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

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

        visible = true;
    }

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

    public void setVisible(boolean visible)
    {
        this.visible = visible;
    }

    public Color getColor()
    {
        return color;
    }

    public int getCol()
    {
        return col;
    }

    public int getRow()
    {
        return row;
    }

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

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

    public MovePiece(Color color, int x, int y)
    {
        this.color = color;
        this.x = x;
        this.y = y;
    }

    public void moveTo(int destX, int destY)
    {
        x = destX;
        y = destY;
    }

    public Color getColor()
    {
        return color;
    }

    public int getX()
    {
        return x;
    }

    public int getY()
    {
        return y;
    }

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

```java
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 bleu ou rouge)
    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 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(Piece p, int destCol, int destRow)
    {
        inMove = null;
        p.setVisible(true);

        // 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 beginMove(Piece p, int x, int y)
    {
        inMove = new MovePiece(p.getColor(), x, y);
        p.setVisible(false);
    }

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





        // suite ... prochaine page
```

```java
    public void draw(Graphics g, int offsetLeft, int offsetTop, 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(offsetLeft + c, offsetTop + r, cellSize, cellSize);
                g.setColor(Color.BLACK);
                g.drawRect(offsetLeft + c, offsetTop + 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, offsetLeft, offsetTop, cellSize);

        // dessiner la piece en mouvement
        if (inMove != null)
            inMove.draw(g, offsetLeft, offsetTop, 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;
    private int offsetLeft = 0;
    private int offsetTop = 0;

    public int getOffsetLeft()
    {
        return offsetLeft;
    }

    public int getOffsetTop()
    {
        return offsetTop;
    }

    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);

        // Attention au pixel en plus utilisé par la méthode drawRect !
        int l = Math.min(w, h) - 1;
        squareSide = l / 8;
        offsetLeft = (w - (squareSide*8+1)) / 2;
        offsetTop = (h - (squareSide*8+1)) / 2;

        if (checkers != null)
            checkers.draw(g, offsetLeft, offsetTop, squareSide);
    }
// Skipped: ... initComponents { ... }
// Skipped: ... graphic attributes
}
```

```java
public class MainFrame extends javax.swing.JFrame
{
    private Checkers checkers = null;
    private Piece srcPiece = 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
    {//GEN-HEADEREND:event_drawPanelMousePressed
        if (checkers != null)
        {
            int x, y;
            x = evt.getX() - drawPanel.getOffsetLeft();
            y = evt.getY() - drawPanel.getOffsetTop();

            int s = drawPanel.getSquareSide();

            int col, row;
            if (evt.getX() < drawPanel.getOffsetLeft())
                col = -1;
            else
                col = x / s;

            if (evt.getY() < drawPanel.getOffsetTop())
                row = -1;
            else
                row = y / s;

            srcPiece = checkers.getPieceAt(col, row);
            if (srcPiece != null)
            {
                checkers.beginMove(srcPiece, x, y);
                msgLabel.setText("-");
            }
            else
                msgLabel.setText("No piece at the position!");
            repaint();
        }
    }//GEN-LAST:event_drawPanelMousePressed

    private void drawPanelMouseDragged(java.awt.event.MouseEvent evt)//GEN-FIRST:event_drawPanelMouseDragged
    {//GEN-HEADEREND:event_drawPanelMouseDragged
        if (checkers == null)
            return;

        int x, y;
        x = evt.getX() - drawPanel.getOffsetLeft();
        y = evt.getY() - drawPanel.getOffsetTop();

        checkers.doMove(x, y);
        repaint();
    }//GEN-LAST:event_drawPanelMouseDragged

    private void drawPanelMouseReleased(java.awt.event.MouseEvent evt)//GEN-FIRST:event_drawPanelMouseReleased
    {//GEN-HEADEREND:event_drawPanelMouseReleased
        if (srcPiece == null)
            return;

        int x, y;
        x = evt.getX() - drawPanel.getOffsetLeft();
        y = evt.getY() - drawPanel.getOffsetTop();

        int s = drawPanel.getSquareSide();

        int col, row;
        if (evt.getX() < drawPanel.getOffsetLeft())
            col = -1;
        else
            col = x / s;

        if (evt.getY() < drawPanel.getOffsetTop())
            row = -1;
        else
            row = y / s;

        if (checkers.move(srcPiece, col, row))
            msgLabel.setText("-");
        else
            msgLabel.setText("Invalid move");

        // terminé
        srcPiece = null;
        repaint();
    }//GEN-LAST:event_drawPanelMouseReleased
// Skipped: ... Look & Feel
// Skipped: ... graphic attributes
}
```