Print Manager 1.0 Component Specification

1. Design

The Print Manager component provides the ability to print a java.awt.Component. It provides a framework for the print process: choosing the page formats, splitting the image in pieces and print.

1.1 Design Patterns

Strategy: The user can plug-in different mechanisms for the ComponentVisibleMaker and PrintFormatRetriever. Currently there are only one concrete default implementation of each interface, but others are possible.

1.2 Industry Standards

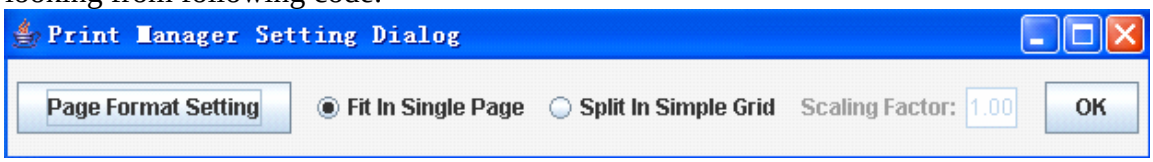
N/A

1.3 Required Algorithms

This section describes four algorithm used in this component.

1.3.1 The Print Setting UI Dialog (DefaultPrintFormatRetriever class)

This is the pseudocode for the DefaultPrintFormatRetriever.initDialog() method. The developer is encouraged to optimize this pseudocode. And this is the UI looking from following code:




```
this.setLayout(new FlowLayout());

final PrinterJob job = PrinterJob.getPrinterJob();
                                    pageFormat = job.defaultPage();
// Page Format Setting Button
JButton pageFormatButton = new JButton("Page Format Setting");
pageFormatButton.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent event) {
            pageFormat = job.pageDialog(pageFormat);
      }
});
this.add(pageFormatButton);

// Scaling Factor text field
final JTextField textField = new JTextField();
textField.setText("1.00");
final JLabel label = new JLabel("Scaling Factor:");
final JPanel factorPanel = new JPanel();
factorPanel.setLayout(new FlowLayout());
```

```
factorPanel.add(label);
factorPanel.add(textField);

// Print Choice (Fit In Single Page or Split In Simple Grid) Setting
JRadioButton fitInPageRadio = new JRadioButton("Fit In Single Page");
fitInPageRadio.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent event) {
		fitInPage = true;
		textField.setEnabled(false);
		label.setEnabled(false);
	}
});
fitInPageRadio.setSelected(true);
textField.setEnabled(false);
label.setEnabled(false);

JRadioButton splitInGridRadio = new JRadioButton("Split In Simple Grid");
splitInGridRadio.addActionListener(new ActionListener(){
	public void actionPerformed(ActionEvent event) {
		fitInPage = false;
		textField.setEnabled(true);
		label.setEnabled(true);
	}
});

ButtonGroup group = new ButtonGroup();
group.add(fitInPageRadio);
group.add(splitInGridRadio);

JPanel printChoicePanel = new JPanel();
printChoicePanel.setLayout(new FlowLayout());
printChoicePanel.add(fitInPageRadio);
printChoicePanel.add(splitInGridRadio);
printChoicePanel.add(factorPanel);
this.add(printChoicePanel);

JButton okButton = new JButton("OK");
okButton.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent event) {
		if (!fitInPage) {
			scaleFactor = Double.parseDouble(textField.getText());
		}
		setVisible(false);
	}
});
this.add(okButton);
```

this.pack();

1.3.2 *The print method in FitInPagePrint class*

This is the pseudocode for the FitInPagePrint.print() method. The developer is encouraged to optimize this pseudocode.

1. if given pageIndex is greater than 0, return NO_SUCH_PAGE;
2. set the graphics relative to pageFormat.getImagableX(), pageFormat.getImagableY() point:

```
Graphics2D g2d = (Graphics2D) graphics;
```

3. calculate the print area

```
if (rectangle != null) {
   areaWidth = rectangle.getWidth();
   areaHeight = rectangle.getHeight();
   areaX = rectangle.getX();
   areaY = rectangle.getY();
} else {
   //the printed area is the component it self
   areaWidth = component.getWidth();
   areaHeight = component.getHeight();
   areaX = 0;
   areaY = 0;
}
```

4. Get the max width and height allow to print:

```
double pageWidth = pageFormat.getImageableWidth();
double pageHeight = pageFormat.getImageableHeight();
//left top coordinate of the page's printable area
double pageX = pageFormat.getImageableX();
double pageY = pageFormat.getImageableY();
```

5. calculate the proper scale factor:

```
double scale = Math.min(pageWidth / areaWidth, pageHeight / areaHeight);
```

6. adjust the origin

```
//scale the graphics
g2d.scale(scale, scale);
g2d.translate(((pageX / scale) - areaX), ((pageY / scale) - areaY));
//adjust origin
g2d.setClip((int) areaX, (int) areaY, (int) areaWidth, (int) areaHeight);
```

7. print the component:

```
component.print(graphics);
```

1.3.3 *The print method in ScalingPrint class*

This is the pseudocode for the ScalingPrint.print() method. The developer is encouraged to optimize this pseudocode.

1. if 'grids' array is null, then initialize 'grids' array:
   a) Calculate the print area as above
   b) calculate the grids rows and colums split

```
int numberRow = (int) Math.ceil(areaHeight / pageHeight);
int numberColumn = (int) Math.ceil(areaWidth / pageWidth);
```

c) split the print area into grids

```
grids = new Rectangle[numberRow * numberColumn];

double x, y, w, h; //x,y, width, height of the split grid

for (int row = 0; row < numberRow; row++) {
    y = areaY + (row * pageHeight);

    //ensure the height of the marginal grid will not exceed the
printed area
    h = Math.min(pageHeight, maxAreaY - y);

    for (int col = 0; col < numberColumn; col++) {
        x = areaX + (col * pageWidth);

        //ensure the width of the marginal grid will not exceed
the printed area
        w = Math.min(pageWidth, maxAreaX - x);
        grids[(row * numberColumn) + col] = new Rectangle((int)
x, (int) y, (int) w, (int) h);
    }
}
```

2. if given pageIndex is greater than grids.length, return NO_SUCH_PAGE;
3. get current grid to print

```
Rectangle grid = grids[pageIndex];
```

4. Get the left top coordinate of the page's printable area:

```
double pageX = pageFormat.getImageableX();
double pageY = pageFormat.getImageableY();
```

5. Get the left top coordinate of the grid:

```
double gridX = grid.getX();
double gridY = grid.getY();
```

6. adjust the origin, scale, print area

```
g2d.scale(scaleFactor, scaleFactor); //scale the graphics
g2d.translate(((pageX / scaleFactor) - gridX), ((pageY /
scaleFactor) - gridY)); //adjust origin
g2d.setClip(grid); //set current grid as printed area
```

6. print the component:

```
component.print(graphics);
```

1.3.4 *The print method in PrintManager class*

Main algorithm for print component with rectangle:

1. if double buffered on, close double buffered
2. if component is not visible, use componentVisibleMaker to make it visible
3. create FitInPagePrint or ScalingPrint printable according to printFormatRetriever
4. get PrinterJob, and print the printable
5. restore the component's previous visibility
6. restore the component's previous double buffered

1.4 Component Class Overview

PrintManager

This is the main class of Print Manager component. It has four main methods which accept java.awt.Component instance(s) and java.awt.Rectangle instance(s) (optional). Before calling these request methods, user need call specifyPrintFormat() method to retrieve user's preference for Page Format, Print Choice (fitting given component to single page or scaling given component and splitting it using simple grid) and Print given component within given Rectangle range if it is provided.

AbstractPrint

This is abstract class which implements java.awt.print.Printable interface. This class aim to provide common elements for the requirements of this component. It has two subclasses: FitInPagePrint and ScalingPrint. The aim of this class is to maintain java.awt.Component and java.awt.Rectangle instance.

FitInPagePrint

This class extends from AbstractPrint class. This class aims to print given component into a single page, which means if the area of given component to print is too large for one page, we need to scaling it to fit in just a single page.

ScalePrint

This class extends from AbstractPrint class. This class aims to split given component into several simple grids and print each grid into a page. And we will use scaleFactor to scale given component before print.

ComponentVisibleMaker

This interface aims to make invisible component to visible first and let it to be printed. And after printing, it will make the component back to original invisible state.

DefaultComponentVisibleMaker

This class implements the ComponentVisibleMaker interface. This is the default implementation used by PrintManager and DefaultPageFormatRetriever class. It will store original state information of component in makeVisible() method and in the makeInvisible() method it will use the information stored to put the component back to its original state.

PrintFormatRetriever

This is the interface used by PrintManager to retrieve user's preference of print page format and print scaling type. The retrievePrintFormat() method will do the main job to retrieve user's preference of print page format and print scaling type. Implementation should implement it by its own manner to retrieve these information. For instance, it can prompt user a dialog to retrieve these information or it can read these information from configuration file.

DefaultPrintFormatRetriever

This is class extends from JFrame and implements PrintFormatRetriever interface. And it is used for prompting user with a dialog where user can set Page Format, Print Choice (fitting given component to single page or scaling given component and splitting it using simple grid).

1.5 Component Exception Definitions

ConfigurationException

The ConfigurationException exception is used to wrap any specific exception or indicate any errors when creating ComponentVisibleMaker instance. These exceptions are thrown by the PrintManager class.

PrintingException

The PrintingException exception is used to wrap any specific exception or indicate any errors when printing given java.awt.Component. These exceptions are thrown by the DefaultPageFormatRetriever class.
PrintFormatRetrieveException

PrintFormatRetrieveException

The PrintFormatRetrieveException exception is used to wrap any specific exception or indicate any errors when retrieving user's preference of Print Format or Scaling type. These exceptions are thrown by the PrintFormatRetriever implementations and PrintManager which uses PrintFormatRetriever implementations.

1.6 Thread Safety

This component is not thread-safe, and there is no requirement for it to be thread-safe. Thread safety will be provided by the application using these implementations.

2. Environment Requirements

2.1 Environment

JDK 1.5

2.2 TopCoder Software Components

- **Configuration Manager v2.1.5** used by PrintManager class to receive the class name of ComponentVisibleMaker implementation
- **BaseException v1.0** used by the exceptions to provide a common exception base for future changes
- **Object Factory V2.0.1** used by PrintManager to create ComponentVisibleMaker and printFormatRetriever instance.

2.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Names

com.topcoder.swing.print
com.topcoder.swing.print.impl

3.2 Configuration Parameters

Parameter	Description	Values
objectFactoryNamespace	Used to create the ObjectFactory object, required.	Must be non-empty string
componentVisibleMaker	Represent the key used to create the ComponentVisibleMaker object from ObjectFactory, required.	Must be non-empty string
printFormatRetriever	Represents the key used to create the	Must be non-empty

	PrintFormatRetriever object from ObjectFactory, required	string
hasPrintDialog	Represents whether to prompt user with the print dialog when calling print method, optional	true or false (case insensitive)

3.3 Dependencies Configuration

The configuration for ObjectFactory please refer to Object Factory component.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

None

4.3 Demo

The demo will demonstrate the usage of this component. Supposing component and rectangle and componets and rectangles is initialized.

```
        //create a new PrintManager
       PrintManager manager = new PrintManager();

       //create a custom PrintManager(see Demo.xml)
       //custom PrintFormatRetriever and ComponentVisibleMaker can be
used
       PrintManager customPrintManager = new PrintManager("demo");

       //get the user' preference for print format and scaling type
       manager.specifyPrintFormat();

       //print the whole component
       manager.print(components.get(0));

       //print the whole component within rectangle area
       manager.print(components.get(0), rectangles.get(0));

       //print the whole component list
       manager.print(components);

       //print the whole component list within rectangle area list
corresponding
       manager.print(components, rectangles);

       //print a component without print dialog
       manager.setHasPrintDialog(false);
       manager.print(components.get(0));
```

5. Future Enhancements

Provide more ComponentVisibleMaker or PrintFormatRetriever implementations

and provide the preview functionality.