Lesson 6: Inheritance

Exercise 1: Shapes Application

Task: Create an application according to the following specification.

Use Case Model

Shapes

The system provides for the user three main services. First the user can draw various shapes such as rect, elipse, triangle. The user can select border color of the shapes that are drawn otherwise default border color is used. Second, user can fill selected shape with a selected fill color. The boder color and the fill color of the shape can be different. Third, the user can erase selected shape by specifying the index of that shape.

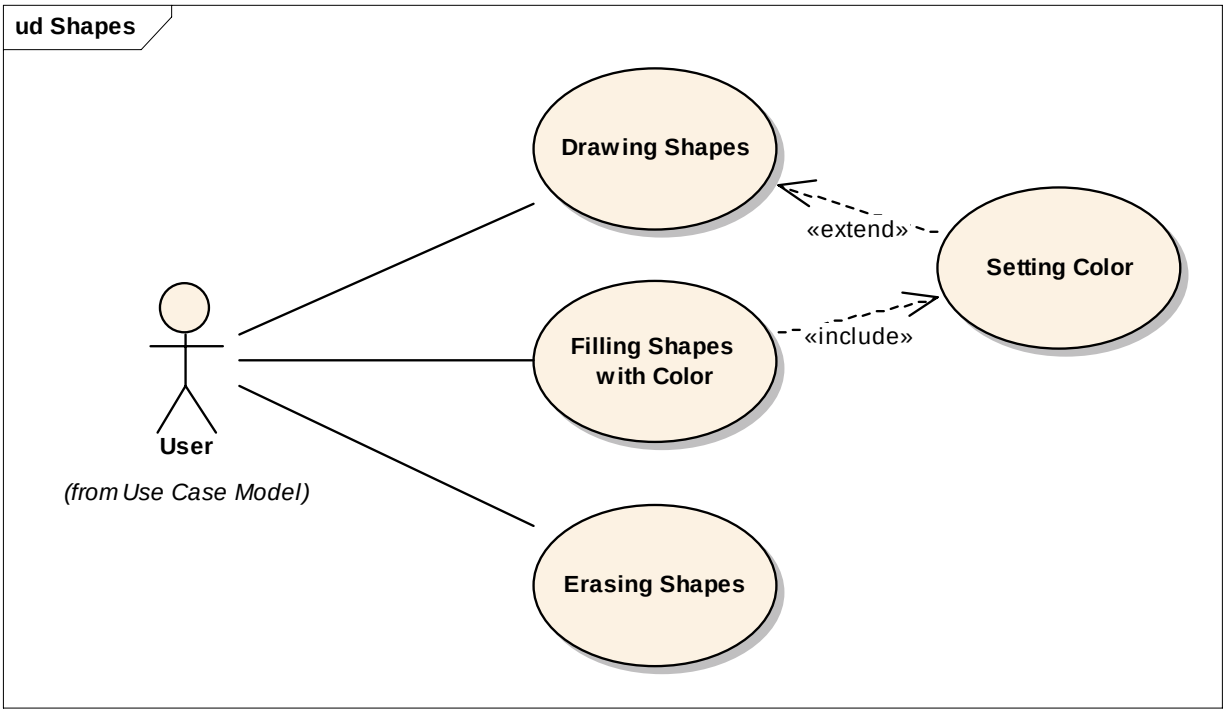

Figure 1 : Shapes

Console

Type: *public* **MessageEndpoint**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 27.3.2006 11:09:51. Modified on 27.3.2006 11:11:09. Author: Pavel Čech

EditBox

Type: *public* **MessageEndpoint**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 27.3.2006 11:00:52. Modified on 27.3.2006 11:11:41. Author: Pavel Čech

Vector

Type: *public* **MessageEndpoint**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 27.3.2006 11:07:31. Modified on 27.3.2006 11:07:49. Author: Pavel Čech

Drawing Shapes

Type: *public* **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 23.3.2006 22:21:02. Modified on 26.3.2006 22:21:48. Author: Pavel Čech

Drawing shape with default color.

Connections

- Extend link from usecase *Setting Color*
- Used by actor *User <Use Case Model>*

Scenarios

Drawing {Basic Path}.

1. User selects the shape to be drawn.
2. System asks for coordinates and sizes.
3. User sets the coordinates and sizes.
4. System draws the shape.

On the drawing interaction participates the following classes: ShapesApp, Drawing, Shape. ShapesApp will handles the user action. It creates the particular shape that the user has selected. Then necessary params are obtained from the user and set to the shape. Next the shape is added to the drawing and all the shapes are redrawn. Last the window is forced to repaint itself.

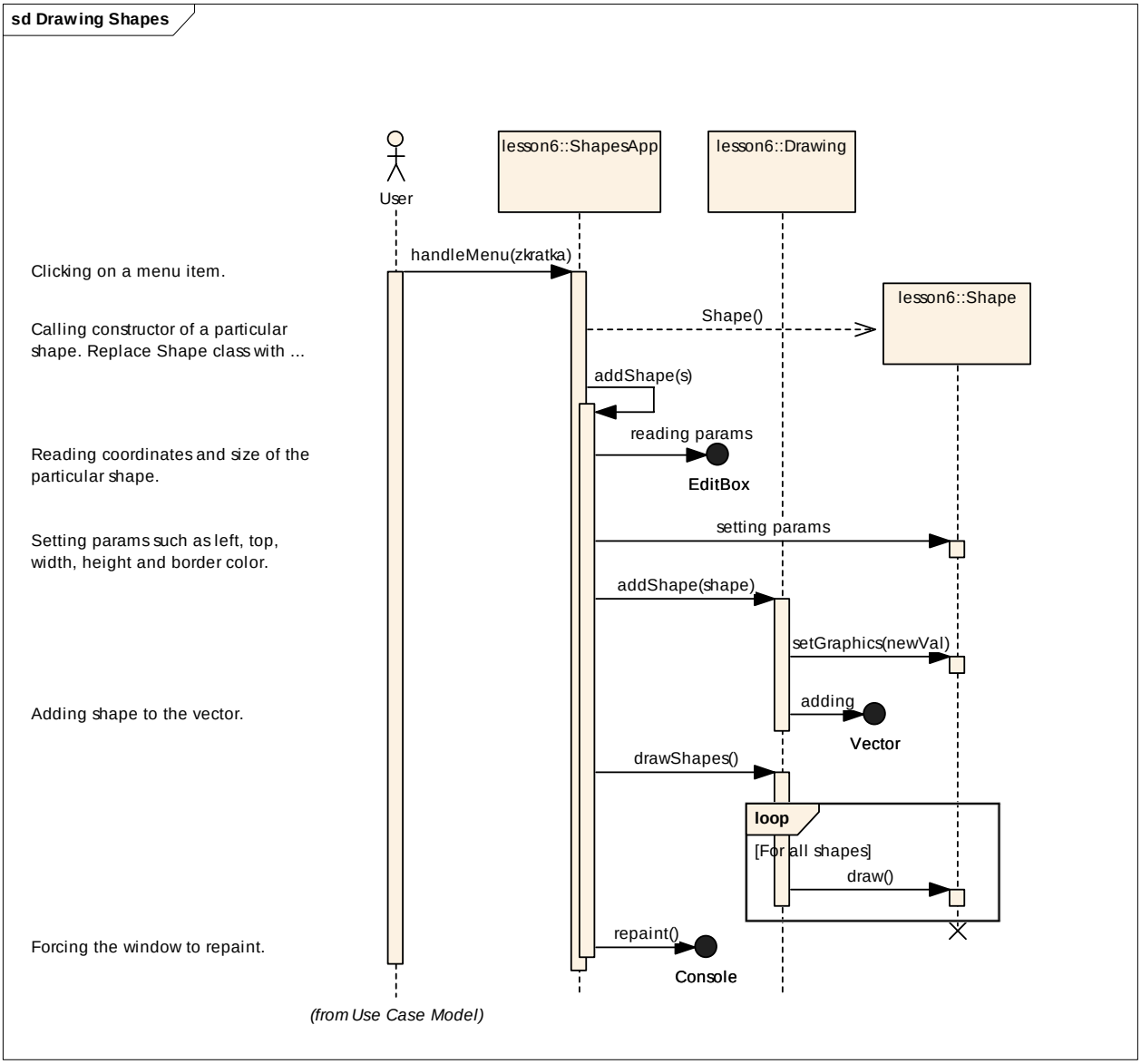

Figure 2 : Drawing Shapes

Drawing Shapes Messages

ID	Message	From Object	To Object	Notes
1	handleMenu(int)	User	ShapesApp	Clicking on a menu item.
2	Shape()	ShapesApp	Shape	Calling constructor of a particular shape. Replace Shape class with one of its subclasses such as Rectangle, Elipse or Triangle.
3	addShape(Shape)	ShapesApp	ShapesApp	
4	reading params	ShapesApp	EditBox	Reading coordinates and size of the particular shape.
5	setting params	ShapesApp	Shape	Setting params such as left, top, width, height and border color.
6	addShape(Shape)	ShapesApp	Drawing	

7	setGraphics(Graphics2D)	Drawing	Shape	
8	adding	Drawing	Vector	Adding shape to the vector.
9	drawShapes()	ShapesApp	Drawing	
10	draw()	Drawing	Shape	
11	repaint()	ShapesApp	Console	Forcing the window to repaint.

<anonymous>

Type: *public* **InteractionFragment**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 27.3.2006 11:12:00. Modified on 27.3.2006 11:12:37. Author: Pavel Čech

Erasing Shapes

Type: *public* **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 23.3.2006 22:22:17. Modified on 26.3.2006 22:22:22. Author: Pavel Čech

Erasing selected shape.

Connections

- Used by actor *User <Use Case Model>*

Scenarios

Erasing {Basic Path}.
1. User selects the menu item for erasing one shape
2. System asks for an index of the shape.
3. User sets the index of the shape to be erased.
4. System erases the shape.

Filling Shapes with Color

Type: *public* **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 23.3.2006 22:24:21. Modified on 27.3.2006 10:56:39. Author: Pavel Čech

Filling shape with selected color.

Connections

- Include link to usecase *Setting Color*
- Used by actor *User <Use Case Model>*

Scenarios

Filling shape with color {Basic Path}.
1. User selects the menu item for filling one shape
2. System asks for an index of the shape.
3. User sets the index of the shape to be erased.

4. "Setting Fill Color" is called
5. System fills the shape.

Setting Color

Type: *public* **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Shapes
Details: Created on 23.3.2006 22:25:34. Modified on 26.3.2006 22:24:03. Author: Pavel Čech

Setting a selected color.

Connections

- Extend link to usecase *Drawing Shapes*
- Include link from usecase *Filling Shapes with Color*

Scenarios

Setting color {Alternate}.
1. User wants color to be changed.
2. System displays dialog to set the color.
3. User sets the color.
4. System changes the color.

Class Model

lesson6

The core of the application is represented by the ShapesApp class. It provides the I/O functionality with the user. It is responsible for creating the Drawing class which is the collection of the shapes beign drawn on the graphical device. Thus Drawing holds the references to all the instances of the Shape subclass. Shape class encapsulate the common attributes and operations of particular shapes. Hence Rectangle, Elipse and Triangle classes are the children of the general Shape class. Shapes children redefines only those methods that are different for particular child.

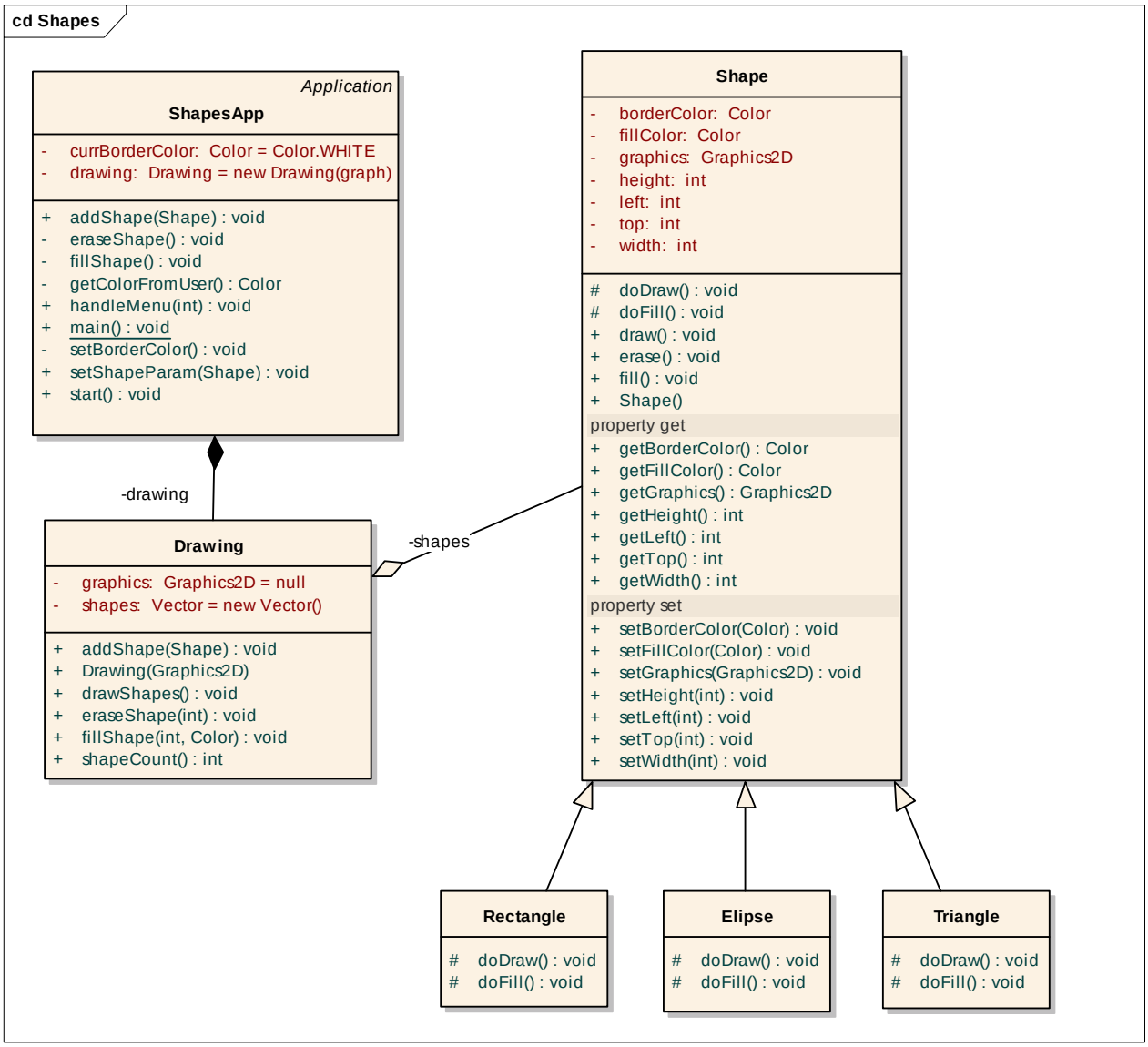

Figure 3 : Shapes

lesson6::Drawing

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: lesson6
Details: Created on 23.3.2006 22:06:13. Modified on 26.3.2006 20:54:25. Author: Pavel Čech

Class representing the collection of shapes. It holds the references to all shapes using collection object. It provides the handling routines for adding, drawing, filling and erasing shapes. It provides graphical device for all shapes on which they are drawn.

Connections

- Aggregation link to class *ShapesApp*
- Aggregation link from class *Shape*
- Association link to class *Vector<java.util>*
- Association link to class *Graphics2D<java.awt>*

lesson6::Drawing Attributes

Attribute	Type	Notes
graphics	private : *Graphics2D*	Graphical device to be drawn to. Initial Value: null;
shapes	private : *Vector*	Collection of the shapes. Initial Value: new Vector();

lesson6::Drawing Methods

Method	Type	Notes
addShape (*Shape*)	public: *void*	param: shape [Shape - in] shape Sets a graphics (Graphics2D) device to a shape and adds a shape into the shapes collection.
Drawing (*Graphics2D*)	public:	param: graphics [Graphics2D - in] Device on which the drawing i.e. shapes will be displayed. Creates a new instance and sets graphics device.
drawShapes ()	public: *void*	Draws all the shapes in the shapes collection. It iterates for all shape items in the shapes collanection and call draw() method for each shape object.
eraseShape (*int*)	public: *void*	param: index [int - in] Index of the shape to be erased. Erases a shape given by the index. It sets the color of the border and the fill color to the background color and calls draw() and fill().
fillShape (*int, Color*)	public: *void*	param: index [int - in] Index of the shape to be filled in. param: color [Color - in] Color to fill in the shape. Fills a shape given by the index. It gets the particular shape from the shapes collection, sets the color and calls the fill() method.

shapeCount ()	public: *int*	The count of the shapes in the shapes collection. It calls the size() method of the shapes collection.

lesson6::Elipse

Type: *public* **Class**
Extends: *Shape.*
Status: Proposed. Version 1.0. Phase 1.0.
Package: lesson6
Details: Created on 23.3.2006 22:06:52. Modified on 26.3.2006 21:18:09. Author: Pavel Čech

Class representing the elipses. It is the child of the shape class. It redefines doDraw() and doFill().

Connections

- Generalization link to class *Shape*

lesson6::Elipse Methods

Method	Type	Notes
doDraw ()	protected: *void*	
doFill ()	protected: *void*	

lesson6::Rectangle

Type: *public* **Class**
Extends: *Shape.*
Status: Proposed. Version 1.0. Phase 1.0.
Package: lesson6
Details: Created on 23.3.2006 22:06:29. Modified on 26.3.2006 21:17:51. Author: Pavel Čech

Class representing the rectangles. It is the child of the shape class. It redefines doDraw() and doFill().

Connections

- Generalization link to class *Shape*

lesson6::Rectangle Methods

Method	Type	Notes
doDraw ()	protected: *void*	
doFill ()	protected: *void*	

lesson6::Shape

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: lesson6
Details: Created on 23.3.2006 22:06:22. Modified on 26.3.2006 21:08:05. Author: Pavel Čech

Parent for all particular shapes. It declares variables for coordinates, sizes and colors. It also holds the reference to graphical device. It provides the common basic functionality such as getters and setters and setting colors before the actual drawing or filling is done.

Connections

- Aggregation link to class *Drawing*
- Generalization link from class *Triangle*
- Generalization link from class *Elipse*
- Generalization link from class *Rectangle*

lesson6::Shape Attributes

Attribute	Type	Notes
borderColor	private : *Color*	Color for borders.
fillColor	private : *Color*	Fill color of the shape.
graphics	private : *Graphics2D*	Graphical device for drawing the shape.
height	private : *int*	Height of the shape.
left	private : *int*	Horizontanal coordinate of the shape.
top	private : *int*	Vertical coordinate of the shape.
width	private : *int*	Width of the shape.

lesson6::Shape Methods

Method	Type	Notes
doDraw ()	protected: *void*	Does the actual drawing of the shape using graphics device.
doFill ()	protected: *void*	Does the actual filling of the shape using graphics device.
draw ()	public: *void*	Sets the color to the border color and calls the doDraw().
erase ()	public: *void*	Sets the color to the background color and calls the doDraw() and doFill().
fill ()	public: *void*	Sets the color to the fill color and calls the doFill().
getBorderColor ()	«property get» public: *Color*	attribute_name = 'borderColor'
getFillColor ()	«property get» public: *Color*	attribute_name = 'fillColor'
getGraphics ()	«property get» public: *Graphics2D*	attribute_name = 'graphics'
getHeight ()	«property get» public: *int*	attribute_name = 'height'
getLeft ()	«property get» public: *int*	attribute_name = 'left'
getTop ()	«property get» public: *int*	attribute_name = 'top'
getWidth ()	«property get» public: *int*	attribute_name = 'width'
setBorderColor (*Color*)	«property set»	param: newVal [Color - in]

	public: *void*	attribute_name = 'borderColor'
setFillColor (*Color*)	«property set» public: *void*	param: newVal [Color - in] attribute_name = 'fillColor'
setGraphics (*Graphics2D*)	«property set» public: *void*	param: newVal [Graphics2D - in] attribute_name = 'graphics'
setHeight (*int*)	«property set» public: *void*	param: newVal [int - in] attribute_name = 'height'
setLeft (*int*)	«property set» public: *void*	param: newVal [int - in] attribute_name = 'left'
setTop (*int*)	«property set» public: *void*	param: newVal [int - in] attribute_name = 'top'
setWidth (*int*)	«property set» public: *void*	param: newVal [int - in] attribute_name = 'width'
Shape ()	public:	

lesson6::ShapesApp

Type: *public* **Class**
Extends: *Application.*
Status: Proposed. Version 1.0. Phase 1.0.
Package: lesson6
Details: Created on 23.3.2006 22:03:25. Modified on 27.3.2006 11:28:04. Author: Pavel Čech

Application class providing the I/O functionality. Provides menu and the graphical device on which the drawing takes place. It also provides a dialog box for quering the user. It is responsible for handling the menu clicks and responding with appropriate action. It creates the instance of the Drawing class and delegates commands to that class. When asked it creates also instances of particular shapes based on the user selection and forwards them to the Drawing class.

Connections

- Aggregation link from class *Drawing*
- Generalization link to class *Application<fim.utils>*

lesson6::ShapesApp Attributes

Attribute	Type	Notes
drawing	private : *Drawing*	Instance variable of the Drawing class. Initial Value: new Drawing(graph);
currBorderColor	private : *Color*	The border color of shapes. New shapes are drawn with this color of the borders.

		Initial Value: Color.WHITE;

lesson6::ShapesApp Methods

Method	Type	Notes
handleMenu (*int*)	public: *void*	param: zkratka [int - in] zkratka Handles the users commands set through menu.
eraseShape ()	private: *void*	Erases shape by asking the user for an index. It then calls the eraseShape() of the Drawing object.
fillShape ()	private: *void*	Fills a specified shape with a specified color. The shape will be specified by the user by asking him for an index of the shape. Also the color will be ask for by the user.
setBorderColor ()	private: *void*	Sets the border color by asking the user for selecting a color.
getColorFromUser ()	private: *Color*	Method asks the user for a color. The selected color is returned.
main ()	public static: *void*	
setShapeParam (*Shape*)	public: *void*	param: shape [Shape - in] shape Sets the parameters of the shape by asking the user to provide the left, top, width and height values. Also the current border color is assigned to the shape border color attribute. throws = 'FIMReadException' - @exception FIMReadException FIMReadException
start ()	public: *void*	
addShape (*Shape*)	public: *void*	param: s [Shape - in] s Adds the shape to the drawing. This includes setting the parameters such as left, top, width, height and also border color. The methods will also ask for drawing all shapes in the drawing.

lesson6::Triangle

Type: *public* **Class**
Extends: *Shape.*
Status: Proposed. Version 1.0. Phase 1.0.
Package: lesson6
Details: Created on 23.3.2006 22:08:12. Modified on 26.3.2006 21:18:25. Author: Pavel Čech

Class representing the triangles. It is the child of the shape class. It redefines doDraw() and doFill().

Connections

- Generalization link to class *Shape*

lesson6::Triangle Methods

Method	Type	Notes
doDraw ()	protected: *void*	
doFill ()	protected: *void*	