```
import java.awt.BorderLayout;
import java.awt.Image;
import java.awt.event.*;
import javax.swing.*;

/**
 * Game Core for Applet and Application
 * <P>
 *
 * @author Owen Astrachan
 */
public class GameCore extends JPanel implements IGameView
{
    protected GameModel myModel;
    protected GamePanel myGamePanel;
    protected JLabel myMessageLabel;
    protected JLabel myScoreLabel;
    /**
     * Create a view with a model
     * @param model is the model for this view
     */

    public GameCore(GameModel model, GamePanel panel)
    {
        setLayout(new BorderLayout());

        myModel = model;
        myGamePanel = panel;
        myMessageLabel = new JLabel(" ");
        myScoreLabel = new JLabel("",SwingConstants.RIGHT);
        setScore(0);
        add(myScoreLabel,  BorderLayout.NORTH);
        add(myGamePanel, BorderLayout.CENTER);
        add(myMessageLabel,BorderLayout.SOUTH);

        myModel.addView(this); // draws board
    }

    public void newGame()
    {
        myModel.initialize();
    }

    public void setScore(int score)
    {
        myScoreLabel.setText(score+" Score");
    }

    /**
     * Required by interface, show the grid that's the model
     * @param list represents the model
     */
    public void showGrid(GridPoint[][] list)
    {
        myGamePanel.setGrid(list);
    }

    public void highlight(GridPoint[] list, boolean show)
    {
        myGamePanel.highlight(list,show);
    }

    public void showMessage(String s)
    {
        myMessageLabel.setText(s);
    }

    public void gameOver()
    {
```



```
        JOptionPane.showMessageDialog(this,"game over!");
    }
}
```

```
import java.awt.Dimension;
import javax.swing.*;

/**
 * Clickomania Applet
 * <P>
 * @author Owen Astrachan
 */
public class ClickomaniaApplet extends JApplet
{
    private Dimension getDimension()
    {
        int rows = 6;
        int cols = 15;
        try{
            rows = Integer.parseInt(getParameter("rows"));
            cols = Integer.parseInt(getParameter("cols"));
        }
        catch (Exception e){
            // in case format bad or params missing
        }
        return new Dimension(rows,cols);
    }
    /**
     */
    public void init()
    {
        Dimension d = getDimension();
        GameModel model = new ClickomaniaModel(d.height,d.width);
        JPanel panel = new GameCore(model,new ClickomaniaPanel(model));
        setContentPane(panel);
        setVisible(true);
    }
}
```

```
import java.awt.event.ActionEvent;
import javax.swing.*;

/**
 * Clickomania main
 * <P>
 *
 * @author Owen Astrachan
 */
public class Clickomania
{
    public static void main(String args[])
    {
        ClickomaniaModel model = new ClickomaniaModel(10,10);
        ClickomaniaPanel panel = new ClickomaniaPanel(model);
        GameCore core = new GameCore(model, panel);
        GameGui gui = new GameGui(model, core, "CPS 108 Clickomania");
    }

}
```

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.awt.GridLayout;
import java.awt.Color;
import java.awt.Image;

public abstract class GamePanel extends JPanel
{
    protected JButton myButtons[][];
    protected GameModel myModel;
    protected Border myEmptyBorder;
    protected Border myShowBorder;
    /**
     * Create a button panel for the model
     */

    GamePanel(GameModel model)
    {
        // construct superclass and make button array
        super(new GridLayout(model.getRows(),model.getCols(),0,0));
        myButtons = new JButton[model.getRows()][model.getCols()];
        myModel = model;
        myEmptyBorder = BorderFactory.createEmptyBorder();
        myShowBorder = BorderFactory.createMatteBorder(4,4,4,4,Color.BLUE);
        initialize();
        makeButtons();
    }

    protected abstract void initialize();
    protected abstract JButton makeButton(int row, int col, int count);

    /**
     * Makes the buttons for this GUI/view. The number
     * of buttons is determined my the size of the array
     * myButtons. This helper function takes some of
     * the busy work out of the GamePanel constructor.
     */
    protected  void makeButtons()
    {
        int count = 0;
        for(int row = 0; row < myButtons.length; row++){
            for(int col = 0; col < myButtons[0].length; col++) {
                myButtons[row][col] = makeButton(row,col,count);
                count++;
                myButtons[row][col].setBorder(myEmptyBorder);
                add(myButtons[row][col]);
            }
        }
    }

    public abstract void doHighlight(GridPoint[] list, boolean show);

    public void showBorder(JComponent widget, boolean show)
    {
        if (show){
            widget.setBorder(myShowBorder);
        }
        else {
            widget.setBorder(myEmptyBorder);
        }
    }

    public void highlight(GridPoint[] list, boolean show)
    {
        doHighlight(list,show);
        repaint();
    }
```



```
    public abstract void doSetGrid(GridPoint[][] list);

    /**
     * Set all the buttons by re-displaying them all
     * in the right order. First we remove all the components
     * in this panel (that's the buttons). Then we add the
     * buttons to this panel in the right order based on what
     * the ordering of the buttons in the model is.
     * Finally, we revalidate so the buttons are shown (GUI
     * will redraw as a result of the revalidate).
     */
    public void setGrid(GridPoint[][] list)
    {
        doSetGrid(list);
        revalidate();
    }
}
```

```
import java.util.*;

public abstract class GameModel
{
    public static int INVALID = -1;
    protected static int DEFAULT_RANGE = 5;

    protected int myRows;
    protected int myCols;
    protected GridPoint[][] myGrid;
    protected IGameView myView;
    protected int myRange;
    protected ArrayList myList;
    protected GridPoint[] myArray;
    protected int  myScore;

    public GameModel(int rows, int cols, boolean paired)
    {
        this(rows,cols,DEFAULT_RANGE,paired);
    }

    public GameModel(int rows, int cols)
    {
        this(rows,cols,false);
    }

    public GameModel(int rows, int cols, int range)
    {
        this(rows,cols,range, false);
    }

    public GameModel(int rows, int cols, int range, boolean paired)
    {
        myRows = rows;
        myCols = cols;
        myRange = range;
        myGrid = new GridPoint[rows][cols];
        myList = new ArrayList();
        myArray = new GridPoint[0];
        initialize(paired);
    }

    public void initialize()
    {
        initialize(false);
    }

    public void initialize(boolean paired)
    {
        java.util.Random rando = new java.util.Random();
        clear();

        int total = getRows()*getCols();
        int incr = 1;
        if (paired){
            incr++;
        }
        for(int k=0; k < total; k += incr){
            int val = rando.nextInt(myRange);
            myList.add(new Integer(val));
            if (paired){
                myList.add(new Integer(val));
            }
        }
        Collections.shuffle(myList);

        int count = 0;
        for(int j=0; j < myRows; j++) {
            for(int k=0; k < myCols; k++) {
```



```
                myGrid[j][k] = new GridPoint(0,j,k);
                myGrid[j][k].setValue(((Integer)myList.get(count)).intValue());
                count++;
            }
        }
        clear();
        myScore = 0;
        showGrid();
        setScore();
    }

    protected boolean inBounds(int row, int col)
    {
        return 0 <= row && row < getRows() && 0 <= col && col < getCols();
    }

    public int getRows()
    {
        return myRows;
    }

    public int getCols()
    {
        return myCols;
    }

    public abstract boolean tryMove(Move move);

    public abstract boolean makeMove(Move move);

    protected void showMessage(String message)
    {
        myView.showMessage(message);
    }

    protected void clear()
    {
        myList.clear();
        myArray = new GridPoint[0];
    }

    public void addView(IGameView view)
    {
        myView = view;
        showGrid();
        setScore();
    }

    public void highlight(boolean value)
    {
        if (myView != null){
            myView.highlight(myArray,value);
        }
    }

    public void showGrid()
    {
        if (myView != null){
            myView.showGrid(myGrid);
        }
    }

    public void setScore()
    {
        if (myView != null){
            myView.setScore(myScore);
        }
    }
}
```