Fondamenti di informatica L-B: esercitazione 05

Autore: Andrea Urbini urbo83@tiscali.it

Matricola: 167064

Descrizione problema

L'esercitazione chiede la realizzazione di un programma per disegnare al computer. Deve essere rivolto agli anziani per cui non è molto sofisticato.
In particolare la finestra del programma deve essere divisa in tre parti, quelle superiore e inferiore destinate a contenere i pulsanti per disegnare e quella centrale destinata ad essere il luogo in cui disegnare. I comandi di disegno devono essere, in particolare: quattro pulsanti per muovere il cursore, un pulsante per uscire dal programma, un campo di testo in cui è possibile definire la griglia e un pulsante di Start/Stop utilizzato per disegnare righe sul pannello di disegno. Deve poi essere possibile muovere il cursore tramite il mouse. L'utente deve avere la possibilità di scegliere il colore del cursore e quindi della linea.

Analisi

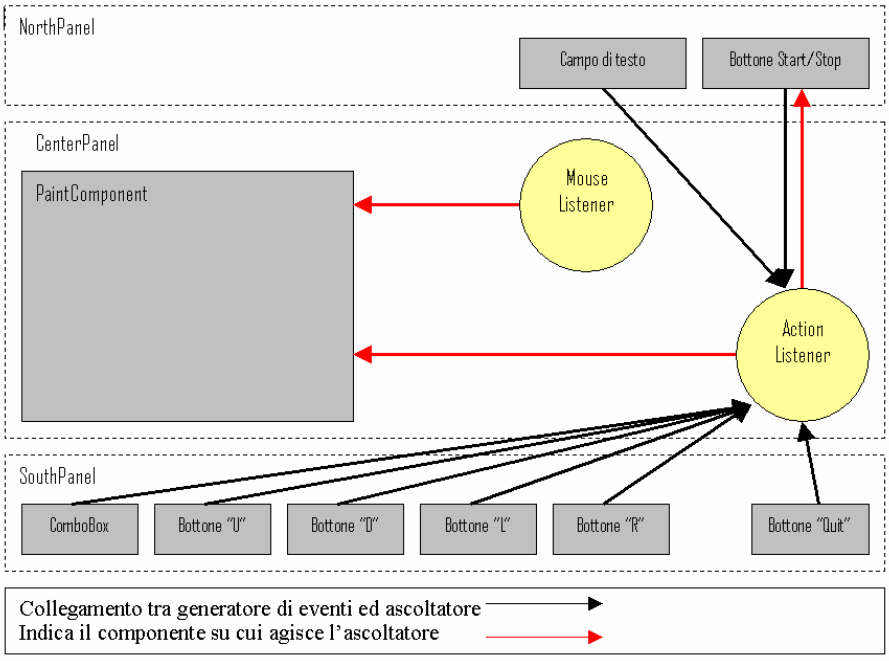
La realizzazione di questo programma interessa due grandi campi della programmazione Java: la grafica e gli eventi.

La finestra del programma (Frame di nome *ACMEPaint*) deve essere di dimensione definita (400x400) e all'avvio si deve posizionare al centro dello schermo. Come è già stato accennato nella descrizione del problema, la finestra deve essere suddivisa in tre parti (pannelli) chiamate, dall'alto al basso, *NorthPanel*, *CenterPanel*, *SouthPanel*.

- ***NorthPanel:*** in questo pannello devono essere posizionati un'etichetta di nome "Griglia", un campo di testo su cui l'utente può inserire la dimensione della griglia di disegno e un bottone "Start" che da il via al tracciamento di una riga poi, questo bottone si trasforma in uno chiamato "Stop" che termina le operazioni di tracciamento di righe.
- ***CenterPanel:*** questo è il pannello di disegno un cui deve essere disegnata una croce (il cursore) inizialmente posizionata al centro del pannello.
- ***SouthPanel:*** in questo pannello devono essere posizionati quattro pulsanti che permettono di muovere il cursore nelle quattro direzioni, il pulsante "Quit" utilizzato per terminare il programma e il menù a tendina dedicato alla scelta dei colori.

Gli eventi da gestire sono di due tipi:

- ***ActionEvent*:** questa categoria racchiude gli eventi generati dai bottoni, dai campi di testo e da altri componenti.
- ***MouseEvent*:** questa categoria di eventi contiene gli eventi generati dal mouse.



Quasi tutti gli eventi comportano modifiche al pannello centrale per cui mi sembra più semplice e adeguato aggiungere al pannello centrale i metodi necessari per l'intercettazione degli eventi generati dal mouse e dagli altri componenti. Per fare ciò il pannello centrale deve implementare le interfacce *ActionLis*tener (ascoltatore di *ActionEvent*) e *MouseListener* (ascoltatore di *MouseEvent*). I pannelli *north* e *south* devono conoscere chi ascolta gli eventi per cui è necessario "passargli",

all'atto di creazione, il pannello centrale che loro interpreteranno come ascoltatore degli eventi generati dai loro componenti.
Nella figura viene illustrato lo schema di interazione tra componenti grafici (bottoni…), ascoltatori e pannelli stessi. Gli ascoltatori sono rappresentati da un cerchio giallo; le frecce nere indicano il legame fra il componente generatore di evento e l'ascoltatore, mentre le frecce rosse indicano il pannello su cui l'ascoltatore agisce ogni qualvolta ascolta un evento.

Bisogna dunque realizzare un componente software (di nome *ACMEPaint*), che verrà lanciato per avviare il programma, contenente le operazioni di costruzione della finestra principale del programma. La finestra è divisa in tre parti (pannelli) che devono essere specificati in altrettante classi (*NorthPanel*, *CenterPanel* e *SouthPanel*) contenti anche i metodi necessari all'ascolto degli eventi.

Le linee disegnate sul pannello centrale devono permanere una volta premuto il pulsante *"Stop"*, quindi devono essere memorizzate di un vettore e richiamate ogni volta che si ridisegna il pannello. Ogni linea è caratterizzata da informazioni sul colore e da quattro coordinate che definiscono i due punti estremi del segmento; queste informazioni vanno immagazzinate in una nuova entità definita come astrazione di una linea.

Progetto

1. **ACMEPaint:** questa classe è il componente software del programma, cioè la classe che deve essere lanciata per avviare il programma. Deve quindi contenere il metodo *main*. La classe costruisce una finestra (*JFrame*) dal titolo "*ACMEPaint*" di dimensione 400x400 e posta al centro dello schermo. Per posizionarla al centro è necessario ricavare le dimensioni dello schermo in termini di risoluzione, questa operazione può essere fatta creando un oggetto *Toolkit* e richiamando il metodo *getScreenSize*. La finestra deve essere suddivisa in tre parti dal *BorderLayout*: *North*, *Center*, *South*. Queste parti devono essere riempite da tre diversi pannelli costruiti in altrettante classi. I panelli vanno poi inseriti nel *Container* caratteristico di ogni *JFrame*.

2. **SouthPanel:** questa classe deve costruire un pannello derivato dall'oggetto *JPanel*. Nel costruttore si fa riferimento al costruttore della superclasse e poi si inseriscono tutti i componenti richiesti dal problema. Nell'ordine, si devono inserire, un *JComboBox* in cui saranno indicati i colori con cui disegnare (nero, rosso, giallo, verde, blu), cinque *JButton* indicati con le stringhe "U", "D", "L", "R", "Quit" , i primi quattro permettono di muovere la crocetta , mentre l'ultimo permette di uscire dal programma. A tutti i componenti deve essere assegnato un oggetto ascoltatore, in questo caso si tratta del pannello centrale memorizzato dal costruttore in un campo di nome *ascoltatore*. Il legame tra componente e ascoltatore avviene tramite il metodo *addActionListener(ascoltatore).*

3. **NorthPanel:** questa classe deve costruire un pannello derivato dall'oggetto *JPanel*. Nel costruttore si fa riferimento al costruttore della superclasse e poi si inseriscono tutti i componenti richiesti dal problema. Nell'ordine, si devono inserire, un *JLabel* indicato con la stringa "Griglia", un *JTextField* in cui l'utente può inserire la dimensione della griglia (default=5) e un *JButton* "Start" che permettere di dare il via alle operazioni di disegno di una linea, quando queste operazioni sono in corso il bottone deve essere indicato con una stringa "Stop" e premendolo nuovamente termina le operazioni di disegno di linea. A tutti i componenti deve essere assegnato un oggetto ascoltatore, in questo caso si tratta del pannello centrale memorizzato dal costruttore in un campo di nome *ascoltatore*. Il legame tra componente e ascoltatore avviene tramite il metodo *addActionListener(ascoltatore).*

4. **CenterPanel:** questa classe deve costruire un pannello derivato dall'oggetto *JPanel*. Questo pannello, essendo un ascoltatore di eventi, deve implementare le interfacce *MouseListener* e *ActionListener*. I campi contenuti in questa classe sono i seguenti:

Tipo	Nome	Cosa indica
boolean	*disegna*	true se è stato premuto il bottone "Start", false se è stato premuto il bottone "Stop". Valore iniziale false.
int	*startX, startY*	Coordinate della crocetta quando viene premuto il bottone "Start"
int	*xMax, yMax*	Larghezza massima del pannello.
int	*grid*	Indica il valore della griglia. Valore iniziale 5.
java.awt.Color	*colore*	Indica il colore della crocetta. Valore iniziale black.
int	*x,y*	Coordinate della crocetta. Valori iniziali: x=200, y=140.
java.util.Vector	*listaLinee*	Contiene le linee disegnate.
Line	*linea*	Indica la linea che viene disegnata dal mouse.

Nel costruttore si fa riferimento al costruttore della superclasse *JPanel*. Su questo pannello l'utente deve poter disegnare, quindi occorre implementare il metodo *paintComponent(Graphics g)* in cui si richiamano tutte le funzioni necessarie per il disegno. In particolare ogniqualvolta viene disegnato il *paintComponent* si disegnano tutte le linee contenute nel vettore *listaLinee*. Attraverso un ciclo si estraggono una ad una tutte le linee dal vettore e si disegnano sul pannello. Successivamente va poi anche disegnata la crocetta formata da due segmenti, di lunghezza 10 pixel, posti inizialmente al centro del pannello. La classe contiene anche metodi modificatori per cambiare le coordinate della crocetta:

Signature	Cosa fa:
public void moveUp()	Sottrae alla y il valore di grid. Se è stato premuto "Start" crea e traccia una linea.
public void moveDown()	Somma alla y il valore di grid. Se è stato premuto "Start" crea e traccia una linea.
public void moveLeft()	Sottrae alla x il valore di grid. Se è stato premuto "Start" crea e traccia una linea.
public void moveRight()	Somma alla x il valore di grid. Se è stato premuto "Start" crea e traccia una linea.

Ogni modificatore per rendere visibile la modifica deve invocare il metodo *repaint()* che ridisegna il panello.
In particolare questo tipo di eventi viene ascoltato dal metodo *actionPerformed* dell'interfaccia *ActionListener* che deve essere implementata da questa classe. Quindi, oltre ai metodi descritti in precedenza, va definito il metodo *actionPerformed* che ascolta gli eventi generati dai componenti degli altri pannelli. Questo metodo deve identificare quale componente ha generato l'evento e intervenire nel modo più appropriato secondo la seguente tabella:

Generatore di evento	Cosa fa l'ascoltatore
JComboBox "colorList"	Assegna al campo *colore* l'oggetto di tipo *Color* derivante dalla stringa scelta dall'utente che viene recuperata tramite il comando *getSelectedItem*.
JButton "U"	Richiama il metodo *moveUp*.
JButton "D"	Richiama il metodo *moveDown*.
JButton "L"	Richiama il metodo *moveLeft*.
JButton "R"	Richiama il metodo *moveRight.*
JButton "Quit"	Termina il programma tramite *System.exit.*
JTextField "text"	Assegna al campo *grid* il valore ricavato dalla stringa contenuta nel *JTextField*. Inoltre viene riposizionata la crocetta sulla griglia.
JButton "Start"	Cambia la propria stringa identificativa in "Stop" e da il via alla fase di disegno settando a *true* il campo *disegna.* Inoltre salva nei campi *startX* e *startY* le coordinate correnti della crocetta.
JButton "Stop"	Cambia la propria stringa identificativa in "Start" e termina la fase di disegno settando a *false* il campo. Inoltre salva nel vettore *listaLinee* l'oggetto *linea* corrente.

In particolare l'identificazione del componente avviene controllando la stringa identificativa: per i bottoni è quella che appare su di loro, mentre per il *JComboBox* è "*comboBoxChanged*". Per identificare il componente *JTextField* si va ad esclusione.
L'utente deve avere la possibilità di spostare la crocetta anche con un click del mouse. Per questo la classe *CenterPanel* deve implementare l'interfaccia *MouseListener* che gestisce gli eventi del mouse. In particolare occorre implementare tutti i metodi dell'interfaccia (anche se inutilizzati). L'unico metodo utilizzato è *mouseClicked(mouseEvent e)* che cattura gli eventi generati quando l'utente clicca con il mouse sul pannello. Occorre, poi, recuperare le coordinate della freccia del mouse, se le coordinate sono dei multipli della griglia allora le cose vanno bene, altrimenti vanno approssimate al multiplo della griglia più vicino. Dopo aver effettuato questo controllo la crocetta va spostata in quel punto. Se la variabile *disegna* è *false* allora lo spostamento avviene normalmente, altrimenti viene creato un oggetto *Line*, che rappresenta una linea tra i punti (startX, startY) e (x, y), assegnato alla variabile *linea*. Per rendere visibile la modifica viene invocato il metodo *repaint()*. Nel *paintComponent*, oltre a disegnare tutte le linee contenute nel vettore, si disegna anche l'oggetto assegnato al riferimento *linea*.

5. **Line:** questa classe si occupa della creazione e della gestione di un oggetto di tipo Line, un'astrazione di una linea. In particolare questo oggetto contiene le informazioni relative al colore e alle quattro coordinate necessarie per identificare un segmento. Queste informazioni sono contenute in altrettanti campi privati della classe. All'atto di costruzione di una linea devono essere "passati" al costruttore tutte le informazioni caratteristiche della linea stessa: colore, x1, y1, x2, y2. La classe poi contiene i seguenti selettori:

Signature	Cosa fa:
public int getX1()	Ritorna la coordinata x1 dell'oggetto.
public int getY1()	Ritorna la coordinata y1 dell'oggetto.
public int getX2()	Ritorna la coordinata x2 dell'oggetto.
public int getY2()	Ritorna la coordinata y2 dell'oggetto.
public Color GetColor()	Ritorna il colore contenuto nel campo colore.

Implementazione

Riporto il codice della classe ACMEPaint:

```
package es05;

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 ACMEPaint{

	private static int x=400,y=400;

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

	public static void main(String[] args){
		//creo un oggetto Toolkit per ottenere le dimensioni dello schermo
		Toolkit tk=Toolkit.getDefaultToolkit();
		Dimension d=tk.getScreenSize();
		//creo la finestra
		JFrame finestra=new JFrame("ACMEPaint");
		finestra.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		finestra.setSize(new Dimension(x,y));
		//centro la finestra nello schermo
		finestra.setLocation((int)(d.getWidth()-x)/2,(int)(d.getHeight()-y)/2);
		finestra.setResizable(false);
		//Estraggo il container dal JFrame
		Container c=finestra.getContentPane();
		//Definisco il layout del container
		c.setLayout(new BorderLayout());
		//creo i tre pannelli e li aggiungo al container
		CenterPanel center=new CenterPanel();
		c.add(center,BorderLayout.CENTER);
		SouthPanel south=new SouthPanel(center);
		c.add(south,BorderLayout.SOUTH);
		NorthPanel north=new NorthPanel(center);
		c.add(north,BorderLayout.NORTH);
		//mostro la finestra sul desktop
		finestra.show();
	}
}
```

Riporto il codice della classe NorthPanel:

```
package es05;

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

/**
 * Questa classe estende la classe JPanel definendo un nuovo panello che
 * corrisponde alla parte superiore della finestra di ACMEPaint.
 * @author Andrea Urbini
 */

public class NorthPanel extends JPanel{

	CenterPanel ascoltatore;
```

```
	/**
	 * Questo metodo costruisce un oggetto NorthPanel.
	 * @param center pannello centrale, cioe' quello su cui si agisce premendo i bottoni.
	 */

	public NorthPanel(CenterPanel ascoltatore){
		super();
		this.ascoltatore=ascoltatore;
		//creo una JLabel
		JLabel lb=new JLabel("Griglia:");
		add(lb);
		//creo un campo di testo in cui il testo e' allineato a destra
		JTextField text=new JTextField("5",3);
		text.setHorizontalAlignment(JTextField.RIGHT);
		text.addActionListener(ascoltatore);
		add(text);
		//creo un JButton
		JButton button=new JButton("Start");
		button.addActionListener(ascoltatore);
		add(button);
	}

}
```

Riporto il codice della classe CenterPanel:

```
package es05;

import java.awt.*;
import java.awt.event.*;
import java.awt.Color;
import javax.swing.*;
import java.util.Vector;

/**
 * Questa classe estende la classe JPanel definendo un nuovo panello che
 * corrisponde alla parte centrale della finestra di ACMEPaint.
 * La classe implementa anche l'interfaccia MouseListener quindi si rende
 * ascoltatrice degli eventi generati dal mouse; implementa l'interfaccia
 * ActionListener ascoltando gli eventi provenienti dagli altri due pannelli.
 * La classe, rapresentando il pannello su cui si disegna, contiene anche i
 * metodi per spostare il cursore e disegnare linee.
 * @author Andrea Urbini
 */

public class CenterPanel extends JPanel implements MouseListener,ActionListener {

	private int xMax,yMax;
	private int grid=5;
	private boolean disegna=false;
	private Color colore=Color.black;
	private int y=140;
	private int x=200;
	private Vector listaLinee;
	private Line linea;
	private int startX,startY;

	/**
	 * Crea un oggetto di tipo CenterPanel. Un pannello di questo tipo genera
	 * eventi di tipo MouseEvent che vengono ascoltati dal pannello stesso.
	 */

	public CenterPanel(){
		super();
		addMouseListener(this);
		setBackground(Color.white);
		listaLinee=new Vector();
	}

	/**
	 * In questo metodo sono contenute le operazioni di disegno.
	 */

	public void paintComponent(Graphics g){
		super.paintComponent(g);
		xMax=getWidth();
		yMax=getHeight();
		//disegno le linee presenti nel vettore
```

```
		for (int i=0;i<listaLinee.size();i++){
			Line temp=(Line)listaLinee.get(i);
			g.setColor(temp.getColor());
			g.drawLine(temp.getX1(),temp.getY1(),temp.getX2(),temp.getY2());
		}
		//se esiste, disegno la variablie linea, e se viene premuto "stop" la
		//linea viene memorizzata nel vettore
		if (linea!=null){
			g.setColor(linea.getColor());
			g.drawLine(linea.getX1(),linea.getY1(),linea.getX2(),linea.getY2());
		}
		//disegno la crocetta
		g.setColor(colore);
		g.drawLine(x,y+5,x,y-5);
		g.drawLine(x+5,y,x-5,y);
	}

	/**Con questo metodo si sposta la crocetta in alto*/

	public void moveUp(){
		if(y-grid>=0){
			y-=grid;
			//se è stato premuto "Start" viene disegnata una linea tra il punto corrente e
il punto precedente
			if (disegna){
				linea=new Line(startX,startY,x,y,colore);
			}
			repaint();
		}
	}

	/**Con questo metodo si sposta la crocetta in basso*/

	public void moveDown(){
		if(y+grid<=yMax){
			y+=grid;
			//se è stato premuto "Start" viene disegnata una linea tra il punto corrente e
il punto precedente
			if (disegna){
				linea=new Line(startX,startY,x,y,colore);
			}
			repaint();
		}
	}

	/**Con questo metodo si sposta la crocetta verso sinistra*/

	public void moveLeft(){
		if(x-grid>=0){
			x-=grid;
			//se è stato premuto "Start" viene disegnata una linea tra il punto corrente e
il punto precedente
			if (disegna){
				linea=new Line(startX,startY,x,y,colore);
			}
			repaint();
		}
	}

	/**Con questo metodo si sposta la crocetta verso destra*/

	public void moveRight(){
		if(x+grid<=xMax){
			x+=grid;
			//se è stato premuto "Start" viene disegnata una linea tra il punto corrente e
il punto precedente
			if (disegna){
				linea=new Line(startX,startY,x,y,colore);
			}
			repaint();
		}
	}

	/**
	 * Questo metodo ascolta eventi di tipo ActionEvent provenienti da JButton,
	 * JTextField, JComboBox.
	 * @param e evento da ascoltare
	 */
```

```
	public void actionPerformed(ActionEvent e){
		//recupero il nome dell'oggetto che ha generato l'evento
		String name=e.getActionCommand();
		if (name.equals("Quit")){
			System.exit(0);
		}else if (name.equals("U")){
			moveUp();
		}else if (name.equals("D")){
			moveDown();
		}else if (name.equals("L")){
			moveLeft();
		}else if (name.equals("R")){
			moveRight();
		}else if (name.equals("Start")){
			//recupero l'oggetto che ha generato l'evento
			JButton button=(JButton)e.getSource();
			//cambio il testo del bottone
			button.setText("Stop");
			disegna=true;
			//fisso le coordinate da cui partira' la linea.
			startX=x;
			startY=y;
		}else if (name.equals("Stop")){
			//recupero l'oggetto che ha generato l'evento
			JButton button=(JButton)e.getSource();
			//cambio il testo del bottone
			button.setText("Start");
			disegna=false;
			//aggiungo l'oggetto assegnato a linea al vettore
			listaLinee.add(linea);
		}else if (name.equals("comboBoxChanged")){
			//recupero l'oggetto che ha generato l'evento
			JComboBox colorList=(JComboBox)e.getSource();
			//recupero il colore scelto dall'utente, e' in formato stringa, lo devo
convertire
			String scelta=(String)colorList.getSelectedItem();
			if (scelta.equals("Nero"))colore=Color.black;
			if (scelta.equals("Rosso"))colore=Color.red;
			if (scelta.equals("Giallo"))colore=Color.yellow;
			if (scelta.equals("Verde"))colore=Color.green;
			if (scelta.equals("Blu"))colore=Color.blue;
			repaint();
		}else{
			//recupero l'oggetto che ha generato l'evento
			JTextField text=(JTextField)e.getSource();
			//recupero il valore della griglia
			grid=Integer.parseInt(text.getText());
			//approssimo la x ai valori della griglia
			if (x%grid!=0){
				x=(x/grid)*grid;
			}
			//approssimo la y ai valori della griglia
			if (y%grid!=0){
				y=(y/grid)*grid;
			}
			repaint();
		}
	}

	/**
	 * Ascoltatore del mouse. Assegna alla variabili x,y le coordinate su cui il
	 * mouse ha cliccato.
	 * @param e evento generato dal mouse
	 */

	public void mouseClicked(MouseEvent e){
		x=e.getX();
		y=e.getY();
		//approssimo la x ai valori della griglia
		if (x%grid!=0){
			x=(x/grid)*grid;
		}
		//approssimo la y ai valori della griglia
		if (y%grid!=0){
			y=(y/grid)*grid;
		}
```

```
                //se è stato premuto "Start" viene disegnata una linea tra il punto corrente e il
punto precedente
                if (disegna){
                        linea=new Line(startX,startY,x,y,colore);
                }
                repaint();
        }

        /**metodo inutilizzato*/
        public void mousePressed(MouseEvent e){}
        /**metodo inutilizzato*/
        public void mouseReleased(MouseEvent e){}
        /**metodo inutilizzato*/
        public void mouseEntered(MouseEvent e){}
        /**metodo inutilizzato*/
        public void mouseExited(MouseEvent e){}
}
```

Riporto il codice della classe SouthPanel:

```
package es05;

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

/**
 * Questa classe estende la classe JPanel definendo un nuovo panello che
 * corrisponde alla parte inferiore della finestra di ACMEPaint.
 * @author Andrea Urbini
 */

public class SouthPanel extends JPanel{

        CenterPanel ascoltatore;

        /**
         * Questo metodo costruisce un oggetto SouthPanel.
         * @param center pannello centrale, cioe' quello su cui si agisce premendo i bottoni.
         */

        public SouthPanel(CenterPanel ascoltatore){
                super();
                this.ascoltatore=ascoltatore;
                String[] colori=new String[]{"Nero","Rosso","Giallo","Verde","Blu"};
                //creazione della JComboBox dei colori: non e' editabile dall'utente e
                //gli eventi che genera sono acoltati da questa classe.
                JComboBox colorList=new JComboBox(colori);
                colorList.setEditable(false);
                colorList.addActionListener(ascoltatore);
                add(colorList);
                //creazione del bottone
                JButton up=new JButton("U");
                up.addActionListener(ascoltatore);
                add(up);
                //creazione del bottone
                JButton down=new JButton("D");
                down.addActionListener(ascoltatore);
                add(down);
                //creazione del bottone
                JButton left=new JButton("L");
                left.addActionListener(ascoltatore);
                add(left);
                //creazione del bottone
                JButton right=new JButton("R");
                right.addActionListener(ascoltatore);
                add(right);
                //creazione del bottone
                JButton quit=new JButton("Quit");
                quit.addActionListener(ascoltatore);
                add(quit);
        }

}
```

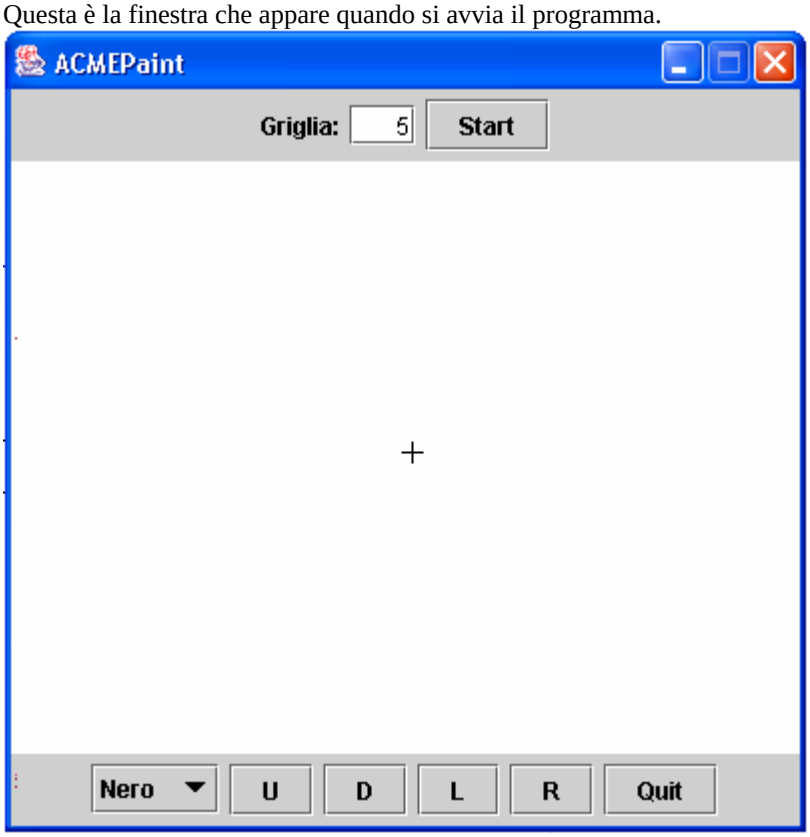
Riporto il codice della classe Line:

```
package es05;

import java.awt.Color;
```

```
/**
 * Questa classe rappresenta l'astrazione di un segmento caratterizzato da 4
 * coordinate (i due punti estremi) e da un colore.
 * @author andrea urbini
 */

public class Line{

	private int x1,x2,y1,y2;
	private Color colore;

	/**
	 * Costruttore
	 * @param x1 ascissa del primo estremo.
	 * @param y1 ordinata del primo estremo.
	 * @param x2 ascissa del secondo estremo.
	 * @param y2 ordinata del secondo estremo.
	 * @param colore colore del segmento
	 */

	public Line(int x1,int y1,int x2,int y2,Color colore){
		this.x1=x1;
		this.x2=x2;
		this.y1=y1;
		this.y2=y2;
		this.colore=colore;
	}

	/**
	 * Ritorna l'ascissa del primo estremo
	 * @return x1;
	 */

	public int getX1(){
		return x1;
	}

	/**
	 * Ritorna lordinata del primo estremo
	 * @return y1;
	 */

	public int getY1(){
		return y1;
	}

	/**
	 * Ritorna l'ascissa del secondo estremo
	 * @return x2;
	 */

	public int getX2(){
		return x2;
	}

	/**
	 * Ritorna l'ordinata del secondo estremo
	 * @return y2;
	 */

	public int getY2(){
		return y2;
	}

		/**
	 * Ritorna il colore del segmento
	 * @return colore;
	 */

	public Color getColor(){
		return colore;
	}
}
```

Casi d'uso

Questa è la finestra che appare quando si avvia il programma.



Concetti e tecniche acquisiti

Questa esercitazione mi ha permesso di conoscere due grandi campi della programmazione in Java: la grafica e la gestione degli eventi. La gestione della grafica non si è rivelata difficile anche se richiede molto tempo da dedicare allo studio dei metodi forniti, infatti non sono riuscito a dare un allineamento diverso ai bottoni. Questa carenza non la ritengo grave in quanto la grafica è passata in secondo piano rispetto alla gestione degli eventi. Durante questa fase mi sono accorto di numerosi errori progettuali (ogni pannello aveva un proprio listener) che mi hanno costretto all'intera riprogettazione del sistema. Fortunatamente le cose ora funzionano abbastanza bene.