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

    public Piece(Color color, int col, int row)
    {
        this.color = color;
        this.col = col;
        this.row = row;
    }

    public void moveTo(int destCol, int destRow)
    {
        col = destCol;
        row = destRow;
    }

    public Color getColor()
    {
        return color;
    }

    public int getCol()
    {
        return col;
    }

    public int getRow()
    {
        return row;
    }

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

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

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

    public boolean move(int startCol, int startRow, int destCol, int destRow)
    {
        Piece p = getPieceAt(startCol, startRow);
        if (p == null)
            return false;           // Pas de pièce à cette position de départ

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

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

        return false;
    }

    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);
    }
}
```

```java
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();

        g.setColor(Color.WHITE);
        g.fillRect(0, 0, w, h);

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

        if (checkers != null)
            checkers.draw(g, squareSide);
    }
// Skipped: ... initComponents { ... }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
}
```

```java
import java.awt.Point;
public class MainFrame extends javax.swing.JFrame
{
    private Checkers checkers = null;
    private Point startPoint = 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
        startPoint = evt.getPoint();
    }//GEN-LAST:event_drawPanelMousePressed

    private void drawPanelMouseReleased(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_drawPanelMouseReleased
        if (checkers != null)
        {
            int s = drawPanel.getSquareSide();
            int srcCol = startPoint.x / s;
            int srcRow = startPoint.y / s;
            int destCol = evt.getX() / s;
            int destRow = evt.getY() / s;

            if (checkers.move(srcCol, srcRow, destCol, destRow))
            {
                msgLabel.setText("-");
            }
            else
            {
                msgLabel.setText("Invalid move!");
            }

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