getText());
                        if (temperaturaCorrente>temperaturaSoglia){
                                String messaggio=data()+" il valore della temperatura ha superato il
valore \n soglia ("+temperaturaSoglia+"): "+temperaturaCorrente+"\n\n";
                                ascoltatore1.addMessage(messaggio);
                                ascoltatore2.addMessage(messaggio);
                        }
                        if (temperaturaCorrente*pressioneCorrente>prodottoSoglia){
                                String messaggio=data()+" il valore del prodotto temperatura*pressione
ha superato il valore \n soglia ("+prodottoSoglia+"):
"+temperaturaCorrente*pressioneCorrente+"\n\n";
                                ascoltatore1.addMessage(messaggio);
                                ascoltatore2.addMessage(messaggio);
                        }
                }
```

```
	}

	/**
	 * Ascoltatore degli eventi generati dal componente JSlider della classe
	 * Sensor. Se il valore indicato sullo JSlider supera il valore della
	 * pressione di soglia manda un messaggio al visualizzatore.
	 * @param e evento generato dal JSlider
	 */

	public void stateChanged(ChangeEvent e){
		JSlider presValore=(JSlider)e.getSource();
		pressioneCorrente=presValore.getValue();
		if (pressioneCorrente>pressioneSoglia){
			String messaggio=data()+" il valore della pressione massima ha superato il
valore \n soglia ("+pressioneSoglia+"): "+pressioneCorrente+"\n\n";
			ascoltatore1.addMessage(messaggio);
			ascoltatore2.addMessage(messaggio);
		}
		if (temperaturaCorrente*pressioneCorrente>prodottoSoglia){
				String messaggio=data()+" il valore del prodotto temperatura*pressione
ha superato il valore \n soglia ("+prodottoSoglia+"):
"+temperaturaCorrente*pressioneCorrente+"\n\n";
				ascoltatore1.addMessage(messaggio);
				ascoltatore2.addMessage(messaggio);
		}

	}

	/**
	 * Questo metodo crea una stringa contente le informazioni sulla data nel
	 * seguente formato:
	 * Giorno della settimana, giorno del mese, mese, anno, ora, minuti, secondi.
	 * @return stringa contente la data.
	 */

	public String data(){
		String s=" ";
		//creo un oggetto Calendar
		Calendar calendario=Calendar.getInstance();
		//recupero il giorno della settimana e gli assegno il nome italiano
		int giorno=calendario.get(Calendar.DAY_OF_WEEK);
		if (giorno==1) s+="Domenica";
		else if(giorno==2) s+="Lunedi'";
		else if(giorno==3) s+="Martedi'";
		else if(giorno==4) s+="Mercoledi'";
		else if(giorno==5) s+="Giovedi'";
		else if(giorno==6) s+="Venerdi'";
		else s+="Sabato";
		//recupero il giorno del mese
		s+=" "+calendario.get(Calendar.DAY_OF_MONTH)+" ";
		//recupero il mese e gli assegno il nome italiano
		int mese=calendario.get(Calendar.MONTH);
		if (mese==0) s+="Gennaio";
		else if(mese==1) s+="Febbraio";
		else if(mese==2) s+="Marzo";
		else if(mese==3) s+="Aprile";
		else if(mese==4) s+="Maggio";
		else if(mese==5) s+="Giugno";
		else if(mese==6) s+="Luglio";
		else if(mese==7) s+="Agosto";
		else if(mese==8) s+="Settembre";
		else if(mese==9) s+="Ottobre";
		else if(mese==10) s+="Novembre";
		else s+="Dicembre";
		//recupero l'anno
		s+=" "+calendario.get(Calendar.YEAR);
		//trasformo in stringa l'orario
		s+=" -
"+calendario.get(Calendar.HOUR_OF_DAY)+":"+calendario.get(Calendar.MINUTE)+":"+calendario.get(Calend
ar.SECOND)+" - ";
		return s;
	}
}
```

Riporto il codice della classe Viewer:

```
package es06;

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

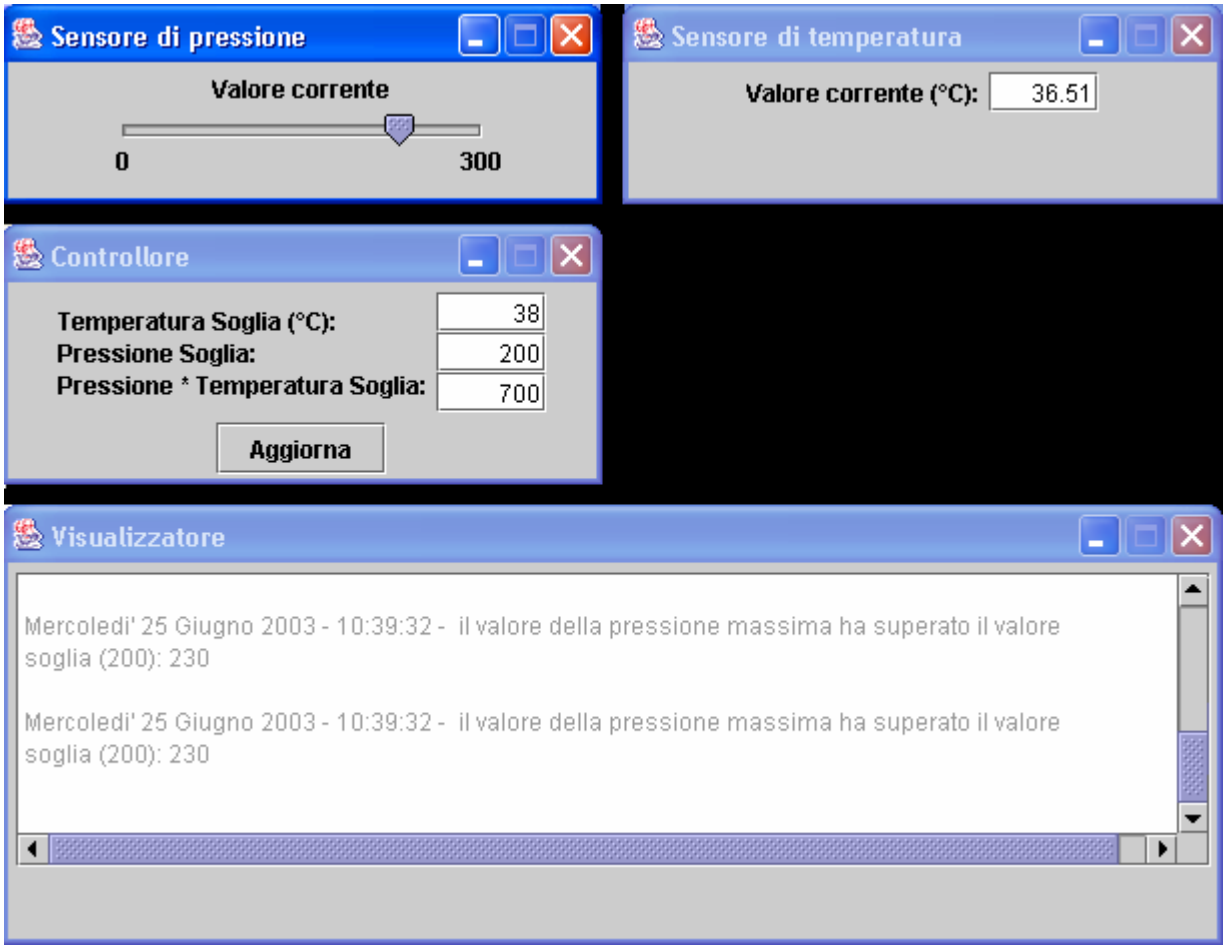
```
/**
 * Questa classe crea la finestra corrispondente al visualizzatore.
 * @author Andrea Urbini
 */

public class Viewer{

	JTextArea viewerText;
	JPanel viewerPanel;

	/**
	 * Costruisce il JFrame corrispondente al componente di visualizzazione.
	 */

	public Viewer(){
		JFrame viewerFrame=new JFrame("Visualizzatore");
		viewerFrame.setBounds(10,260,610,220);
		viewerFrame.setResizable(false);
		viewerFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container cv=viewerFrame.getContentPane();
		viewerPanel=new JPanel();
		viewerText=new JTextArea(9,54);
		viewerText.setEnabled(false);
		JScrollPane viewerScroll=new JScrollPane(viewerText);
		viewerScroll.setWheelScrollingEnabled(true);
		viewerPanel.add(viewerScroll);
		cv.add(viewerPanel);
		viewerFrame.show();
	}

	/**
	 * Questo metodo scrive sulla TextArea del pannello la stringa message
	 * passata in ingresso.
	 * @param message messaggio da visualizzare sul pannello.
	 */

	public void addMessage(String message){
		//aggiungo alla JTextArea il messaggio
		viewerText.append(message);
		//ridisegno il pannello
		viewerPanel.repaint();
	}
}
```

Riporto il codice della classe Saver:

```
package es06;

import java.io.*;

/**
 * Questa classe salva su file di testo i messaggi visualizzati inviati dalla
 * classe Controll.
 * @author Andrea Urbini
 */

public class Saver{

	public Saver(){}

	/**
	 * Questo metodo scrive sul file di testo alarm.txt la stringa messaggio
	 * passata in ingresso.
	 * @param message messaggio da salvare su disco.
	 */

	public void addMessage(String messaggio){
	FileWriter file=null;
		try{
			//Crea un oggetto di tipo FileWriter; e' abilitata la funzione append
			file=new FileWriter("alarm.txt",true);
		}catch(IOException e1){
			//Se si verifica una eccezione allora stampo un messaggio di errore
			System.out.println("Impossibile aprire file");
			System.exit(1);
		}
		try{
			//scrive sul file la stringa messaggio
			file.write(messaggio);
```

```
                        //Chiude lo stream
                        file.close();
                }catch(IOException e2){
                        //Se si verifica una eccezione allora stampo un messaggio di errore
                        System.out.println("Errore di output");
                        System.exit(2);
                }
        }
}
```

Testing e Casi d'uso



Concetti e tecniche acquisiti

Questa esercitazione mi ha permesso di approfondire ulteriormente due grandi capitoli della programmazione di Java: la grafica e gli eventi. Come nella passata esercitazione la grafica non mi ha dato problemi.
Il progetto iniziale (prima di stendere la relazione) prevedeva di utilizzare gli eventi anche tra Controllore e Visualizzatori. Purtroppo non sono riuscito ha generare eventi in corrispondenza del superamento di una soglia per cui ho dovuto ripiegare su un altro modo (la classica invocazione di metodi tra classi). Tutto sommato sono riuscito a far funzionare nel modo migliore il programma.