Fondamenti di informatica L-B: Esercitazione 03

Autore: Andrea Urbini urbo83@tiscali.it

Matricola: 167064

Descrizione problema

L'esercitazione consiste nella costruzione di un nuovo "Plotter virtuale", chiamato *ACMEPlotter*, basato sulla classe *Plotter* della quale è disponibile solo la documentazione e il file compilato. Queste classi utilizzano per rappresentare i punti il modello *Point*, definito in un'ulteriore classe di cui è disponibile il listato, la documentazione e il file compilato.
La classe *Plotter* deve quindi essere estesa aggiungendo le seguenti funzionalità:

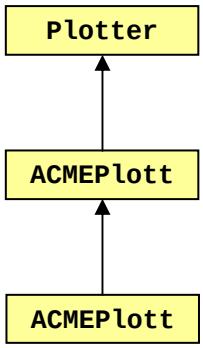
- *drawLine:* disegna una linea tra due punti specificati;
- *drawPolygon:* disegna un poligono generico collegando i punti contenuti in un array in ingresso;
- *drawCircle:* disegna una circonferenza conoscendo il centro e il raggio.

Un'estensione della classe *ACMEPlotter* è la classe *ACMEPlotterLog* che, per ogni suo metodo stampa su standard output una stringa di spiegazione sull'azione eseguita.

Inoltre si può utilizzare l'ereditarietà per modellare e categorizzare i possibili tipi di figura disegnabili su un *Plotter*.

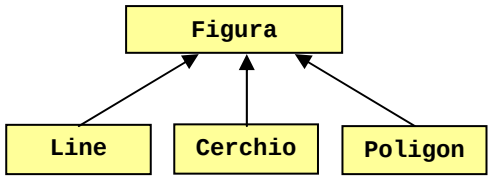
Analisi

Per aggiungere alla classe Plotter le funzionalità richieste non è più sufficiente (ed efficiente) creare una nuova classe e copiare il listato della precedente aggiungendo le parti che definiscono le nuove funzionalità. Bisogna sfruttare il principio dell'ereditarietà fornito da Java: è possibile estendere una classe (ad esempio *Plotter*) attraverso la creazione di una nuova classe derivata (*ACMEPlotter*) formata da un costruttore e dai nuovi metodi che si intendevano aggiungere alla classe di base.
Creando poi un oggetto di tipo *ACMEPlotter* esso avrà tutti i metodi di un oggetto di tipo *Plotter* e tutti i nuovi metodi definiti nella classe *ACMEPlotter*.
Le nuove funzionalità cui si faceva riferimento sono quelle specificate nella "Descrizione del Problema"; per implementarle occorre utilizzare i metodi forniti da *Plotter* combinati opportunamente tra loro.



La classe *ACMEPlotterLog* deve re-implementare solo i metodi della classe Plotter, mantenendo la possibilità per un oggetto *ACMEPlotterLog* di disporre dei metodi di un oggetto *ACMEPlotter*, per questo non può essere una classe derivata direttamente da *Plotter*. I metodi re-implementati richiamano le funzioni della classe *Plotter* e stampano su standard output una stringa di spiegazione.

Per categorizzare le possibili figure disegnabili su un *Plotter* attraverso l'ereditarietà, bisogna definire un oggetto di base, una figura geometrica generica, da cui derivare tutte le altre figure particolari (linea, cerchio, poligono…), che differiscono dalla prima solo nel modo di disegnarsi. In questo modo sarà sempre possibile aggiungere nuove figure senza estendere la classe *Plotter*. Va anche sviluppato un componente software di testing, di nome *TestFigure*, che interagisca con gli oggetti derivati (Cerchio, Line, Poligono).



Occorre anche sviluppare una classe di testing per l'ADT *ACMEPlotter* chiamata *TestMyPlotter*. Questo componente software deve creare un nuovo oggetto *ACMEPlotter* e utilizzare i metodi implementati nell'ADT dimostrando il loro corretto funzionamento.
La stessa cosa va fatta per l'ADT *ACMEPlotterLog* creando una classe *TestMyPlotterLog* allo stesso modo del componente software specificato sopra.

Progetto

1. **La classe derivata *ACMEPlotter*** deve avere la seguente signature per indicare la superclasse da cui deriva:
   public class ACMEPlotter extends Plotter
   In questo modo il compilatore sa che l'oggetto che si andrà a creare deve avere anche i metodi della superclasse *Plotter*.
   Il costruttore deve, quindi, far riferimento a quello della classe base.
   L'ADT *ACMEPlotter* non necessita di ulteriori campi rispetto alla classe base *Plotter*.
   I metodi aggiunti sono i seguenti:

   public void drawLine(Point p1,Pont p2)
   sposta il pennino virtuale nel punto p1 e lo fa muovere fino a p2 tracciando il percorso.

   public void drawPolygon(Point[] vertici)
   sposta il pennino virtuale nel punto vertici[0] e tramite un ciclo *for* collega tutti gli altri vertici con un segmento. Alla fine collega l'ultimo punto con il primo.

   public void drawCircle(Point centro,double r)
   Prima calcola le coordinate x0 e y0 del centro grazie ai metodi getX() e getY() della classe Point; Con un ciclo *for* calcola alcuni punti della circonferenza variando tra 0 e 2Π con passi di 0.01 l'angolo alfa delle seguenti formule:
   x=x0+r*cos(alfa); y=y0+r*sin(alfa);
   Con queste coordinate si costruiscono dei punti che vengono a mano a mano collegati fra di loro disegnando una circonferenza.
   La costante Π è definita dalla costante *Math.PI* e le funzioni coseno e seno delle due formule sono rispettivamente i metodi *Math.cos* e *Math.sin*.

2. **La classe derivata ACMEPlotterLog** re-implementa i metodi di Plotter (setColor, setThickness, moveTo, up, down) mantenendo inalterata la loro signature, richiamando il costruttore della classe base e aggiungendo un comando di stampa su standard output riguardante una stringa caratterista della funzione eseguita.
   Il costruttore fa riferimento a quello della classe base, che in questo caso è *ACMEPlotter*.

3. Questo diverso approccio alla soluzione del problema necessita della creazione quattro nuove classi: *Figura* (è la superclasse), *Line* (permette di disegnare una linea), *Poligono* (disegna un poligono) e *Cerchio* (disegna una circonferenza).Tutte le figure hanno in comune la possibilità di definire il colore e lo spessore del pennino che le disegnerà. Questi parametri vanno definiti al momento della creazione dell'oggetto corrispondente e quindi sono i parametri di ingresso di ogni costruttore. Nelle classi derivate questi parametri sono affiancati da quelli caratteristi della figura (come il centro e il raggio per il Cerchio).
   Ogni figura può essere disegnata su un Plotter, quindi ogni classe deve avere un metodo
   public void disegna(Plotter tavola)
   che la disegni sul plotter.
   E' in questo metodo che per ogni figura (anche per quella generica) vanno settati il colore e lo spessore del pennino. Queste operazioni sono state implementate solo nel metodo *disegna* della superclasse ed ereditati dalle classi derivate.
   Le operazioni di disegno vero e proprio delle figure sono identiche a quelle utilizzate nella classe ACMEPlotter.

4. **I componenti software** *TestMyPlotter* e *TestMyPlotterLog* creano un oggetto relativo all'ADT da testare e lo assegnato alla variabile tavola rispettivamente di tipo *ACMEPlotter* e *ACMEPlotterLog*. Le classi di testing, dopo averlo creato, interagiscono con esso invocando la costruzione di una retta, di un poligono e di un cerchio, cambiando colore e spessore al pennino. La classe *TestFigure* crea un oggetto *Plotter* su cui andare a disegnare i nuovi oggetti (di tipo *Cerchio, Line, Poligono*).

Implementazione

Di seguito riporto il listato della classe *ACMEPlotter*:

```
/**
 * Questa classe estende la classe Plotter aggiungendo nuovi metodi per disegnare
 * segmenti (drawLine), per disegnare poligoni (drawPolygon) e per disegnare
 * circonferenze (drawCircle).
 *
 * @author Andrea Urbini
 */
public class ACMEPlotter extends Plotter{

	/**
	 * Costruisce un oggetto ACMEPlotter
	 */

	public ACMEPlotter(){
		super();
	}

	/**
	 * Disegna un segmento tra i due punti ingresso. Alla fine del tracciamento
	 * il pennino si trova nel secondo punto.
	 * @param p1 primo punto
	 * @param p2 secondo punto
	 */

	public void drawLine(Point p1,Point p2){
		up();
		moveTo(p1);
		down();
		moveTo(p2);
	}

	/**
	 * Disegna un cerchio. Alla fine del tracciamento il pennino si trova sul
	 * centro della circonferenza. .
	 * @param centro centro del cerchio
	 * @param r raggio del cerchio
	 */

	public void drawCircle(Point centro,double r){
		//Calcola le coordinate del centro
		double x0=centro.getX();
		double y0=centro.getY();
		//Sposta il pennino nel punto corrispondente ad alfa=0
		up();
		Point start=new Point(x0+r*Math.cos(0),y0+r*Math.sin(0));
		moveTo(start);
		down();
		//Calcola i punti al variare di alfa (precisione 0.01) e li collega
		for (double alfa=0;alfa<=2*Math.PI;alfa+=0.01){
			Point punto=new Point(x0+r*Math.cos(alfa),y0+r*Math.sin(alfa));
			moveTo(punto);
		}
		//Collega l'ultimo punto con il primo.
		moveTo(start);
		//Sposta il pennino al centro
		up();
		moveTo(centro);
	}

	/**
	 * Disegna un poligono
	 * @param vertici array contenete i vertici del poligono
	 */

	public void drawPolygon(Point[] vertici){
		//Sposta il pennino al primo vertice del poligono
		up();
		moveTo(vertici[0]);
		down();
		//Collega tutti i vertici
		for (int i=1;i<vertici.length;i++){
			moveTo(vertici[i]);
```

```
                }
                //Collega l'ultimo vertice con il primo.
                moveTo(vertici[0]);
        }
}
```

Di seguito riporto il listato della classe *ACMEPlotterLog*:

```
/**
 * Questa classe estende la classe ACMEPlotter modificando i metodi della classe
 * base: ad ognuno di essi è stato aggiunta una linea di codice che stampa su
 * standard output quello che fanno.
 *
 * @author Andrea Urbini
 */
public class ACMEPlotterLog extends ACMEPlotter{

        /**
         * Costruisce un oggetto ACMEPlotterLog
         */

        public ACMEPlotterLog(){
                super();
        }

        /**
         * Alza il pennino virtuale
         */

        public void up(){
                super.up();
                System.out.println("E' stato alzato il pennino virtuale");
        }

        /**
         * Abbassa il pennino viruale
         */

        public void down(){
                super.down();
                System.out.println("E' stato abbasato il pennino virtuale");
        }

        /**
         * Sposta il pennino virtuale a punto p
         * @param p punto di arrivo del pennino
         */

        public void moveTo(Point p){
                super.moveTo(p);
                System.out.println("E' stato spostato il pennino virtuale alla posizione
"+p.toString());
        }

        /**
         * Definisce il colore del pennino
         * @param r tonalita' di rosso
         * @param g tonalita' di verde
         * @param b tonalita' di blu
         */

        public void setColor(double r,double g,double b){
                super.setColor(r,g,b);
                System.out.println("E' stato definito il seguente colore: rosso="+r+", verde="+g+",
blu="+b);
        }

        /**
         * Definisce lo spessore del pennino
         * @param s spessore del tratto del pennino
         */

        public void setThickness(int s){
                super.setThickness(s);
                System.out.println("E' stato definito uno spessore di "+s);
        }
```

```
}
```

Di seguito riporto il codice delle classi *Figura*, *Line*, *Cerchio*, *Poligono*.

```
/**
 * Questa classe rappresenta l'astrazione di una figura geometrica generica
 * caratterizzata da un colore e uno spessore del pennino che la disegna.
 *
 * @author Andrea Urbini
 */

public class Figura{

	//Campi privati
	private int spessore;
	private double r,g,b;

	/**
	 * Costruisce un oggetto di tipo Figura
	 * @param r tonalita' di rosso
	 * @param g tonalita' di verde
	 * @param b tonalita' di blu
	 * @param spessore spessore del pennino
	 */

	public Figura(double r,double g,double b,int spessore){
		this.r=r;
		this.g=g;
		this.b=b;
		this.spessore=spessore;
	}

	/**
	 * Disegna la figura generica. In questo caso setta solo il colore del
	 * pennino e il suo spessore
	 * @param tavola plotter su cui va disegnata la figura.
	 */

	public void disegna(Plotter tavola){
		tavola.setColor(r,g,b);
		tavola.setThickness(spessore);
	}
}

**
 * Questa classe rappresenta l'astrazione di una linea caratterizzata da un
 * colore e uno spessore del pennino che la disegna. Inoltre vanno definiti
 * anche i due punti estremi.
 *
 * @author Andrea Urbini
 */

public class Line extends Figura{

	private Point p1,p2;

	/**
	 * Costruisce un oggetto di tipo Line
	 * @param p1 primo punto del segmento
	 * @param p2 secondo punto del segmento
	 * @param r tonalita' di rosso
	 * @param g tonalita' di verde
	 * @param b tonalita' di blu
	 * @param spessore spessore del pennino
	 */


	public Line(Point p1,Point p2,double r,double g,double b,int spessore){
		super(r,g,b,spessore);
		this.p1=p1;
		this.p2=p2;
	}

	/**
	 * Disegna la linea sul plotter in ingresso.
	 * @param tavola plotter su cui va disegnata la linea.
	 */
```

```
	public void disegna(Plotter tavola){
		super.disegna(tavola);
		tavola.up();
		tavola.moveTo(p1);
		tavola.down();
		tavola.moveTo(p2);
	}
}

/**
 * Questa classe rappresenta l'astrazione di un cerchio caratterizzato da un
 * colore e uno spessore del pennino che lo disegna. Inoltre vanno definiti
 * anche il centro  e il raggio.
 *
 * @author Andrea Urbini
 */


public class Cerchio extends Figura{

	private double raggio;
	private Point centro;

	/**
	 * Costruisce un oggetto di tipo Cerchio
	 * @param centro centro del cerchio
	 * @param raggio raggio del cerchio
	 * @param r tonalita' di rosso
	 * @param g tonalita' di verde
	 * @param b tonalita' di blu
	 * @param spessore spessore del pennino
	 */

	public Cerchio(Point centro,double raggio,double r,double g,double b,int spessore){
		super(r,g,b,spessore);
		this.raggio=raggio;
		this.centro=centro;
	}

	/**
	 * Disegna il cerchio sul plotter in ingresso.
	 * @param tavola plotter su cui va disegnato il cerchio.
	 */

	public void disegna(Plotter tavola){
		//Calcola le coordinate del centro
		double x0=centro.getX();
		double y0=centro.getY();
		//Sposta il pennino nel punto corrispondente ad alfa=0
		tavola.up();
		Point start=new Point(x0+raggio*Math.cos(0),y0+raggio*Math.sin(0));
		tavola.moveTo(start);
		tavola.down();
		//Calcola i punti al variare di alfa (precisione 0.01) e li collega
		for (double alfa=0;alfa<=2*Math.PI;alfa+=0.01){
			Point punto=new Point(x0+raggio*Math.cos(alfa),y0+raggio*Math.sin(alfa));
			tavola.moveTo(punto);
		}
		//Colleaga l'ultimo punto con il primo.
		tavola.moveTo(start);
		//Sposta il pennino al centro
		tavola.up();
		tavola.moveTo(centro);
	}

}

/**
 * Questa classe rappresenta l'astrazione di un poligono caratterizzato da un
 * colore e uno spessore del pennino che lo disegna. Inoltre va definito
 * anche l'array dei suoi vertici.
 *
 * @author Andrea Urbini
 */

public class Poligono extends Figura{

	private Point[] vertici;
```

```
	/**
	 * Costruisce un oggetto di tipo Poligono
	 * @param vertici array contenente i vertici del poligono
	 * @param r tonalita' di rosso
	 * @param g tonalita' di verde
	 * @param b tonalita' di blu
	 * @param spessore spessore del pennino
	 */

	public Poligono(Point[] vertici,double r,double g,double b,int spessore){
		super(r,g,b,spessore);
		this.vertici=vertici;
	}

	/**
	 * Disegna il poligono sul plotter in ingresso.
	 * @param tavola plotter su cui va disegnato il poligono.
	 */

	public void disegna(Plotter tavola){
		super.disegna(tavola);
		//Sposta il pennino al primo vertice del poligono
		tavola.up();
		tavola.moveTo(vertici[0]);
		tavola.down();
		//Collega tutti i vertici
		for (int i=1;i<vertici.length;i++){
			tavola.moveTo(vertici[i]);
		}
		//Collega l'ultimo vertice con il primo.
		tavola.moveTo(vertici[0]);
	}
}
```

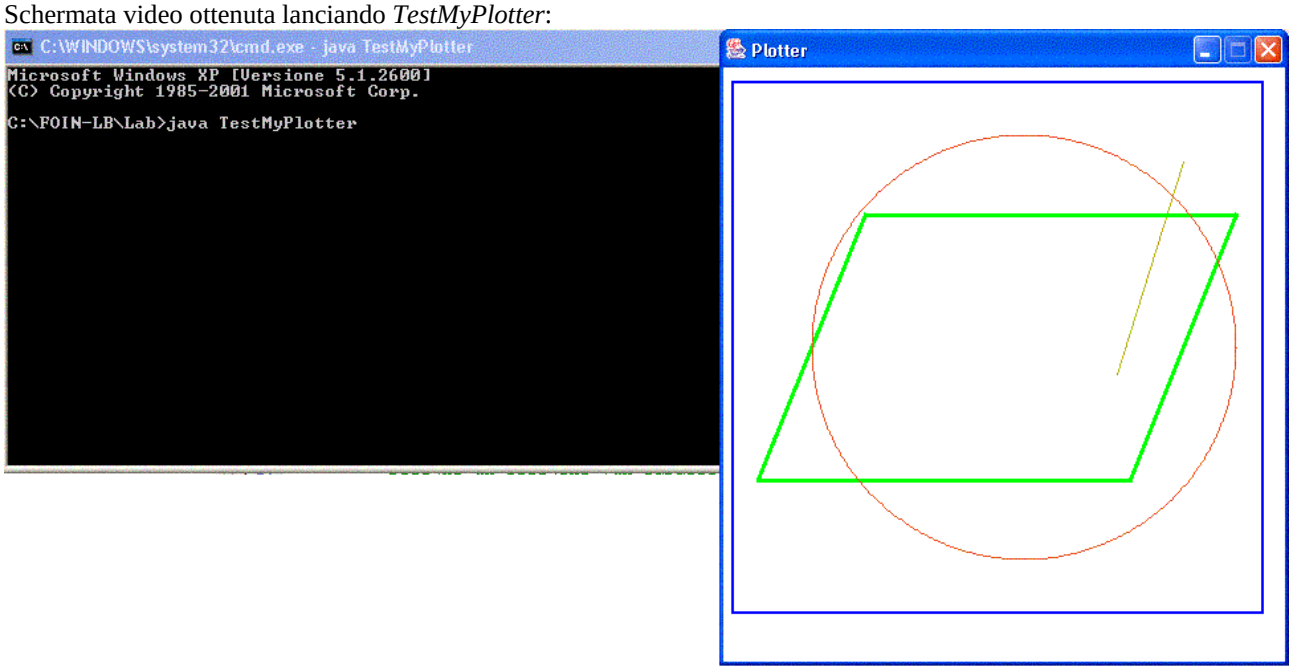
Di seguito riporto il listato della classe *TestMyPlotter*:

```
/**
 * Componente software di testing della classe ACMEPlotter.
 * @author Andrea Urbini
 */

public class TestMyPlotter{

public static void main(String[] args){
		//Creo un nuovo oggetto ACMEPlotter
		ACMEPlotter tavola=new ACMEPlotter();
		//Setto il colore giallo
		tavola.setColor(0.7,0.7,0.0);
		//Disegno una linea
		tavola.drawLine(new Point(0.7,0.7),new Point(0.45,-0.1));
		//Setto lo spessore
		tavola.setThickness(3);
		//Setto il colore verde
		tavola.setColor(0.0,1.0,0.0);
		//Disegno un poligono (un parallelogramma)
		tavola.drawPolygon(new Point[]{new Point(0.9,0.5),new Point(0.5,-0.5),new Point(-
0.9,-0.5),new Point(-0.5,0.5)});
		//Setto il colore rosso
		tavola.setColor(0.9,0.3,0.1);
		//Setto lo spessore
		tavola.setThickness(1);
		//Disegno un cerchio
		tavola.drawCircle(new Point(0.1,0.0),0.8);
	}

}
```

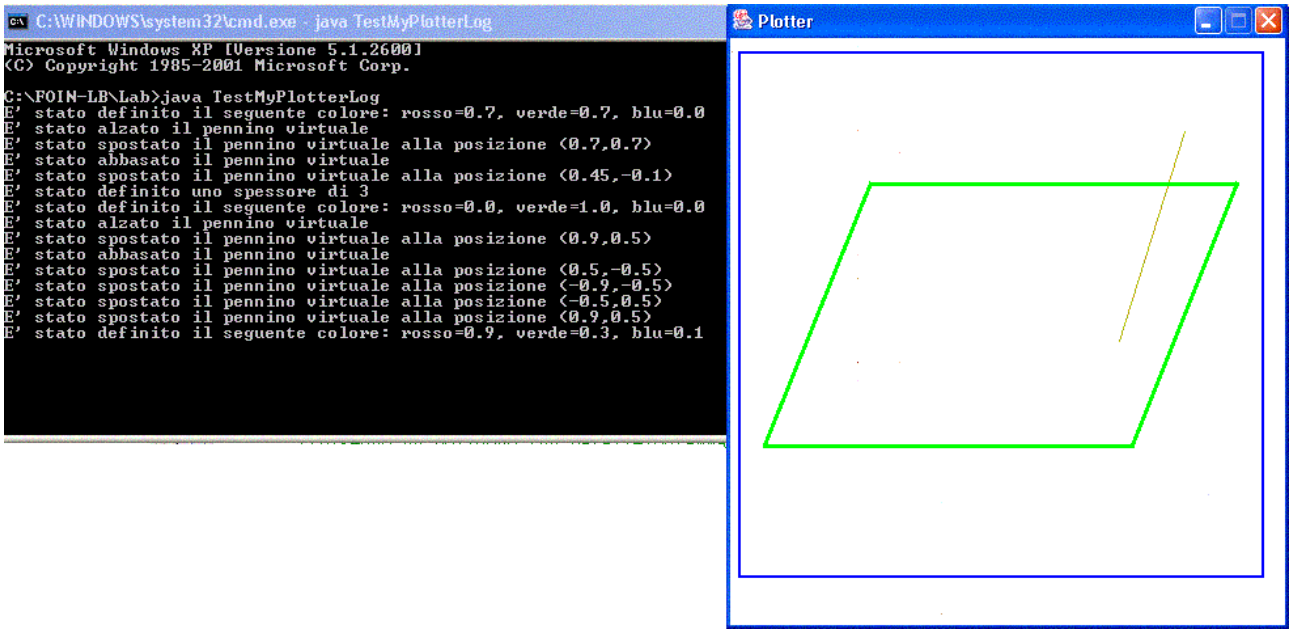
Il listato della classe *TestMyPlotterLog* è praticamente identico a questo quindi non lo riporto, è comunque disponibile nel file *src\TestMyPlotterLog.java*.

Casi d'uso

Schermata video ottenuta lanciando *TestMyPlotter*:



Schermata video ottenuta lanciando *TestMyPlotterLog*:



Concetti e tecniche acquisiti

Questa esercitazione mi ha permesso di comprendere nel miglior modo (spero) l'ereditarietà, le sue caratteristiche e i suoi vantaggi. Senza l'ereditarietà sarebbe stata impossibile (o comunque molto difficoltosa) l'espansione della classe *Plotter* di cui non è disponibile il listato. Tramite la classe *ACMEPlotter* ho potuto implementare nuovi metodi, mentre con la classe *ACMEPlotterLog* ho dovuto re-implementare metodi già esistenti nella classe *Plotter*.
Il nuovo approccio al problema proposto mi ha permesso di comprendere un ulteriore pregio dell'ereditarietà: la categorizzazione della realtà in oggetti sempre più specifici.