Print Manager 1.0 Requirements Specification

1. Scope

1.1 Overview

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 previewing.

1.2 Logic Requirements

1.2.1 PrintManager

This manager provides methods to print a java.awt.Component or an array of Components. One method receives the Component and one receives the Component and the rectangle area that should be printed.

1.2.2 Page format and pages

The component will create Printable instance(s) using the Component provided. It will provide a pluggable way to determine the pages and the page format for each page. These could be obtained from the user using a dialog.

A simple implementation using the print and page dialogs from PrinterJob should be provided. The idea is that the user should be able to print the component on the same page (fitting the component to the page), or splitting it using a simple grid, while the first piece conforms to the scaling chosen by the user.

There should be a way to add a custom viewer for the user to preview and determine the exact pages and page formats.

1.2.3 Double buffering

The component will turn the global double buffering off before printing, and on after printing. Any other optimizations are welcome.

1.2.4 Non visible Component

The Component passed to the methods might not be visible, in which case they will not render anything. The component will provide a pluggable way to make the component visible while printing. The default way will be a dialog showing that printing is in progress. The component will be added to the dialog with a size of (0,0) - it only needs to be visible, not to be fully displayed.

To determine if a Component is visible, the Graphics object is checked whether it is null. Note that the Component might belong to a parent, in which case, it should be added back to that container in the same position after printing (this is not an entirely safe strategy, but it will do).

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used in the TopCoder UML Tool to print the diagrams.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

The design must follow the interface found in the class diagram with the component interfaces. The designer is encouraged to add to the existing interface, but not to remove anything.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5

2.1.4 Package Structure

com.topcoder.swing.print

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None

3.2.2 TopCoder Software Component Dependencies:

None

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram

- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of Poseidon.