Overview

Ultimately, a user of The Big Pixel will be able to draw filled rectangles (called “big pixels”). At this stage in the development process our goal is to create a GUI component that can draw a simple grid.

When rendered with both the grid and headings, the GUI component (constructed using the default constructor) must appear as follows.

![Grid Diagram](image)
Engineering Design

The relationships between the various classes is illustrated in the following UML diagrams.
In addition to the specifications in this UML diagram, these classes/interfaces must comply with the following specifications.

**The GridComponent Class**

A GridComponent is a JComponent that renders a grid with column and row "indexes" that are both Integer objects. It may or may not include column/row headers (containing the "indexes") and margins.
The default constructor must initialize the columns, rows, and margin to 10, 10, and 20, respectively.

The `getMinimumSize()` method must assume that cells are 10 pixels by 10 pixels. The `getPreferredSize()` method must assume that cells are 30 pixels by 30 pixels.

The `paintGrid()` method must render the grid itself, and the `paintHeader()` method must render the column and row headers in the top and left margins. Each column label must be centered above the relevant column, just above the top grid line. Each row label must be centered just to the left of the relevant row and must be aligned flush right. Note that both of these methods will need to use the `toPixels()` method in a `GridConverter` object but they should never pass it invalid parameters. Hence, they should never need to handle an `InvalidCoordinateException`. **These methods must use an assertion to ensure that this is not the case during testing.**

The `setGridVisibility()` method must call `repaint()` after it changes the object’s state.