JAVA NOTES

GRAPHICAL USER INTERFACES

Terry Marris July 2001

APPENDIX F PRINTING

We see how to print text on the printer device.

To make a print out we need to

- define a page that implements the *Printable* interface
- create a book and add the page to it
- start a print job

The *Printable* interface has just one method:

```
int print(Graphics g, PageFormat pf, int page)
```

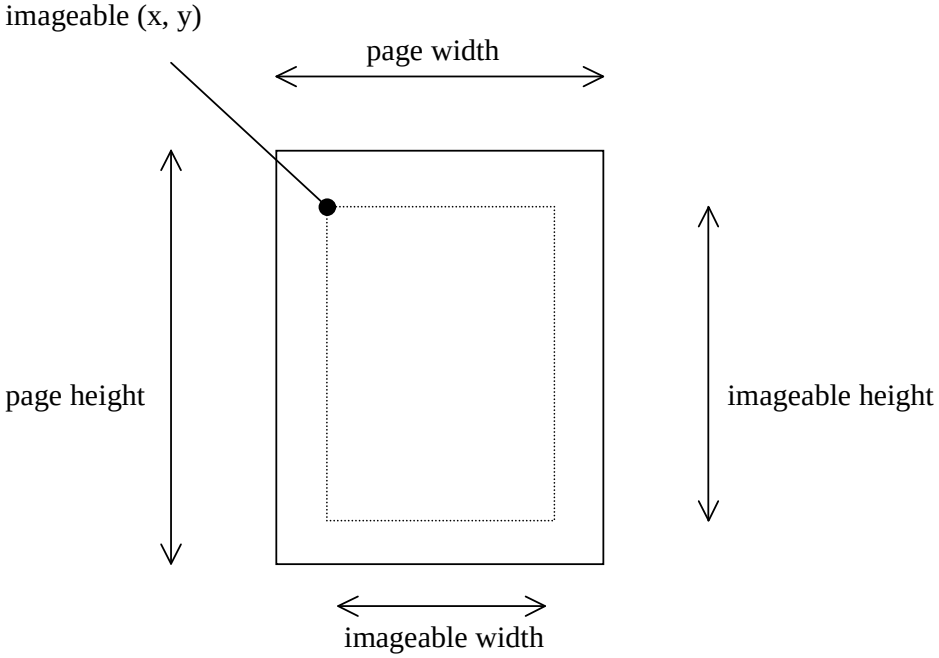
This method is called whenever the print engine needs to format a page for printing. We implement this method by writing the code to print a page.

```
class Page implements Printable {
...
  public int print(Graphics g, PageFormat pf, int page) throws
                               PrinterException
  {
    if (page >= getPageCount())
      return Printable.NO_SUCH_PAGE;

    Graphics2D g2 = (Graphics2D)g;
    g2.translate(pf.getImageableX(), pf.getImageableY());
    g2.setFont(new Font("Monospaced", Font.PLAIN, 14));
    g2.setPaint(Color.black);
    g2.drawString(pageHeading + " Page #: " + (page+1),10,10);

    return Printable.PAGE_EXISTS;
  }
}
```

A print job will keep on calling *print(Graphic, PageFormat, int)* for as long as *PAGE_EXISTS* is returned. We programmers need to calculate in advance how many pages there are to be printed, perhaps by knowing how many lines there are to be printed, and then dividing this by the number of lines we fit on a page. Printing stops when *NO_SUCH_PAGE* is returned.

We use a two-dimensional graphics object for printing.

The *PageFormat* parameter contains information about the printed page. The methods *getWidth()* and *getHeight()* returns the paper size in points. There are 72 points to an inch, and A4 paper size is approximately 595 by 842 points. Printers cannot print on a small area around the edges of the paper. The methods *getImageableWidth()* and *getImageableHeight()* tell you the dimensions of the paper you can actually use. The top left-hand corner of this area is identified by *getImageableX()* and *getImageableY()*.



The *PrinterJob* class starts a print job. First, we call the static *getPrinterJob()* method. Then we call the *defaultPage()* method to obtain a default page format and then use the *setPrintable(Printable, PageFormat)* method.

```
PrinterJob printJob = PrinterJob.getPrinterJob();
printJob.setPageable(makeBook());
try {
  printJob.print();
}
catch (PrinterException exception) {
  JOptionPane.showMessageDialog(this, exception);
}
```

We create a new book and add the required number of pages to it.

```
public Book makeBook()
{
  PrinterJob printJob = PrinterJob.getPrinterJob();
  pageFormat = printJob.defaultPage();

  Book book = new Book();
  Page page = new Page();
  int pageCount = page.getPageCount();
  book.append(page, pageFormat, pageCount);
  return book;
}
```

The program shown below prints two pages, numbered one and two, each with the heading *Report*.

```
/* PrintBookFrame.java
   Terry Marris  16 July 2001
*/

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


class Page implements Printable {
  private String pageHeading;
  private int pageCount;

  public Page()
  {
    pageHeading = "Report";
    pageCount = 2;
  }


  public int getPageCount()
  {
    return pageCount;
  }


  public int print(Graphics g, PageFormat pf, int page) throws
                                PrinterException
  {
    if (page >= getPageCount())
      return Printable.NO_SUCH_PAGE;

    Graphics2D g2 = (Graphics2D)g;
    g2.translate(pf.getImageableX(), pf.getImageableY());
    g2.setFont(new Font("Monospaced", Font.PLAIN, 14));
    g2.setPaint(Color.black);
    g2.drawString(pageHeading + " Page #: " + (page+1),10,10);

    return Printable.PAGE_EXISTS;
  }
}
```

```
public class PrintBookFrame extends JFrame implements
ActionListener {
  private JButton printButton;
  private PageFormat pageFormat;

  public PrintBookFrame()
  {
    setTitle("Print Book Frame");

    Toolkit tk = Toolkit.getDefaultToolkit();
    Dimension screen = tk.getScreenSize();
    setLocation(screen.width/4, screen.height/4);
    setSize(screen.width/2, screen.height/2);

    addWindowListener(new WindowAdapter()
      {
        public void windowClosing(WindowEvent e)
        {
          System.exit(0);
        }
     });

    JPanel buttonPanel = new JPanel();
    printButton = new JButton("Print");
    buttonPanel.add(printButton);
    printButton.addActionListener(this);
    getContentPane().add(buttonPanel);
  }


  public Book makeBook()
  {
    PrinterJob printJob = PrinterJob.getPrinterJob();
    pageFormat = printJob.defaultPage();

    Book book = new Book();
    Page page = new Page();
    int pageCount = page.getPageCount();
    book.append(page, pageFormat, pageCount);
    return book;
  }
```

```
  public void actionPerformed(ActionEvent event)
  {
    Object source = event.getSource();

    if (source == printButton) {
      PrinterJob printJob = PrinterJob.getPrinterJob();
      printJob.setPageable(makeBook());
      try {
        printJob.print();
      }
      catch (PrinterException exception) {
        JOptionPane.showMessageDialog(this, exception);
      }
    }
  }


  public static void main(String[] s)
  {
    new PrintBookFrame().show();
  }
}
```

java.awt.Printable

Methods		
int	print(Graphics g, PageFormat pf, int pageNumber)	renders (i.e. formats and prints) a page and returns PAGE_EXISTS or NO_SUCH_PAGE.

java.awt.print.PrinterJob

Methods		
static PrinterJob	getPrinterJob()	returns a PrinterJob object.
PageFormat	defaultPage()	returns the default page format for this printer.
void	setPageable(Pageable p)	sets a Pageable (such as a Book) to be printed.
void	setPrintable(Printable p)	sets the printable for this print job.
void	setPrintable(Printable p, PageFormat pf)	sets the printable for this print job with the given page format.
void	print()	prints the current Printable by repeatedly calling its print() method and sending the rendered pages to the printer until no more pages are available.

java.awt.print.PageFormat

Methods		
double double	getWidth() getHeight()	returns the width and height of the page.
double double	getImageableWidth() getImageableHeight()	returns the width and height of the imageable area of the page.
double double	getImageableX() getImageableY()	returns the position of the top left corner of the imageable area.

java.awt.print.Book

Methods		
void	append(Printable p, PageFormat pf)	appends the first page to this book.
void	append(Printable p, PageFormat pf, int pageCount)	appends a section to this book that contains the given number, pageCount, of pages.
Printable	getPrintable(int page)	returns the printable for the given page.

Further reading:

HORSTMANN & CORNELL Core Java 2 Volume 1 pp 278
HORSTMANN & CORNELL Core Java 2 Volume 2 pp 542