

Esercitazione su Java

I primi esercizi

- Scrivere un programma che stampa una stringa costante sullo schermo
- Modificare il programma in modo tale da acquisire la stringa dalla linea di comando con cui il programma viene eseguito
- Definire una classe Punto e scrivere un main che crea un punto, lo sposta in uno spazio bidimensionale, ne stampa le coordinate



L'editor di figure geometriche

- Un editor di figure geometriche consente la costruzione di immagini complesse, date dalla composizione di più figure elementari
- L'editor consente di stampare sullo schermo una figura complessa
- Esso inoltre consente di spostare la figura nello spazio bidimensionale
- Si facciano le seguenti ipotesi semplificative:
 - Ci si concentra sull'implementazione del nucleo del programma, trascurando gli aspetti relativi all'interfaccia utente
 - Le figure geometriche da considerare nella prima implementazione del sistema sono i rettangoli e le linee
 - Sono già date la classe Screen, la classe Point, ed una classe per la gestione delle eccezioni



Interfaccia della classe Screen

- Costruttori
 - Screen(int w, int h)
 - Screen()
- Metodi
 - writePoint(Point p) throws OutOfBoundsException
 - erasePoint(Point p) throws OutOfBoundsException
 - writeLine(Point p0, Point p1) throws OutOfBoundsException
 - eraseLine(Point p0, Point p1) throws OutOfBoundsException
 - boolean isOutOfBounds(Point p)
 - refresh()
 - clear()
 - NB: writePoint, writeLine, erasePoint, eraseLine non causano un refresh dell'immagine che viene vista dall'utente. Il refresh deve essere effettuato esplicitamente



Interfacce di Point e OutOfBoundsException

Point

- Costruttori
 - Point(int x, int y)
 - Point ()
- Metodi
 - int getX()
 - int getY()
 - set(int x, int y)
 - String toString()

OutOfRangeException

- Costruttore
 - OutOfBoundsException(Point p, int x, int y)
- Metodo
 - String toString()



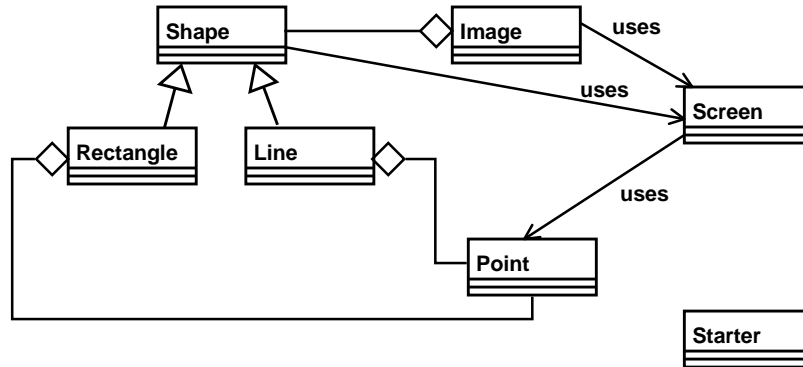
Il progetto del nucleo dell'editor di figure geometriche (1)

- Le classi
 - Screen (già data)
 - Point (già data)
 - OutOfBoundsException
 - Image (rappresenta l'immagine complessa)
 - Line
 - Rectangle
 - Shape (è la superclasse comune tra Line e Rectangle)
 - E` una classe o un'interfaccia?
 - Starter (la classe che ospita il main che ci serve per testare il nucleo dell'editor)



Il progetto del nucleo dell'editor di figure geometriche (2)

- Le *relazioni* tra le classi



Image

- Costruttore
 - Image(Screen s)
- Metodi
 - void addShape(Shape s)
 - void removeShape(Shape s)
 - Enumeration elements()
 - void show()
- Struttura dati privata
 - Vector shapes
 - Screen s



Shape

- Metodi:
 - void move(int dx, int dy)
 - void draw(Screen s) throws OutOfBoundsException
 - void erase(Screen s) throws OutOfBoundsException
- NB: l'eccezione che viene sollevata da Screen non viene gestita da draw ed erase ed è a carico degli utilizzatori di Shape
- Tutti i suoi metodi sono abstract
- ☞ E' possibile definirla come una interfaccia

- Rectangle e Line implementano Shape



La classe Starter: implementazione del main (1)

```
public class Starter {
public static void main(String[] args) {
    Image image;
    Screen screen=new Screen(50,20);
    image=new Image(screen);
    try{
        image.addShape(new Rectangle(new Point (3,7),new Point(40,18)));
        image.addShape(new Rectangle(new Point (7,9),new Point(33,16)));
        image.addShape(new Line(new Point(33,16),new Point (7,9)));
        image.show();
        Thread.sleep(3000);
    }
}
```



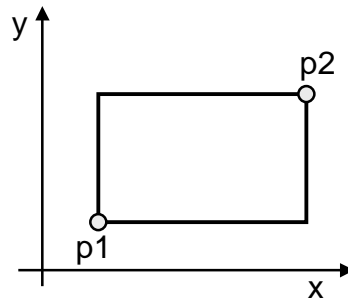
La classe Starter: implementazione del main (2)

```
for(int j=0;j<5;j++) {
    for(Enumeration e=image.elements();e.hasMoreElements();)
        ((Shape)(e.nextElement())).move(0,1);
    image.show();
    Thread.sleep(3000);
}
} catch(Exception e) {
    e.printStackTrace();
}
}
```



Implementazione di Rectangle (1)

```
public class Rectangle implements Shape{
    private Point p1,p2;
    public Rectangle(Point p1,Point p2) {
        this.p1=p1; this.p2=p2;
    }
    public void move(int dx, int dy){
        p1.set(p1.getX()+dx,p1.getY()+dy);
        p2.set(p2.getX()+dx,p2.getY()+dy);
    }
}
```



Implementazione di Rectangle (1)

```
public void draw(Screen s) throws DrawIt.OutOfBoundsException {
    s.drawLine(p1,new Point(p2.getX(),p1.getY()));
    s.drawLine(new Point(p2.getX(),p1.getY()),p2);
    s.drawLine(p2, new Point(p1.getX(), p2.getY()));
    s.drawLine(new Point(p1.getX(), p2.getY()),p1);
}
public void erase(Screen s) throws OutOfBoundsException{
    s.eraseLine(p1,new Point(p2.getX(),p1.getY()));
    s.eraseLine(new Point(p2.getX(),p1.getY()),p2);
    s.eraseLine(p2, new Point(p1.getX(), p2.getY()));
    s.eraseLine(new Point(p1.getX(), p2.getY()),p1);
}
}
```



Implementazione di Image (1)

```
import java.util.Vector;
import java.util.Enumeration;

public class Image {
    private Vector shapes;
    private Screen s;

    public Image(Screen s){
        this.s=s;
        shapes=new Vector();
    }

    public void addShape(Shape s){
        shapes.addElement(s);
    }

    public void removeShape(Shape s){
        shapes.removeElement(s);
    }

    public Enumeration elements(){
        return shapes.elements();
    }
}
```



Implementazione di Image (2)

```
public void show(){
    s.clear();
    for(Enumeration e=shapes.elements();e.hasMoreElements();)
    {
        try{
            ((Shape)e.nextElement()).draw(s);
        } catch(OutOfBoundsException ex) {}
    }
    s.refresh();
}
}
```

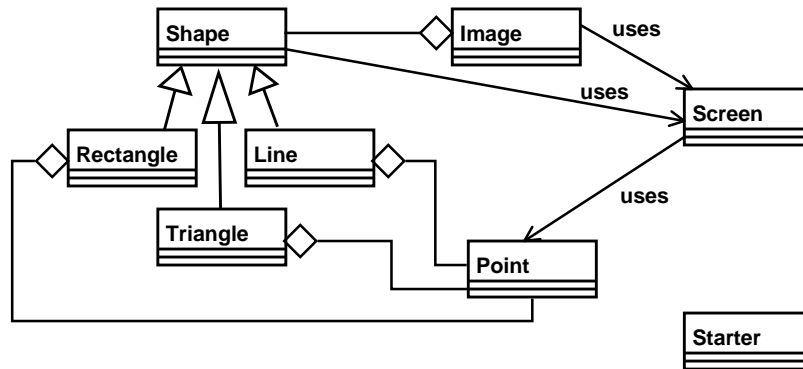


Estensioni

- Si vuole introdurre una nuova figura geometrica. Per esempio, il triangolo
- Si vuole salvare le immagini su file



Introduzione di una nuova figura geometrica: riprogettazione



- Una nuova classe deve essere implementata. Il resto del sistema non subisce modifiche (eccetto per il main in cui verranno create istanze di Triangle)



Salvataggio delle immagini su file

- Alternative
 - Image, tutte le classi che implementano Shape, la classe Point e la classe Screen definiscono due metodi:
 - Un metodo di scrittura su file: `writeData(DataOutputStream out)`
 - Un metodo di lettura da file: `readData(DataInputStream in)`
 - Questi metodi vengono invocati per scrivere/leggere le informazioni relative ad ogni singolo oggetto
 - Si sfrutta il meccanismo di serializzazione degli oggetti che viene fornito dalla JDK
 - Tutte le classi le cui istanze devono essere memorizzate su file devono implementare l'interfaccia `Serializable`
 - Si usano i metodi `writeObject` e `readObject` messi a disposizione da `ObjectOutputStream` e `ObjectInputStream`

