```java
/*
 * TCSS 305 - Autumn 2016
 * Assignment 6a - Tetris
 */

package view;

import java.awt.Color;

/**
 * The game panel were the blocks will be shown on.
 *
 * @author Leah Ruisenor
 * @version December 2016.
 */
public class TetrisGamePanel extends JPanel implements Observer {

    /** Serialization. */
    private static final long serialVersionUID = 1159971900757430935L;

    /** Constant for calculating shifts. */
    private static final int FIVE = 5;

    /** Constant for calculating shifts. */
    private static final int THIRTY = 30;

    /** Constant for calculating shifts. */
    private static final int FOURTY = 40;

    /** A list of blocks for drawing the tetris pieces. */
    private final List<Block[]> myBlockList;

    /**
     * Constructs the tetris board.
     */
    public TetrisGamePanel() {
        super();
        myBlockList = new ArrayList<Block[]>();
        setBackground(Color.YELLOW.darker().darker());
    }

    /**
     * Draws tetris pieces on the game panel.
     *
     * {@inheritDoc}
     */
    @Override
    protected void paintComponent(final Graphics theGraphics) {
        super.paintComponent(theGraphics);
        final Graphics2D g2d = (Graphics2D) theGraphics;
        g2d.setPaint(Color.YELLOW.darker().darker().darker());
```

```
65          //System.out.print(getPreferredSize());
66
67          // blockScale will be 20, 30, or 40
68          final int blockScale = getWidth() / 10;
69          int tinyBlockScale = FIVE; // 5
70          if (blockScale == THIRTY) {
71              tinyBlockScale = FIVE + 2; // 7
72          } else if (blockScale == FOURTY) {
73              tinyBlockScale = FIVE * 2 - 1; // 9
74          }
75
76          for (int i = 0; i < myBlockList.size(); i++) {
77
78              final Block[] blocks = myBlockList.get(i);
79
80              for (int j = 0; j < blocks.length; j++) {
81                  final Point pt = new Point(j * blockScale, i * blockScale);
82
83                  g2d.setPaint(Color.YELLOW.darker().darker().darker());
84                  if (blocks[j] != null) {
85                      g2d.fill(new Rectangle2D.Double(pt.x(),
86                                                      getHeight() - pt.y() -
  blockScale,
87                                                      blockScale, blockScale));
88                      g2d.setPaint(Color.BLACK);
89                      g2d.drawRect(pt.x(), getHeight() - pt.y() - blockScale,
90                                   blockScale, blockScale);
91                      g2d.setPaint(Color.YELLOW.darker().darker());
92                      g2d.fillRect(pt.x() + tinyBlockScale, getHeight() - pt.y() -
  blockScale
93                                   / 2 - tinyBlockScale, blockScale / 2,
  blockScale / 2);
94                      g2d.setPaint(Color.BLACK);
95                      g2d.drawRect(pt.x() + tinyBlockScale, getHeight() - pt.y() -
  blockScale
96                                   / 2 - tinyBlockScale, blockScale / 2,
  blockScale / 2);
97                  }
98              }
99          }
100     }
101
102     /**
103      * Getting the data for the tetris game board.
104      *
105      * {@inheritDoc}
106      */
107     @Override
108     public void update(final Observable theObservable, final Object theData) {
109
110         if (theObservable instanceof Board && theData instanceof List) {
```

```
111             myBlockList.clear();
112             for (int i = 0; i < ((List<?>) theData).size(); i++) {
113                 final Block[] blockArray = (Block[]) ((List<?>) theData).get(i);
114                 myBlockList.add(blockArray);
115             }
116             repaint();
117         }
118     }
119 }
```