```java
import javax.swing.JOptionPane;

import java.io.IOException;

/** JabberPoint Main Program
 * <p>This program is distributed under the terms of the accompanying
 * COPYRIGHT.txt file (which is NOT the GNU General Public License).
 * Please read it. Your use of the software constitutes acceptance
 * of the terms in the COPYRIGHT.txt file.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class JabberPoint {
    protected static final String IOERR = "IO Error: ";
    protected static final String JABERR = "Jabberpoint Error ";
    protected static final String JABVERSION = "Jabberpoint 1.6 - OU version";

    /** The main program */
    public static void main(String[] argv) {

        Style.createStyles();
        Presentation presentation = new Presentation();
        new SlideViewerFrame(JABVERSION, presentation);
        try {
            if (argv.length == 0) { //a demo presentation
                Accessor.getDemoAccessor().loadFile(presentation, "");
            } else {
                new XMLAccessor().loadFile(presentation, argv[0]);
            }
            presentation.setSlideNumber(0);
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null,
                    IOERR + ex, JABERR,
                    JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

```java
import java.util.Vector;
import java.io.File;
import java.io.IOException;
```

```java
import java.io.PrintWriter;
import java.io.FileWriter;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.NodeList;


/** XMLAccessor, reads and writes XML files
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class XMLAccessor extends Accessor {

    /** Default API to use. */
    protected static final String DEFAULT_API_TO_USE = "dom";

    /** Names of xml tags of attributes */
    protected static final String SHOWTITLE = "showtitle";
    protected static final String SLIDETITLE = "title";
    protected static final String SLIDE = "slide";
    protected static final String ITEM = "item";
    protected static final String LEVEL = "level";
    protected static final String KIND = "kind";
    protected static final String TEXT = "text";
    protected static final String IMAGE = "image";

    /** Text of messages */
    protected static final String PCE = "Parser Configuration Exception";
    protected static final String UNKNOWNTYPE = "Unknown Element type";
    protected static final String NFE = "Number Format Exception";


    private String getTitle(Element element, String tagName) {
        NodeList titles = element.getElementsByTagName(tagName);
        return titles.item(0).getTextContent();
```

```
    }

    public void loadFile(Presentation presentation, String filename) throws
IOException {
        int slideNumber, itemNumber, max = 0, maxItems = 0;
        try {
            DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document document = builder.parse(new File(filename)); //Create a
JDOM document
            Element doc = document.getDocumentElement();
            presentation.setTitle(getTitle(doc, SHOWTITLE));

            NodeList slides = doc.getElementsByTagName(SLIDE);
            max = slides.getLength();
            for (slideNumber = 0; slideNumber < max; slideNumber++) {
                Element xmlSlide = (Element) slides.item(slideNumber);
                Slide slide = new Slide();
                slide.setTitle(getTitle(xmlSlide, SLIDETITLE));
                presentation.append(slide);

                NodeList slideItems = xmlSlide.getElementsByTagName(ITEM);
                maxItems = slideItems.getLength();
                for (itemNumber = 0; itemNumber < maxItems; itemNumber++) {
                    Element item = (Element) slideItems.item(itemNumber);
                    loadSlideItem(slide, item);
                }
            }
        }
        catch (IOException iox) {
            System.err.println(iox.toString());
        }
        catch (SAXException sax) {
            System.err.println(sax.getMessage());
        }
        catch (ParserConfigurationException pcx) {
            System.err.println(PCE);
        }
    }

    protected void loadSlideItem(Slide slide, Element item) {
        int level = 1; // default
        NamedNodeMap attributes = item.getAttributes();
        String leveltext = attributes.getNamedItem(LEVEL).getTextContent();
        if (leveltext != null) {
            try {
                level = Integer.parseInt(leveltext);
            }
```

```
            catch(NumberFormatException x) {
                System.err.println(NFE);
            }
        }
        String type = attributes.getNamedItem(KIND).getTextContent();
        if (TEXT.equals(type)) {
            slide.append(new TextItem(level, item.getTextContent()));
        }
        else {
            if (IMAGE.equals(type)) {
                slide.append(new BitmapItem(level, item.getTextContent()));
            }
            else {
                System.err.println(UNKNOWNTYPE);
            }
        }
    }

    public void saveFile(Presentation presentation, String filename) throws
IOException {
        PrintWriter out = new PrintWriter(new FileWriter(filename));
        out.println("<?xml version=\"1.0\"?>");
        out.println("<!DOCTYPE presentation SYSTEM \"jabberpoint.dtd\">");
        out.println("<presentation>");
        out.print("<showtitle>");
        out.print(presentation.getTitle());
        out.println("</showtitle>");
        for (int slideNumber=0; slideNumber<presentation.getSize();
slideNumber++) {
            Slide slide = presentation.getSlide(slideNumber);
            out.println("<slide>");
            out.println("<title>" + slide.getTitle() + "</title>");
            Vector<SlideItem> slideItems = slide.getSlideItems();
            for (int itemNumber = 0; itemNumber<slideItems.size();
itemNumber++) {
                SlideItem slideItem = (SlideItem)
slideItems.elementAt(itemNumber);
                out.print("<item kind=");
                if (slideItem instanceof TextItem) {
                    out.print("\"text\" level=\"" + slideItem.getLevel() +
"\">");
                    out.print( ( (TextItem) slideItem).getText());
                }
                else {
                    if (slideItem instanceof BitmapItem) {
                        out.print("\"image\" level=\"" + slideItem.getLevel()
+ "\">");
                        out.print( ( (BitmapItem) slideItem).getName());
```

```
                    }
                    else {
                        System.out.println("Ignoring " + slideItem);
                    }
                }
                out.println("</item>");
            }
            out.println("</slide>");
        }
        out.println("</presentation>");
        out.close();
    }
}
```

```
import java.awt.Rectangle;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.font.TextLayout;
import java.awt.font.TextAttribute;
import java.awt.font.LineBreakMeasurer;
import java.awt.font.FontRenderContext;
import java.awt.geom.Rectangle2D;
import java.awt.image.ImageObserver;
import java.text.AttributedString;
import java.util.List;
import java.util.Iterator;
import java.util.ArrayList;

/** <p>A text item.</p>
 * <p>A text item has drawing capabilities.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class TextItem extends SlideItem {
    private String text;
```

```
    private static final String EMPTYTEXT = "No Text Given";

//A textitem of int level with text string
    public TextItem(int level, String string) {
        super(level);
        text = string;
    }

//An empty textitem
    public TextItem() {
        this(0, EMPTYTEXT);
    }

//Returns the text
    public String getText() {
        return text == null ? "" : text;
    }

//Returns the AttributedString for the Item
    public AttributedString getAttributedString(Style style, float scale) {
        AttributedString attrStr = new AttributedString(getText());
        attrStr.addAttribute(TextAttribute.FONT, style.getFont(scale), 0,
text.length());
        return attrStr;
    }

//Returns the bounding box of an Item
    public Rectangle getBoundingBox(Graphics g, ImageObserver observer,
            float scale, Style myStyle) {
        List<TextLayout> layouts = getLayouts(g, myStyle, scale);
        int xsize = 0, ysize = (int) (myStyle.leading * scale);
        Iterator<TextLayout> iterator = layouts.iterator();
        while (iterator.hasNext()) {
            TextLayout layout = iterator.next();
            Rectangle2D bounds = layout.getBounds();
            if (bounds.getWidth() > xsize) {
                xsize = (int) bounds.getWidth();
            }
            if (bounds.getHeight() > 0) {
                ysize += bounds.getHeight();
            }
            ysize += layout.getLeading() + layout.getDescent();
        }
        return new Rectangle((int) (myStyle.indent*scale), 0, xsize, ysize );
    }

//Draws the item
    public void draw(int x, int y, float scale, Graphics g,
```

```
            Style myStyle, ImageObserver o) {
        if (text == null || text.length() == 0) {
            return;
        }
        List<TextLayout> layouts = getLayouts(g, myStyle, scale);
        Point pen = new Point(x + (int)(myStyle.indent * scale),
                y + (int) (myStyle.leading * scale));
        Graphics2D g2d = (Graphics2D)g;
        g2d.setColor(myStyle.color);
        Iterator<TextLayout> it = layouts.iterator();
        while (it.hasNext()) {
            TextLayout layout = it.next();
            pen.y += layout.getAscent();
            layout.draw(g2d, pen.x, pen.y);
            pen.y += layout.getDescent();
        }
      }

    private List<TextLayout> getLayouts(Graphics g, Style s, float scale) {
        List<TextLayout> layouts = new ArrayList<TextLayout>();
        AttributedString attrStr = getAttributedString(s, scale);
        Graphics2D g2d = (Graphics2D) g;
        FontRenderContext frc = g2d.getFontRenderContext();
        LineBreakMeasurer measurer = new
LineBreakMeasurer(attrStr.getIterator(), frc);
        float wrappingWidth = (Slide.WIDTH - s.indent) * scale;
        while (measurer.getPosition() < getText().length()) {
            TextLayout layout = measurer.nextLayout(wrappingWidth);
            layouts.add(layout);
        }
        return layouts;
    }

    public String toString() {
        return "TextItem[" + getLevel()+","+getText()+"]";
    }
}
```

```
import java.awt.Color;
import java.awt.Font;

/** <p>Style stands for Indent, Color, Font and Leading.</p>
 * <p>The link between a style number and a item level is hard-linked:
 * in Slide the style is grabbed for an item
 * with a style number the same as the item level.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
```

```
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class Style {
    private static Style[] styles; // de styles

    private static final String FONTNAME = "Helvetica";
    int indent;
    Color color;
    Font font;
    int fontSize;
    int leading;

    public static void createStyles() {
        styles = new Style[5];
        // De styles zijn vast ingecodeerd.
        styles[0] = new Style(0, Color.red,   48, 20);  // style voor item-
level 0
        styles[1] = new Style(20, Color.blue,  40, 10); // style voor item-
level 1
        styles[2] = new Style(50, Color.black, 36, 10); // style voor item-
level 2
        styles[3] = new Style(70, Color.black, 30, 10); // style voor item-
level 3
        styles[4] = new Style(90, Color.black, 24, 10); // style voor item-
level 4
    }

    public static Style getStyle(int level) {
        if (level >= styles.length) {
            level = styles.length - 1;
        }
        return styles[level];
    }

    public Style(int indent, Color color, int points, int leading) {
        this.indent = indent;
        this.color = color;
        font = new Font(FONTNAME, Font.BOLD, fontSize=points);
        this.leading = leading;
    }

    public String toString() {
```

```
        return "["+ indent + "," + color + "; " + fontSize + " on " + leading
+"]";
    }

    public Font getFont(float scale) {
        return font.deriveFont(fontSize * scale);
    }
}
```

```
import java.awt.Dimension;
import java.awt.event.WindowEvent;
import java.awt.event.WindowAdapter;
import javax.swing.JFrame;

/**
 * <p>The applicatiewindow for a slideviewcomponent</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
*/

public class SlideViewerFrame extends JFrame {
    private static final long serialVersionUID = 3227L;

    private static final String JABTITLE = "Jabberpoint 1.6 - OU";
    public final static int WIDTH = 1200;
    public final static int HEIGHT = 800;

    public SlideViewerFrame(String title, Presentation presentation) {
        super(title);
        SlideViewerComponent slideViewerComponent = new
SlideViewerComponent(presentation, this);
        presentation.setShowView(slideViewerComponent);
        setupWindow(slideViewerComponent, presentation);
    }

//Setup the GUI
    public void setupWindow(SlideViewerComponent
            slideViewerComponent, Presentation presentation) {
        setTitle(JABTITLE);
        addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {
                    System.exit(0);
```

```
                }
            });
        getContentPane().add(slideViewerComponent);
        addKeyListener(new KeyController(presentation)); //Add a controller
        setMenuBar(new MenuController(this, presentation)); //Add another
controller
        setSize(new Dimension(WIDTH, HEIGHT)); //Same sizes a slide has
        setVisible(true);
    }
}
```

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Rectangle;
import javax.swing.JComponent;
import javax.swing.JFrame;


/** <p>SlideViewerComponent is a graphical component that ca display
Slides.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class SlideViewerComponent extends JComponent {

    private Slide slide; //The current slide
    private Font labelFont = null; //The font for labels
    private Presentation presentation = null; //The presentation
    private JFrame frame = null;

    private static final long serialVersionUID = 227L;

    private static final Color BGCOLOR = Color.white;
    private static final Color COLOR = Color.black;
    private static final String FONTNAME = "Dialog";
    private static final int FONTSTYLE = Font.BOLD;
    private static final int FONTHEIGHT = 10;
```

```
    private static final int XPOS = 1100;
    private static final int YPOS = 20;

    public SlideViewerComponent(Presentation pres, JFrame frame) {
        setBackground(BGCOLOR);
        presentation = pres;
        labelFont = new Font(FONTNAME, FONTSTYLE, FONTHEIGHT);
        this.frame = frame;
    }

    public Dimension getPreferredSize() {
        return new Dimension(Slide.WIDTH, Slide.HEIGHT);
    }

    public void update(Presentation presentation, Slide data) {
        if (data == null) {
            repaint();
            return;
        }
        this.presentation = presentation;
        this.slide = data;
        repaint();
        frame.setTitle(presentation.getTitle());
    }

//Draw the slide
    public void paintComponent(Graphics g) {
        g.setColor(BGCOLOR);
        g.fillRect(0, 0, getSize().width, getSize().height);
        if (presentation.getSlideNumber() < 0 || slide == null) {
            return;
        }
        g.setFont(labelFont);
        g.setColor(COLOR);
        g.drawString("Slide " + (1 + presentation.getSlideNumber()) + " of " +
                  presentation.getSize(), XPOS, YPOS);
        Rectangle area = new Rectangle(0, YPOS, getWidth(), (getHeight() -
YPOS));
        slide.draw(g, area, this);
    }
}
```

```
import java.awt.Rectangle;
import java.awt.Graphics;
import java.awt.image.ImageObserver;
```

```
/** <p>The abstract class for items on a slide.<p>
 * <p>All SlideItems have drawing capabilities.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
*/

public abstract class SlideItem {
    private int level = 0; //The level of the SlideItem

    public SlideItem(int lev) {
        level = lev;
    }

    public SlideItem() {
        this(0);
    }

//Returns the level
    public int getLevel() {
        return level;
    }

//Returns the bounding box
    public abstract Rectangle getBoundingBox(Graphics g,
            ImageObserver observer, float scale, Style style);

//Draws the item
    public abstract void draw(int x, int y, float scale,
            Graphics g, Style style, ImageObserver observer);
}
```

```
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.image.ImageObserver;
import java.util.Vector;

/** <p>A slide. This class has drawing functionality.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
```

```
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class Slide {
    public final static int WIDTH = 1200;
    public final static int HEIGHT = 800;
    protected String title; //The title is kept separately
    protected Vector<SlideItem> items; //The SlideItems are kept in a vector

    public Slide() {
        items = new Vector<SlideItem>();
    }

    //Add a SlideItem
    public void append(SlideItem anItem) {
        items.addElement(anItem);
    }

    //Return the title of a slide
    public String getTitle() {
        return title;
    }

    //Change the title of a slide
    public void setTitle(String newTitle) {
        title = newTitle;
    }

    //Create a TextItem out of a String and add the TextItem
    public void append(int level, String message) {
        append(new TextItem(level, message));
    }

    //Returns the SlideItem
    public SlideItem getSlideItem(int number) {
        return (SlideItem)items.elementAt(number);
    }

    //Return all the SlideItems in a vector
    public Vector<SlideItem> getSlideItems() {
        return items;
    }

    //Returns the size of a slide
```

```
    public int getSize() {
        return items.size();
    }

    //Draws the slide
    public void draw(Graphics g, Rectangle area, ImageObserver view) {
        float scale = getScale(area);
        int y = area.y;
    //The title is treated separately
        SlideItem slideItem = new TextItem(0, getTitle());
        Style style = Style.getStyle(slideItem.getLevel());
        slideItem.draw(area.x, y, scale, g, style, view);
        y += slideItem.getBoundingBox(g, view, scale, style).height;
        for (int number=0; number<getSize(); number++) {
          slideItem = (SlideItem)getSlideItems().elementAt(number);
          style = Style.getStyle(slideItem.getLevel());
          slideItem.draw(area.x, y, scale, g, style, view);
          y += slideItem.getBoundingBox(g, view, scale, style).height;
        }
      }

    //Returns the scale to draw a slide
    private float getScale(Rectangle area) {
        return Math.min(((float)area.width) / ((float)WIDTH),
((float)area.height) / ((float)HEIGHT));
    }
}
```

```
import java.util.ArrayList;


/**
 * <p>Presentations keeps track of the slides in a presentation.</p>
 * <p>Only one instance of this class is available.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class Presentation {
    private String showTitle; //The title of the presentation
```

```
    private ArrayList<Slide> showList = null; //An ArrayList with slides
    private int currentSlideNumber = 0; //The number of the current slide
    private SlideViewerComponent slideViewComponent = null; //The view
component of the slides

    public Presentation() {
        slideViewComponent = null;
        clear();
    }

    public Presentation(SlideViewerComponent slideViewerComponent) {
        this.slideViewComponent = slideViewerComponent;
        clear();
    }

    public int getSize() {
        return showList.size();
    }

    public String getTitle() {
        return showTitle;
    }

    public void setTitle(String nt) {
        showTitle = nt;
    }

    public void setShowView(SlideViewerComponent slideViewerComponent) {
        this.slideViewComponent = slideViewerComponent;
    }

    //Returns the number of the current slide
    public int getSlideNumber() {
        return currentSlideNumber;
    }

    //Change the current slide number and report it the the window
    public void setSlideNumber(int number) {
        currentSlideNumber = number;
        if (slideViewComponent != null) {
            slideViewComponent.update(this, getCurrentSlide());
        }
    }

    //Navigate to the previous slide unless we are at the first slide
    public void prevSlide() {
        if (currentSlideNumber > 0) {
            setSlideNumber(currentSlideNumber - 1);
```

```
        }
    }

    //Navigate to the next slide unless we are at the last slide
    public void nextSlide() {
        if (currentSlideNumber < (showList.size()-1)) {
            setSlideNumber(currentSlideNumber + 1);
        }
    }

    //Remove the presentation
    void clear() {
        showList = new ArrayList<Slide>();
        setSlideNumber(-1);
    }

    //Add a slide to the presentation
    public void append(Slide slide) {
        showList.add(slide);
    }

    //Return a slide with a specific number
    public Slide getSlide(int number) {
        if (number < 0 || number >= getSize()){
            return null;
        }
            return (Slide)showList.get(number);
    }

    //Return the current slide
    public Slide getCurrentSlide() {
        return getSlide(currentSlideNumber);
    }

    public void exit(int n) {
        System.exit(n);
    }
}
```

```
import java.awt.MenuBar;
import java.awt.Frame;
import java.awt.Menu;
import java.awt.MenuItem;
import java.awt.MenuShortcut;
import java.awt.event.ActionListener;
```

```java
import java.awt.event.ActionEvent;
import java.io.IOException;

import javax.swing.JOptionPane;

/** <p>The controller for the menu</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */
public class MenuController extends MenuBar {

    private Frame parent; //The frame, only used as parent for the Dialogs
    private Presentation presentation; //Commands are given to the
presentation

    private static final long serialVersionUID = 227L;

    protected static final String ABOUT = "About";
    protected static final String FILE = "File";
    protected static final String EXIT = "Exit";
    protected static final String GOTO = "Go to";
    protected static final String HELP = "Help";
    protected static final String NEW = "New";
    protected static final String NEXT = "Next";
    protected static final String OPEN = "Open";
    protected static final String PAGENR = "Page number?";
    protected static final String PREV = "Prev";
    protected static final String SAVE = "Save";
    protected static final String VIEW = "View";

    protected static final String TESTFILE = "testPresentation.xml";
    protected static final String SAVEFILE = "savedPresentation.xml";

    protected static final String IOEX = "IO Exception: ";
    protected static final String LOADERR = "Load Error";
    protected static final String SAVEERR = "Save Error";

    public MenuController(Frame frame, Presentation pres) {
        parent = frame;
        presentation = pres;
        MenuItem menuItem;
        Menu fileMenu = new Menu(FILE);
        fileMenu.add(menuItem = mkMenuItem(OPEN));
```

```
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                presentation.clear();
                Accessor xmlAccessor = new XMLAccessor();
                try {
                    xmlAccessor.loadFile(presentation, TESTFILE);
                    presentation.setSlideNumber(0);
                } catch (IOException exc) {
                    JOptionPane.showMessageDialog(parent, IOEX + exc,
                    LOADERR, JOptionPane.ERROR_MESSAGE);
                }
                parent.repaint();
            }
        } );
        fileMenu.add(menuItem = mkMenuItem(NEW));
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                presentation.clear();
                parent.repaint();
            }
        });
        fileMenu.add(menuItem = mkMenuItem(SAVE));
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Accessor xmlAccessor = new XMLAccessor();
                try {
                    xmlAccessor.saveFile(presentation, SAVEFILE);
                } catch (IOException exc) {
                    JOptionPane.showMessageDialog(parent, IOEX + exc,
                            SAVEERR, JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        fileMenu.addSeparator();
        fileMenu.add(menuItem = mkMenuItem(EXIT));
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                presentation.exit(0);
            }
        });
        add(fileMenu);
        Menu viewMenu = new Menu(VIEW);
        viewMenu.add(menuItem = mkMenuItem(NEXT));
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                presentation.nextSlide();
            }
        });
```

```
        viewMenu.add(menuItem = mkMenuItem(PREV));
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                presentation.prevSlide();
            }
        });
        viewMenu.add(menuItem = mkMenuItem(GOTO));
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                String pageNumberStr =
JOptionPane.showInputDialog((Object)PAGENR);
                int pageNumber = Integer.parseInt(pageNumberStr);
                presentation.setSlideNumber(pageNumber - 1);
            }
        });
        add(viewMenu);
        Menu helpMenu = new Menu(HELP);
        helpMenu.add(menuItem = mkMenuItem(ABOUT));
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                AboutBox.show(parent);
            }
        });
        setHelpMenu(helpMenu);      //Needed for portability (Motif, etc.).
    }

//Creating a menu-item
    public MenuItem mkMenuItem(String name) {
        return new MenuItem(name, new MenuShortcut(name.charAt(0)));
    }
}
```

```
import java.awt.event.KeyEvent;
import java.awt.event.KeyAdapter;

/** <p>This is the KeyController (KeyListener)</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
*/
```

```
public class KeyController extends KeyAdapter {
    private Presentation presentation; //Commands are given to the
presentation

    public KeyController(Presentation p) {
        presentation = p;
    }

    public void keyPressed(KeyEvent keyEvent) {
        switch(keyEvent.getKeyCode()) {
            case KeyEvent.VK_PAGE_DOWN:
            case KeyEvent.VK_DOWN:
            case KeyEvent.VK_ENTER:
            case '+':
                presentation.nextSlide();
                break;
            case KeyEvent.VK_PAGE_UP:
            case KeyEvent.VK_UP:
            case '-':
                presentation.prevSlide();
                break;
            case 'q':
            case 'Q':
                System.exit(0);
                break; //Should not be reached
            default:
                break;
        }
    }
}
```

```
/** A built-in demo presentation
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

class DemoPresentation extends Accessor {

    public void loadFile(Presentation presentation, String unusedFilename) {
        presentation.setTitle("Demo Presentation");
```

```
        Slide slide;
        slide = new Slide();
        slide.setTitle("JabberPoint");
        slide.append(1, "The Java prestentation tool");
        slide.append(2, "Copyright (c) 1996-2000: Ian Darwin");
        slide.append(2, "Copyright (c) 2000-now:");
        slide.append(2, "Gert Florijn and Sylvia Stuurman");
        slide.append(4, "Calling Jabberpoint without a filename");
        slide.append(4, "will show this presentation");
        slide.append(1, "Navigate:");
        slide.append(3, "Next slide: PgDn or Enter");
        slide.append(3, "Previous slide: PgUp or up-arrow");
        slide.append(3, "Quit: q or Q");
        presentation.append(slide);

        slide = new Slide();
        slide.setTitle("Demonstration of levels and styles");
        slide.append(1, "Level 1");
        slide.append(2, "Level 2");
        slide.append(1, "Again level 1");
        slide.append(1, "Level 1 has style number 1");
        slide.append(2, "Level 2 has style number 2");
        slide.append(3, "This is how level 3 looks like");
        slide.append(4, "And this is level 4");
        presentation.append(slide);

        slide = new Slide();
        slide.setTitle("The third slide");
        slide.append(1, "To open a new presentation,");
        slide.append(2, "use File->Open from the menu.");
        slide.append(1, " ");
        slide.append(1, "This is the end of the presentation.");
        slide.append(new BitmapItem(1, "JabberPoint.jpg"));
        presentation.append(slide);
    }

    public void saveFile(Presentation presentation, String unusedFilename) {
        throw new IllegalStateException("Save As->Demo! called");
    }
}
```

```
import java.awt.Rectangle;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.awt.image.ImageObserver;
```

```java
import java.io.File;

import javax.imageio.ImageIO;

import java.io.IOException;


/** <p>The class for a Bitmap item</p>
 * <p>Bitmap items are responsible for drawing themselves.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
*/

public class BitmapItem extends SlideItem {
  private BufferedImage bufferedImage;
  private String imageName;

  protected static final String FILE = "File ";
  protected static final String NOTFOUND = " not found";


    //level indicates the item-level; name indicates the name of the file with
the image
    public BitmapItem(int level, String name) {
        super(level);
        imageName = name;
        try {
            bufferedImage = ImageIO.read(new File(imageName));
        }
        catch (IOException e) {
            System.err.println(FILE + imageName + NOTFOUND) ;
        }
    }

    //An empty bitmap item
    public BitmapItem() {
        this(0, null);
    }

    //Returns the filename of the image
    public String getName() {
        return imageName;
    }
```

```
    //Returns the bounding box of the image
    public Rectangle getBoundingBox(Graphics g, ImageObserver observer, float
scale, Style myStyle) {
        return new Rectangle((int) (myStyle.indent * scale), 0,
                (int) (bufferedImage.getWidth(observer) * scale),
                ((int) (myStyle.leading * scale)) +
                (int) (bufferedImage.getHeight(observer) * scale));
    }

    //Draws the image
    public void draw(int x, int y, float scale, Graphics g, Style myStyle,
ImageObserver observer) {
        int width = x + (int) (myStyle.indent * scale);
        int height = y + (int) (myStyle.leading * scale);
        g.drawImage(bufferedImage, width, height,(int)
(bufferedImage.getWidth(observer)*scale),
                (int) (bufferedImage.getHeight(observer)*scale), observer);
    }

    public String toString() {
        return "BitmapItem[" + getLevel() + "," + imageName + "]";
    }
}
```

```
import java.io.IOException;

/**
 * <p>An Accessor makes it possible to read and write data
 * for a presentation.</p>
 * <p>Non-abstract subclasses should implement the load and save methods.</p>
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public abstract class Accessor {
    public static final String DEMO_NAME = "Demo presentation";
    public static final String DEFAULT_EXTENSION = ".xml";

    public static Accessor getDemoAccessor() {
        return new DemoPresentation();
```

```
    }

    public Accessor() {
    }

    abstract public void loadFile(Presentation p, String fn) throws
IOException;

    abstract public void saveFile(Presentation p, String fn) throws
IOException;

}
```

```
import java.awt.Frame;
import javax.swing.JOptionPane;

/**The About-box for JabberPoint.
 * @author Ian F. Darwin, ian@darwinsys.com, Gert Florijn, Sylvia Stuurman
 * @version 1.1 2002/12/17 Gert Florijn
 * @version 1.2 2003/11/19 Sylvia Stuurman
 * @version 1.3 2004/08/17 Sylvia Stuurman
 * @version 1.4 2007/07/16 Sylvia Stuurman
 * @version 1.5 2010/03/03 Sylvia Stuurman
 * @version 1.6 2014/05/16 Sylvia Stuurman
 */

public class AboutBox {
    public static void show(Frame parent) {
        JOptionPane.showMessageDialog(parent,
                "JabberPoint is a primitive slide-show program in Java(tm).
It\n" +
                "is freely copyable as long as you keep this notice and\n" +
                "the splash screen intact.\n" +
                "Copyright (c) 1995-1997 by Ian F. Darwin,
ian@darwinsys.com.\n" +
                "Adapted by Gert Florijn (version 1.1) and " +
                "Sylvia Stuurman (version 1.2 and higher) for the Open" +
                "University of the Netherlands, 2002 -- now.\n" +
                "Author's version available from http://www.darwinsys.com/",
                "About JabberPoint",
                JOptionPane.INFORMATION_MESSAGE
        );
    }
}
```