

```
import java.awt.Color;
import java.awt.Graphics;
public class MovePiece
{
    private Color color;
    private int x;
    private int y;

    public MovePiece(Color pColor, int pX, int pY)
    {
        color = pColor;
        x = pX;
        y = pY;
    }

    public void moveTo(int pX, int pY)
    {
        x = pX;
        y = pY;
    }

    public Color getColor()
    {
        return color;
    }

    public int getX()
    {
        return x;
    }

    public int getY()
    {
        return y;
    }

    public void draw(Graphics g, int squareSide)
    {
        g.setColor(color);
        g.fillOval(x - squareSide / 2, y - squareSide / 2, squareSide, squareSide);
    }
}
```

```
import java.awt.Color;
import java.awt.Graphics;
public class Piece
{
    private Color color;
    private int col;
    private int row;

    public Piece(Color pColor, int pCol, int pRow)
    {
        color = pColor;
        col = pCol;
        row = pRow;
    }

    public void moveTo(int pCol, int pRow)
    {
        col = pCol;
        row = pRow;
    }

    public Color getColor()
    {
        return color;
    }

    public void setColor(Color color)
    {
        this.color = color;
    }

    public int getCol()
    {
        return col;
    }

    public int getRow()
    {
        return row;
    }

    public void draw(Graphics g, int squareSide)
    {
        g.setColor(color);
        g.fillRect(col * squareSide, row * squareSide, squareSide, squareSide);
    }
}
```

```

import java.util.ArrayList;
import java.awt.Color;
import java.awt.Graphics;
public class Checkers
{
    private ArrayList<Piece> alPieces = new ArrayList<>();

    // Pièce en déplacement/mouvement (colorée en gris)
    private MovePiece inMove = null;

    public Checkers()
    {
        for (int r = 0; r < 8; r++)
        {
            for (int c = 0; c < 8; c++)
            {
                if ((r + c) % 2 == 1)
                    if (r < 3)
                        alPieces.add(new Piece(Color.BLUE, c, r));
                    else if (r >= 5)
                        alPieces.add(new Piece(Color.RED, c, r));
            }
        }
    }

    public Piece getPieceAt(int pCol, int pRow)
    {
        Piece p = null;
        boolean found = false;
        int i = 0;
        while (!found && i < alPieces.size())
        {
            p = alPieces.get(i);
            if ((pCol == p.getCol()) && (pRow == p.getRow()))
                found = true;
            else
                i++;
        }
        if (found)
            return alPieces.get(i);
        else
            return null;
    }

    public boolean move(int pStartCol, int pStartRow, int pDestCol, int pDestRow)
    {
        Piece p = getPieceAt(pStartCol, pStartRow);
        if (p == null)
            return false; // Pas de pièce à cette position de départ

        // Enlever la pièce en mouvement
        inMove = null;

        // Pièce existe
        if ((pDestCol < 0) || (pDestRow < 0) || (pDestCol > 7) || (pDestRow > 7))
        {
            // Position en dehors du damier --> effacer la pièce
            alPieces.remove(p);
            return true;
        }

        if (getPieceAt(pDestCol, pDestRow) == null)
        {
            // Pas de pièce à la position d'arrivée --> on bouge la pièce
            p.moveTo(pDestCol, pDestRow);
            return true;
        }

        return false;
    }

    public void beginMove(Color color, int pX, int pY)
    {
        inMove = new MovePiece(color, pX, pY);
    }

    public void doMove(int pX, int pY)
    {
        // Bouge la pièce en déplacement vers la nouvelle position
        if (inMove != null)
            inMove.moveTo(pX, pY);
    }

    // Suite page suivante...

```

```
public void draw(Graphics g, int cellSize)
{
    // Dessiner la grille
    int c, r;
    for (int i = 0; i < 8; i++)
    {
        r = (i * cellSize);
        for (int j = 0; j < 8; j++)
        {
            c = (j * cellSize);
            if ((i + j) % 2 == 1)
                g.setColor(Color.GRAY);
            else
                g.setColor(Color.WHITE);
            g.fillRect(c, r, cellSize, cellSize);
            g.setColor(Color.BLACK);
            g.drawRect(c, r, cellSize, cellSize);
        }
    }

    // Dessiner les pièces... chaque pièce se dessine elle-même!
    for (int i = 0; i < alPieces.size(); i++)
        alPieces.get(i).draw(g, cellSize);

    // Dessiner la pièce en déplacement
    if (inMove != null)
        inMove.draw(g, cellSize);
}
```

```
import java.awt.Color;
import java.awt.Graphics;
public class DrawPanel extends javax.swing.JPanel
{
    private Checkers checkers = null;
    private int squareSide = 0;

    public void setCheckers(Checkers checkers)
    {
        this.checkers = checkers;
    }

    public int getSquareSide()
    {
        return squareSide;
    }

    public DrawPanel()
    {
        initComponents();
    }

    public void paintComponent(Graphics g)
    {
        int w = getWidth();
        int h = getHeight();

        // Peindre l'arrière fond
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, w, h);

        int l = Math.min(w, h) - 1;
        squareSide = l / 8;

        // Dessiner le damier et les pièces... s'il existe
        if (checkers != null)
            checkers.draw(g, squareSide);
    }
}
// Skipped: ... initComponents { ... }
// Variables declaration - do not modify//GEN-BEGIN:variables
// End of variables declaration//GEN-END:variables
```

```

import java.awt.Color;
public class MainFrame extends javax.swing.JFrame
{
    private Checkers checkers = null;
    private Piece piece = null;
    private Color realColor = null;

    public MainFrame()
    {
        initComponents();
    }
// Skipped: ... initComponents { ... }
    private void newButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_newButtonActionPerformed
    { //GEN-HEADEREND:event_newButtonActionPerformed
        checkers = new Checkers();
        drawPanel.setCheckers(checkers);

        repaint();
    } //GEN-LAST:event_newButtonActionPerformed

    private void drawPanelMousePressed(java.awt.event.MouseEvent evt) { //GEN-FIRST:event_drawPanelMousePressed
        if (checkers != null)
        {
            int col = evt.getX() / drawPanel.getSquareSide();
            int row = evt.getY() / drawPanel.getSquareSide();

            piece = checkers.getPieceAt(col, row);
            if (piece != null)
            {
                realColor = piece.getColor();
                piece.setColor(realColor.darker().darker());
                checkers.beginMove(realColor, evt.getX(), evt.getY());
                msgLabel.setText("-");
            }
            else
            {
                msgLabel.setText("No piece at that position!");
            }
            repaint();
        }
        else
        {
            msgLabel.setText("No game started!");
        }
    } //GEN-LAST:event_drawPanelMousePressed

    private void drawPanelMouseReleased(java.awt.event.MouseEvent evt) { //GEN-FIRST:event_drawPanelMouseReleased
        if ((checkers != null) && (piece != null))
        {
            int col = evt.getX() / drawPanel.getSquareSide();
            int row = evt.getY() / drawPanel.getSquareSide();

            if (checkers.move(piece.getCol(), piece.getRow(), col, row))
                msgLabel.setText("-");
            else
                msgLabel.setText("Invalid move!");
            piece.setColor(realColor);
            piece = null;

            repaint();
        }
    } //GEN-LAST:event_drawPanelMouseReleased

    private void drawPanelMouseDragged(java.awt.event.MouseEvent evt) //GEN-FIRST:event_drawPanelMouseDragged
    { //GEN-HEADEREND:event_drawPanelMouseDragged
        if ((checkers != null) && (piece != null))
        {
            checkers.doMove(evt.getX(), evt.getY());
            repaint();
        }
    } //GEN-LAST:event_drawPanelMouseDragged
// Skipped: ... Look & Feel
// Variables declaration - do not modify //GEN-BEGIN:variables
    private DrawPanel drawPanel;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel msgLabel;
    private javax.swing.JButton newButton;
// End of variables declaration //GEN-END:variables
}

```